IBM WebSphere Business Integration Server Express e
Express Plus

Guida allo sviluppo delle mappe

Versione 44

IBM WebSphere Business Integration Server Express e
Express Plus

Guida allo sviluppo delle mappe

Versione 44

Nota!
Prima di utilizzare queste informazioni ed il prodotto supportato, consultare “Informazioni particolari” a pagina 531.

22 aprile 2005

Questa edizione del documento si applica a IBM WebSphere Business Integration Server Express versione *4.4*, IBM WebSphere Business Integration Server Express Plus versione 4.4, Toolset Express versione *4.4* e a tutti i rilasci e modifiche successive se non diversamente indicato nelle nuove edizioni.

Per inviare commenti su questa documentazione, inviare un messaggio di posta elettronica all’indirizzo doc-comments@us.ibm.com. Siamo in attesa di ricevere i vostri commenti.

Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente dall’IBM e diventeranno esclusiva della stessa.

Indice

Informazioni

I prodotti IBM(R) WebSphere Business Integration Server Express e IBM(R) WebSphere Business Integration Server Express Plus comprendono i seguenti componenti: Interchange Server Express, Toolset Express associato, CollaborationFoundation e una serie di adattatori di integrazione software. Gli strumenti di Toolset Express consentono di creare, modificare e gestire i processi di business. E' possibile scegliere tra diversi adattatori preconfigurati per i processi di business che utilizzano le applicazioni. La maschera dei processi standard, CollaborationFoundation, consente di creare rapidamente processi personalizzati.

Questo documento fornisce un'introduzione all'utilizzo delle mappe e delle relazioni e descrive il modo in cui utilizzare Map Designer Express e Relationship Designer Express per la creazione e la modifica di mappe e relazioni.

Se non diversamente indicato, tutte le informazioni contenute in questa guida si applicano sia a WebSphere Business Integration Server Express sia a IBM WebSphere Business Integration Server Express Plus. Il termine "WebSphere Business Integration Server Express" e le relative varianti fa riferimento a entrambi i prodotti.

A chi è rivolto questo manuale

Questo documento è rivolto agli sviluppatori di connettori, di collaborazioni e ai consulenti di IBM WebSphere che creano o modificano le definizioni degli oggetti business o le mappe.

Utilizzo di questo manuale

Questo manuale è organizzato nel modo seguente.

Parte I: Mappe	
Capitolo 1, "Introduzione allo sviluppo delle mappe"	E' una panoramica delle mappe e degli strumenti di mappatura di WebSphere Business Integration.
Capitolo 2, "Creazione di mappe"	Fornisce un'introduzione all'utilizzo di Map Designer Express per la creazione e la modifica delle mappe.
Capitolo 3, "Utilizzo delle mappe"	Descriva alcune funzioni avanzate di Map Designer Express da utilizzare dopo la creazione delle mappe.
Capitolo 4, "Compilazione e verifica delle mappe"	Descrive il modo in cui compilare una mappa nella relativa forma eseguibile e il modo in cui eseguire un test per verificare la correttezza della mappa stessa.
Capitolo 5, "Personalizzazione di una mappa"	Descrive il modo in cui implementare le mappe.
Parte II: Relazioni	
Capitolo 6, "Introduzione alle relazioni"	Fornisce un'introduzione alle relazioni, compresi i tipi di relazioni supportate da WebSphere Business Integration e il modo in cui il sistema implementa una relazione.

Documentazione correlata

La serie completa di manuali descrive le funzioni e i componenti comuni a tutte le installazioni di WebSphere Business Integration Server Express e WebSphere Business Integration Server Express Plus e comprende i materiali di riferimento per componenti specifici.

È possibile scaricare, installare e visualizzare la documentazione al seguente indirizzo: http://www.ibm.com/websphere/wbiserverexpress/infocenter.

Nota: Le informazioni rilevanti su questo prodotto sono disponibili nel documento Technical Support Technotes and Flashes emesso in seguito alla pubblicazione di questo manuale. Tali documenti sono disponibili sul sito Web di supporto di WebSphere Business Integration all’indirizzo http://www.ibm.com/software/integration/websphere/support/. Selezionare l’area dei componenti di interesse e passare alle sezioni Technotes and Flashes.

Convenzioni tipografiche

Questo manuale utilizza le seguenti convenzioni:

`carattere courier`	Indica un valore letterale, come ad esempio il nome di un comando, le informazioni immesse o le informazioni che il sistema stampa sullo schermo.
corsivo o `corsivo`	Indica il nome di una variabile, il nome del titolo o il nuovo termine la prima volta in cui viene visualizzato

riquadro blu	Uno schema blu, visibile solo quando si visualizza il manuale in linea, indica un collegamento ipertestuale a riferimento incrociato. Fare clic all'interno del riquadro per passare all'oggetto del riferimento.
ProductDir	Rappresenta la directory in cui viene installato il prodotto.

Novità di questo rilascio

Questa sezione descrive le funzioni nuove e modificate di IBM WebSphere Business Integration Server Express e IBM WebSphere Business Integration Server Express Plus ed i relativi programmi associati per lo sviluppo delle mappe e delle relazioni riportate in questo documento.

Novità nel rilascio 4.4

Per questo rilascio sono state apportate le seguenti modifiche al manuale

- Supporto fornito per la creazione dell'istanza nella scheda Diagramma.
- Metodo updateParticipantByInstance() non supportato e relativa descrizione eliminata dalla classe Relationship.
- Miglioramento del blocco gruppo Activity Editor con rappresentazione ad icona per migliore gestione dei componenti durante riutilizzo del gruppo attività.
- Supporto fornito per la configurazione di blocchi funzione standard nella finestra Preferenze per uso diretto in Map Designer Express.
- Supporto fornito per la funzione lingua bidirezionale per la denominazione delle mappe.
- Ulteriore supporto fornito per la gestione di funzioni primitive nella classe DtpDataConversion.
- Nuovo capitolo fornito per la classe CwBidiEngine.
- Nuovo capitolo fornito per la classe CxExecutionContext.

Novità nel rilascio 4.3.1

Il manuale non è stato modificato nel rilascio 4.3.1.

Novità del rilascio 4.3

Questo è il primo rilascio di questo manuale.

Parte 1. Mappe

Capitolo 1. Introduzione allo sviluppo delle mappe

Benvenuti nel processo di sviluppo delle mappe. Questo capitolo fornisce un sommario della mappatura dei dati, introduce gli strumenti da utilizzare per implementare le mappe e descrive le definizioni di relazione e di mappa.

Questo capitolo descrive i seguenti argomenti:

Informazioni sulla mappatura dati

La mappatura dati è il processo di trasformazione (o mappatura) dei dati da un formato applicativo specifico ad un altro. L'operazione di mappatura è fondamentale nel processo di trasferimento di informazioni tra diverse applicazioni e nel fornire collaborazioni (processi business) che sono indipendenti da applicazioni specifiche. Con la mappatura di dati tra oggetti business specifici dell'applicazione e oggetti business generici, WebSphere crea l'ambiente che consente l'utilizzo di applicazioni "avanzate". Il sistema WebSphere Business Integration Server Express fornisce un'architettura modulare ed estensibile per una più facile manutenzione delle mappe.

Il sistema di sviluppo delle mappe WebSphere fornisce un supporto completo per la mappatura tra oggetti business, con l'aggiunta delle seguenti funzioni:

- Trasformazione dei valori dati da uno o più attributi in un oggetto business sorgente ad uno o più attributi in un oggetto business di destinazione
- Realizzazione e conservazione delle relazioni tra entità dati equivalenti ma rappresentate diversamente e non direttamente trasformabili
- Abilitazione dell'accesso a risorse di mappatura esterne, come prodotti di mappatura di terze parti e database per l'esecuzione delle query

Quando si imposta la mappatura dei dati per diverse applicazioni, un evento in un'applicazione viene eseguito in tutte le altre applicazioni a cui è mappato. Un evento può essere la creazione, il richiamo, l'aggiornamento o l'eliminazione di dati.

La funzione di mappatura utilizza le *mappe* che definiscono il trasferimento (o trasformazione) dei dati tra gli oggetti business sorgenti e di destinazione. In un ambiente di sviluppo delle mappe, i dati vengono mappati da un oggetto business specifico dell'applicazione a un oggetto business generico oppure da un oggetto business generico a un oggetto business specifico dell'applicazione. Tabella 1 elenca i tipi di mappatura richiesti.

Tabella 1. Requisiti di mappatura

Direzione dell'oggetto business	Oggetto business sorgente	Oggetto business di destinazione	Tipo di mappa
Connettore a collaborazione	Specifico dell'applicazione	Generico	Mappa in entrata

Tabella 1. Requisiti di mappatura (Continua)

Direzione dell'oggetto business	Oggetto business sorgente	Oggetto business di destinazione	Tipo di mappa
Collaborazione a connettore	Generico	Specifico dell'applicazione	Mappa in uscita

Esempio: Figura 1 illustra come viene eseguita la mappatura durante la fase di runtime, utilizzando una collaborazione Employee Management di finzione come esempio.

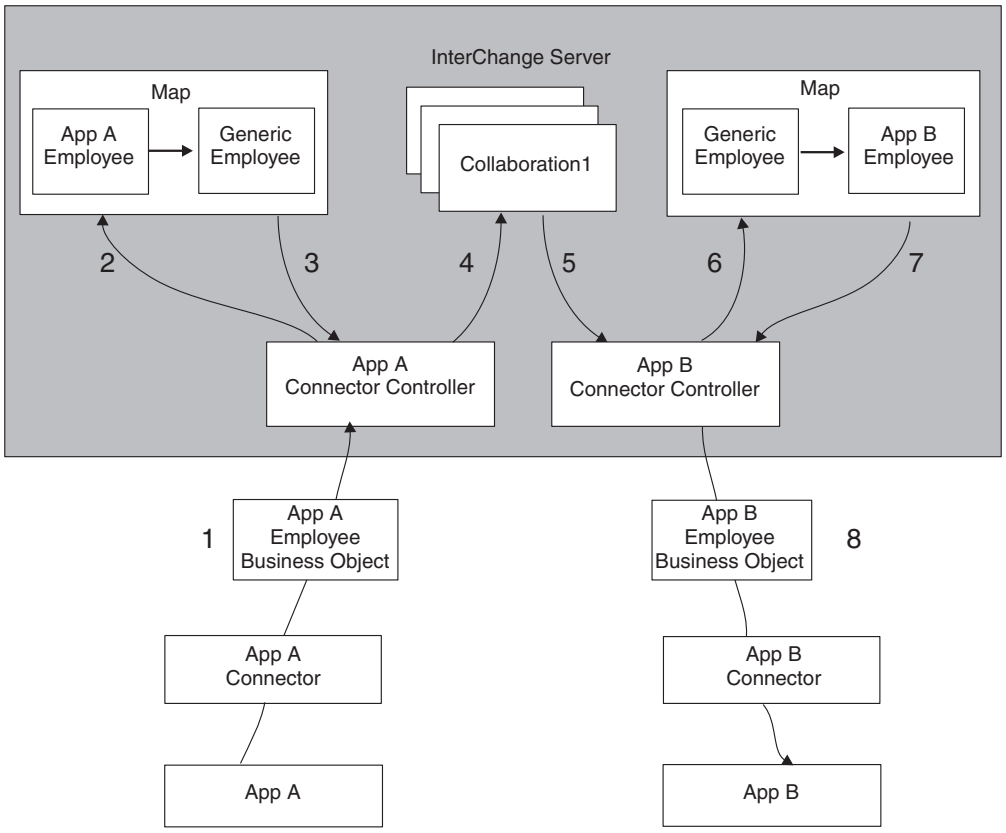


Figura 1. Mappatura dati in fase runtime

La collaborazione Employee Management (`Collaboration1`) riceve un oggetto business `Employee` dal connettore sorgente (`App A`), quindi invia un oggetto business `Employee` al connettore di destinazione (`App B`). Figura 1 illustra la sequenza riportata di seguito (i numeri corrispondono a quelli presenti nella figura):

1. Si è verificato un evento in `App A`. Il connettore `App A` crea un oggetto business `App A Employee` e lo invia al controllore del connettore `App A`.
2. Il controllore del connettore `App A` invia l'oggetto business `App A Employee` alla funzione di collaborazione Employee Management (`Collaboration1`), che risiede in InterChange Server Express, per la mappatura. La richiesta include il nome della mappa dati che il server deve utilizzare, in base al nome mappa specificato nella configurazione del connettore.
3. La mappa in entrata restituisce l'oggetto business `Employee` generico al controllore del connettore `App A`.

4. Il controllore del connettore App A esamina le collaborazioni che hanno sottoscrizioni per l'oggetto business Employee generico. In questo caso, Collaboration1 ha una sottoscrizione, per cui il controllore del connettore cede l'oggetto business a Collaboration1.
5. La collaborazione esegue delle elaborazioni, quindi genera un altro oggetto business Employee generico come output, che invia al controllore del connettore App B.
6. Il controllore del connettore App B invia l'oggetto business generico a InterChange Server Express, richiedendo la mappatura per l'oggetto business App B Employee.
7. La mappa in uscita restituisce l'oggetto business Employee specifico dell'applicazione al controllore del connettore App B.
8. Il controllore del connettore App B trasferisce l'oggetto App B Employee al connettore App B, che può quindi trasferire i dati dell'oggetto business a App B.

La figura mostra due tipi di mappe in uso:

- Una mappa in entrata dall'oggetto business App A Employee all'oggetto business Employee generico utilizzata dalla funzione di collaborazione.
- Una mappa in uscita dall'oggetto business Employee generico all'oggetto business App B Employee

I dati Employee si spostano in una sola direzione, da Application A verso Application B. Se si desidera scambiare i dati Employee in entrambe le direzioni tra le due applicazioni, sono richieste due ulteriori mappe:

- Una mappa in entrata dall'oggetto business specifico dell'applicazione di Application B all'oggetto business generico
- Una mappa in uscita dall'oggetto business generico all'oggetto business specifico dell'applicazione di Application A

Mappe: un controllo più dettagliato

Come Tabella 2 illustra, una mappa è un'entità composta da due parti, una definizione mappe e un oggetto runtime.

Definizione della mappa

Il sistema di sviluppo delle mappe utilizza la funzione di *definizione della mappa*. Le definizioni delle mappe sono memorizzate nei progetti del System Manager. Il programma Map Designer Express fornisce delle finestre di dialogo che consentono di creare le definizioni delle mappe (spesso indicate semplicemente come mappe). Inoltre, gestisce la memorizzazione della definizione mappa completa nei progetti del System Manager.

Per ulteriori informazioni su come utilizzare Map Designer Express per creare le definizioni delle mappe, consultare "Creazione di una mappa: Passi di base" a pagina 33..

Nella definizione di una mappa vengono fornite le seguenti informazioni:

- Il nome della mappa
- Gli oggetti sorgenti e di destinazione della mappa
- Le trasformazioni della mappa

Nome di definizione della mappa

Una definizione di mappa è semplicemente una maschera o descrizione della mappa. Fornisce informazioni sulle modalità di trasformazione degli attributi da un oggetto business ad un altro. Pertanto, il nome definizione della mappa dovrebbe indicare la direzione della mappa e gli oggetti business che trasforma.

Oggetti business sorgenti e di destinazione

Le mappe sono composte da uno o più oggetti business sorgenti e da uno o più oggetti business di destinazione. Gli *oggetti business sorgenti* sono gli oggetti da trasformare; gli *oggetti business di destinazione* sono gli oggetti generati con dati provenienti dagli oggetti business sorgenti.

Trasformazioni delle mappe

Il resto della mappa è costituito da una serie di operazioni di trasformazione. Un'*operazione di trasformazione* è un segmento del codice Java che restituisce il valore di un attributo di destinazione. Una mappa contiene un'operazione di trasformazione per ogni attributo di destinazione trasformato. Le trasformazioni sono realizzate come codice Java e sono pertanto memorizzate in un file sorgente Java (`.java`).

Tabella 2 mostra alcune delle trasformazioni che si possono eseguire su un oggetto business di destinazione. Le trasformazioni standard includono Imposta valore, Sposta, Aggiungi, Dividi, Mappa secondaria e Riferimento incrociato. E' possibile creare delle trasformazioni personalizzate con blocchi di funzioni grafiche, così come con il codice Java per: ″Relazioni,″ ″Logica basata sul contenuto,″ ″Conversione data″ e ″Trasformazioni stringhe.″

Tabella 2. Trasformazioni di una mappa

Trasformazione	Descrizione	Per ulteriori informazioni
Trasformazioni standard	Trasformazioni per cui il Map Designer Express può generare automaticamente il codice	
Imposta valore	Specifica di un valore per l'attributo di destinazione	"Specifica del valore per un attributo" a pagina 40
Sposta (Copia)	Copia di un attributo sorgente nell'attributo di destinazione	"Copia di un attributo di origine ad un attributo di destinazione" a pagina 42
Aggiungi	Unione di due o più attributi sorgente in un unico attributo di destinazione	"Aggiunta di attributi" a pagina 43
Dividi	Suddivisione di un attributo sorgente in due o più attributi di destinazione	"Divisione degli attributi" a pagina 45
Mappa secondaria	Chiamata di una mappa per un oggetto business secondario	"Trasformazione con una mappa secondaria" a pagina 47
Riferimento incrociato	Gestione delle relazioni d'identita per gli oggetti business	"Relazioni di identità con riferimento incrociato" a pagina 51
Trasformazioni personalizzate	Creazione di una trasformazione diversa da una delle trasformazioni standard sopra elencate	"Creazione di una trasformazione personalizzata" a pagina 52
Relazione	Associazione di oggetti business di cui non si può eseguire direttamente la mappatura poiché ogni applicazione conserva i dati nel proprio formato	Capitolo 8, "Implementazione delle relazioni", a pagina 271
Logica basata sul contenuto	Trasformazione di un attributo di destinazione basato sul contenuto dell'attributo sorgente	"Logica basata sul contenuto" a pagina 183
Conversione della data	Conversione di una data dal proprio formato nell'attributo sorgente al formato nell'attributo di destinazione	"Formattazione della data" a pagina 188

Tabella 2. Trasformazioni di una mappa (Continua)

Trasformazione	Descrizione	Per ulteriori informazioni
Stringa	Esecuzione di trasformazioni base di una stringa, come la conversione maiuscolo/minuscolo e l'ottenimento delle stringhe secondarie	"Utilizzo del generatore di espressioni per le trasformazioni delle stringhe" a pagina 191

Quando si verifica una chiara corrispondenza tra l'attributo sorgente e quello di destinazione, l'operazione di trasformazione copia semplicemente il valore sorgente nell'attributo di destinazione. Altre trasformazioni possono interessare il calcolo, l'elaborazione delle stringhe, la conversione del tipo dati e una qualsiasi altra funzione logica che è possibile codificare utilizzando Java.

Figura 2 illustra alcuni particolari tipi di trasformazioni di attributo:

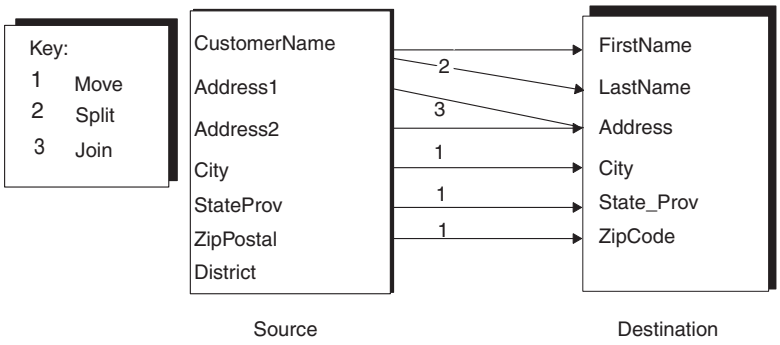


Figura 2. Trasformazioni particolari di attributo

Come Figura 2 mostra, gli attributi dall'oggetto business sorgente sono:

- Copiati in un attributo di destinazione (`City`, `StateProv`, `ZipPostal`).
- Divisi in più attributi di destinazione (`CustomerName`).
- Aggiunti (uniti) in un attributo di destinazione (`Address1`, `Address2`).
- Ignorati quando l'oggetto di destinazione non ha un attributo equivalente (`District`).

Per trasformazioni semplici come la copia di un valore in un attributo, la suddivisione di un valore in due o più attributi oppure l'unione di due o più valori in un attributo, è possibile specificare la procedura in modo grafico e Map Designer Express genera il codice Java. Per trasformazioni più complesse, è possibile personalizzare la trasformazione con un editor grafico o scrivere il proprio codice Java.

Istanza mappa

La definizione della mappa è una maschera per la creazione istanza runtime della mappa, l'*istanza mappa*. Durante l'esecuzione della mappa, il sistema di sviluppo delle mappe crea le istanze della mappa in base alla definizione della mappa e al codice di trasformazione.

Ogni istanza della mappa fornisce le seguenti informazioni:

- La funzionalità di base come la registrazione, la traccia, le connessioni e la gestione delle eccezioni tramite i metodi della classe `BaseDLM`

- Il contesto di esecuzione della mappa

  Per ulteriori informazioni, consultare “Contesti di esecuzione delle mappe” a pagina 200.

Un'istanza mappa è rappresentata nella mappatura API da un'istanza della classe `BaseDLM`.

Strumenti per lo sviluppo delle mappe

Tabella 3 mostra i due strumenti di progettazione grafica della mappatura.

Tabella 3. Componenti principali del sistema di mappatura dati

Strumento di progettazione	Componente mappatura	Descrizione
Map Designer Express	Mappa	Utilizza il codice Java per specificare in che modo trasformare gli attributi da uno o più oggetti business sorgenti ad uno o più oggetti business di destinazione. Normalmente si crea una mappa per ogni oggetto business sorgente che si desidera trasformare, sebbene sia anche possibile suddividere una mappa in diverse mappe secondarie.
Relationship Designer Express	Relazione	Stabilisce un'associazione tra due o più entità di dati nel sistema di sviluppo delle mappe. Le definizioni di relazione associano frequentemente due o più oggetti business. Le definizioni di relazione vengono utilizzate per trasformare dati equivalenti presenti negli oggetti business ma che vengono rappresentati diversamente. Ad esempio, un codice per lo Stato del Michigan potrebbe essere rappresentato come `MI` in un'applicazione e `MICH` in un'altra. Questi dati sono equivalenti ma rappresentati diversamente in ciascuna applicazione. La maggior parte delle mappe utilizzano una o alcune definizioni di relazione.

Questi strumenti di grafica vengono eseguiti su Windows 2003 e Windows XP. Pertanto, queste piattaforme vengono utilizzate per lo sviluppo di mappe.

Tabella 4 elenca gli strumenti aggiuntivi supportati per lo sviluppo delle mappe.

Tabella 4. Strumenti per lo sviluppo delle mappe

Programma	Descrizione
Mappatura API	Un insieme di classi Java con le quali è possibile personalizzare il codice di mappatura creato.
System Manager	Fornisce delle finestre grafiche per la configurazione di un'istanza mappe oltre alla configurazione di un oggetto di relazione.

Map Designer Express

Map Designer Express crea e compila le mappe. E' possibile avviare il Map Designer Express da System Manager selezionandolo dal menu degli strumenti. Per ulteriori informazioni sulle modalità di avvio di Map Designer Express, consultare “Avvio di Map Designer Express” a pagina 16.. Map Designer Express fornisce una finestra schede per visualizzare le informazioni sulla mappa. Questa finestra visualizza una di quattro schede: Tabella, Diagramma, Messaggi o Testo.

Figura 3 mostra una mappa visualizzata nella scheda Diagramma di Map Designer Express.

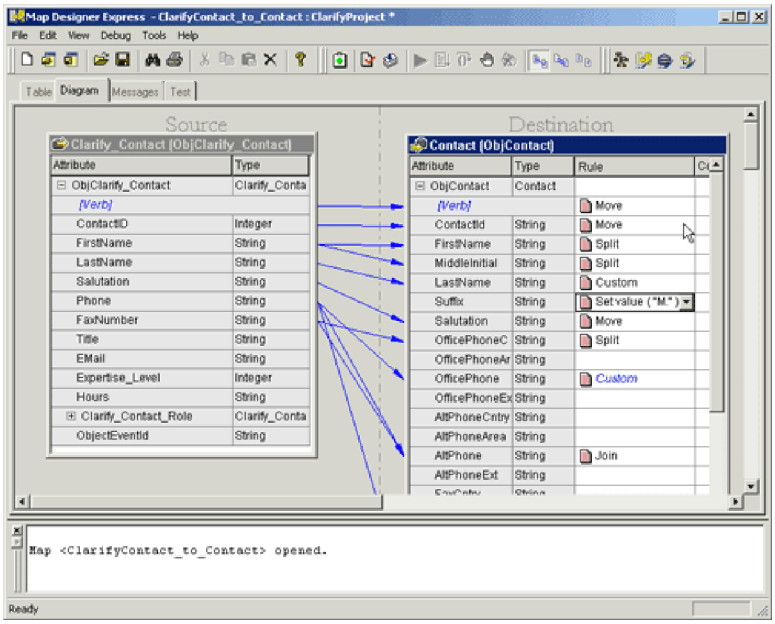


Figura 3. Map Designer Express

Per ulteriori informazioni su come utilizzare il Map Designer Express per creare una mappa, consultare Capitolo 2, “Creazione di mappe”, a pagina 15.

Relationship Designer Express

Relationship Designer Express crea le definizioni di relazione che memorizzano i dati dell’istanza di relazione in fase di runtime. E’ possibile avviare il Relationship Designer Express da System Manager selezionandolo dal menu degli strumenti. Figura 4 mostra diverse relazioni visualizzate in Relationship Designer Express.

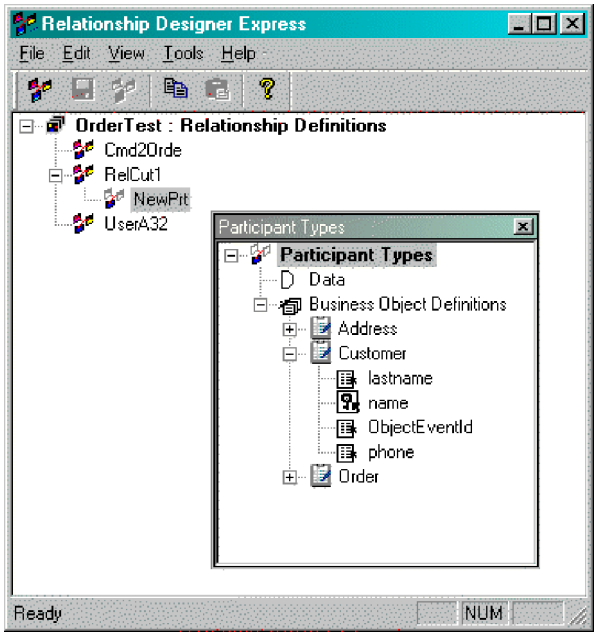


Figura 4. Relationship Designer Express

Per ulteriori informazioni su come utilizzare il Relationship Designer Express, consultare Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249.

Mappatura API

Molte fasi di trasformazione possono essere programmate utilizzando i metodi Java standard. Per rendere più facile le fasi di trasformazioni della scrittura, il sistema di sviluppo delle mappe fornisce un’API di mappatura (descrizione dettagliata in Parte 3, “Riferimento della mappatura API”, a pagina 331), con metodi per gestire le più comuni condizioni di trasformazione dati. La mappatura API include le seguenti classi:

- Le classi DTP (Data Transformation Package) forniscono dei metodi per l’elaborazione delle stringhe, la conversione del tipo dati, l’elaborazione della data, la chiamata alle mappe secondarie e l’esecuzione della query SQL. Le classi sono:
  - `DtpConnection` (obsoleto)
  - `DtpDataConversion`
  - `DtpDate`
  - `DtpMapService`
  - `DtpSplitString`
  - `DtpUtils`
- Le classi di oggetti business sono utilizzate per lo sviluppo di collaborazioni e la mappatura. Le classi sono:
  - `BusObj`
  - `BusObjArray`
- Le classi di gestione delle relazioni forniscono dei metodi per la creazione e la gestione delle istanze di relazione. Le classi sono:
  - `Participant`
  - `Relationship`
  - `IdentityRelationship`
- Le classi di connessione del database forniscono dei metodi per l’esecuzione della query SQL. Queste classi sono:
  - `CwDBConnection`
  - `CwDBStoredProcedureParam`
  - `DtpConnection` (obsoleto)
  - `UserStoredProcedureParam` (obsoleto)
- Le classi di utilità assistono l’utente nella gestione degli errori, nel debug e nell’impostazione di importanti valori di runtime per le mappe. Le classi sono:
  - `BaseDLM`
  - `MapExeContext`

System Manager

System Manager è uno strumento grafico che fornisce un’interfaccia con InterChange Server Express e l’archivio. System Manager consente di gestire le mappe e di configurare una definizione mappe. E’ possibile:

- Impostare alcune proprietà generali di una definizione mappe, includendo il livello di traccia e di convalida dei dati.
- Visualizzare gli oggetti business sorgenti e di destinazione di una mappa.

- Compilare una definizione di mappa.

Nota: Il System Manager consente di avviare il Map Designer Express. Per ulteriori informazioni, consultare “Avvio di Map Designer Express” a pagina 16.

Inoltre, System Manager consente di gestire le relazioni. E' possibile:

- Impostare alcune proprietà generali di una relazione, includendo l'ubicazione delle tabelle di relazione.
- Visualizzare i partecipanti di una relazione.

Nota: Il System Manager consente anche di avviare il Relationship Designer Express. Per ulteriori informazioni, consultare “Avvio di Relationship Designer Express” a pagina 249.

Panoramica sullo sviluppo delle mappe

In questa sezione viene fornita una panoramica del processo di sviluppo mappe, che comprende le seguenti attività di livello avanzato:

1. Installazione e configurazione del software per lo sviluppo mappe e l'installazione del Java Development Kit.
2. Progettazione e realizzazione di una mappa.

Requisiti per la configurazione di un ambiente di sviluppo

Prima di avviare il processo di sviluppo, devono sussistere le seguenti condizioni:

- Il software per lo sviluppo delle mappe è installato su una macchina che è possibile accedere.

  Per informazioni su come installare e avviare il sistema software per lo sviluppo delle mappe, consultare la guida all'installazione del sistema.
- L'IBM JDK (Java Development Kit) viene installato dal CD del prodotto.

  Accertarsi di aggiornare la la variabile d'ambiente `PATH` per includere la directory Java installata. Riavviare InterChange Server Express dopo aver aggiornato il percorso.
- System Manager è in esecuzione.

  Per informazioni sull'avvio di System Manager, consultare la guida all'installazione del sistema.
- Map Designer Express è aperto e collegato a System Manager.

  Per informazioni su come avviare Map Designer Express, consultare “Panoramica di Map Designer Express” a pagina 15.

Progettazione e realizzazione della mappa

Per progettare e realizzare le mappe, effettuare quanto segue:

1. Conoscere i formati dati utilizzati da tutti gli oggetti business riguardanti la mappa.
2. Creare la mappa con Map Designer Express.
3. Personalizzare una qualsiasi regola di trasformazione richiesta.
4. Definire le relazioni all'interno di Relationship Designer Express richieste dalla mappa.
5. Personalizzare la trasformazione della mappatura per l'esecuzione della gestione di relazione.

6. Implementare la gestione degli errori e dei messaggi.
7. Creare il file `.java` e il codice compilato. Il codice compilato è una classe Java eseguibile. Per ulteriori informazioni, consultare "File di sviluppo delle mappe" a pagina 12.
8. Verificare ed eseguire il debug della mappa, con ricodifica se necessario.

Figura 4 fornisce una panoramica visiva dello sviluppo delle mappe e fornisce un rapido riferimento ai capitoli in cui è possibile trovare le informazioni sugli argomenti specifici.

Consiglio: se un team è disponibile per lo sviluppo delle mappe, le principali attività di sviluppo della mappa possono essere svolte in parallelo con differenti membri del team di sviluppo.

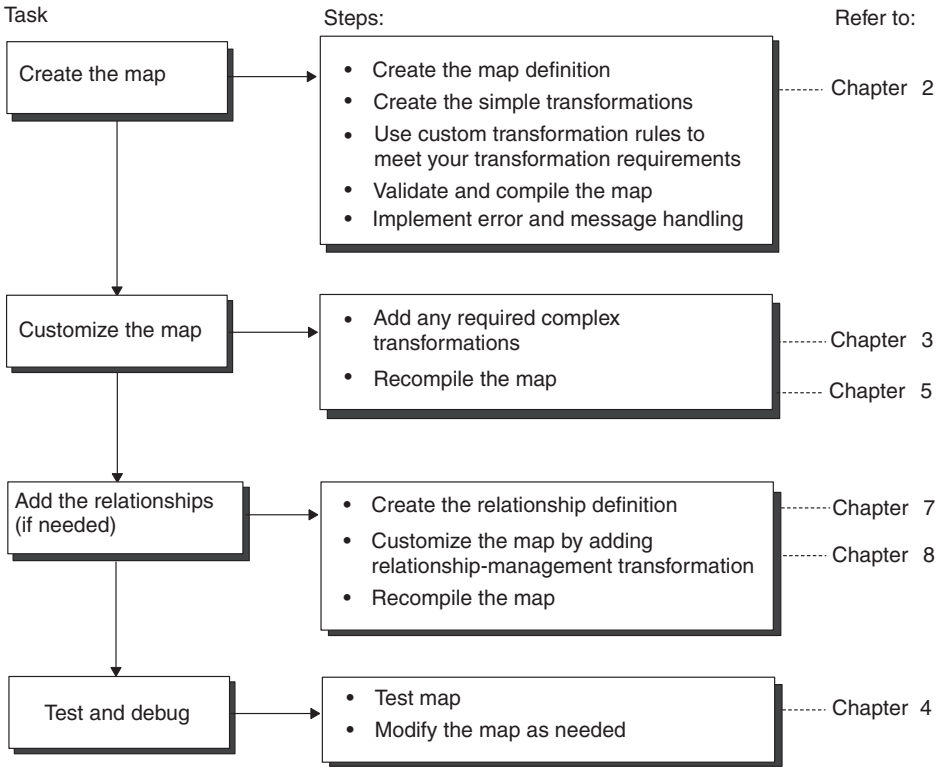


Figura 5. Panoramica dell'attività di sviluppo delle mappe

File di sviluppo delle mappe

Le seguenti informazioni costituiscono la base della mappa:

- Quando si compila una mappa, Map Designer Express crea due tipi di file (`.java`, `.class`) o un file di messaggi opzionale (.txt) (nel caso siano definiti dei messaggi con mappe specificate nella mappa). Questi file vengono salvati nel progetto in System Manager.
- Map Designer Express crea una definizione di mappa quando si salva una mappa di un progetto in System Manager. La definizione di mappa contiene informazioni generali sulla mappa (come le proprietà), oltre ad informazioni su come vengono mappati gli attributi di destinazione.

Attenzione: Non modificare il file *nomemappa*`.java`. In caso contrario, le modifiche non sono apportate nella progettazione della mappa, che è

memorizzata nel progetto in System Manager. Pertanto questi cambiamenti non sono modificabili in Map Designer Express. Map Designer Express legge solo la definizione di mappa.

Anche Relationship Designer Express memorizza le definizioni di relazione in formato XML in System Manager. Durante la fase di distribuzione, System Manager crea gli schemi tabella nel database di relazione per contenere i dati delle istanze runtime della relazione. E' possibile specificare, per ciascuna relazione, l'ubicazione di tutte le relative tabelle di relazione. L'ubicazione predefinita per queste tabelle è l'archivio di IBM WebSphere Business Integration Server Express.

Tabella 5 elenca i tipi di file che Map Designer Express può generare (`.java`, `.class`, `.cwm`, `.bo`, `.txt`) e le relative ubicazioni nell'area di lavoro di System Manager.

Tabella 5. Tipi di file mappa

Tipo file	Descrizione	Ubicazione relativa all'area di lavoro System Manager
`.java`	Codice Java generato, creato da Map Designer Express durante la compilazione di una mappa.	Memorizzato in `ProjectName\Maps\Src`.
`.class`	Codice Java compilato, creato da Map Designer Express durante la compilazione di una mappa.	Memorizzato in `ProjectName\Maps\Classes`.
`.cwm`	File di definizione della mappa, generato da Map Designer Express quando si salva una definizione della mappa.	Salvato in `ProjectName\Maps` quando "Salvato" in System Manager.
`.bo`	File di testo non codificato, utilizzato per salvare e caricare i dati di esecuzione delle prove e salvarne i risultati.	E' possibile salvare questi file in qualsiasi ubicazione.
`.txt`	File di messaggi, creato da Map Designer Express in base alle informazioni nella scheda Messaggi durante la compilazione della mappa.	Memorizzato in `ProjectName\Maps\Messages`.

Capitolo 2. Creazione di mappe

Questo capitolo contiene una panoramica di Map Designer Express, oltre ad una descrizione del relativo utilizzo. Map Designer Express per creare le mappe.

Nota: In questo capitolo vengono scambievolmente utilizzati i termini *mappa* e *definizione di mappe*. Il termine *mappa* fa riferimento alla definizione delle mappe (a cosa è possibile accedere mediante Map Designer Express).

In questo capitolo sono illustrati i seguenti argomenti:

Per informazioni secondarie sul modo in cui WebSphere Business Integration Express utilizza le mappe, fare riferimento a Capitolo 1, "Introduzione allo sviluppo delle mappe", a pagina 3.

Panoramica di Map Designer Express

Map Designer Express è uno strumento di sviluppo grafico per la creazione e la modifica delle mappe. Una *mappa* è costituita da una serie di passi di trasformazione che definiscono il modo in cui calcolare il valore per ciascun attributo dell'oggetto business di destinazione. La creazione di una mappa è il processo mediante il quale si specificano i passi di trasformazione per ciascun attributo di destinazione da trasformare.

Utilizzando Map Designer Express, è possibile specificare i passi di trasformazione semplice, come ad esempio la copia di un attributo di origine ad un attributo di destinazione dello stesso tipo di dati in modo interattivo utilizzando il trascinamento. Map Designer Express genera automaticamente il codice Java necessario per eseguire la trasformazione.

Per consentire altre trasformazioni comuni, come ad esempio la separazione di un attributo di origine in più attributi di destinazione o l'unione di più attributi di origine in un unico attributo di destinazione, Map Designer Express richiede alcune informazioni, come ad esempio il delimitatore per separare o unire, quindi genera il codice Java necessario. Per specificare trasformazioni più complesse, è possibile definire graficamente le attività utilizzando Activity Editor in una regola di trasformazione personalizzata, modificare direttamente il codice Java nella finestra Activity Editor o scrivere i passi di trasformazione per proprio conto da un volume di lavoro.

Questa sezione illustra gli argomenti di seguito riportati per la presentazione di Map Designer Express:

• "Utilizzo delle funzioni di Map Designer Express" a pagina 27

Avvio di Map Designer Express

Per avviare Map Designer Express, procedere nel modo seguente:

- Da System Manager, eseguire una delle azioni indicate:
  - Dal menu Strumenti, selezionare Map Designer Express.
  - Fare clic su una cartella di mappa in un progetto per abilitare l'icona Map Designer Express nella barra degli strumenti di System Manager. Quindi, fare clic sull'icona Map Designer Express.
  - Fare clic con il tastino destro del mouse sulla cartella Mappa in un progetto, quindi selezionare Crea nuova mappa dal menu Contesto.
  - Fare clic doppio clic con il tastino destro del mouse su una mappa per avviare Map Designer Express con la mappa selezionata aperta.
- Da uno strumento di sviluppo, come ad esempio Business Object Designer Express, Relationship Designer Express o Process Designer Express, effettuare una delle seguenti azioni:
  - Dal menu Strumenti, selezionare Map Designer Express.
  - Nella barra degli strumenti Programmi, fare clic sul pulsante Map Designer Express.

  Limitazione: Process Designer Express è disponibile solo in WebSphere Business Integration Server Express Plus.
- Utilizzare un collegamento di sistema:

  ```
  Avvio > Programmi >IBM WebSphere Business Integration
  Server Express > Toolset Express  >
  ```

  ```
   Sviluppo > Map Designer Express
  ```

Importante: Affinché Map Designer Express sia in grado di accedere alle mappe memorizzate in System Manager, Map Designer Express deve essere connesso ad un'istanza di System Manager. I passi precedenti suppongono che System Manager sia già stato avviato. Se System Manager è già in esecuzione, Map Designer Express si collega automaticamente.

Map Designer Express viene visualizzato nella propria finestra dell'applicazione. E' possibile avviare più di un'istanza di Map Designer Express per volta per modificare più di una mappa.

Attività con i progetti

Map Designer Express visualizza, modifica ed edita le mappe memorizzate in System Manager in base al *progetto*. Un *progetto* è semplicemente un raggruppamento logico di entità per la gestione e la distribuzione degli obiettivi. System Manager consente di creare più progetti.

Quando Map Designer Express stabilisce una connessione a System Manager, ottiene un elenco di oggetti business definiti nel progetto corrente. Se si aggiunge o si elimina un oggetto business utilizzando Business Object Designer Express, System Manager invia una notifica a Map Designer Express, che aggiorna in modo dinamico l'elenco delle definizioni dell'oggetto business.

Prima di effettuare attività con le mappe, è necessario selezionare il progetto in cui si trova la mappa, immettendo il nome del progetto nel dialogo Apri mappa da un progetto. Prima di passare ad un altro progetto, è necessario salvare le mappe

modificate nel progetto corrente. Per ulteriori informazioni sull'apertura di una mappa da un progetto e il salvataggio di una mappa in un progetto, consultare "Procedura per l'apertura di una mappa da un progetto in System Manager" a pagina 60 e "Salvataggio di una mappa in un progetto" a pagina 53, rispettivamente.

Layout di Map Designer Express

Quando si apre Map Designer Express per la prima volta senza specificare una mappa, la finestra della scheda di Map Designer Express è vuota e la finestra di output non viene visualizzata. Quando si apre una mappa esistente, la finestra di Map Designer Express visualizza la scheda Mappa nella finestra.

Tabella 6 descrive ciascun componente nella Finestra principale di Map Designer Express.

Tabella 6. Componenti della finestra di Map Designer Express

Area finestra	Descrizione	Per ulteriori informazioni
Menu	Fornisce le opzioni per accedere alle funzioni di Map Designer Express.	"Menu a discesa di Map Designer Express" a pagina 28
Barra degli strumenti	Al momento contiene tre barre degli strumenti separate, ciascuna delle quali fornisce una serie di pulsanti per accedere alle funzioni di Map Designer Express.	"Barre degli strumenti di Map Designer Express" a pagina 30
Finestra della scheda di Map Designer Express	Visualizza le informazioni sulla mappa per una mappa aperta in una delle quattro schede Mappa.	"Scheda Tabella" a pagina 17 "Scheda Diagramma" a pagina 20 "Scheda Messaggi" a pagina 22 "Scheda Test" a pagina 22
Finestra di output	Visualizza i risultati per la compilazione di una nappa e di altri messaggi di stato. Se la finestra di output non è visualizzata quando Map Designer Express genera un messaggio di stato, la finestra viene aperta automaticamente. E' possibile annullare il contenuto della finestra di output con l'opzione Cancella output del Menu Visualizza. **Suggerimento:** è possibile controllare se il pannello della finestra di output viene visualizzato come parte della finestra principale di Map Designer Express con l'opzione Finestra di output del Menu Visualizza.	N/D
Barra di stato	Visualizza i messaggi di stato di Map Designer Express. **Suggerimento:** è possibile controllare se la barra di stato viene visualizzata come parte della finestra Map Designer Express con l'opzione Barra di stato del Menu Visualizza.	N/D

Le sezioni di seguito riportate descrivono il layout generico di ciascuna delle schede che vengono visualizzate nella finestra della scheda di Map Designer Express.

Scheda Tabella

La scheda Tabella di Map Designer Express visualizza le informazioni sulla mappatura in formato tabellare che elenca tutti gli attributi e le trasformazioni della mappatura.

La scheda Tabella è costituita dalle seguenti aree:

- Tabella di trasformazione degli attributi
- Pannello oggetto business

Tabella di trasformazione degli attributi: La tabella di trasformazione degli attributi presenta in formato tabellare tutte le trasformazioni associate alla mappa. Tabella 7 illustra le colonne che costituiscono questa tabella.

Tabella 7. Colonne della tabella di trasformazione degli attributi

Nome colonna	Descrizione
Ordine di esecuz.	L'ordine di esecuzione per l'attributo di destinazione. Quando si aggiunge una trasformazione alla fine di questa tabella, Map Designer Express assegna automaticamente il relativo ordine di esecuzione come ultimo della tabella. E' possibile modificare l'ordine di esecuzione di un attributo immettendo il numero di ordine desiderato nel campo Ordine di esecuzione. **Nota:** E' possibile specificare il modo in cui Map Designer Express gestisce l'ordine di esecuzione degli attributi di destinazione con l'opzione `Definizione mappa: regola automaticamente l'ordine di esecuzione`. Per impostazione predefinita, questa opzione è disabilitata. Quando l'opzione è abilitata, Map Designer Express regola automaticamente l'ordine di esecuzione degli altri attributi. E' possibile modificare l'impostazione di questa opzione nella scheda Generale del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.
Attributo di origine	Il nome dell'attributo di origine per la trasformazione. Questo campo fornisce una casella combinata contenente un elenco di tutti gli oggetti business di destinazione e di origine con i relativi attributi elencati. Fare clic sull'attributo di origine appropriato dall'elenco. E' possibile selezionare più attributi di origine facendo clic sulla voce Più attributi nell'elenco della casella combinata. Map Designer visualizza la finestra di dialogo Più attributi da cui è possibile selezionare gli attributi. **Nota:** E' possibile specificare il modo in cui Map Designer Express visualizza il nome dell'attributo di origine con l'opzione `Definizione mappa: illustra il percorso completo dell'attributo`. Per impostazione predefinita, questa opzione è disabilitata e Map Designer Express visualizza tutti i nomi dell'attributo di origine come *...AttrName*. Quando l'opzione è abilitata, Map Designer Express visualizza il percorso completo dell'attributo: `Obj`*`SrcBusObj.AttrName`*. E' possibile modificare l'impostazione di questa opzione nella scheda Generale del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.
Tipo di origine	Il tipo di dati dell'attributo di origine. Questo campo è in sola lettura.
Attributo di destinazione	Il nome dell'attributo di destinazione per la trasformazione. Questo campo fornisce una casella combinata contenente un elenco di tutti gli oggetti business di destinazione e di origine con i relativi attributi elencati. Fare clic sull'attributo di destinazione appropriato dall'elenco. **Nota:** E' possibile specificare il modo in cui Map Designer Express visualizza il nome dell'attributo di destinazione con l'opzione `Definizione mappa: illustra il percorso completo dell'attributo`. Per impostazione predefinita, questa opzione è disabilitata e Map Designer Express visualizza tutti i nomi dell'attributo di destinazione come *...AttrName*. Quando l'opzione è abilitata, Map Designer Express visualizza il percorso completo dell'attributo: `Obj`*`DestBusObj.AttrName`*. E' possibile modificare l'impostazione di questa opzione nella scheda Generale del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.

Tabella 7. Colonne della tabella di trasformazione degli attributi (Continua)

Nome colonna	Descrizione
Tipo di dest.	Il tipo di dati dell'attributo di destinazione. Questo campo è in sola lettura.
Regola di trasformazione	La regola di trasformazione e codice per questo passo di trasformazione. Questo campo fornisce una casella combinata contenente un elenco delle trasformazioni standard: • Nessuna (nessuna trasformazione) • Aggiungi • Sposta • Dividi • Imposta valore • Mappa secondaria • Riferimento incrociato • Personalizzata Fare clic sulla trasformazione appropriata da questo elenco per immetterla nel campo. Per ulteriori informazioni, consultare "Specifica delle trasformazioni degli attributi standard" a pagina 40.
Commento	Una descrizione informativa della trasformazione dell'attributo. Consultare "Impostazione dei commenti nel campo commenti dell'attributo" a pagina 57.

Passi per la definizione di una mappa dalla scheda Tabella: per definire una mappa dalla scheda Tabella, effettuare i passi di seguito riportati:

1. Fare clic su una cella vuota nella colonna Attributo di origine. Dalla casella combinata disponibile, fare clic sull'attributo di origine da trasformare.
2. Fare clic sulla cella corrispondente nella colonna Attributo di destinazione. Fare clic sull'attributo di destinazione dalla casella combinata disponibile.
3. Fare clic nella cella corrispondente nella colonna Regola di trasformazione. Questa colonna fornice una casella combinata:
   - Per una trasformazione standard (Aggiungi, Sposta, Dividi, Imposta valore, Mappa secondaria o Riferimento incrociato), selezionare l'opzione associata dall'elenco. Map Designer Express genera il codice per queste trasformazioni standard. E' possibile ignorare questo codice, se necessario. Per ulteriori informazioni, consultare "Specifica delle trasformazioni degli attributi standard" a pagina 40.
   - Per una trasformazione che *non* si trova in questa casella combinata, selezionare Personalizza dall'elenco ed aggiungere il codice Java in Activity Editor. Per ulteriori informazioni, consultare "Creazione di una trasformazione personalizzata" a pagina 52.
4. Fare clic sulla cella corrispondente nella colonna Commento. Per ulteriori informazioni, consultare "Impostazione dei commenti nel campo commenti dell'attributo" a pagina 57.

Riquadro oggetti business: Il riquadro Oggetti business presenta un elenco di tutti gli oggetti business di origine e di destinazione associati alla mappa. Nell'area a sinistra vengono visualizzati gli oggetti business di origine, nell'area a destra vengono visualizzati gli oggetti business di destinazione. Se la mappa contiene un

oggetto business temporaneo, il riquadro Oggetti business contiene tre aree: Oggetto business di origine, Oggetto business temporaneo e Oggetto business di destinazione.

Suggerimento: è possibile controllare se il riquadro degli oggetti business viene visualizzato come parte della scheda Tabella con l'opzione Riquadro oggetti business del Menu Visualizza.

Scheda Diagramma

La scheda Diagramma di Map Designer Express fornisce un'interfaccia per fare clic e trascinare per la definizione e l'esame delle trasformazioni. E' possibile visualizzare e progettare le mappe nello spazio di lavoro della mappa, che si trova a destra nella finestra.

La scheda Diagramma è costituita dalle seguenti aree:

- Browser oggetto business, che viene visualizzato nel riquadro del progetto, a sinistra della finestra. Questo browser utilizza un formato gerarchico per elencare gli oggetti business del progetto in System Manager quando Map Designer Express è connesso a System Manager. Per aggiornare l'elenco degli oggetti business nel browser degli oggetti business, fare clic con il tastino destro del mouse nel browser dell'oggetto business e selezionare Aggiorna tutto dal menu Contesto. Map Designer Express esegue una query a System Manager ed aggiorna il browser degli oggetti business con gli oggetti business correnti.

  Nota: Se si aggiunge o si elimina un oggetto business dal progetto in System Manager, System Manager aggiorna in modo dinamico l'elenco delle definizioni degli oggetti business.

  Suggerimento: è possibile controllare se il browser degli oggetti business viene visualizzato come parte della vista Diagramma con l'opzione Riquadro progetto del menu a discesa Visualizza.
- Lo spazio di lavoro della mappa, che visualizza sempre le informazioni sulla mappa corrente.

  Quando si apre una mappa, lo spazio di lavoro della mappa visualizza una finestra dell'oggetto business per ciascun oggetto business di origine e di destinazione utilizzato nella mappa. Ciascuna finestra dell'oggetto business elenca alcuni o tutti gli attributi definiti nell'oggetto business, in base alla modalità di visualizzazione al momento selezionata. Nel caso di un oggetto business di destinazione o di un oggetto business temporaneo, la finestra dell'oggetto business elenca anche la regola di trasformazione e i commenti associati all'attributo. Nello spazio di lavoro della mappa, è possibile aggiungere, eliminare o modificare le trasformazioni nella mappa. Le righe che collegano gli attributi rappresentano le trasformazioni tra gli attributi.

  Suggerimento: è possibile controllare gli attributi visualizzati negli oggetti business di origine e di destinazione nella scheda Diagramma con le opzioni del menu secondario Visualizza > Diagramma. Questo menu secondario consente di selezionare se visualizzare tutti gli attributi, solo gli attributi collegati (mappati) o solo gli attributi non collegati (non mappati).

  Nella vista Diagramma, è inoltre possibile aggiungere più istanze per gli oggetti business child con cardinalità multipla e mappare graficamente gli attributi a tali istanze. Ciò è realmente utile quando si mappa un oggetto business non gerarchico ad un oggetto business gerarchico.

Per creare un'istanza, fare clic con il tastino destro del mouse sull'oggetto business child, quindi selezionare Aggiungi istanza dal menu Contesto che viene visualizzato. Inoltre, è possibile eliminare un'istanza da questo menu selezionando Rimuovi istanza.

Nota: E' possibile solo aggiungere o rimuovere le istanze dal livello di oggetto business con cardinalità multipla. L'aggiunta o l'eliminazione dell'istanza avviene solo in modo sequenziale. Ad esempio, l'ultima istanza creata sarà la prima ad essere eliminata.

Figura 6 illustra la creazione di più istanze per l'oggetto business child PhoneInfo da parte dell'origine.

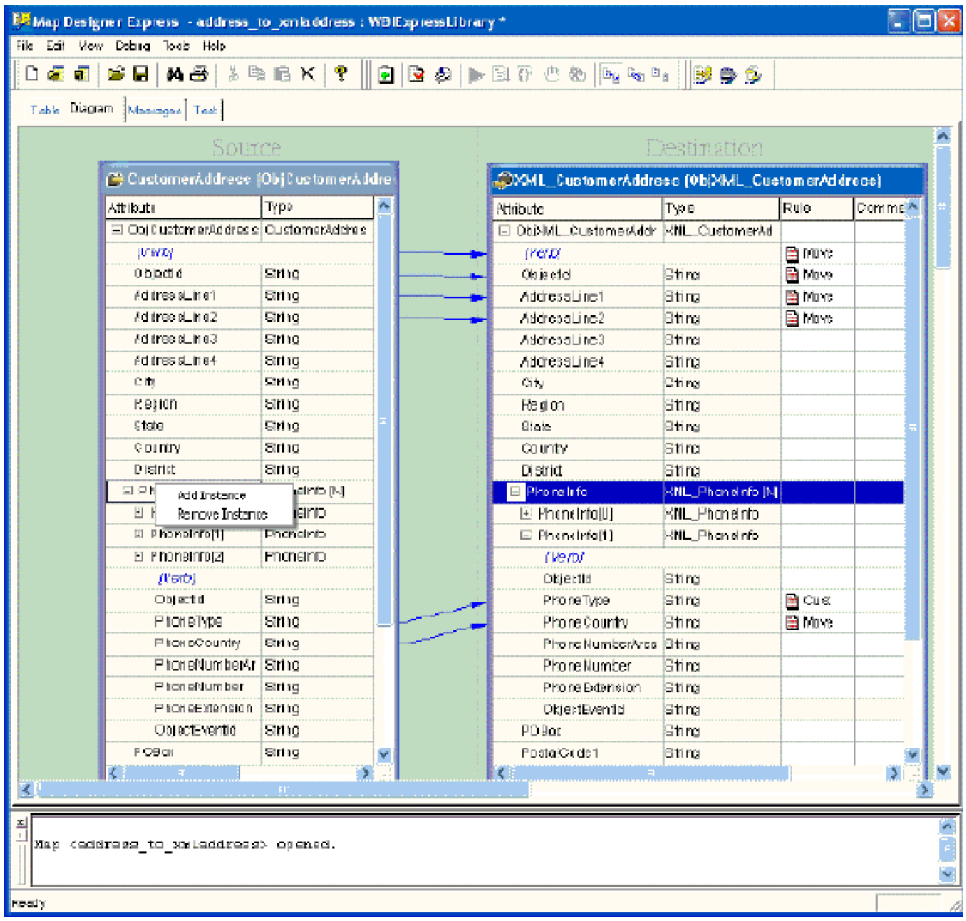


Figura 6. Creazione dell'istanza

Quando si espandono gli oggetti business con cardinalità multipla, viene visualizzato l'elenco delle istanze e non gli attributi. Per visualizzare gli attributi, è necessario espandere le singole istanze, come illustrato nell'istanza estesa PhoneInfo[2] dell'origine. E' possibile eseguire la mappatura per gli attributi presenti nelle istanze in qualunque parte della destinazione, comprese le altre istanze della destinazione.

Inoltre, in Figura 6 viene illustrata la mappatura tra le istanze. Gli attributi PhoneType e PhoneCountry di PhoneInfo Instance[2] dell'origine sono mappati agli attributi PhoneType e PhoneCountry di PhoneInfo Instance[1] della destinazione.

Scheda Messaggi

La scheda Messaggi visualizza i messaggi della mappa. Un messaggio è costituito da un ID messaggio e dal relativo testo del messaggio associato.

La scheda Messaggi è divisa in due riquadri. Il riquadro superiore è la griglia del messaggio, costituita da tre colonne: la colonna ID messaggio, la colonna Messaggio e la colonna Spiegazione (per i commenti di tutto il file del messaggio). La parte inferiore o riquadro Descrizione è un campo riservato all'immissione del testo semplice. Quando si immette il testo nel riquadro Descrizione, il testo viene aggiunto nella parte superiore del file di messaggio generato come commenti. Map Designer Express salva eventuali modifiche effettuate ai messaggi della mappa nel progetto di System Manager.

Per ulteriori informazioni sui messaggi e sul relativo utilizzo, consultare Capitolo 27, "File di messaggi", a pagina 517. Per informazioni sul formato dei messaggi, consultare "Formato per i messaggi di mappe" a pagina 521.

Quando si compila una nuova mappa, Map Designer Express genera un file di messaggi, in base alle informazioni immesse nella scheda Messaggi. Questo file di messaggi viene salvato nella directory del messaggio.

Attenzione: E' necessario effettuare tutte le modifiche nei messaggi della mappa con la scheda Messaggi di Map Designer Express. *Non* utilizzare un editor di testo esterno per effettuare le modifiche al file di messaggi generato. Eventuali modifiche effettuate mediante un editor esterno *non* potranno essere visualizzate da Map Designer Express, poiché *non* verranno memorizzate nella definizione della mappa del progetto. Inoltre, tali modifiche verranno sovrascritte la volta successiva in cui viene compilata la mappa.

Scheda Test

La scheda Test fornisce un'interfaccia per eseguire il test delle mappe e visualizzarne i risultati. In questa scheda, è possibile eseguire i test per verificare che le trasformazioni funzionino correttamente.

La scheda Test è costituita dalle seguenti aree:

- Diagramma di percorso del test

  Il diagramma di percorso del test in alto nella finestra illustra il test della mappa come una serie di icone:

  – La freccia Dati di test di origine indica la direzione di trasformazione della mappa ed è etichettata con il tipo di oggetto business per l'oggetto business di origine che partecipa al test della mappa.

  – L'icona Mappa rappresenta la mappa al momento aperta, che viene utilizzata nel test.

  – La freccia Dati di test di destinazione indica la direzione di trasformazione della mappa ed è etichettata con il tipo di oggetto business per l'oggetto business di destinazione risultante dal test della mappa.

- Riquadro dei dati del test di origine

  L'area di dati del test di origine nella finestra in basso a sinistra utilizza un formato gerarchico per elencare gli attributi di un oggetto business di origine che partecipano alla mappa. Fare clic sul simbolo (+) accanto all'oggetto business di origine per espanderlo. In questa area, immettere i dati di test per l'oggetto business di origine.

- Riquadro dati di test di destinazione

L'area di dati del test di destinazione nella finestra in basso a destra utilizza un formato gerarchico per elencare gli attributi di un oggetto business di destinazione risultanti dalla mappa. Fare clic sul simbolo (+) accanto all'oggetto business di destinazione per espanderlo. In questa area, visualizzare i risultati del test per l'oggetto business di destinazione.

Nota: Map Designer Express visualizza i risultati dal test eseguita dalla mappa nella finestra di output.

Per ulteriori informazioni su come utilizzare la scheda Test, consultare "Test delle mappe" a pagina 91.

Assegnazione delle preferenze

La finestra di dialogo Preferenze consente di personalizzare il comportamento dello strumento Map Designer Express. Per visualizzare la finestra di dialogo Preferenze:

- Dal menu Visualizza, selezionare Preferenze.
- Utilizzare la combinazione di tasti di accesso rapido `Ctrl+U`.

Figura 7 illustra la finestra di dialogo Preferenze.

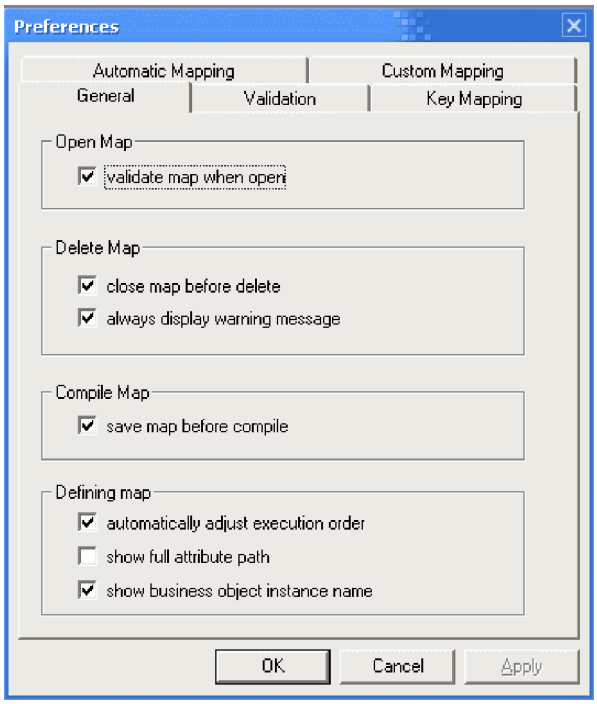


Figura 7. Dialogo Preferenze

Map Designer Express salva le impostazioni relative alle preferenze nel registro di Windows. Pertanto, restano attive per la sessione corrente e quelle successive di Map Designer Express. Il dialogo Preferenze dispone delle seguenti schede:

- Generale
- Convalida
- Mappatura chiave
- Mappatura automatica
- Mappatura personalizzata

Specifica delle Preferenze generali

La scheda Generale della finestra di dialogo Preferenze visualizza le preferenze generali che è possibile specificare per il modo in cui Map Designer Express gestisce le mappe.

Tabella 8. Generale Map Designer Express Preferenze

Preferenze generali	Descrizione	Per ulteriori informazioni
Apri mappa		
convalida mappa quando aperta	Quando viene abilitata questa opzione, Map Designer Express convalida la mappa all'apertura della stessa. **Suggerimento:** se una mappa utilizza gli oggetti business con molti attributi, ovvero più di mille attributi, abilitando questa opzione, la mappa verrà aperta dopo un lungo intervallo di tempo. Se si verifica questa situazione, disabilitare l'opzione.	"Apertura di una mappa" a pagina 59
Elimina mappa		
chiudi mappa prima di eliminare	Quando viene abilitata questa opzione, Map Designer Express chiude sempre la mappa al momento aperta prima di visualizzare la finestra di dialogo Elimina mappa.	"Procedura per l'eliminazione delle mappe" a pagina 81
visualizza sempre i messaggi di avviso	Quando viene abilitata questa opzione, Map Designer Express visualizza sempre una conferma prima di eliminare la mappa.	"Procedura per l'eliminazione delle mappe" a pagina 81
Compila mappa		
salva mappa prima di compilare	Quando viene abilitata questa opzione, Map Designer Express salva sempre la mappa corrente nel progetto in System Manager prima di compilarlo.	"Compilazione di una mappa" a pagina 89
Definizione mappa		
regola automaticamente l'ordine di esecuzione	Quando viene abilitata questa opzione, Map Designer Express rinumera automaticamente l'ordine di esecuzione degli attributi di destinazione nella scheda Tabella quando cambia l'ordine di esecuzione dell'attributo esistente.	"Utilizzo dell'ordine di esecuzione" a pagina 83
mostra percorso completo attributi	Quando viene abilitata questa opzione, Map Designer Express illustra il percorso completo dell'attributo per i nomi degli attributi di origine e di destinazione nella scheda Tabella.	"Scheda Tabella" a pagina 17
mostra nome istanza oggetto business	Quando viene abilitata questa opzione, Map Designer Express visualizza i nomi dell'oggetto business di origine e di destinazione *e* i relativi nomi della variabile. Quando viene disabilitata questa opzione Map Designer Express omette i nomi delle variabili dell'oggetto business in entrambe le schede Tabella e Diagramma.	"Passi per la modifica delle variabili dell'oggetto business" a pagina 179

Specifica della convalida

La scheda Convalida della finestra di dialogo Preferenze fornisce le opzioni da selezionare affinché Map Designer Express possa eseguire le convalide nella mappa al momento del salvataggio. Di seguito sono riportate le opzioni:

- Visualizza avvertenza se il verbo non viene mappato

- Visualizza avvertenza se l'attributo chiave non viene mappato
- Visualizza avvertenza se l'attributo richiesto non viene mappato
- Visualizza avvertenza se l'oggetto business child non viene mappato

Map Designer Express effettua la convalida selezionata per tutte le regole di trasformazione in quel livello.

Esempio: se è mappato il percorso a.b.c, Map Designer Express esegue queste convalide a livello degli oggetti business a, a.b e a.b.c.

Per ulteriori informazioni, consultare "Convalida di una mappa" a pagina 88.

Specifica della mappatura chiave

La scheda Mappatura chiave del dialogo Preferenze visualizza le mappature chiave per varie trasformazioni standard nella scheda Diagramma.

Tabella 9. Mappatura chiave Map Designer Express Preferenze

Mappa chiave	Descrizione	Per ulteriori informazioni
Sposta/Aggiungi/Mappa secondaria	Mappa chiave da utilizzare durante la creazione di una trasformazione Sposta, Aggiungi o Maschera secondaria. Map Designer Express distingue tra le trasformazioni per tipo e numero degli attributi di origine:	
	• Sposta — un attributo di origine che *non* è un oggetto business child	"Copia di un attributo di origine ad un attributo di destinazione" a pagina 42
	• Aggiungi — più di un attributo di origine che *non* è un oggetto business child	"Aggiunta di attributi" a pagina 43
	• Mappa secondaria — uno o più attributi di origine che sono un oggetto business child	"Trasformazione con una mappa secondaria" a pagina 47
Dividi/Riferimento incrociato	Mappa di chiave da utilizzare durante la creazione di una trasformazione Dividi o per la conservazione delle relazioni di identità	"Divisione degli attributi" a pagina 45, "Relazioni di identità con riferimento incrociato" a pagina 51
Personalizza	Mappa di chiave da utilizzare durante la creazione di una trasformazione Crea.	"Creazione di una trasformazione personalizzata" a pagina 52

La scheda Mappatura chiave fornisce le seguenti funzioni:

- Per modificare una mappatura di chiave, fare clic nel campo di trasformazione appropriato e selezionare la mappa di chiave desiderata per questa trasformazione dalla casella combinata. Fare clic su OK.
- Per ripristinare le mappature di chiave ai valori predefiniti, fare clic su Utilizza valore predefinito, quindi fare clic su OK.

Specifica della mappatura automatica

La scheda Mappatura automatica del dialogo Preferenze fornisce le opzioni che è possibile selezionare per Map Designer Express da utilizzare durante la ricerca dei nomi degli attributi corrispondente negli oggetti business per l'automatizzazione della mappa. Di seguito sono riportate le opzioni:

- Ignora maiuscolo/minuscolo - per corrispondenze non sensibili al maiuscolo/minuscolo nella stringa di ricerca
- Ignora tipi di dati incompatibili - per eseguire corrispondenze di nomi con tipi di dati incompatibili nella stringa di ricerca

  Nota: La selezione di questa opzione potrebbe causare una perdita di dati.

Per ulteriori informazioni, consultare “Utilizzo dell’automazione della mappa” a pagina 70.

Specifica della mappatura personalizzata

La scheda Mappatura personalizzata del dialogo Preferenze consente di configurare i blocchi di funzione standard da utilizzare direttamente in Map Designer Express.

Per ulteriori informazioni, consultare “Suggerimento: Utilizzo dei blocchi funzione direttamente in Map Designer Express” a pagina 119

Personalizzazione della finestra principale

Map Designer Express consente di personalizzare la finestra principale nel modo seguente:

- “Selezione della visualizzazione delle finestre”
- “Spostamento di una finestra divisibile” a pagina 27

Selezione della visualizzazione delle finestre

Quando si apre Map Designer Express senza specificare una mappa, la finestra principale è vuota con le barre degli strumenti e di stato visibili. Quando si apre una mappa, Map Designer Express visualizza la scheda Diagramma nella finestra e apre la finestra di output. Per impostazione predefinita, Map Designer Express visualizza ciascuna delle schede della mappa nel modo seguente:

- Scheda Tabella — il riquadro oggetti business viene visualizzato nella tabella di trasformazione dell’attributo.
- Scheda Diagramma — viene visualizzata l’area dello spazio di lavoro della mappa ed è vuota.
- Schede Messaggi e Test — come descritte in “Scheda Messaggi” a pagina 22 e in “Scheda Test” a pagina 22 rispettivamente.

E’ possibile personalizzare l’aspetto della finestra principale e le schede Mappa con le opzioni del menu Visualizza. Tabella 10 descrive le opzioni del menu a discesa Visualizza e il modo in cui condizionano l’aspetto della finestra di Map Designer Express.

Tabella 10. Opzioni del menu visualizza per la personalizzazione della finestra di Map Designer Express

Opzioni del menu Visualizza	**Elemento visualizzato**
Barre degli strumenti	Un menu secondario con le opzioni per ciascuna barra degli strumenti di Map Designer Express: • barra degli strumenti Standard • barra degli strumenti Designer • barra di strumenti Programmi
Barra di stato	Un riquadro con un’unica riga in cui vengono visualizzate le informazioni sullo stato di Map Designer Express.
Pannello oggetto business	Un riquadro che visualizza gli oggetti business di origine e di destinazione nella scheda Tabella di Map Designer Express.
Riquadro Progetto	Un riquadro che visualizza il browser dell’oggetto business nella scheda Diagramma di Map Designer Express.

Tabella 10. Opzioni del menu visualizza per la personalizzazione della finestra di Map Designer Express (Continua)

Opzioni del menu Visualizza	Elemento visualizzato
Diagramma	Un menu secondario con le opzioni per gli attributi da visualizzare negli oggetti business di origine e di destinazione nelle finestre dell'oggetto business della scheda Diagramma: • Tutti gli attributi • Attributi collegati • Attributi non collegati La La barra degli strumenti Designer dispone anche di icone per la visualizzazione di questi attributi.
Finestra di output	Una piccola finestra nella parte inferiore della finestra di Map Designer Express. L'opzione Cancella output del menu Visualizza annulla tutto il testo contenuto nella finestra di output.

Suggerimento: quando viene visualizzata un'opzione di menu con un contrassegno a sinistra, viene visualizzato anche l'elemento associato. Per disattivare la visualizzazione dell'elemento, selezionare l'opzione di menu associata. Il contrassegno non viene più visualizzato indicando che l'elemento non viene al momento visualizzato. Invece, è possibile attivare la visualizzazione di un elemento non visualizzato, selezionando l'opzione di menu associata. In questo caso, il contrassegno viene visualizzato accanto all'elemento di visualizzazione.

Spostamento di una finestra divisibile

Map Designer Express supporta le seguenti funzioni come finestre divisibili:

- Barre degli strumenti nella finestra principale:
  - barra degli strumenti Standard
  - barra degli strumenti Designer
  - barra di strumenti Programmi

  Per ulteriori informazioni sulle funzioni di tali barre degli strumenti, consultare "Barre degli strumenti di Map Designer Express" a pagina 30.
- Finestra di output
- Pannello di controllo Trova. Per ulteriori informazioni, consultare "Ricerca di informazioni in una mappa" a pagina 77.

Suggerimento: per impostazione predefinita, una finestra divisibile è, in genere, posizionata lungo il margine della finestra principale e si sposta come parte di quest'ultima. Quando si sposta una finestra divisibile, la si stacca dalla finestra principale, consentendo di funzionare come finestra indipendente. Per spostare una finestra divisibile, tenere premuto il pulsante sinistro del mouse, spostare il bordo della finestra e trascinarlo fuori della finestra principale o sul desktop.

Utilizzo delle funzioni di Map Designer Express

E' possibile accedere alle funzioni di Map Designer Express utilizzando gli elementi di seguito riportato:

- Menu a discesa
- Menu Contesto
- Pulsanti della Barra degli strumenti
- Tasti di accesso rapido

Menu a discesa di Map Designer Express

Map Designer Express dispone dei seguenti menu a discesa:

- Menu File
- Menu Modifica
- menu Visualizza
- Menu Debug
- menu Strumenti
- Menu Guida

Le sezioni di seguito riportate descrivono le opzioni di ciascuno di questi menu.

Funzioni del menu File: Il menu a discesa File di Map Designer Express dispone delle opzioni illustrate in Tabella 11.

Tabella 11. Opzioni del menu File di Map Designer Express

Opzione del menu File	Descrizione	Per ulteriori informazioni
Nuovo	Crea un nuovo file di mappa, annullando eventuali mappe esistenti dallo spazio di lavoro della mappa.	"Creazione di una mappa: Passi di base" a pagina 33
Apri	Apre una mappa esistente da un Progetto o da un File	"Apertura di una mappa" a pagina 59
Chiudi	Chiude la mappa corrente	"Chiusura di una mappa" a pagina 62
Salva	Salva la mappa corrente con lo stesso nome nel Progetto o nel File	"Salvataggio delle mappe" a pagina 53
Salva con nome	Salva la mappa corrente con un nome diverso in un Progetto o in un File	"Salvataggio delle mappe" a pagina 53
Elimina	Elimina la mappa specificata	"Eliminazione di oggetti" a pagina 80
Convalida mappa	Convalida la mappa corrente	"Convalida di una mappa" a pagina 88
Compila	Compila la mappa corrente	"Compilazione di una mappa" a pagina 89
Compila con mappe secondarie	Compila la mappa corrente e le relative mappe secondarie	"Compilazione di una mappa" a pagina 89
Compila tutto	Compila tutto o una serie secondaria di mappe definite	"Compilazione di una serie di mappe" a pagina 90
Crea documento mappa	Crea file HTML che descrivono la mappa tra gli oggetti business	"Procedura per la creazione di un documento della mappa" a pagina 68
Visualizza documento mappa	Visualizza il file del documento mappa HTML nel browser HTML	"Visualizzazione di un documento della mappa" a pagina 69
Stampa configurazione, Stampa anteprima, Stampa	Fornisce le opzioni per l'anteprima, la stampa e le impostazioni di stampa di un lavoro di stampa	"Stampa di una mappa" a pagina 79
Esci	Esce da Map Designer Express	N/D

Funzioni del menu Modifica: Il menu a discesa Modifica di Map Designer Express dispone delle seguenti opzioni:

- Opzioni di modifica di Windows standard — Taglia, Copia e Incolla
- Elimina selezione corrente — Elimina l'oggetto al momento selezionato
- Seleziona tutto — Nella scheda diagramma, seleziona le trasformazioni tra gli oggetti di origine e di destinazione
- Inserisci riga — Inserisce una riga prima della riga corrente nella tabella di trasformazione dell'attributo della scheda Tabella

- Aggiungi oggetto business — Visualizza il dialogo Aggiungi oggetto business per aggiungere gli oggetti business (di origine, destinazione e temporanei) alla mappa
- Elimina oggetto business — Visualizza il dialogo Elimina oggetto business per eliminare un oggetto business
- Trova — Ricerca un nome di attributo o un codice di trasformazione mediante il testo o il codice di trasformazione per gli attributi non mappati
- Sostituisci — Ricerca e sostituisce nel codice Java personalizzato o i commenti
- Proprietà mappa — Visualizza la finestra Proprietà mappa

Funzioni del menu Visualizza: Il menu a discesa Visualizza di Map Designer Express dispone delle seguenti opzioni di visualizzazione:

- Riquadro oggetto business — Quando abilitato, visualizza gli oggetti business di origine e di destinazione in basso nel riquadro della scheda Tabella della finestra di Map Designer Express
- Diagramma — Fornisce le opzioni per la visualizzazione degli attributi nelle finestre Oggetto business della scheda Diagramma
- Riquadro progetto — sempre abilitato, visualizza il browser dell'oggetto business come riquadro sinistro della scheda Diagramma nella finestra Map Designer Express
- Cancella output — Cancella il contenuto della finestra di output
- Finestra di output — Quando abilitata, visualizza i messaggi di stato, compresi i messaggi sull'apertura, la convalida, il salvataggio, la compilazione e il test eseguendo la mappa
- Barre degli strumenti — dispongono delle opzioni per la visualizzazione delle barre degli strumenti di Map Designer Express: Standard, Designer e Programmi
- Barra di stato — Quando abilitata, visualizza un messaggio di stato di una sola riga in basso nella finestra principale
- Preferenze — Visualizza il dialogo Preferenze, da cui è possibile impostare le preferenze di Map Designer Express

Per informazioni sulle opzioni del menu Visualizza che controllano la visualizzazione, consultare "Selezione della visualizzazione delle finestre" a pagina 26.

Funzioni del menu Debug: Il menu a discesa Debug fornisce l'accesso alle utilità di debug di Map Designer Express. Fornisce le seguenti opzioni:

- Esegui test — Effettua la connessione al server e avvia il test della mappa che viene aperta da un progetto
- Continua — Continua l'esecuzione dopo l'arresto ad un punto di interruzione
- Ignora — Continua l'esecuzione dopo l'arresto ad un punto di interruzione, ma termina l'esecuzione prima dell'esecuzione dell'attributo successivo
- Arresta esecuzione test - Arresta l'esecuzione del test di una mappa
- Avanzate - fornisce le opzioni per la connessione al server per il test di una mappa che si trova in un server (Collega) e la disconnessione da un server e la chiusura di una mappa (Scollega)
- Attiva/disattiva punto di interruzione — Imposta un punto di interruzione in una mappa, che mette in pausa l'esecuzione appena prima della trasformazione dell'attributo selezionato
- Punti di interruzione — Visualizza tutti i punti di interruzione per la mappa

- Cancella tutti i punti di interruzione — Cancella tutti i punti di interruzione nella mappa

Per ulteriori informazioni sull'utilizzo delle funzioni di test e di debug di Map Designer Express, consultare "Test delle mappe" a pagina 91.

Funzioni del menu Strumenti: Il menu a discesa Strumenti di Map Designer Express fornice le opzioni per avviare ciascuno degli strumenti, compresi gli strumenti di Automazione delle mappe:

- Mappatura automatica
- Inverti mappa
- Process Designer Express

  Limitazione: questo strumento è disponibile solo in WebSphere Business Integration Server Express Plus.
- Map Designer Express
- Business Object Designer Express
- Relationship Designer Express

Funzioni del Menu Guida: Il menu Guida fornisce le opzioni standard della guida di Windows:

- Argomenti della guida
- Documentazione
- Informazioni su Map Designer Express

Menu Contesto

Il menu Contesto è un menu di scelta rapida disponibile, facendo clic con il tastino destro del mouse, da svariate posizioni, come ad esempio la colonna della regola di trasformazione, l'intestazione delle righe nella vista Tabella, l'oggetto business child nel riquadro di test di origine o la casella modifica in una dialogo. Viene aperto un menu contenente comandi utili, che cambiano in base alla posizione in cui si fa clic.

Esempio: Facendo clic nella colonna relativa alla regola di trasformazione viene aperto un menu Contesto che fornisce le seguenti opzioni:

- Apri — Apre la finestra di dialogo corrispondente per la regola di trasformazione, come ad esempio Aggiungi, Dividi e Mappa secondaria. Per le trasformazioni personalizzate, apre Activity Editor.
- Apri in una nuova finestra — Per le trasformazioni personalizzate, apre una nuova istanza di Activity Editor per visualizzare i dettagli della regola di trasformazione.
- Visualizza origine — Visualizza il codice Java corrispondente alla trasformazione in Activity Editor. In base alla natura della trasformazione, il codice potrebbe essere in sola lettura.

Nota: L'azione predefinita quando si fa doppio clic sulla cella della trasformazione è Apri. Se Apri non è disponibile per questa trasformazione, nella barra di stato viene visualizzato un messaggio indicante che l'azione non è disponibile.

Barre degli strumenti di Map Designer Express

Map Designer Express dispone di tre barre degli strumenti per le attività comuni da eseguire:

- barra degli strumenti Standard

- barra degli strumenti Designer
- barra di strumenti Programmi

Queste barre degli strumenti sono divisibili, ovvero è possibile dividerle dalla tavolozza della finestra principale e spostarle fuori della finestra principale o sul desktop.

Suggerimento: per identificare lo scopo di ciascun pulsante della barra degli strumenti, passare con il cursore su ciascun pulsante.

Barra degli strumenti standard: Figura 8 illustra la barra degli strumenti standard.

Figura 8. barra degli strumenti Standard

L'elenco di seguito riportato fornisce la funzione di ciascun pulsante della barra degli strumenti standard, da sinistra a destra:

1. Nuova mappa
2. Apri
3. Salva nel progetto
4. Apri da file
5. Salva nel file
6. Trova nella mappa
7. Stampa mappa
8. Taglia
9. Copia
10. Incolla
11. Elimina
12. Guida

Barra degli strumenti Designer: Figura 9 illustra la barra degli strumenti Designer.

Figura 9. barra degli strumenti Designer

L'elenco di seguito riportato fornisce le funzioni di ciascun pulsante della barra degli strumenti di Designer, da sinistra a destra:

1. Aggiungi oggetto business
2. Convalida
3. Compila
4. Esegui test
5. Continua
6. Ignora
7. Attiva/disattiva punti di interruzione
8. Cancella tutti i punti di interruzione
9. Tutti gli attributi
10. Attributi collegati

11. Attributi non collegati

Barra degli strumenti Programmi: Figura 10 illustra la barra degli strumenti Programmi.

Figura 10. barra di strumenti Programmi

L'elenco di seguito riportato fornisce le funzioni di ciascun pulsante della barra degli strumenti Programmi, da sinistra a destra:

1. Process Designer Express

   Limitazione: questa barra degli strumenti è disponibile solo in WebSphere Business Integration Server Express Plus.
2. Map Designer Express
3. Business Object Designer Express
4. Relationship Designer Express

Tasti di accesso rapido

Map Designer Express dispone dei tasti di accesso rapido illustrati in Tabella 12 per gran parte delle opzioni di menu.

Tabella 12. Tasti di accesso rapido per Map Designer Express

Tasti di accesso rapido	Descrizione	Per ulteriori informazioni
`Ctrl+E`	Salva la definizione della mappa corrente in un file di definizione della mappa	"Salvataggio di una mappa in un file" a pagina 55
`Ctrl+T`	Visualizza il pannello di controllo Trova per ricercare il testo o gli attributi non collegati nella mappa (utilizzare `Ctrl+H` per Sostituisci)	"Ricerca di informazioni in una mappa" a pagina 77
`Ctrl+H`	Visualizza il dialogo Sostituisci per ricercare e sostituire il testo nel codice Java personalizzato e i commenti delle regole di trasformazione.	"Ricerca e sostituzione del testo" a pagina 79
`Ctrl+I`	Apre un file di definizione della mappa	"Procedura per l'apertura di una mappa da un file" a pagina 61
`Ctrl+M`	Visualizza un documento della mappa	"Visualizzazione di un documento della mappa" a pagina 69
`Ctrl+N`	Visualizza la procedura guidata Nuova mappa che consente di creare una nuova mappa	"Creazione di una mappa: Passi di base" a pagina 33
`Ctrl+A`	Apre una definizione di mappa dal progetto in System Manager	"Procedura per l'apertura di una mappa da un progetto in System Manager" a pagina 60
`Ctrl+P`	Stampa la definizione della mappa	"Stampa di una mappa" a pagina 79
`Ctrl+S`	Nella finestra principale di Map Designer Express — Salva la definizione della mappa corrente in un progetto in System Manager	"Salvataggio di una mappa in un progetto" a pagina 53
`Ctrl+U`	Visualizza il dialogo Preferenze per impostare le preferenze di Map Designer Express	"Assegnazione delle preferenze" a pagina 23
`Ctrl+Alt+F`	Salva la definizione della mappa corrente in un file di definizione mappa con un nome diverso (Salva con nome)	"Salvataggio di una mappa in un file" a pagina 55
`Ctrl+Alt+S`	Slava la definizione della mappa corrente nel progetto in System Manager con un nome diverso (Salva con nome)	"Salvataggio di una mappa in un progetto" a pagina 53

Tabella 12. Tasti di accesso rapido per Map Designer Express (Continua)

Tasti di accesso rapido	Descrizione	Per ulteriori informazioni
`Ctrl+Maiusc+P`	Visualizza il dialogo Impostazioni di stampa per specificare le informazioni relative alla stampa della definizione di mappa	"Stampa di una mappa" a pagina 79
`Ctrl+Invio`	Visualizza il dialogo Proprietà della mappa, da cui è possibile impostare le proprietà generali e dell'oggetto business della mappa	"Specifica delle informazioni sulla proprietà della mappa" a pagina 62
`F7`	Compila la mappa corrente	"Compilazione di una mappa" a pagina 89
`Alt+F4`	Chiude la mappa corrente	"Chiusura di una mappa" a pagina 62
`Canc`	Elimina l'entità al momento selezionata	N/D
`F1`	Visualizza la guida contestuale per la finestra o dialogo corrente	N/D
`Ctrl+F7`	Compila tutte o una serie secondaria di mappe definite in System Manager	"Compilazione di una serie di mappe" a pagina 90
`F8`	Durante l'esecuzione di un test, continua una mappa messa in pausa continuando l'esecuzione fino alla fine della mappa o di un altro punto di interruzione	"Passi per l'elaborazione dei punti di interruzione" a pagina 98
`F9`	Attiva/disattiva lo stato di un punto di interruzione di una regola di trasformazione	"Impostazione dei punti di interruzione (punti di interruzione)" a pagina 95
`F10`	Durante l'esecuzione di un test, continua una mappa messa in pausa eseguendo l'unico passo successivo	"Passi per l'elaborazione dei punti di interruzione" a pagina 98

Creazione di una mappa: Passi di base

Tabella 13 fornisce una panoramica delle attività secondarie per la creazione di una nuova mappa.

Tabella 13. Attività secondarie per la creazione di una nuova mappa

Attività secondaria		Procedura associata (consultare. . .)
1.	Creazione di un nuovo file di mappa con la procedura guidata Nuova mappa che specifica il progetto, gli oggetti di origine e di destinazione e il nome della nuova mappa.	"Passi per la creazione della definizione della mappa" a pagina 34.
2.	Impostazione del verbo per ciascun oggetto di destinazione. In gran parte dei casi, gli oggetti business di destinazione dispongono dello stesso verbo degli oggetti business di origine. Inoltre, è possibile impostare il valore del verbo sempre come valore specifico.	"Impostazione del verbo dell'oggetto business di destinazione" a pagina 39.
3.	Specifica dei passi di trasformazione per ciascun attributo di destinazione che si desidera mappare. Il modo di effettuare questa operazione dipende dal tipo di trasformazione richiesta.	"Specifica delle trasformazioni degli attributi standard" a pagina 40.
4.	Specifica del commento per l'attributo di destinazione. Sebbene queste informazioni siano facoltative, ottimizzano la lettura delle informazioni sulla mappa in Map Designer Express.	"Impostazione dei commenti nel campo commenti dell'attributo" a pagina 57.
5.	Salvataggio della mappa.	"Salvataggio delle mappe" a pagina 53.
6.	Verifica del completamento, della convalida e della compilazione della mappa.	"Verifica del completamento" a pagina 56, "Convalida di una mappa" a pagina 88 e "Compilazione di una mappa" a pagina 89
7.	Test e debug della mappa	"Test delle mappe" a pagina 91

Passi per la creazione della definizione della mappa

Map Designer Express dispone di una procedura guidata Nuova mappa per consentire la creazione di una definizione della mappa. Effettuare i seguenti passi per creare una definizione mappa utilizzando la procedura guidata Nuova mappa :

1. Avviare la procedura guidata Nuova mappa in uno dei seguenti modi:
   - Dal Menu File, selezionare Nuovo per creare una nuova mappa.
   - Utilizzare la combinazione di tasti di accesso rapido `Ctrl+N`.
   - Nella Barra degli strumenti standard, fare clic sul pulsante Nuova mappa.

   Risultato: Map Designer Express visualizza la prima finestra della procedura guidata Nuova mappa.

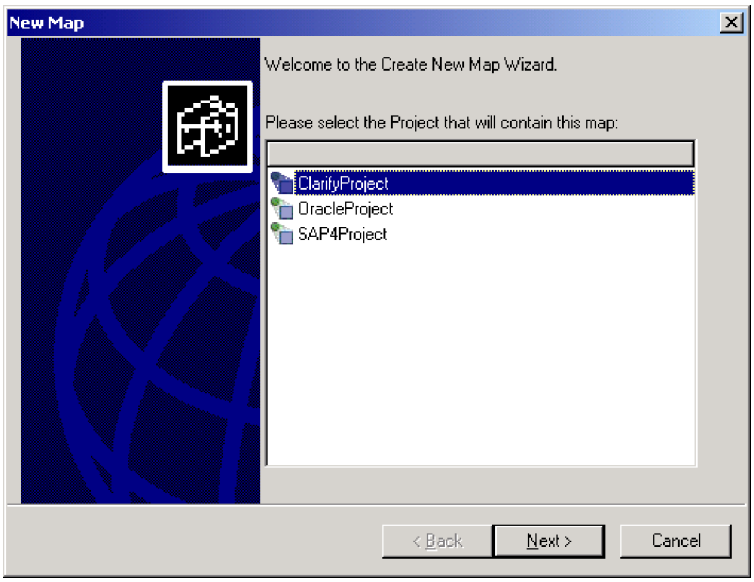


Figura 11. Finestra di Benvenuto della procedura guidata Nuova mappa

2. Dalla casella dell'elenco, selezionare il nome del progetto per cui si desidera creare la mappa.
3. Selezionare l'oggetto business da utilizzare come oggetto business di origine per la mappa. E' possibile selezionare uno o più oggetti business di origine facendo clic nella colonna Utilizza per ciascun oggetto business desiderato. Quindi, fare clic su Avanti per continuare.

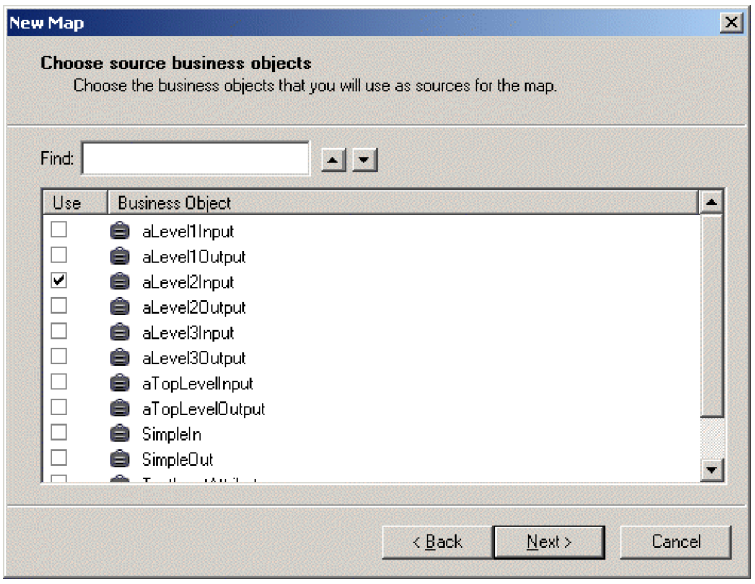


Figura 12. Selezione degli oggetti business di origine

Suggerimento: per ricercare un determinato oggetto business, immetterne il nome nel campo Trova. Le frecce Su e Giù scorrono l'elenco degli oggetti business. Fare clic su Avanti per continuare.

La procedura guidata Nuova mappa *non* richiede di specificare l'oggetto business di origine. E' possibile fare clic su Avanti senza selezionare l'oggetto business di origine per specificare in seguito la definizione di tale oggetto. E' possibile specificarla in seguito nello spazio di lavoro della mappa della scheda Diagramma. Per ulteriori informazioni, consultare "Creazione degli oggetti business di origine e di destinazione" a pagina 37.

Nota: Se si aggiunge o si elimina un oggetto business da System Manager, l'elenco delle definizioni degli oggetti business viene aggiornato dinamicamente.

4. Selezionare il tipo di oggetto business da utilizzare come oggetto business di destinazione per la mappa. E' possibile selezionare uno o più oggetti business di destinazione facendo clic nella colonna Utilizza per ciascun oggetto business desiderato. Quindi, fare clic su Avanti per continuare.

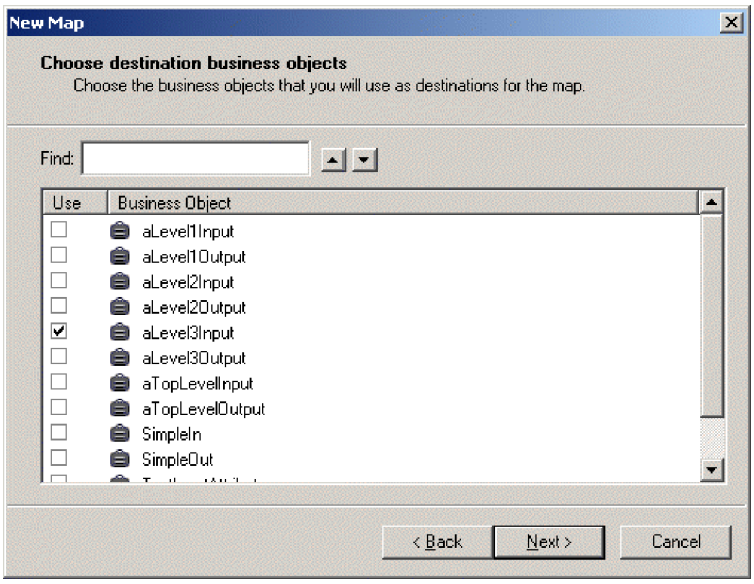


Figura 13. Selezione di un oggetto business di destinazione

Suggerimento: per ricercare un determinato oggetto business, immetterne il nome nel campo Trova. Le frecce Su e Giù scorrono l'elenco degli oggetti business. Fare clic su Avanti per continuare.

La procedura guidata Nuova mappa non richiede la specifica dell'oggetto business di destinazione. E' possibile fare clic su Avanti senza selezionare l'oggetto business di destinazione per specificare in seguito la definizione di tale oggetto. E' possibile specificarla in seguito nello spazio di lavoro della mappa della scheda Diagramma. Per ulteriori informazioni, consultare "Creazione degli oggetti business di origine e di destinazione" a pagina 37.

Nota: Se si aggiunge o si elimina un oggetto business da System Manager, l'elenco delle definizioni degli oggetti business viene aggiornato dinamicamente.

5. Specificare il nome da associare alla mappa.

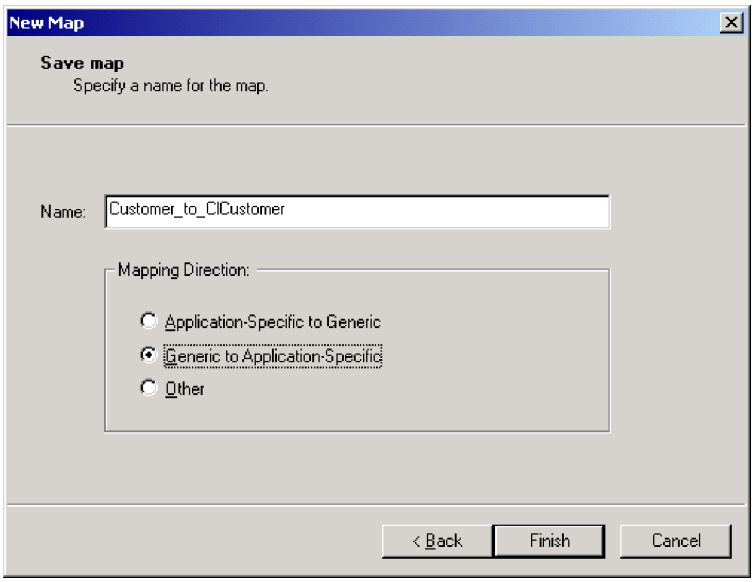


Figura 14. Salvataggio di una nuova mappa

Regola: Un nome mappa deve essere inferiore o uguale a 76 caratteri alfanumerici e trattini bassi (_). *Non può* contenere spazi o alcuni simboli di punteggiatura, come ad esempio punti, parentesi quadre aperte([), parentesi quadre chiuse (]), apici singoli o doppi apici.

La procedura guidata Nuova mappa *non* richiede che sia specificato il nome della mappa. E' possibile fare clic sul Fine senza immettere il nome della mappa e rimandare questa definizione della mappa. Quando si salva la mappa, Map Designer Express visualizza il dialogo Salva mappa con nome che richiede di specificare il nome della mappa. Per ulteriori informazioni, consultare "Salvataggio di una mappa in un progetto" a pagina 53.

Specificare se la mappa è in entrata o in uscita. Questo ruolo della mappa è necessario per generare automaticamente i codici di relazione.

6. Fare clic su Fine per salvare la nuova definizione della mappa con gli oggetti di origine e di destinazione specificati.

   Risultato: Map Designer Express visualizza le informazioni sulla nuova mappa nella scheda Diagramma.

Creazione degli oggetti business di origine e di destinazione

Se non si specificano gli oggetti business di origine e di destinazione della mappa nella procedura guidata Nuova mappa, è possibile specificarli nella finestra di dialogo Aggiungi oggetto business o nella scheda Diagramma nel browser dell'oggetto business.

Passi per la specifica di oggetti business dal dialogo Aggiungi oggetto business

Effettuare i passi di seguito riportati per aggiungere un oggetto business di origine o di destinazione ad una mappa dalla scheda Generale del dialogo Aggiungi oggetto business.

1. Visualizzare il dialogo Aggiungi oggetto business in uno dei seguenti modi:
   - Dal Menu Modifica di Map Designer Express, selezionare Aggiungi oggetto business.
   - Nella barra degli strumenti di Designer, fare clic sul pulsante Aggiungi oggetto business.
   - Dalla scheda Tabella, fare clic con il tastino destro del mouse nell'area vuota del riquadro degli oggetti business, quindi selezionare Aggiungi oggetto business dal menu Contesto.
   - Dalla scheda Diagramma, fare clic con il tastino destro del mouse nello spazio di lavoro della mappa e selezionare Aggiungi oggetto business dal menu Contesto.
2. Per specificare un oggetto business di origine.
   - Fare clic sull'oggetto business nell'elenco degli oggetti business.
   - Fare clic sul pulsante Aggiungi all'origine.

   Suggerimento: per ricercare un determinato oggetto business, immetterne il nome nel campo Trova. Le frecce Su e Giù scorrono l'elenco degli oggetti business.
3. Per specificare un dell'oggetto business di destinazione:
   - Fare clic sull'oggetto business nell'elenco degli oggetti business.
   - Fare clic sul pulsante Aggiungi destinazione.

Suggerimento: per ricercare un determinato oggetto business, immetterne il nome nel campo Trova. Le frecce Su e Giù scorrono l'elenco degli oggetti business.

4. Per chiudere il dialogo, fare clic su Fine.

Passi per la specifica degli oggetti business dalla scheda Diagramma nel browser degli oggetti business

Dalla scheda Diagramma, è possibile aggiungere un oggetto business di origine o di destinazione ad una mappa. Per effettuare tale operazione, seguire i passi indicati:

1. Trascinare l'oggetto business di origine dal browser degli oggetti business al lato sinistro dello spazio di lavoro della mappa. Viene visualizzato l'oggetto business ed il relativo titolo che inizia con `Src`.
2. Trascinare l'oggetto business di destinazione dal browser degli oggetti business al lato destro dello spazio di lavoro della mappa. Viene visualizzato l'oggetto business ed il relativo titolo che inizia con `Dest`.

Nota: Un limite punteggiato divide le metà destra e sinistra dello spazio di lavoro ed identifica le parti di origine e di destinazione dello spazio di lavoro della mappa. Assicurarsi di rilasciare gli oggetti alla posizione appropriata.

Figura 15 illustra gli oggetti business di origine e di destinazione nello spazio di lavoro della mappa.

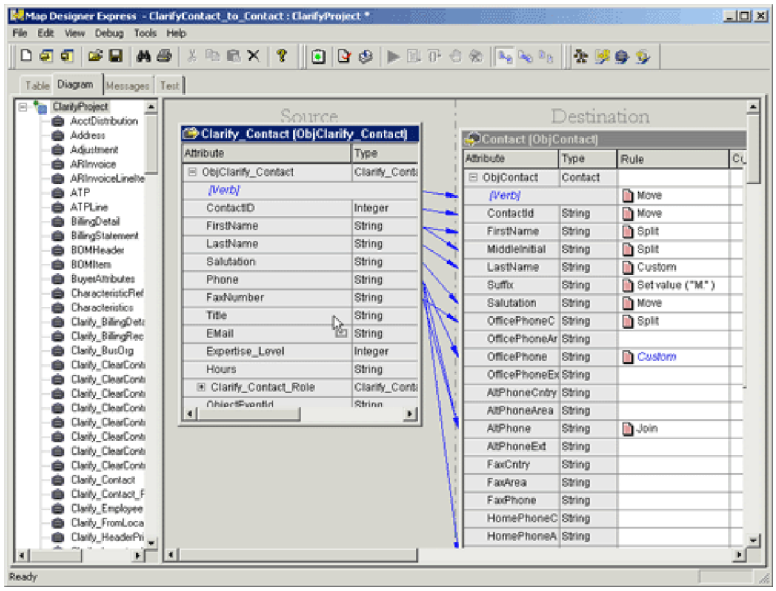


Figura 15. Definizione degli oggetti business di Origine e di Destinazione

Suggerimento: in alternativa, è possibile creare gli oggetti business di origine e di destinazione facendo clic con il tastino destro del mouse sull'oggetto business nel relativo browser, selezionando Copia dal menu Contesto, quindi facendo clic con il tastino destro del mouse nello spazio di lavoro della mappa e selezionando Incolla come oggetto di input o Incolla come oggetto di output.

Map Designer Express crea una finestra denominata *finestra dell'oggetto business*, per gli oggetti di origine e di destinazione. La barra dei titoli di questa finestra visualizza il nome dell'istanza dell'oggetto business. Per una guida sulla

descrizione della barra dei titoli della finestra relativa all'oggetto business, consultare "Utilizzo delle variabili e degli attributi generati dell'oggetto business" a pagina 178. La finestra dell'oggetto business per l'oggetto business di origine contiene colonne per il nome e il tipo di dati di ciascun attributo di origine. La finestra dell'oggetto business per l'oggetto business di destinazione contiene le colonne per il nome, il tipo di dati, la regola di trasformazione (che identifica il passo di trasformazione) e un commento facoltativo.

Linee guida: se si effettua un errore trascinando l'oggetto business errato o specificandolo come oggetto di output invece che come oggetto di input, è possibile eliminare l'oggetto dallo spazio di lavoro della mappa e riprovare. Per eliminare un oggetto business dallo spazio di lavoro della mappa, è possibile:

- Selezionare l'oggetto business da eliminare e dal menu Modifica Elimina selezione corrente (o premere il tasto `Canc`).
- Fare clic con il tastino destro del mouse sulla barra dei titoli della finestra degli oggetti business, quindi selezionare Elimina dal menu Contesto.

Impostazione del verbo dell'oggetto business di destinazione

Il verbo indica il modo in cui il sistema elabora i dati dell'oggetto business. Quando una mappa viene eseguita, il sistema deve conoscere il verbo da assegnare a ciascun oggetto di destinazione creato.

Se la mappa dispone solo di un oggetto business di origine e un oggetto business di destinazione, il verbo per l'oggetto business di destinazione è, in genere, uguale al verbo per l'oggetto business di origine.

In questo caso, è necessario copiare il verbo dall'oggetto business di origine all'oggetto business di destinazione (consultare Figura 15 a pagina 38), definendo una regola di trasformazione *Sposta* con l'attributo di origine come verbo dell'oggetto business di origine e l'attributo di destinazione come verbo dell'oggetto business di destinazione. Per ulteriori informazioni, consultare "Copia di un attributo di origine ad un attributo di destinazione" a pagina 42.

Suggerimento: inoltre, è possibile trascinare e rilasciare il verbo dall'oggetto business di origine a quello di destinazione per definire il valore del verbo.

Se una mappa dispone di un oggetto business di destinazione con un verbo che non è stato trovato nell'oggetto business di origine, è necessario impostare il verbo su un valore costante, definendo una regola di trasformazione *Imposta valore* con l'attributo di destinazione come verbo dell'oggetto business di destinazione. Nella finestra di dialogo Imposta valore, immettere il valore del verbo della costante. Per ulteriori informazioni, consultare "Specifica del valore per un attributo" a pagina 40.

Talvolta, le mappa dispongono di più di un oggetto business di origine o di destinazione e tali oggetti dispongono di vari oggetti business child. In tali casi, è necessario considerare attentamente il verbo da assegnare a ciascun oggetto business di destinazione. Alcuni oggetti business di destinazione potrebbero richiedere alcune logiche di personalizzazione per impostare il verbo in base ai verbi di uno o più oggetti business di origine.

Specifica delle trasformazioni degli attributi standard

E' possibile specificare varie trasformazioni degli attributi standard in modo interattivo in Map Designer Express durante la scrittura del codice Java. Tabella 14 illustra le trasformazioni standard che è possibile specificare in Map Designer Express.

Tabella 14. Trasformazioni degli attributi comuni

Nome	Passo di trasformazione	Scopo
Imposta valore	"Specifica del valore per un attributo"	Per un attributo nell'oggetto business di destinazione che non viene trovato nell'oggetto business di origine, ma che viene richiesto nell'applicazione di destinazione
Sposta	"Copia di un attributo di origine ad un attributo di destinazione" a pagina 42	Per un attributo che è uguale sia negli oggetti di origine sia negli oggetti di destinazione
Aggiungi	"Aggiunta di attributi" a pagina 43	Per un attributo nell'oggetto business di destinazione che è una combinazione di vari attributi nell'oggetto business di origine
Dividi	"Divisione degli attributi" a pagina 45	Per un attributo nell'oggetto business di destinazione che è: • Solo una parte di un attributo nell'oggetto business di origine • Costituito di vari campi, ma con diversi delimitatori rispetto a quelli nell'oggetto business di origine
Mappa secondaria	"Trasformazione con una mappa secondaria" a pagina 47	Per gli attributi negli oggetti business di destinazione che contengono oggetti business child
Riferimento incrociato	"Relazioni di identità con riferimento incrociato" a pagina 51	Per conservare le relazioni di identità per gli oggetti business
Personalizza	"Creazione di una trasformazione personalizzata" a pagina 52	Per un attributo che richiede le trasformazioni non fornite dalle trasformazioni generate automaticamente

Per informazioni sulle ulteriori trasformazioni che è possibile eseguire, consultare "Ulteriori metodi di trasformazione attributi" a pagina 183.

Nella scheda Diagramma, è possibile selezionare gli attributi da visualizzare nelle finestre degli oggetti aziendali con le opzioni del menu Visualizza > Diagramma. E' possibile scegliere di visualizzare tutti gli attributi, solo quelli collegati (mappati) o solo quelli non collegati (non mappati).

Suggerimenti: Gli attributi vengono visualizzati nello stesso ordine in cui vengono visualizzati nella definizione dell'oggetto business. Per localizzare un determinato attributo in un lungo elenco di attributi, selezionare Trova dal menu Modifica (oppure utilizzare la combinazione di tasti di accesso rapido `Ctrl+F`). Per ulteriori informazioni, consultare "Ricerca di informazioni in una mappa" a pagina 77.

Specifica del valore per un attributo

Alcuni valori degli attributi di destinazione non dipendono da un attributo di origine e possono essere compilati con un valore della costante. Ciò è particolarmente vero se l'oggetto business di destinazione contiene più attributi che non sono stati trovati nell'oggetto business di origine, ma sono richiesti nell'applicazione di destinazione. Alcuni esempio di valori predefiniti per gli attributi sono `CustomerStatus = "active"` o `AddressType = "business"`.

Questo tipo di trasformazione è denominata *Imposta valore*. Impostare il valore di un attributo di destinazione con il dialogo Imposta valore, illustrato in Figura 16.

Passi per la specifica di una trasformazione Imposta valore: Effettuare i passi di seguito riportati per specificare una trasformazione Imposta valore:

1. Visualizzare il dialogo Imposta valore in uno dei seguenti modi:
   - Dalla scheda Tabella, effettuare i seguenti passi:
     a. Selezionare l'attributo di destinazione cui si desidera impostare il valore.
     b. Fare clic su Imposta valore dall'elenco che si trova nella colonna Regola di trasformazione.
   - Dalla Scheda Diagramma, effettuare i seguenti passi:
     a. Selezionare l'attributo di destinazione cui si desidera impostare il valore.
     b. Fare clic su Imposta valore dall'elenco che si trova nella colonna Regola dell'oggetto business di destinazione.
   - Se una trasformazione Imposta valore è già definita, è possibile visualizzare il dialogo Imposta valore per riconfigurare la trasformazione, compresa la modifica del relativo codice di trasformazione, in uno dei seguenti modi:
     – Fare doppio clic nella cella corrispondente della colonna della regola di trasformazione.
     – Fare clic sull'icona della bitmap Imposta valore che si trova nella colonna della regola di trasformazione.

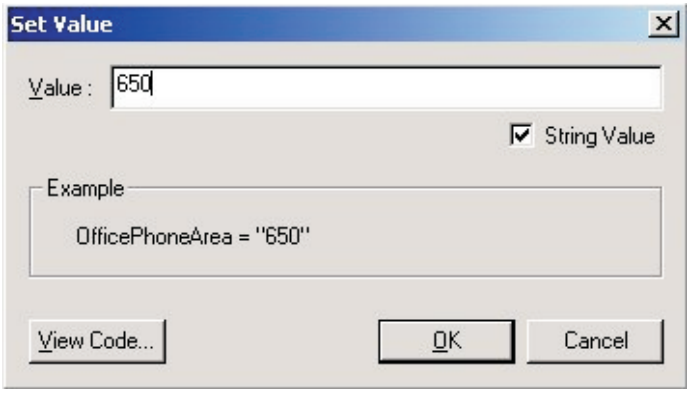


Figura 16. Dialogo Imposta valore

2. Mediante il dialogo Imposta valore, impostare il valore della costante da assegnare all'attributo di destinazione. Il dialogo Imposta valore dispone delle seguenti funzioni:
   - Per specificare il valore della costante, immetterlo nel campo Valore. Per i valori numerici, immettere semplicemente il numero ed assicurarsi che la casella di controllo Valore della stringa non sia selezionata. Per i valori della stringa, immettere il valore della stringa nel campo Valore, quindi selezionare la casella di controllo Valore della stringa.

     Nota: Il dialogo Valore della stringa utilizza l'area Esempi per mostrare la visualizzazione del risultante attributo di destinazione.
   - Per modificare il valore immesso, fare clic nel campo Valore, quindi modificare in modo appropriato.

- Per personalizzare il codice generato, fare clic sul pulsante di opzione Visualizza codice.

  Risultato: Map Designer Express apre Activity Editor nella vista Java, contenente un esempio del codice di trasformazione in modalità di sola lettura per l'attributo di destinazione. Per effettuare le modifiche al codice di trasformazione, fare clic su Modifica codice in Activity Editor. Per ulteriori informazioni, consultare "Panoramica di Activity Editor" a pagina 109.

  Nota: Quando si salvano le modifiche in Activity Editor, vengono comunicate a Map Designer Express. Quando si salva la mappa, vengono salvate anche le modifiche.
- Per confermare le impostazioni della trasformazione, fare clic su Ok.

Copia di un attributo di origine ad un attributo di destinazione

Il tipo più semplice del passo di trasformazione è la copia di un attributo di origine in un attributo di destinazione corrispondente. Questo tipo di trasformazione è denominata trasformazione di *Spostamento*.

Passi per la specifica di una trasformazione di spostamento: Effettuare i passi da uno di queste schede della mappa per specificare una trasformazione di Spostamento:

- Dalla scheda Tabella:
  1. Seleziona l'attributo di origine.
  2. Seleziona l'attributo di destinazione
  3. Fare clic per spostare dall'elenco nella colonna Regola di trasformazione.
- Dalla scheda Diagramma:
  1. Seleziona l'attributo di origine.
  2. Utilizzare `Ctrl+Trascina` per spostare nell'attributo di destinazione, ovvero tenere premuto il tasto `Ctrl` e trascinare l'attributo sull'attributo di destinazione nella finestra dell'oggetto business di destinazione. Continuare a tenere premuto il tasto `Ctrl` fino a quando non viene rilasciato il pulsante del mouse, altrimenti l'operazione non riesce.

     Risultato: Map Designer Express crea una freccia blu dall'oggetto di origine a quello di destinazione. Se la trasformazione coinvolge un singolo attributo di origine che *non* è un oggetto business child, Map Designer Express suppone che tale trasformazione è uno spostamento e la assegna automaticamente Sposta alla colonna Regola dell'attributo di destinazione.

  Suggerimento: è possibile personalizzare la sequenza di chiavi utilizzate per iniziare una trasformazione di spostamento nella scheda Diagramma dalla scheda Mappatura chiave del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica della mappatura chiave" a pagina 25.

Risultato: Map Designer Express genera il codice per copiare il valore dell'attributo di origine a quello di destinazione. Se gli attributi di origine e di destinazione sono tipi di dati diversi, Map Designer Express determina se un tipo di conversione è possibile o meno, e se lo è genera il codice per convertire il tipo di origine al tipo di destinazione. Se una conversione del tipo *non* è possibile oppure potrebbe causare una perdita di dati, Map Designer Express visualizza una finestra di dialogo per confermare o annullare l'operazione.

Se si desidera visualizzare un esempio del codice generato per la trasformazione di spostamento, nel menu Contesto della colonna regola, selezionare Visualizza origine.

Aggiunta di attributi

E' possibile concatenare o aggiungere i valori per più di un attributo di origine in un singolo attributo di destinazione. Questo tipo di trasformazione è denominato trasformazione di *Unione*. Ad esempio, l'oggetto business di origine potrebbe memorizzare il codice postale, il numero di telefono e l'estensione in attributi a parte, mentre l'oggetto business di destinazione memorizza tali valori insieme in un attributo.

Oltre all'aggiunta degli attributi, è possibile ordinarli e inserire delimitatori, parentesi o altri caratteri. Ad esempio, quando si aggiungono attributi dei numeri telefonici e codici postali in un solo attributo, è possibile inserire il codice postale tra parentesi.

Suggerimenti: Gli attributi da aggiungere possono talvolta trovarsi in più di un oggetto business di origine, come ad esempio in un oggetto business parent e in uno degli oggetti business child. Inoltre, è possibile aggiungere un attributo ad una variabile definita. (Per ulteriori informazioni sulla definizione delle variabili, consultare "Creazione di variabili temporanee" a pagina 180.)

Aggiungere più attributi di origine in un attributo di destinazione con il dialogo Aggiungi, illustrato in Figura 17.

Passi per la specifica di una trasformazione di aggiunta: Effettuare i passi di seguito riportati per specificare una trasformazione di aggiunta:

1. Visualizzare il dialogo Aggiungi in uno dei seguenti modi:
   - Dalla scheda Tabella:
     a. Selezionare gli attributi di origine da aggiungere.

        Suggerimento: è possibile fare clic su Più attributi nella casella combinata per visualizzare il dialogo Più attributi. In questo dialogo, è possibile selezionare più attributi d'origine. Per ricercare un determinato oggetto business, immetterne il nome nel campo Trova. Le frecce Su e Giù scorrono l'elenco degli oggetti business. Una volta selezionati gli attributi di origine, fare clic su OK per chiudere il dialogo.
     b. Selezionare l'attributo di destinazione singolo.
     c. Fare clic su Aggiungi dall'elenco nella colonna Regola di trasformazione.
   - Dal scheda Diagramma:
     a. Selezionare due o più attributi di origine.
     b. Utilizzare `Ctrl+Trascina` per spostare l'attributo di destinazione, ovvero tenere premuto il tasto `Ctrl` e trascinare gli attributi selezionati nell'attributo di destinazione. Continuare a tenere premuto il tasto `Ctrl` fino a quando non viene rilasciato il pulsante del mouse, altrimenti l'operazione non riesce.

        Risultato: se la trasformazione coinvolge più di un attributo di origine, Map Designer Express suppone che la trasformazione è di aggiunta. Quindi, assegna automaticamente Aggiungi alla colonna Regola dell'attributo di destinazione e visualizza il dialogo Aggiungi.

     Suggerimento: è possibile personalizzare la sequenza di chiavi utilizzate per iniziare una trasformazione di aggiunta nella scheda Digramma della scheda Mappatura chiave del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica della mappatura chiave" a pagina 25.

- Se una trasformazione Aggiungi è già definita, è possibile visualizzare il dialogo Aggiungi per riconfigurare la trasformazione, compresa la modifica del relativo codice di trasformazione, in uno dei seguenti modi:
  - Fare doppio clic nella cella corrispondente della colonna della regola di trasformazione.
  - Fare clic sull'icona della bitmap Aggiungi che si trova nella colonna della regola di trasformazione.

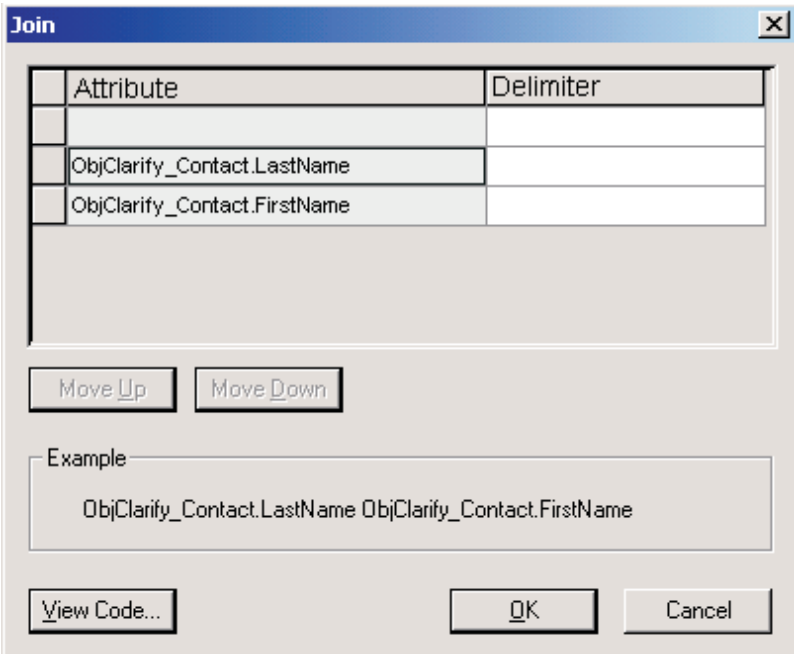


Figura 17. Dialogo Aggiungi

2. Mediante il dialogo Aggiungi, è possibile creare un'espressione per concatenare gli attributi di origine aggiungendo delimitatori, raggruppando con parentesi e riordinando gli attributi se necessario. Il dialogo Aggiungi dispone delle seguenti funzioni:
   - Per inserire un delimitatore o le parentesi, immetterne il valore nel campo Delimitatore associato all'attributo. *Non* inserire i delimitatori tra gli apici. il delimitatore immesso viene aggiunto all'attributo associato. Per i delimitatori principali, immetterne il valore nel campo Delimitatori della prima riga vuota.

     Nota: Il dialogo Aggiungi utilizza l'area Esempi per illustrare il modo in cui verrà visualizzata la stringa risultate dopo l'unione.
   - Per modificare il delimitatore o le parentesi immesse, fare clic nel campo Delimitatore, quindi modificare in modo appropriato.
   - Per riordinare un delimitatore o gli attributi, fare clic sulla colonna a sinistra per selezionare la riga, quindi fare clic su Sposta verso l'alto o Sposta verso il basso per spostare l'intera riga verso l'alto o il basso.
   - Per personalizzare il codice generato, fare clic sul pulsante di opzione Visualizza codice.

     Risultato: Map Designer Express apre Activity Editor nella vista Java, contenente un esempio del codice di trasformazione in modalità di sola lettura per l'attributo di destinazione. Per effettuare le modifiche al codice di

trasformazione, fare clic su Modifica codice in Activity Editor. Per ulteriori informazioni, consultare “Panoramica di Activity Editor” a pagina 109.

Nota: Quando si salvano le modifiche in Activity Editor, vengono comunicate a Map Designer Express. Quando si salva la mappa, vengono salvate anche le modifiche.

- Per confermare le impostazioni della trasformazione, fare clic su Ok.

Risultato: Map Designer Express genera il codice per aggiungere gli attributi di origine. Se un attributo di origine è di un tipo dati diverso dall'attributo di destinazione, Map Designer Express effettua i richiami necessari ai metodo nella classe `DtpDataConversion` per convertire i tipi.

Divisione degli attributi

Per dividere un attributo di origine in due o più attributi di destinazione, specificare la trasformazione separatamente per ciascun attributo di destinazione. Questo tipo di trasformazione è denominata trasformazione *Dividi*. Ad esempio, per dividere un attributo di origine, come ad esempio `phone_number`, in tre separati attributi di destinazione, quali `area_code`, `tel_number` e `estensione`, specificare le trasformazioni per `area_code`, `tel_number` e `estensione` separatamente.

Dividere un attributo di origine in più attributi di destinazione con il dialogo Dividi, illustrato in Figura 18.

Passi per la specifica di una trasformazione di divisione: Effettuare i passi di seguito riportati per specificare una trasformazione di divisione:

1. Visualizzare il dialogo Dividi in uno dei seguenti modi:
   - Dalla scheda Tabella, effettuare i seguenti passi:
     a. Selezionare il singolo attributo di origine da dividere.
     b. Selezionare uno degli attributi di destinazione desiderati.
     c. Fare clic su Dividi dall'elenco nella colonna Regola di trasformazione.
     d. Ripetere questi passi per ciascuno degli attributi di destinazione che riceve un segmento dell'attributo di origine.
   - Dalla Scheda Diagramma, effettuare i seguenti passi:
     a. Selezionare il singolo attributo di origine da dividere.
     b. Utilizzare `Alt+Trascina` per spostarsi in uno degli attributi di destinazione, ovvero tenere premuto il tasto `Alt` e trascinare l'attributo di origine su uno degli attributi di destinazione.

        Risultato: se la trasformazione coinvolge più di un attributo di destinazione, Map Designer Express suppone che la trasformazione è una divisione. Quindi, assegna automaticamente Dividi alla colonna Regola dell'attributo di destinazione e visualizza il dialogo Aggiungi.
     c. Ripetere questi passi per ciascuno degli attributi di destinazione che riceve un segmento dell'attributo di origine.

     Suggerimento: è possibile personalizzare la sequenza di chiavi utilizzate per iniziare una trasformazione di divisione nella scheda Diagramma dalla scheda Mappatura chiave del Dialogo Preferenze. Per ulteriori informazioni, consultare “Specifica della mappatura chiave” a pagina 25.
   - Se una trasformazione Dividi è già definita, è possibile visualizzare il dialogo Dividi per riconfigurare la trasformazione, compresa la modifica del relativo codice di trasformazione, in uno dei seguenti modi:

– Fare doppio clic nella cella corrispondente della colonna della regola di trasformazione.
– Fare clic sull'icona della bitmap Dividi che si trova nella colonna della regola di trasformazione.

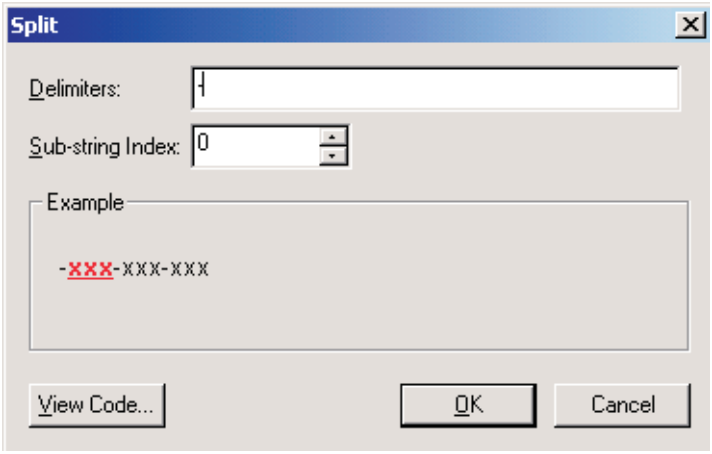


Figura 18. Dialogo Dividi

2. Mediante il dialogo Dividi, dividere un'espressione in segmenti che sono separati da un delimitatore. Ciascun segmento è identificato con un numero di indice, con il primo segmento con un numero di indice zero (0). Il dialogo Dividi dispone delle seguenti funzioni:
   - Per identificare il delimitatore con cui analizzare l'attributo di origine, immetterlo nel campo Delimitatore. *Non* inserire i delimitatori tra gli apici. E' possibile specificare uno o più delimitatori nel campo. La trasformazione utilizza ciascuno dei delimitatori specificati per analizzare la stringa nel segmento. Ad esempio, per dividere `LastName,FirstName`, specificare "`,`" come delimitatore, `LastName` come segmento 0 (il primo segmento) e `FirstName` come segmento 1 (il secondo segmento).

     Nota: Il dialogo Dividi utilizza l'area Esempi per visualizzare la stringa dell'attributo di origine e per indicare il segmento che al momento è acceduto. Il segmento acceduto viene visualizzato in rosso e grassetto.
   - Per modificare il delimitatore o le parentesi immesse, fare clic nel campo Delimitatore, quindi modificare in modo appropriato.
   - Per identificare il segmento dell'attributo di origine che viene copiato nell'attributo di destinazione, immettere il relativo numero di indice nel campo Indice stringa secondaria.
   - Per personalizzare il codice generato, fare clic sul pulsante di opzione Visualizza codice.

     Risultato: Map Designer Express apre Activity Editor nella vista Java, contenente un esempio del codice di trasformazione in modalità di sola lettura per l'attributo di destinazione. Per effettuare le modifiche al codice di trasformazione, fare clic su Modifica codice in Activity Editor. Per ulteriori informazioni, consultare "Panoramica di Activity Editor" a pagina 109.

     Nota: Quando si salvano le modifiche in Activity Editor, vengono comunicate a Map Designer Express. Quando si salva la mappa, vengono salvate anche le modifiche.
   - Per confermare le impostazioni della trasformazione, fare clic su Ok.

Risultato: Map Designer Express genera il codice di trasformazione per l'attributo di destinazione. Il codice generato utilizza metodi della classe `DtpSplitString()` per analizzare l'attributo di origine in segmenti.

Trasformazione con una mappa secondaria

Una *mappa secondaria* è una mappa richiamata all'interno di un'altra mappa, denominata *mappa principale*. Questa sezione fornisce le seguenti informazioni sulle mappe secondarie:

- "Utilizzi delle mappe secondarie"
- "Passi per la specifica della trasformazione di una mappa secondaria" a pagina 49

Utilizzi delle mappe secondarie: E' possibile richiamare una mappa secondari per ottenere un valore per qualunque attributo di destinazione, ma le mappe secondarie vengono più comunemente utilizzate per i seguenti scopi:

- Per rendere una mappa modulare
- Per specificare le trasformazioni tra gli oggetti business child

Ottimizzazione della modularità delle mappe: Utilizzando le mappe secondarie è possibile ottimizzare la modularità delle mappe isolando le trasformazioni comuni che possono essere riutilizzate in più di una mappa. Ad esempio, un oggetto business `Customer` potrebbe disporre di un oggetto business child `Address` che è anche un child di un oggetto business `Order`. Se si crea una mappa secondari per l'oggetto business `Address`, è possibile riutilizzare la mappa secondaria in entrambe le mappe dell'oggetto business `Customer` e `Order`.

Figura 19 illustra il modo in cui una mappa secondaria `MyAddrToGenAddr` può essere riutilizzata da due mappe diverse.

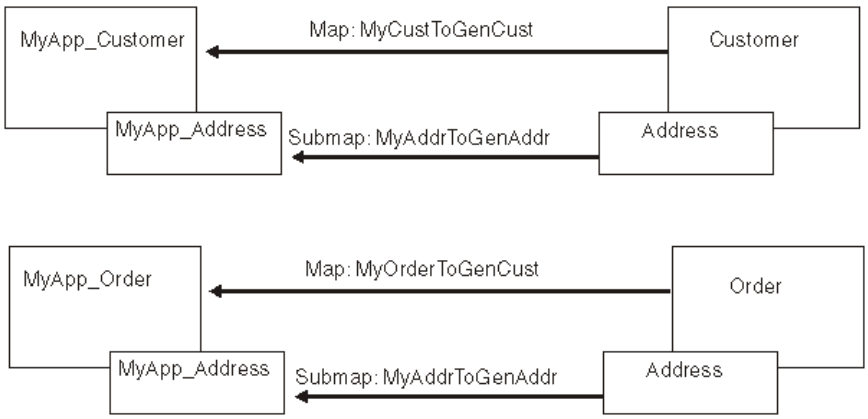


Figura 19. Utilizzo delle mappe secondarie per la modularità

Trasformazione degli oggetti business child: Quando gli attributi di origine e di destinazione contengono oggetti business child con cardinalità multipla, è utile utilizzare una mappa secondaria per specificarne le trasformazioni. Gli esempi tipici degli oggetti business child con cardinalità multipla sono indirizzi multipli di un cliente o voci di riga multiple in un ordine.

Nel caso più semplice, si trasforma ciascun oggetto business child di origine in un singolo oggetto business child di destinazione, in una relazione uno-a-uno.

Figura 20 illustra l'utilizzo di mappe secondarie per un oggetto business `Employee` ed il relativo vettore del business child contenente le istanze di `EmployeeAddress`.

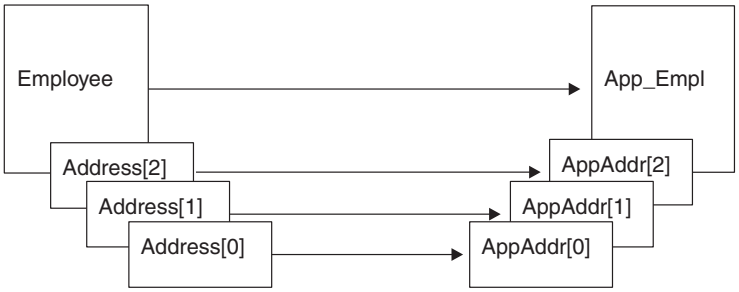


Figura 20. Trasformazione uno-a-uno di vettori dell'oggetto business child

Una mappa secondaria può essere associata ad un'istruzione condizionale che ne regola l'esecuzione. Ad esempio, si consideri Figura 21: l'oggetto business `Order` dispone di un attributo `OrderLine` che contiene un oggetto business child con cardinalità multipla, `OrderLine`. L'oggetto business `OrderLine` dispone di un attributo `DeliverySchedule` contenente un oggetto business child con cardinalità multipla, `DelSched`.

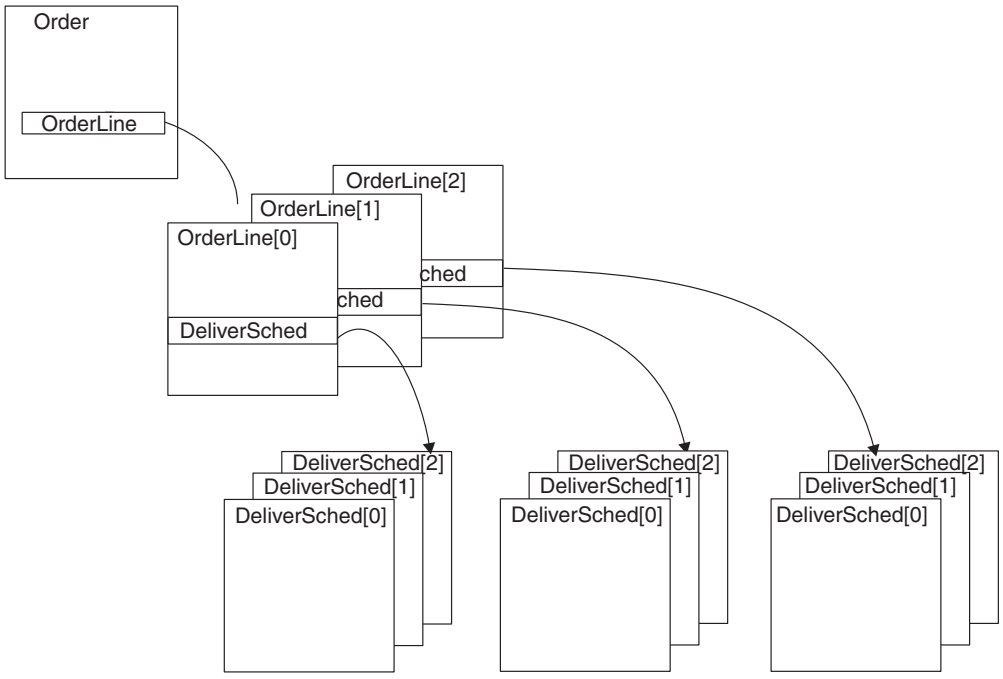


Figura 21. L'oggetto business di origine con oggetto business child con cardinalità multipla

Alcune condizioni che possono essere scritte nella mappa per `Order` possono:

- Eseguire la mappa secondaria che trasforma l'attributo `OrderLine` in `Order` solo se un attributo diverso in `Order` ha un valore particolare.
- Eseguire la mappa secondaria che trasforma l'attributo `DeliverSched` in `OrderLine` solo se un attributo diverso in `OrderLine` ha un valore particolare.
- Eseguire la mappa secondaria che trasforma l'attributo `DeliverSched` in `OrderLine` solo se un attributo in `Order` ha un valore particolare.

Passi per la specifica della trasformazione di una mappa secondaria: Eseguire i passi indicati per specificare una trasformazione di una mappa secondaria:

1. Creare la mappa che si desidera utilizzare come mappa secondaria:

   Suggerimento: procedere nello stesso modo in cui sono state create e salvate le altre mappe. La convenzione di denominazione IBM consiglia di denominare la mappa secondaria con la stringa "Sub_".
2. Salvare la mappa secondaria nel progetto in System Manager e compilare la mappa secondaria.
3. Specificare la trasformazione della mappa secondaria nell'attributo nell'oggetto business parent che deve richiamare la mappa secondaria. Questo attributo di origine contiene un oggetto business child mappato ad un attributo di destinazione contenente un oggetto business child.

   Specificare che una mappa secondaria deve essere richiamata con il dialogo Mappa secondaria illustrato in Figura 22. Visualizzare il dialogo Mappa secondaria in uno dei seguenti modi:

   - Dalla scheda Tabella, effettuare i seguenti passi:
     a. Nella mappa parent, selezionare un attributo di origine (che è un oggetto business child).
     b. Selezionare l'attributo di destinazione desiderato (che è anche un oggetto business child).
     c. Fare clic su Mappa secondaria dall'elenco nella colonna Regola di trasformazione.
     d. Ripetere questi passi per ciascun attributo di origine che è un oggetto business di origine per la mappa secondaria e ciascun attributo di destinazione che è un oggetto business di destinazione per questa mappa secondaria.
   - Dalla Scheda Diagramma, effettuare i seguenti passi:
     a. Nella mappa parent, selezionare l'attributo di origine (che è un oggetto business child).
     b. Utilizzare Ctrl+Trascina per spostare l'attributo di destinazione, ovvero tenere premuto il tasto Ctrl e trascinare l'attributo di origine nell'attributo di destinazione. Continuare a tenere premuto il tasto Ctrl fino a quando non viene rilasciato il pulsante del mouse, altrimenti l'operazione non riesce.

        Se la trasformazione coinvolge un attributo di origine che è un oggetto business child, Map Designer Express suppone che la trasformazione è una Mappa secondaria. Quindi, assegna automaticamente la Mappa secondaria alla colonna Regola dell'attributo di destinazione e visualizza il dialogo Mappa secondaria.

     Suggerimento: è possibile personalizzare la sequenza di chiavi utilizzate per iniziare una trasformazione di Mappa secondaria nella scheda Digramma della scheda Mappatura chiave del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica della mappatura chiave" a pagina 25.
   - Se una trasformazione Mappa secondaria è già definita, è possibile visualizzare il dialogo Mappa secondaria per riconfigurare la trasformazione, compresa la modifica del relativo codice di trasformazione, in uno dei seguenti modi:
     – Fare doppio clic nella cella corrispondente della colonna della regola di trasformazione.
     – Fare clic sull'icona della bitmap Mappa secondaria che si trova nella colonna della regola di trasformazione.

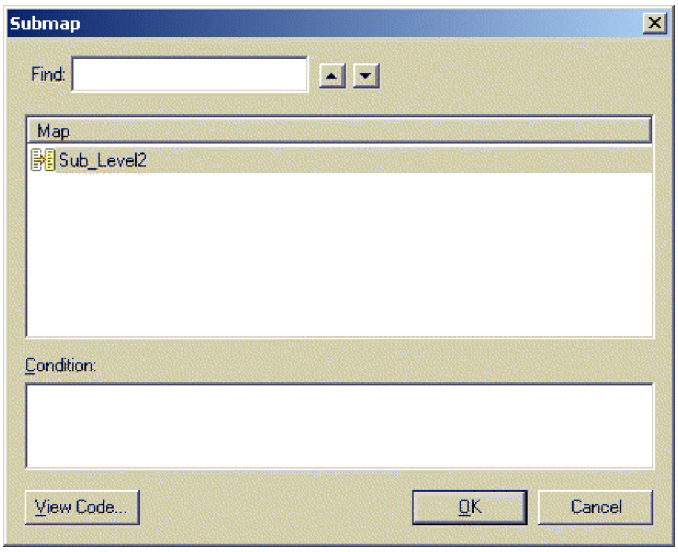


Figura 22. dialogo Mappa secondaria

4. Mediante il dialogo Mappa secondaria, specificare il nome della mappa secondaria da richiamare. Il dialogo Mappa secondaria dispone delle seguenti funzioni:
   - Per identificare una mappa secondaria da richiamare, selezionarne il nome dall'elenco nell'area Mappa. Viene visualizzato l'elenco delle mappe che corrisponde alle seguenti condizioni: la mappa secondaria dispone delle stesse definizioni dell'oggetto business per i relativi oggetti di origine e di destinazione dell'attributo di origine e di destinazione selezionati.

     Suggerimento: per ricercare una determinata mappa secondaria, immetterne il nome nel campo Trova. Le frecce Su e Giù scorrono l'elenco degli oggetti business.
   - Per specificare una condizione per la mappa secondaria, immetterla nell'area Condizione del dialogo Mappa secondaria. E' possibile immettere la condizione ora o chiudere il dialogo per immettere la condizione nel codice generato dell'attributo di destinazione.
   - Per personalizzare il codice generato, fare clic sul pulsante di opzione Visualizza codice.

     Risultato: Map Designer Express apre Activity Editor nella vista Java, contenente un esempio del codice di trasformazione in modalità di sola lettura per l'attributo di destinazione. Per effettuare le modifiche al codice di trasformazione, fare clic su Modifica codice in Activity Editor. Per ulteriori informazioni, consultare "Panoramica di Activity Editor" a pagina 109.

     Nota: Quando si salvano le modifiche in Activity Editor, vengono comunicate a Map Designer Express. Quando si salva la mappa, vengono salvate anche le modifiche.
   - Per confermare le impostazioni della trasformazione, fare clic su Ok.

Risultato: Map Designer Express genera il codice Java per richiamare la mappa secondaria specificata. Quindi, crea automaticamente un richiamo del metodo `runMap()` per richiamare la mappa secondaria.

Nota: In qualunque codice dell'attributo, è possibile utilizzare Expression Builder per inserire un richiamo di esecuzione della mappa. Per ulteriori informazioni, consultare "Utilizzo del Generatore di espressioni per il richiamo di una mappa secondaria" a pagina 211.

Promemoria: Se la regola di trasformazione è stata salvata utilizzando il *codice Java personalizzato*, è necessario gestire *manualmente* le dipendenze dell'oggetto utilizzate nel codice Java personalizzato. Ad esempio, se il codice Java personalizzato richiama una mappa secondaria, è necessario distribuire manualmente la mappa secondaria al server.

Relazioni di identità con riferimento incrociato

In alcuni casi, l'attributo di origine potrebbe essere necessario per fare riferimento a una tabella di relazioni per ricercare il valore da impostare nell'attributo di destinazione. Ciò può essere effettuato utilizzando una trasformazione del *Riferimento incrociato*.

Passi per la specifica di una trasformazione del riferimento incrociato:
Effettuare i passi di seguito riportati per specificare una trasformazione del riferimento incrociato:

1. Selezionare gli attributi di origine e di destinazione in uno dei modi descritti in precedenza per le altre trasformazioni: Entrambi devono essere oggetti business.
2. Selezionare il riferimento incrociato nella cella di trasformazione corrispondente.

   Risultato: viene visualizzato il dialogo Riferimento incrociato:

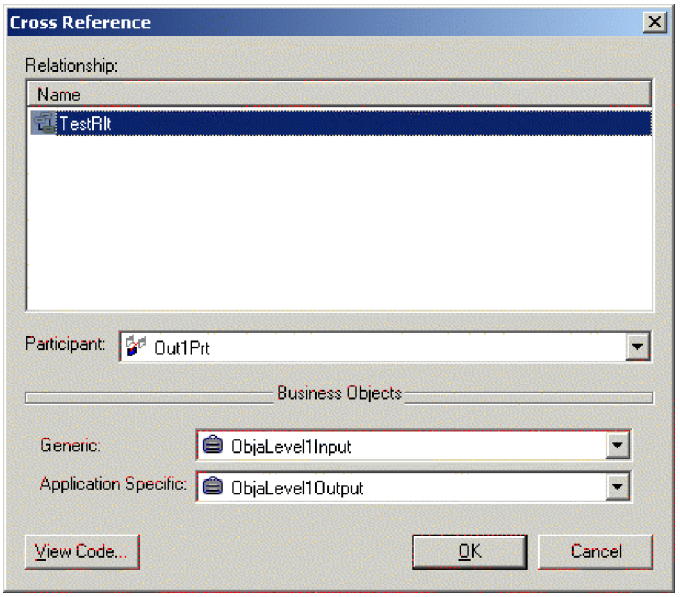


Figura 23. Dialogo Riferimento incrociato

3. In questo dialogo, selezionare il nome della relazione dall'elenco.

   Risultato: la casella combinata partecipante verrà popolata con tutti i partecipanti dalla relazione selezionata. La casella combinata Oggetto business, per impostazione predefinita, verrà popolata in base al ruolo di mappatura definito nella proprietà della mappa: E' possibile modificare le caselle combinate:

Creazione di una trasformazione personalizzata

In una trasformazione *personalizzata*, utilizzare Activity Editor per personalizzare graficamente l'attività per la trasformazione oppure immettere il codice Java per trasformare l'attributo di origine nell'attributo di destinazione.

Suggerimento: se si desidera utilizzare solo un blocco di funzione standard in una trasformazione personalizzata, è possibile configurare il blocco di funzione nel dialogo Preferenze per l'utilizzo diretto in Map Designer Express. Per ulteriori informazioni, consultare "Suggerimento: Utilizzo dei blocchi funzione direttamente in Map Designer Express" a pagina 119.

Passi per la specifica di una trasformazione personalizzata: Effettuare i passi da uno di queste schede della mappa per definire una trasformazione Personalizzata:

- Dalla scheda Tabella:
  1. Seleziona l'attributo di origine.
  2. Selezionare l'attributo di destinazione desiderato.
  3. Fare clic su Personalizza dall'elenco nella colonna Regola di trasformazione.
- Dalla scheda Diagramma:
  1. Seleziona l'attributo di origine.
  2. Selezionare l'attributo di destinazione desiderato.
  3. Trascinare l'attributo di origine sull'attributo di destinazione nella finestra oggetto business di destinazione.
- Se una trasformazione personalizzata è già definita, è possibile modificarne il codice di trasformazione in uno dei seguenti modi:
  - Fare doppio clic nella cella corrispondente della colonna della regola di trasformazione.
  - Fare clic sull'icona della bitmap Personalizza che si trova nella colonna della regola di trasformazione.

Suggerimento: è possibile personalizzare la sequenza di chiavi utilizzate per iniziare una trasformazione personalizzata dalla scheda Mappatura chiave del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica della mappatura chiave" a pagina 25.

Risultato: Map Designer Express visualizza Activity Editor con una vista grafica. Per ulteriori informazioni su Activity Editor, consultare "Panoramica di Activity Editor" a pagina 109.

Promemoria: Se la regola di trasformazione è stata salvata utilizzando il *codice Java personalizzato*, è necessario gestire *manualmente* le dipendenze dell'oggetto utilizzate nel codice Java personalizzato. Ad esempio, se il codice Java personalizzato richiama una mappa secondaria, è necessario distribuire manualmente la mappa secondaria al server.

Tabella 15 elenca le informazioni in questo manuale utili per la definizione di una trasformazione personalizzata.

Tabella 15. Definizione delle trasformazioni personalizzate

Informazioni fornite	Per ulteriori informazioni
Come utilizzare Activity Editor per personalizzare il codice di trasformazione	Capitolo 5, "Personalizzazione di una mappa", a pagina 109
Come creare le relazioni per gli attributi della relazione	Per un'introduzione generale alle relazioni, consultare Capitolo 6, "Introduzione alle relazioni", a pagina 235.

Tabella 15. Definizione delle trasformazioni personalizzate (Continua)

Informazioni fornite		Per ulteriori informazioni
1.	Utilizzare Map Designer Express per creare la mappa per gli oggetti business contenenti le relazioni.	Capitolo 2, "Creazione di mappe", a pagina 15
2.	Utilizzare Relationship Designer Express per definire la relazione.	Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249
3.	Tornare a Map Designer Express per codificare la relazione tra gli attributi.	Capitolo 8, "Implementazione delle relazioni", a pagina 271
Trasformazioni più complesse che possono essere eseguite:		"Ulteriori metodi di trasformazione attributi" a pagina 183
	Logica basata sul contenuto	"Logica basata sul contenuto" a pagina 183
	Formattazione data	"Formattazione della data" a pagina 188
	Elaborazione stringa	"Utilizzo del generatore di espressioni per le trasformazioni delle stringhe" a pagina 191

Nota: Inoltre, è possibile personalizzare una trasformazione esistente modificando il codice generato da Activity Editor. Se si modifica il codice in modalità di aggiornamento automatico, Activity Editor richiede una conferma. Se si conferma la modifica, Activity Editor salva il codice personalizzato. L'etichetta dell'icona di trasformazione nella colonna Regola di trasformazione della scheda Tabella o Diagramma da nera diventa blu in corsivo. Queste etichette di icone blu consentono di distinguere tra il codice che è in modalità di aggiornamento automatico (generato da Map Designer Express) e il codice personalizzato.

E' possibile indicare ad Activity Editor di non confermare l'impostazione nel dialogo Preferenze.

Salvataggio delle mappe

Per conservare la definizione della mappa per un utilizzo successivo, è necessario salvare la mappa. Prima che Map Designer Express salvi una mappa, viene prima convalidata la mappa. Per ulteriori informazioni, consultare "Convalida di una mappa" a pagina 88.

Map Designer Express fornisce due modi per salvare mappa corrente:

- "Salvataggio di una mappa in un progetto" a pagina 53
- "Salvataggio di una mappa in un file" a pagina 55

Importante: Affinché Map Designer Express possa salvare la mappa, quest'ultima deve essere aperta.

Salvataggio di una mappa in un progetto

Una definizione di mappa memorizza le informazioni sulla mappa in un progetto in System Manager. Questa definizione mappa contiene le seguenti informazioni per la mappa:

- Le informazioni generali sulla mappa, che comprendono le proprietà della mappa
- Il progetto della mappa, che comprende le mappature di trasformazione
- Il codice di trasformazione personalizzato

Per salvare una mappa in un progetto in System Manager, è possibile eseguire una delle azioni illustrate in Tabella 16.

Tabella 16. Salvataggio di una mappa in un progetto

Se si desidera. . .	**Quindi. . .**
Salvare la definizione mappa con il nome della mappa al momento aperta.	Effettuare una delle seguenti operazioni: • Selezionare un progetto menu secondario File > Salva. • Utilizzare la combinazione di tasti di accesso rapido `Ctrl+S`. • Nella casella Barra degli strumenti standard, fare clic sul pulsante Salva nel progetto).
Salvare la definizione mappa con un nome diverso da quello della mappa al momento aperta.	Effettuare una delle seguenti operazioni: • Selezionare un progetto menu secondario File > Salva con nome. • Utilizzare la combinazione di tasti di accesso rapido `Ctrl+Alt+S`. **Risultato:** Map Designer Express visualizza il dialogo Salva mappa con nome in cui è possibile specificare il nome della mappa.

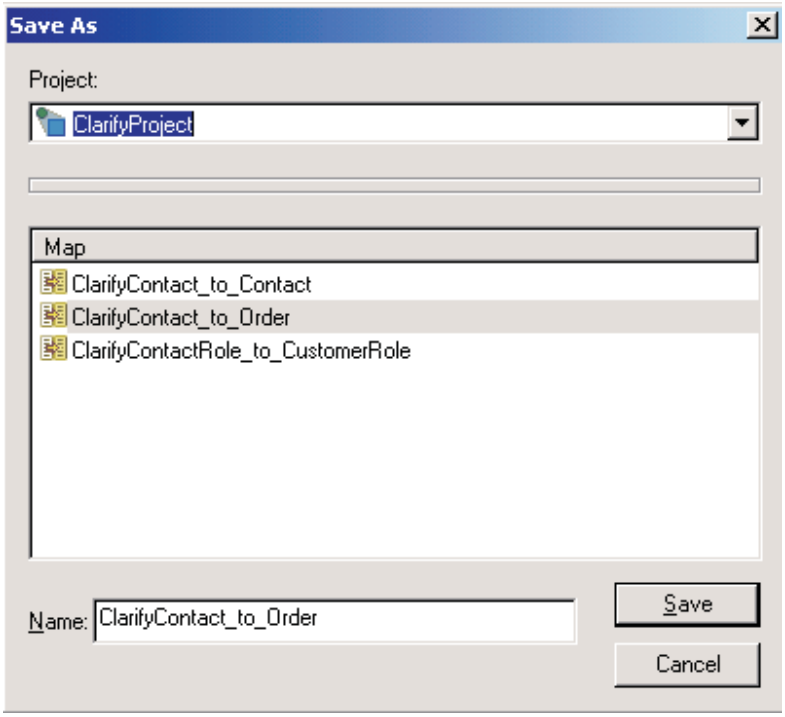


Figura 24. Dialogo Salva con nome

Quando si salva la mappa, Map Designer Express salva la definizione della mappa e il contenuto nel progetto in System Manager. Salva il contenuto della mappa come dati XML.

Nota: E' possibile specificare se Map Designer Express salva automaticamente una mappa nel progetto in System Manager prima di compilare la mappa con l'opzione `Compila mappa: salva mappa prima della compilazione`. Per impostazione predefinita, questa opzione è abilitata. E' possibile modificare

l'impostazione di questa opzione nella scheda Generale del Dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.

Suggerimenti: Per ridenominare una mappa esistente, selezionare Nel progetto dal menu secondario File > Salva con nome.

Salvataggio di una mappa in un file

Una definizione della mappa può essere memorizzata come testo nel file del sistema operativo, denominato *file di definizione mappa*. Un file di definizione mappa contiene la definizione della mappa completa, ovvero questo file utilizza il formato XML (Extended Markup Language) per rappresentare le seguenti parti di una definizione mappa:

- Le informazioni generali sulla mappa, che comprendono le proprietà della mappa
- Il contenuto della mappa, che comprende le mappature della trasformazione in un formato non compresso

Suggerimento: Map Designer Express crea un file di definizione della mappa con un'estensione `.cwm`. E' necessario seguire una convenzione di denominazione per i file di definizione mappa, come ad esempio l'utilizzo dell'estensione file (`.cwm`) per distinguerli.

Impostare una definizione mappa in Map Designer Express aprendo un file di definizione mappa esistente. Per ulteriori informazioni, consultare "Procedura per l'apertura di una mappa da un file" a pagina 61.

E' possibile salvare la mappa al momento aperta in un file di definizione mappa in uno dei modi illustrati in Tabella 17.

Tabella 17. Salvataggio di una mappa in un file di definizione

Se si desidera. . .	**Quindi. . .**
Salvare la mappa con il nome della mappa al momento aperta in formato: *MapName*`.cwm` (dove *MapName* è il nome della mappa al momento aperta) **Nota:** Map Designer Express aprirà sempre il dialogo File Salva se non si apre la mappa dal file.	Effettuare una delle seguenti operazioni: • Selezionare Nel file dal menu secondario File > Salva. • Utilizzare la combinazione di tasti di accesso rapido `Ctrl+E`. • Nella Barra degli strumenti standard, fare clic sul pulsante Salva mappa nel file (consultare Figura 24).
Salvare la mappa in un file di definizione mappa specificato. Map Designer Express visualizza una finestra di dialogo che consente di selezionare il nome del file.	Effettuare una delle seguenti operazioni: • Selezionare Nel file dal menu secondario File > Salva con nome. • Utilizzare la combinazione di tasti di accesso rapido `Ctrl+Alt+F`.

Nota: Quando si seleziona l'opzione Salva nel file dal menu File > Salva o File > Salva con nome, Map Designer Express visualizza una finestra di dialogo che consente di selezionare il nome del file. Questo nome identifica il file. Non è necessariamente il nome della mappa.

Esempio: è possibile salvare MapA in un file denominato fileA.cwm. Questo file fileA contiene la definizione della mappa per MapA. Quando Map Designer Express apre il file di definizione fileA, visualizza la definizione della mappa MapA.

Suggerimento: esportando una mappa quest'ultima viene solo copiata.

Verifica del completamento

Quando si mappano due oggetti business di grandi dimensioni, è facile che sfuggano gli attributi richiesti. E' possibile ricercare gli attributi che non sono mappati per assicurarsi di aver specificato tutte le trasformazioni desiderate. Tali attributi sono denominati *attributi non collegati*.

Eseguire i passi indicati per verificare il completamento:

- Dal Menu Modifica selezionare Trova, quindi fare clic sull'opzione attributi non collegati nel pannello di controllo Trova.

  Risultato: Map Designer Express visualizza un elenco di attributi per cui non è presente il codice di trasformazione. Per ulteriori informazioni, consultare "Ricerca di informazioni in una mappa" a pagina 77.

Nota: Una volta completato il codice, è necessario compilarlo ed eseguirne il test. Per informazioni sulla compilazione di una mappa, consultare "Compilazione di una mappa" a pagina 89. Per informazioni sul test di una mappa, consultare "Test delle mappe" a pagina 91.

Standard di mappatura

In questa sezione vengono illustrate le seguenti procedure standard per le mappe:

- "Suggerimenti per la mappatura di singoli attributi"
- "Impostazione dei commenti nel campo commenti dell'attributo" a pagina 57

Suggerimenti per la mappatura di singoli attributi

Di seguito sono riportati dei punti che forniscono un approccio generale alla mappatura di attributi singoli:

- Se la mappatura dell'attributo *non* comprende la gestione della relazione, avviata dalla copia dell'attributo di origine in quello di destinazione (consultare "Copia di un attributo di origine ad un attributo di destinazione" a pagina 42), quindi modificare il codice generato, come necessario.
- Se la mappatura dell'attributo richiede un richiamo ad un metodo nella mappatura dell'API , scrivere il codice senza copiare l'attributo.
- Se l'attributo di destinazione richiede un valore predefinito quando l'attributo di origine è null, copiare l'attributo verificando che il codice generato comprende due istruzioni if per la verifica dell'attributo di origine. E' possibile:
  - Fornire il valore predefinito in un'istruzione else per entrambe le istruzioni if.
  - Aggiungere un'altra istruzione if all'inizio del codice che verifica l'attributo di origine per null ed aggiungere un valore predefinito. Posizionare il resto del codice nell'istruzione else.

Importante: *Non* mappare l'attributo ObjectEventid. InterChange Server Express riserva ObjectEventId per i propri scopi di elaborazione. Qualunque codice personalizzato con ObjectEventId come attributo di destinazione non verrà eseguito correttamente.

Impostazione dei commenti nel campo commenti dell'attributo

I commenti dell'attributo possono ottimizzare la leggibilità della mappa. Tuttavia, Map Designer Express *non* genera automaticamente una commento per un attributo. Tabella 18 fornisce alcuni standard suggeriti per i commenti dell'attributo in base al tipo di trasformazione associata all'attributo di destinazione.

Tabella 18. Impostazioni per il commento dell'attributo

Situazione	**Impostazione per il commento dell'attributo**
Se l'oggetto business child *non* viene mappato	`=Nessuna mappatura`
Trasformazione Imposta valore	`=SET VALUE(`*`valore`*`)`
Sposta trasformazione	`=MOVE`
Aggiungi trasformazione	`=JOIN(`*`srcAttr1`*`, `*`srcAttr2`*`, ...)`
Dividi trasformazione	`=SPLIT(`*`srcAttr`*`[`*`indice`*`])`
Per gli oggetti business child, quando la mappatura viene effettuata *senza* richiamare una mappa secondaria per indicare che l'oggetto deve essere esteso per visualizzarne gli attributi	`=Mappatura qui`
Se il codice per richiamare la mappa secondaria viene generato	`=SUBMAP(`*`mapName`*`)`
Se la mappatura dell'attributo contiene i richiami della mappatura API che implementano le relazioni, come ad esempio: • `retrieveInstances()` • `retrieveParticipants()` • `maintainSimpleIdentityRelationship()` • `maintainCompositeRelationship()` • Tutti gli altri metodi nella classe `IdentityRelationship` *esclusi* `foreignKeyLookup()` e `foreignKeyXref()`	tipo `=Relationship()` dove *`tipo`* può essere: • `identità` • `ricerca` • `personalizzata`
Se la mappatura dell'attributo contiene `foreignKeyLookup()`	`=foreignKeyLookup()`
Se la mappatura dell'attributo contiene `foreignKeyXref()`	`=foreignKeyXref()`
La trasformazione personalizzata *non* è una di quelle elencate in precedenza (relazione o chiave esterna)	`=CUSTOM(`*`riepilogo`*`)`
Se il codice di un attributo non contiene nulla eccetto che l'impostazione verbo	`=SET VERB`

Capitolo 3. Utilizzo delle mappe

In questo capitolo vengono descritte alcune funzioni avanzate di Map Designer Express che si possono utilizzare dopo la creazione delle mappe.

Il capitolo descrive le seguenti attività:

Apertura e chiusura di una mappa

Map Designer Express visualizza una mappa per volta all'interno della finestra schede. Questa mappa viene definita la *mappa corrente* (a volte anche la "mappa attualmente aperta"). E' possibile controllare quale mappa sia la mappa corrente con le seguenti procedure di Map Designer Express:

Apertura di una mappa

E' necessario aprire una mappa in Map Designer Express prima di poter visualizzare le relative informazioni nella scheda Mappa o modificare queste informazioni. Quando Map Designer Express apre una mappa, se l'opzione `convalida mappa quando aperta` è abilitata, esegue prima una serie di convalide su questa mappa.

Nota: E' possibile specificare se Map Designer Express deve convalidare una mappa alla relativa apertura, con l'opzione `Apri mappa: convalida mappa quando aperta`. Per impostazione predefinita, questa opzione è abilitata.

Se quest'impostazione è abilitata quando viene aperta una mappa che utilizza oggetti business di elevate dimensioni (vale a dire, migliaia di attributi), Map Designer Express potrebbe impiegare molto tempo per aprire la mappa. Si può modificare l'impostazione di quest'opzione dalla scheda Generale della finestra Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.

Di seguito sono riportate le convalide eseguite da Map Designer Express su una mappa:

- Controlla che ciascuna definizione dell'oggetto business utilizzata dalla mappa sia definita nel progetto in System Manager.
- Controlla che tutti gli attributi nella mappa siano presenti nella definizione dell'oggetto business specificato, come definito nel progetto del System Manager.
- Controlla che il tipo di ciascun attributo nella mappa corrisponda al relativo tipo nella definizione dell'oggetto business specificato, come definito nel progetto del System Manager.
- Convalida le trasformazioni:
  - Controlla che l'ordine di esecuzione sia corretto; cioè che l'ordine di esecuzione sia univoco, positivo e consecutivo.
  - Controlla che nessun attributo abbia delle dipendenze cicliche su altri. Se vengono rilevate delle trasformazioni cicliche, Map Designer Express visualizza le regole cicliche nella finestra di output.
  - Verifica le informazioni di trasformazione:

    Trasformazione Sposta—un solo attributo sorgente è interessato.

    Trasformazione Aggiungi—più di un attributo sorgente è interessato.

    Trasformazione Dividi—un solo attributo sorgente è interessato; l'indice di divisione è maggiore di o pari a zero; il delimitatore di divisione non è vuoto.

    Trasformazione Imposta valore—nessun attributo sorgente è interessato; un valore è stato specificato.

    Trasformazione Mappa secondaria—almeno un attributo sorgente è interessato; è specificato il nome della mappa secondaria.

    Trasformazione Riferimento incrociato—un solo attributo sorgente è interessato.

Map Designer Express fornisce i seguenti metodi per aprire una mappa:

- "Procedura per l'apertura di una mappa da un progetto in System Manager" a pagina 60
- "Procedura per l'apertura di una mappa da un file" a pagina 61

Procedura per l'apertura di una mappa da un progetto in System Manager

Effettuare le seguenti operazioni per aprire una mappa da un progetto in System Manager:

1. Aprire la finestra Apri una mappa da un progetto in uno dei seguenti modi:
   - Fare clic su File > Apri > Da progetto.
   - Utilizzare l'accesso rapido della tastiera `Ctrl+O`.
   - Nella barra degli strumenti Standard, fare clic sul pulsante Apri Mappa da Progetto.

   Risultato: Map Designer Express visualizza la finestra Apri mappa.

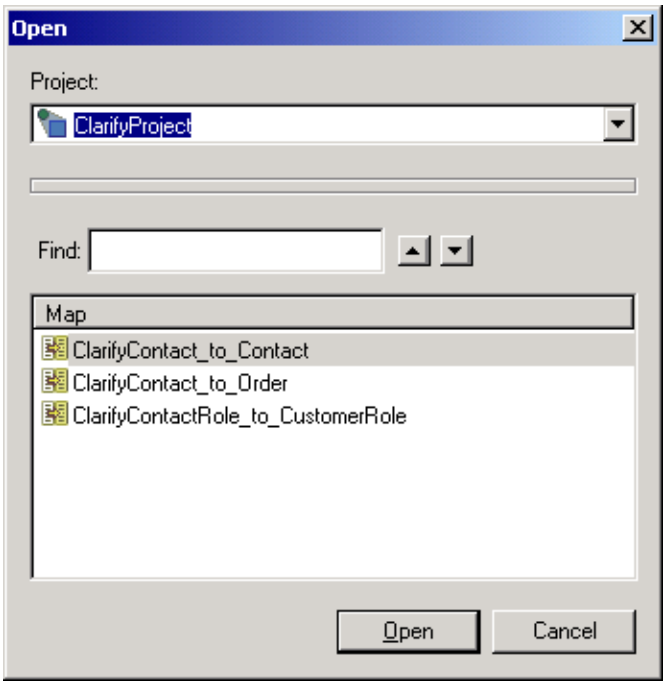


Figura 25. Apertura della finestra di dialogo Mappa

2. Selezionare il progetto.
3. Selezionare il nome della mappa dall'elenco delle mappe attualmente definite nel progetto del System Manager.

   Suggerimento: per individuare un particolare nome di mappa, immetterne il nome nel campo Trova. Le frecce rivolte verso l'alto e verso il basso scorrono l'elenco delle mappe.
4. Fare clic sul pulsante Apri per aprire la mappa dal progetto.

Procedura per l'apertura di una mappa da un file

E' possibile memorizzare una definizione di mappa in formato XML in un file del sistema operativo chiamato *file di definizione della mappa*. Per creare un file di definizione della mappa, salvare la mappa come file di progettazione della mappa (`.cwm`) in Map Designer Express. Per ulteriori informazioni, consultare "Salvataggio di una mappa in un file" a pagina 55.

L'apertura di un file di definizione della mappa viene eseguita in Map Designer Express.

Effettuare le seguenti operazioni per aprire un file di definizione della mappa:

1. Aprire la finestra Apri una mappa da un File in uno dei seguenti modi:
   - Fare clic su File > Apri > da file.
   - Utilizzare l'accesso rapido della tastiera `Ctrl+I`.
   - Nella casella barra degli strumenti Standard, fare clic sul pulsante Apri Mappa da File.

   Risultato: Viene visualizzata la finestra di dialogo Apri file con mappa.
2. Selezionare il file di definizione della mappa che si desidera aprire. Il file deve essere un file `.cwm` creato da Map Designer Express.

Risultato: Map Designer Express apre il file di definizione della mappa. Le informazioni sulla mappa vengono visualizzate nella scheda Mappa.

Importante: L'apertura della mappa in Map Designer Express *non* salva in modo automatico la mappa nel progetto. Per poterla salvare, passare all'operazione 3..

3. Salvare la mappa nel progetto in System Manager. Per ulteriori informazioni, consultare "Salvataggio di una mappa in un progetto" a pagina 53..

   Regola: è necessario salvare la mappa nel progetto in System Manager per essere compilata. Per compilare la mappa, selezionare Compila dal menu File. Per ulteriori informazioni, consultare "Test delle mappe" a pagina 91.

Chiusura di una mappa

Effettuare una delle seguenti operazioni per chiudere la mappa corrente che visualizza la scheda:

- Aprire una nuova mappa in uno dei modi descritti in "Apertura di una mappa" a pagina 59.

  Risultato: Map Designer Express chiude la mappa corrente prima di aprire una nuova.

- Dal menu File, selezionare Chiudi.

  Risultato: Map Designer Express chiude la mappa corrente e cancella la scheda. Per rendere corrente una nuova mappa, è possibile creare una nuova mappa o aprire una mappa esistente.

- Uscire da Map Designer Express in uno dei seguenti modi:
  - Dal menu File, selezionare Esci.
  - Utilizzare l'accesso rapido della tastiera `Alt+F4`.

  Risultato: Map Designer Express chiude in modo automatico la mappa corrente prima di uscire.

Nota: Se è stata modificata la mappa corrente dall'ultimo salvataggio, Map Designer Express visualizza una casella di conferma per la chiusura della mappa.

Specifica delle informazioni sulla proprietà della mappa

Utilizzare la finestra di dialogo Proprietà mappa (consultare Figura 26) per visualizzare e specificare le informazioni sulle proprietà di una mappa. Per visualizzare la finestra di dialogo Proprietà mappa, effettuare una delle seguenti operazioni:

- Dal menu Modifica selezionare Proprietà mappa.
- Utilizzare l'accesso rapido della tastiera `Ctrl+Invio`.
- Nell'area di lavoro della mappa della scheda Diagramma, fare clic con il pulsante destro del mouse e selezionare Proprietà mappa dal menu contestuale.

La finestra Proprietà mappa fornisce le seguenti schede:

- Scheda Generale
- Scheda Oggetti business

Figura 26 a pagina 63 mostra la scheda Generale della finestra Proprietà mappa.

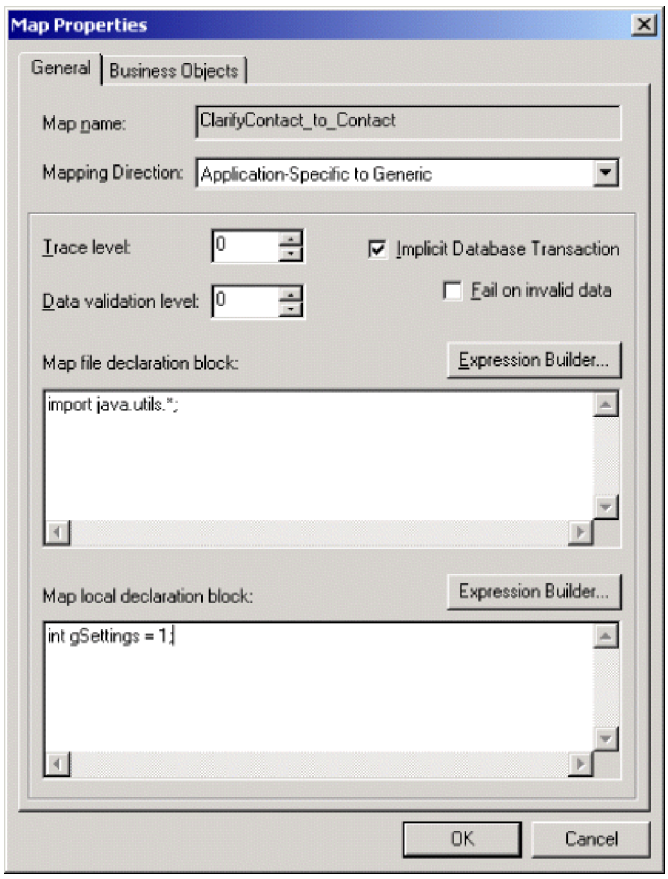


Figura 26. Scheda Generale della finestra di dialogo Proprietà mappa

Definizione delle informazioni generali sulla proprietà

La scheda Generale della finestra Proprietà mappa visualizza le informazioni generali sulle proprietà mostrate in Tabella 19.

Tabella 19. Informazioni generali sulla proprietà della mappa

Proprietà generali della mappa	Descrizione	Per ulteriori informazioni
Nome mappa	Identifica la mappa le cui proprietà vengono visualizzate nella finestra di dialogo. Questo campo viene inizializzato quando si crea una nuova mappa e non è modificabile.	N/D

Tabella 19. Informazioni generali sulla proprietà della mappa (Continua)

Proprietà generali della mappa	Descrizione	Per ulteriori informazioni
Ruolo mappatura	Identifica lo scopo della mappa. Di seguito sono riportati i possibili valori dei ruoli di mappatura: • Da specifica per applicazione a generica • Da generica a specifica per applicazione • Altri (per le mappe che non hanno una specifica direzione di mappatura ad esse associata) **Nota:** Il combo box sarà vuoto per mappe definite in precedenza che non hanno queste informazioni sulle proprietà. Ciò è possibile fino a quando non si utilizzano le regole di trasformazione della relazione (Relationship). Quando si crea inizialmente una regola di trasformazione della relazione e il relativo valore è vuoto, Map Designer Express richiederà l'immissione di questo valore.	
Proprietà in fase di runtime	Specifica le proprietà della mappa (livello della funzione di traccia, livello di convalida dei dati, transazioni database implicite ed errore su dati non validi) che si applicano all'istanza della mappa in fase di runtime. E' possibile specificare queste proprietà nella scheda Generale della finestra Proprietà mappa di Map Designer Express o nella finestra Proprietà mappa che fornisce il System Manager. Le modifiche vengono eseguite al file system locale. La distribuzione della mappa verso il server non aggiornerà l'istanza runtime. **Nota:** E' possibile aggiornare le proprietà della mappa in modo dinamico dalla vista di gestione del componente server, facendo clic con il pulsante destro del mouse su di una mappa e selezionando le proprietà dal menu contestuale. Le modifiche saranno aggiornate automaticamente nel server.	
Livello di traccia	Imposta il livello di traccia della mappa.	"Aggiunta di messaggi di traccia" a pagina 525
Livello di convalida dei dati	Consente il controllo di ogni operazione in una mappa e la registrazione di errori quando i dati nell'oggetto business in entrata non possono essere trasformati.	"Creazione di livelli personalizzati di convalida dati" a pagina 197
Transazione database implicita	Stabilisce se InterChange Server Express deve utilizzare le parentesi per racchiudere le transazioni implicite per transazioni sulle proprie connessioni.	
Errore su dati non validi	Determina l'esito di esecuzione della mappa se i dati non sono validi.	"Creazione di livelli personalizzati di convalida dati" a pagina 197
Dichiarazioni variabile	E' possibile dichiarare le proprie variabili Java da utilizzare nel codice di trasformazione. Per ulteriori informazioni, consultare "Utilizzo delle variabili" a pagina 177.	
Blocco dichiarazione file mappa	Consente di importare pacchetti Java (come `MapUtils`) in una mappa da utilizzare all'interno del codice di trasformazione.	"Importazione di pacchetti Java e di altro codice personalizzato" a pagina 172
Blocco dichiarazione locale mappa	Consente di importare codice Java personalizzato sviluppato esternamente a Map Designer Express in una mappa da utilizzare nel codice di trasformazione.	"Importazione di pacchetti Java e di altro codice personalizzato" a pagina 172

Definizione di oggetti business

La scheda Oggetti business della finestra di dialogo Proprietà mappa visualizza informazioni sugli oggetti business di una mappa. Elenca gli oggetti business

sorgenti e di destinazione così come gli oggetti business temporanei che potrebbero essere definiti. Per ulteriori informazioni, consultare “Passi per la modifica delle variabili dell'oggetto business” a pagina 179.

Progettazione delle mappe per le lingue bidirezionali

WebSphere Business Integration Server Express supporta le lingue bidirezionali. Questo supporto è in un formato standard bidirezionale di tipo Windows (logico da sinistra a destra). Con questo supporto si garantisce che i dati immessi in ambiente InterChange Server Express provenienti da uno dei connettori abilitati per lingue bidirezionali siano in formato lingua bidirezionale uniforme (`CWBF=ILYNN`). Tuttavia, è possibile inserire dati in una mappa da sorgenti diverse da connettori abilitati, ad esempio, un componente che non supporta lingue bidirezionali esportato tramite servizi Web, un adattatore che non supporta lingue bidirezionali oppure i dati importati da qualche sorgente esterna dove il supporto bidirezionale non è conosciuto. Per ulteriori informazioni, consultare “Utilizzo della funzione bidirezionale in Activity Editor” a pagina 170.

L'utilizzo di sorgenti non bidirezionali può determinare incongruenze del formato bidirezionale che causano confronti all'interno di una collaborazione e restituiscono risultati non corretti. Questi tipi di errori possono essere evitati nel modo seguente:

- Accettare gli input solo da sorgenti che rispettano lo stesso formato bidirezionale del sistema WebSphere Business Integration Server Express, come gli adattatori già abilitati per questo supporto.
- Abilitare i connettori che usano questa collaborazione per far rispettare il corretto formato bidirezionale. Per ulteriori informazioni, consultare il manuale *Collaboration Development Guide*.
- Usare le API nella classe CxBidiEngine per trasformare tutti i dati in un formato bidirezionale coerente (consultare Capitolo 12, “Classe CwBidiEngine”, a pagina 379).

Utilizzo di documenti della mappa

E' possibile creare un *documento della mappa* per osservare tutte le trasformazioni di un'unica mappa o tra due mappe. Durante la verifica di una mappa, l'utente potrebbe voler visualizzare tutte le trasformazioni della mappa in un'unica operazione, piuttosto che aprire e visualizzare ciascun attributo separatamente. Per fare ciò, si può creare un documento della mappa che contiene tutte le trasformazioni. Un documento della mappa fornisce un metodo automatizzato di documentare trasformazioni di mappe native.

Questa sezione fornisce le seguenti informazioni:

- Una descrizione dei due file HTML che compongono un documento della mappa
- Come creare un nuovo documento della mappa
- Come visualizzare un documento della mappa
- Come stampare un documento della mappa

Descrizione di un documento della mappa

Un *documento della mappa* è composto da due file HTML che descrivono tutte le trasformazioni di una mappa (o insieme di mappe):

- Un file tabella delle mappe che descrive le trasformazioni di mappa in un formato tabulare.

Il file tabella delle mappe dispone del nome *mapDoc*.HTM.

- Un file di codice Java contenente il codice delle trasformazioni di mappe.
  Il file di codice Java dispone del nome *mapDoc*JavaCode.HTM.

In entrambi questi file HTML, *mapDoc* è il nome specificato dall'utente del documento della mappa.

Il documento della mappa può includere informazioni su tutti gli attributi, solo quegli attributi che dispongono di trasformazioni di mappe oppure solo quegli attributi che non dispongono di trasformazioni di mappe (attributi privi di collegamenti). Se vengono specificati tutti gli attributi, il documento della mappa contiene anche un elenco degli attributi privi di collegamenti negli oggetti business sorgenti e di destinazione.

Nelle seguenti sezioni viene descritto il formato dei due file HTML di un documento della mappa.

Formato file della tabella delle mappe

Il file tabella delle mappe, *mapDoc*.HTM, descrive le trasformazioni di mappe in un formato tabulare:

- Se il documento della mappa descrive *un'unica mappa*, Map Designer Express crea una tabella mappe a mappa singola.
- Se il documento della mappa descrive *due mappe*, Map Designer Express crea una tabella mappe a più mappe.

Tabella mappe a singola mappa: Una *tabella mappe a singola mappa* descrive il flusso di mappatura di una singola mappa; vale a dire, descrive le trasformazioni tra un oggetto business sorgente e un oggetto business di destinazione. La tabella mappe a singola mappa ha le seguenti colonne:

- `Attributo sorgente` mostra i nomi degli attributi dell'oggetto business sorgente.
- `Regola di trasformazione` descrive il tipo di trasformazione della mappatura tra l'attributo presente nell'oggetto business sorgente (nella colonna a sinistra) e quello presente nell'oggetto business di destinazione (nella colonna a destra). Le trasformazioni elencate in questa colonna sono collegamenti di ipertesto per l'ubicazione dell'attributo nel file di codice Java *mapDoc*JavaCode.HTM per la mappa.
- `Attributo di destinazione` mostra i nomi degli attributi dell'oggetto business di destinazione.

Figura 27 mostra il file HTML contenente una tabella mappe a singola mappa.

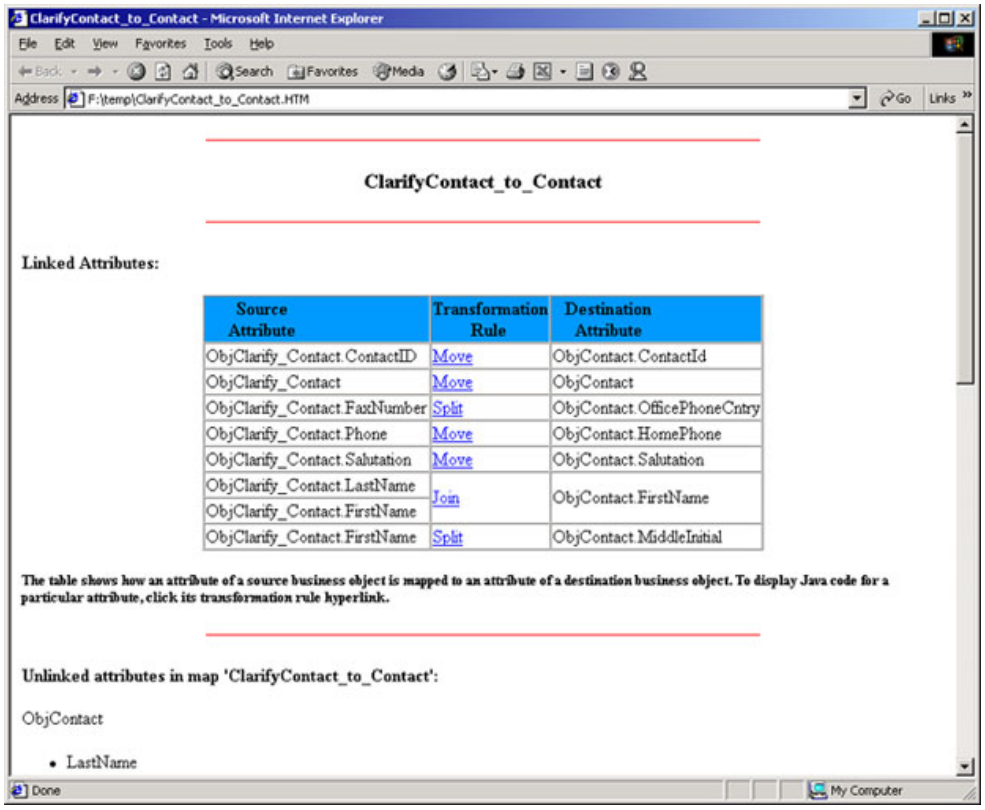


Figura 27. Tabella mappe a singola mappa

Nota: Se è stata abilitata la casella di spunta Commento della finestra Crea documento mappa, la tabella delle mappe contiene una quarta colonna chiamata `Commento` che mostra il commento per ciascuno degli attributi di destinazione nella tabella.

Tabella mappe a più mappe: Una *tabella mappe a più mappe* descrive il flusso di mappatura tra due mappe; vale a dire, descrive le trasformazioni nella mappa in entrata (tra l'oggetto business specifico dell'applicazione e quello generico) e nella mappa in uscita (tra l'oggetto business generico e quello specifico dell'applicazione). La tabella mappe a più mappe ha le seguenti colonne:

- `Attributo sorgente` mostra i nomi degli attributi dell'oggetto business specifico dell'applicazione.
- La prima colonna `Regola di trasformazione` descrive il tipo di trasformazione della mappatura tra l'attributo presente nell'oggetto business specifico dell'applicazione (nella colonna a sinistra) e quello presente nell'oggetto business generico (nella colonna a destra). Le trasformazioni elencate in questa colonna sono collegamenti di ipertesto per l'ubicazione dell'attributo nel file di codice Java *mapDoc*`JavaCode.HTM` per la mappa in entrata (da specifica per applicazione a generica).
- `Attributo comune` mostra i nomi degli attributi dell'oggetto business generico.
- La seconda colonna `Regola di trasformazione` descrive il tipo di trasformazione della mappatura tra l'attributo presente nell'oggetto business generico (nella colonna a sinistra) e quello presente nell'oggetto business specifico dell'applicazione (nella colonna a destra). Le trasformazioni elencate in questa

colonna sono collegamenti di ipertesto per l'ubicazione dell'attributo nel file di codice Java *mapDoc*`JavaCode.HTM` per la mappa in uscita (da generica a specifica dell'applicazione).

- `Attributo di destinazione` mostra i nomi degli attributi dell'oggetto business specifico dell'applicazione.

Figura 28 mostra il file HTML contenente una tabella mappe a più mappe.

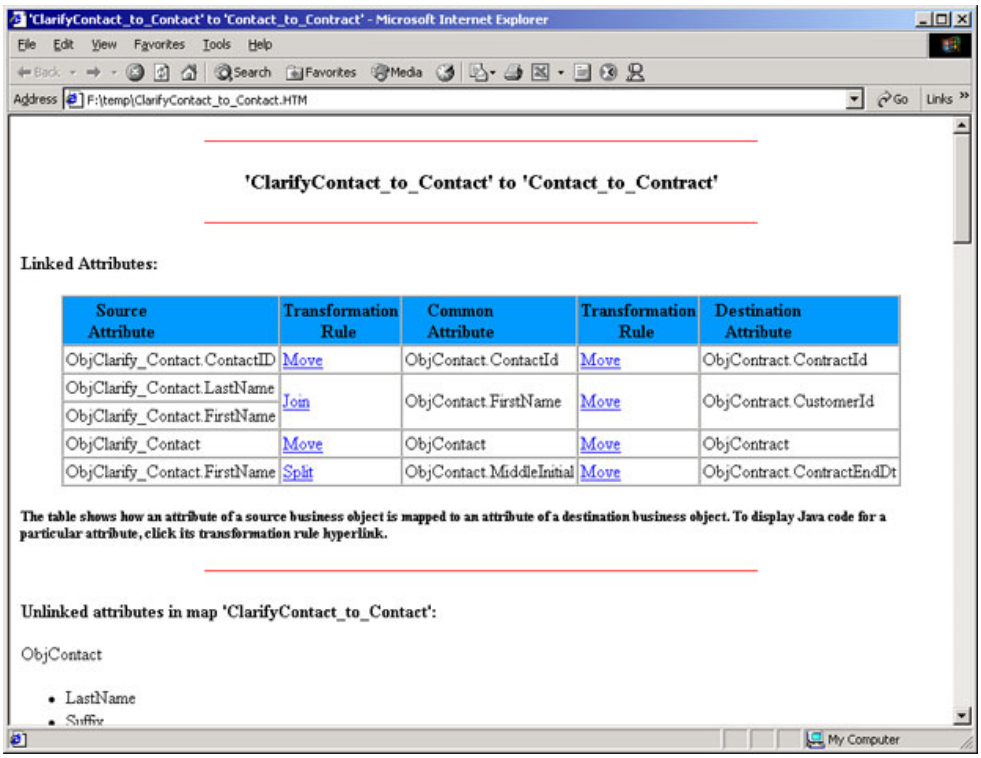


Figura 28. Tabella mappe a più mappe

Formato file di codice Java

Il file di codice Java, *mapDoc*`JavaCode.HTM`, fornisce informazioni più dettagliate sulla mappa. Contiene il codice Java che esegue le trasformazioni. Tale codice è sviluppato in un formato di programmazione standard. Il file di codice Java è utile quando si desidera visualizzare tutte le trasformazioni della mappa in un'unica operazione, piuttosto che aprire e visualizzare ciascun attributo separatamente.

Procedura per la creazione di un documento della mappa

Effettuare le seguenti operazioni per creare un documento della mappa:

1. Dal menu File, selezionare Crea documento mappa.

   Risultato: Map Designer Express visualizza la finestra Crea documento mappa (consultare Figura 29).

2. Selezionare le opzioni di configurazione del documento della mappa dalla finestra Crea documento mappa:
   - Specificare il progetto.
   - Specificare le mappe utilizzate nel documento della mappa.

     Direttiva: Se *non* si seleziona la casella di spunta "Mostra flusso mappatura con due mappe", si può selezionare una sola mappa dall'elenco a discesa.

L'elenco a discesa include tutte le mappe attualmente definite. Se una mappa è attualmente aperta, il nome appare in modo predefinito.

Se si seleziona la casella "Mostra flusso di mappatura con due mappe", viene abilitato il secondo elenco a discesa. Il secondo elenco riporta solo quelle mappe che condividono lo stesso oggetto business generico della prima mappa. Da questo elenco è possibile selezionare il nome della seconda mappa da includere nel documento della mappa.

- Specificare gli attributi nell'oggetto business di destinazione da includere nel documento della mappa.

  Selezionare il pulsante di scelta appropriato per indicare se includere tutti gli attributi, solo gli attributi mappati oppure solo quelli privi di mappatura nel documento della mappa.

- Specificare un nome per il nuovo documento della mappa.

  Direttiva: è possibile fare clic sul pulsante Sfoglia per trovare un'ubicazione per il file documento della mappa. Map Designer Express accoda in modo automatico il suffisso .HTM al nome del documento della mappa immesso. Pertanto, non è necessario specificare un'estensione file.

3. Per avviare la creazione del documento della mappa, selezionare una delle seguenti opzioni:
   - Fare clic su Salva per salvare le mappe selezionate in un documento della mappa.
   - Fare clic su Salva/Visualizza per salvare le mappe selezionate in un documento della mappa e visualizzare questo nuovo documento con un browser HTML.

Figura 29 mostra la finestra di dialogo Crea documento mappa.

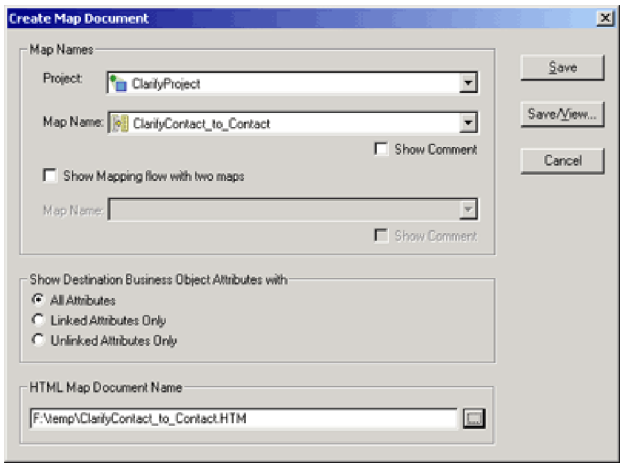


Figura 29. Finestra di dialogo Crea documento mappa

Quando si crea un documento della mappa, Map Designer Express crea il documento come file (*mapDoc*.HTM) HTML (Hypertext Markup Language) e file di codice Java associato (*mapDoc*JavaCode.HTM), dove *mapDoc* è il nome del documento mappa specificato nella finestra di configurazione del documento della mappa.

Visualizzazione di un documento della mappa

E' possibile visualizzare un documento della mappa nei seguenti modi:

- Aprire un documento mappa esistente in uno dei due modi seguenti:

– Dal menu File, selezionare Visualizza documento mappa.
– Utilizzare l'accesso rapido della tastiera `Ctrl+M`.

Risultato: La finestra Apri visualizza i file relativi al documento mappa disponibili. Specificare il documento mappa HTML da consultare e fare clic su Apri.

- Aprire un nuovo documento della mappa facendo clic su Salva/Visualizza nella finestra Configurazione documento mappa.
- Passare alla directory contenente i file relativi al documento mappa e fare doppio clic sul file prescelto.

Risultato: Map Designer Express richiama il browser per visualizzare il file documento della mappa in formato HTML che è stato scelto.

Direttiva: Inoltre, è possibile visualizzare il codice Java associato con una particolare trasformazione facendo clic sull'immissione nella colonna Azione di mappatura della tabella mappe. Il browser visualizza i segmenti di codice Java corrispondenti che implementano la mappatura tra gli attributi sorgenti e di destinazione associati.

Stampa di un documento della mappa

Effettuare le seguenti operazioni per stampare un file documento della mappa:

1. Visualizzare il file desiderato nel browser HTML.

   Per ulteriori informazioni, consultare "Visualizzazione di un documento della mappa" a pagina 69.
2. Stampare il file HTML visualizzato nel browser effettuando una delle seguenti operazioni:
   - Selezionare Stampa dal menu File del browser.
   - Utilizzare l'accesso rapido della tastiera `Ctrl+P`.
   - Fare clic sul pulsante Stampa nel barra degli strumenti Standard.

Utilizzo dell'automazione della mappa

L'automazione della mappa consente di creare in modo automatico le mappe di oggetti business con attributi simili. E' possibile anche generare mappe inverse per tutte le mappe fornite.

Questa sezione descrive le seguenti attività:

- "Creazione di mappe in modo automatico"
- "Creazione di mappe invertite in modo automatico" a pagina 74
- "Utilizzo di sinonimi per l'automazione" a pagina 76

Creazione di mappe in modo automatico

Map Designer Express può generare in modo automatico le mappe di oggetti business che hanno attributi sorgenti e di destinazione con gli stessi nomi. Nonostante gli oggetti business sono diversi, possono avere determinati elementi in comune. Ad esempio, un oggetto business cliente di solito ha gli attributi Nome, Cognome, Indirizzo e CAP per gestire i dati del cliente.

Per eseguire la mappatura di oggetti business automaticamente, Map Designer Express effettua la ricerca degli attributi con nomi corrispondenti negli oggetti business sorgenti e di destinazione e utilizza una trasformazione di spostamento. La mappatura avviene solo ai livelli corrispondenti, cioè, gli attributi di livello

superiore degli oggetti business sorgenti vengono mappati con gli attributi di livello superiore degli oggetti business di destinazione e non a qualsiasi altro livello. Analogamente, gli oggetti business secondari lato sorgente sono interessati dalla sola automazione mappe se vengono trovati oggetti secondari corrispondenti negli oggetti business di destinazione allo stesso livello.

Procedura per la creazione di mappe in modo automatico

Preliminari: è necessario disporre di un file di definizione mappe con gli oggetti business sorgenti e di destinazioni specificati. Per informazioni sulla creazione di un nuovo file di definizione della mappa con la procedura guidata Nuova mappa, consultare “Passi per la creazione della definizione della mappa” a pagina 34.

Effettuare le seguenti operazioni per creare le mappe in modo automatico:

1. Dal menu Strumenti, selezionare Mappatura automatica.

   Risultato: viene visualizzata la finestra Mappatura automatica, che consente di fornire un prefisso o suffisso al programma Map Designer Express per la ricerca degli attributi.

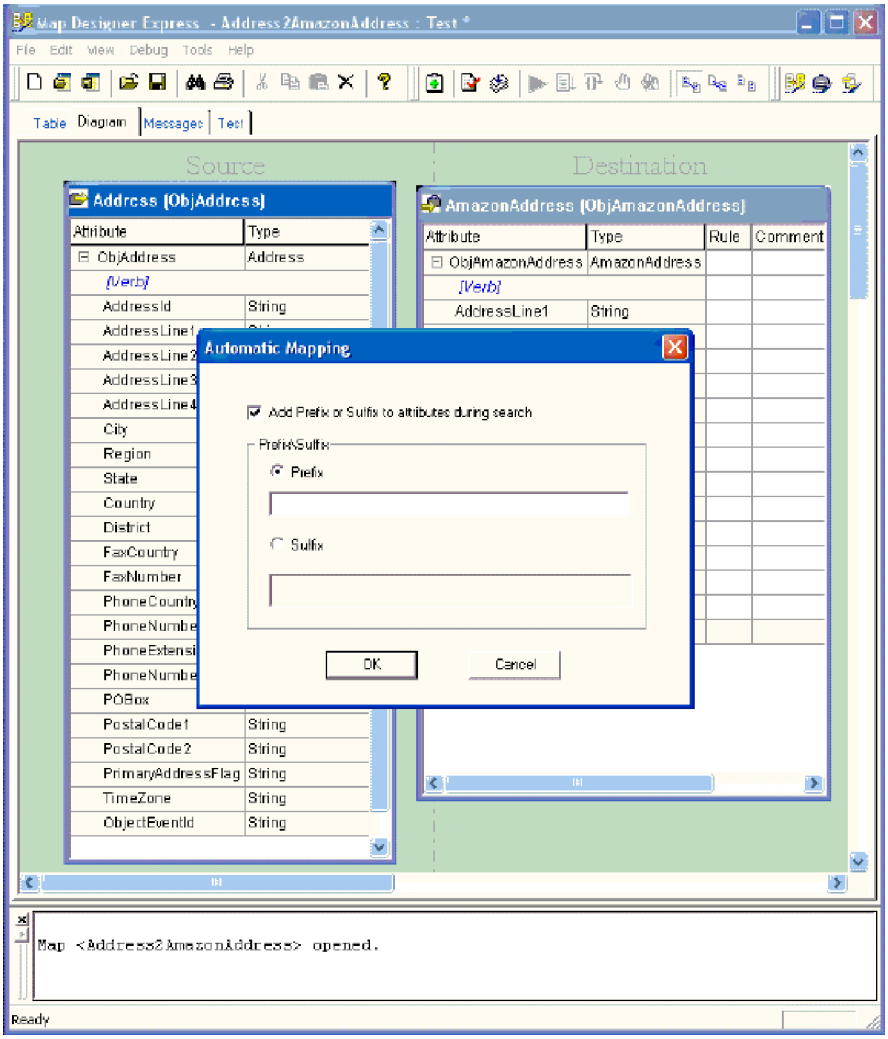


Figura 30. Finestra di impostazione del prefisso e suffisso

2. Per utilizzare quest’opzione, effettuare le seguenti operazioni nella finestra Mappatura automatica:

a. Selezionare la casella di spunta Aggiungi prefisso o suffisso agli attributi durante la ricerca.

   Nota: L'opzione è disabilitata per impostazione predefinita.

b. Selezionare Prefisso o Suffisso; nello spazio fornito, immettere un prefisso o suffisso da aggiungere alla stringa di ricerca per la particolare sessione.

   Limitazione: per ogni istanza fornita, la scelta può essere soltanto un suffisso o un prefisso. Non è possibile utilizzare contemporaneamente entrambi per la ricerca.

c. Fare clic su OK.

   Nota: Map Designer Express utilizzerà anche le preferenze impostate per i tipi di dati e i caratteri sensibili al maiuscolo/minuscolo nella scheda Mappatura automatica della finestra Preferenze.

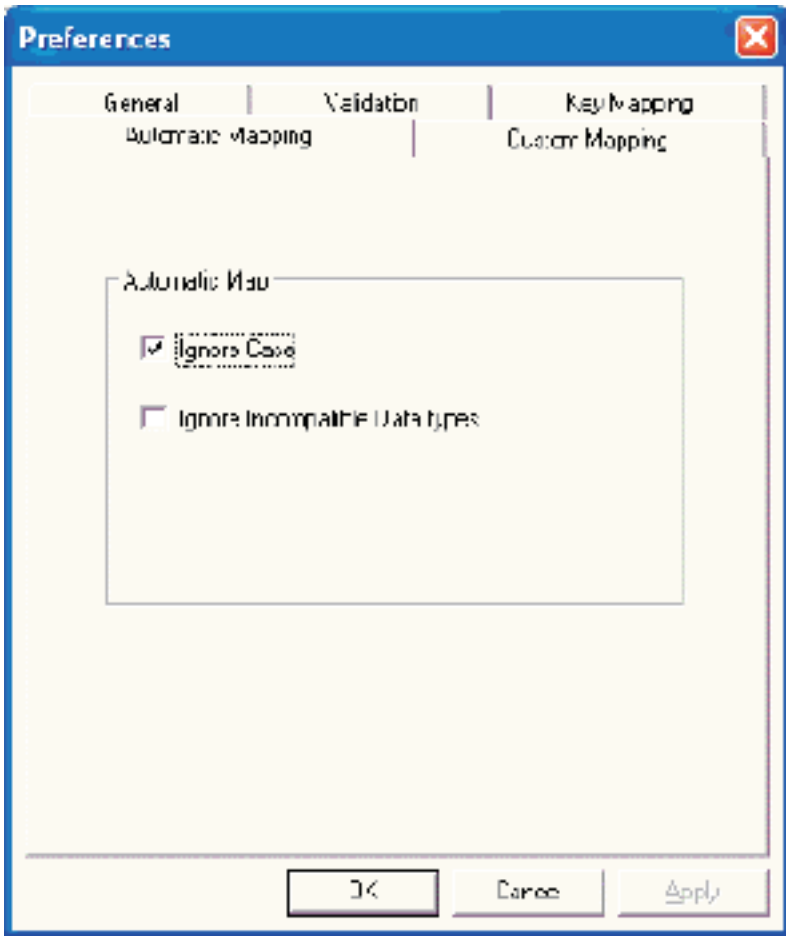


Figura 31. Scheda Mappatura automatica nella finestra Preferenze

Per informazioni sull'impostazione di queste preferenze, consultare "Specifica della mappatura automatica" a pagina 25.

Risultato: Map Designer Express effettuerà una ricerca su ogni attributo lato sorgente con il prefisso o suffisso aggiunto alla stringa di ricerca su lato destinazione. Ogni volta che viene rilevato un attributo corrispondente nell'oggetto

business di destinazione, viene avviata la mappatura automatica tra l'attributo sorgente e l'attributo di destinazione prefissato.

Esempio di mappatura automatica

La seguente descrizione della mappatura automatica include l'aggiunta di un prefisso.

Si supponga che un oggetto business sorgente abbia i seguenti attributi:

1. Nome
2. Cognome
3. Indirizzo
4. CAP

Che l'oggetto business di destinazione abbia questi attributi:

1. ORCL_Nome
2. ORCL_Cognome
3. ORCL_Indirizzo
4. Pin
5. Provincia
6. Paese

Nella finestra Mappatura automatica viene selezionata la casella Aggiungi prefisso o suffisso agli attributi durante la ricerca. Immettere `ORCL_` nello spazio Prefisso e si fa clic su OK.

Nota: Questo esempio presuppone che l'utente abbia precedentemente impostata la preferenza su Ignora maiuscole/minuscole nella Scheda Mappatura automatica della finestra Preferenze in modo da eseguire una ricerca dei nomi non sensibile al maiuscolo/minuscolo.

Risultato: Map Designer Express esegue una ricerca non sensibile al maiuscolo/minuscolo sugli attributi lato sorgente (Nome, Cognome e Indirizzo) con il prefisso `ORCL` aggiunto alla stringa di ricerca lato destinazione (ORCL_Nome, ORCL_Cognome, ORCL_Indirizzo). Ogni volta che viene individuato un attributo corrispondente nell'oggetto business di destinazione, viene avviata la mappatura automatica tra l'attributo sorgente e l'attributo di destinazione prefissato mediante una trasformazione Sposta. Nell'esempio sopra indicato, la mappatura verrà eseguita tra Nome e ORCL_Nome, Cognome e ORCL_Cognome, Indirizzo e ORCL_Indirizzo. Se non esiste corrispondenza tra gli altri attributi, la mappatura non viene eseguita.

Figura 32 illustra questo esempio.

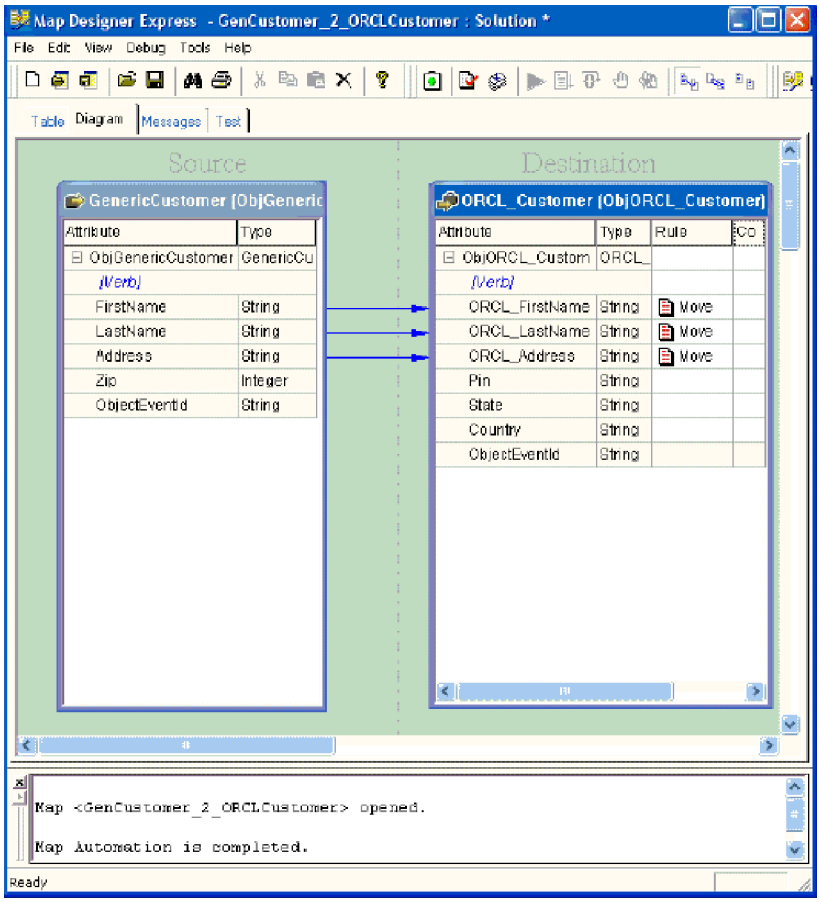


Figura 32. Esempio di aggiunta di un prefisso nella mappatura automatica

Creazione di mappe invertite in modo automatico

Generalmente, le mappe vengono utilizzate in coppie. In molti casi si richiede l'utilizzo di mappe anche nella direzione contraria. L'utilizzo della funzione Inverti mappa automatizza la procedura richiesta per creare una mappa invertita. La seguente tabella mostra le regole di trasformazione standard che Map Designer Express attualmente supporta (colonna Mappa corrente) e le regole di trasformazione che Inverti mappa attualmente include (Colonna Inverti mappa).

Tabella 20. Regole di trasformazioni utilizzate dalla mappa corrente per invertire la mappa

Mappa corrente	Mappa invertita
Sposta	Sposta
Dividi	Aggiungi
Aggiungi	Dividi
Imposta valore	Nessuna mappatura
Personalizza	Nessuna mappatura
Riferimento incrociato	Nessuna mappatura
Mappa secondaria	Mappa secondaria se esistente

Come Tabella 20 riporta, la mappatura invertita include attualmente le trasformazioni Sposta, Dividi, Aggiungi e Mappa secondaria. Le regole di trasformazione Imposta valore, Riferimento incrociato e Personalizza non vengono utilizzate durante una creazione di mappa invertita.

Limitazione: Per l'esecuzione di una mappatura invertita Aggiungi a Dividi, è necessario fornire dei delimitatori. Tuttavia, per una mappatura invertita Dividi ad Aggiungi, i delimitatori sono facoltativi.

Procedura per la creazione delle mappe invertite in modo automatico

Effettuare le seguenti operazioni per creare una mappa invertita in modo automatico.

1. Aprire la mappa di cui si richiede un'inversione.
2. Dal menu Strumenti, selezionare Inverti mappa.

   Risultato: viene visualizzata la finestra Salva con nome.
3. Immettere un nome per la mappa invertita e fare clic su Salva.

   Risultato: Map Designer Express crea una mappa invertita per la mappa attualmente aperta e la apre in una nuova istanza di Map Designer Express.

Esempio di mappatura invertita

Il seguente esempio mostra uno scenario precedente e successivo all'inversione della mappa.

Figura 33 mostra una mappa che richiede un'inversione. Utilizza le trasformazioni Sposta, Personalizza, Aggiungi, Dividi e Imposta valore.

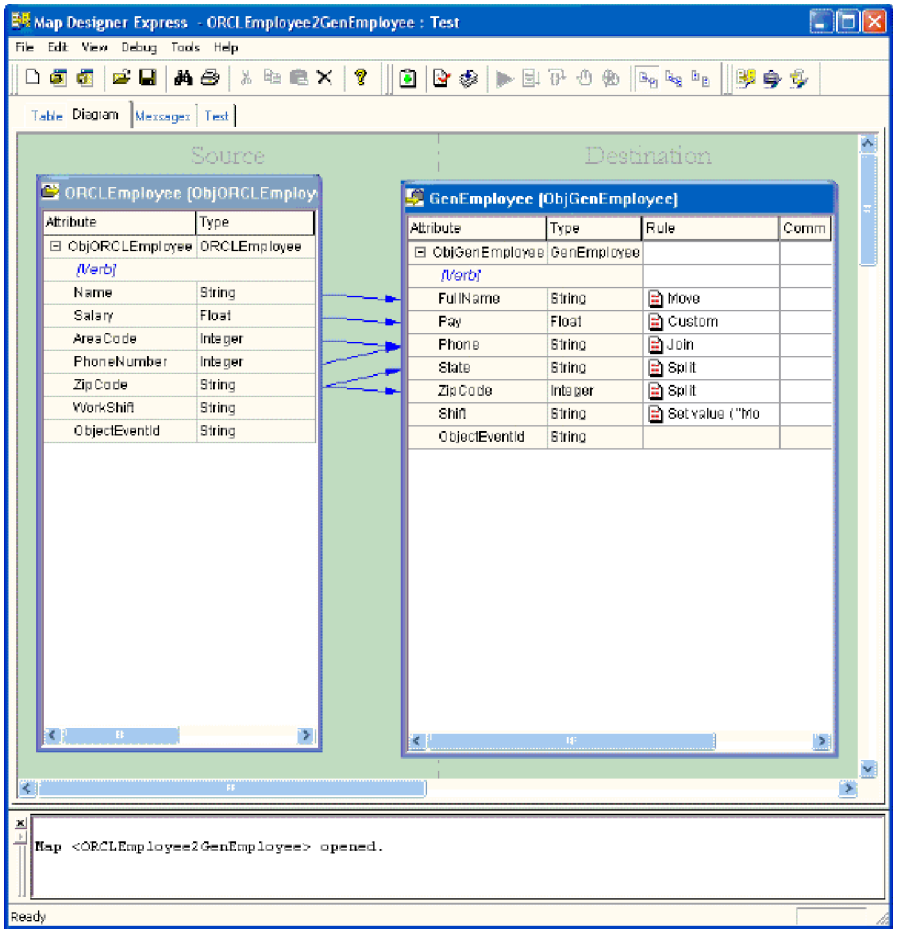


Figura 33. Mappa che richiede un'inversione

Dopo l'esecuzione della procedura per la creazione automatica di una mappa invertita (consultare "Procedura per la creazione delle mappe invertite in modo automatico" a pagina 75), viene aperta la seguente mappa.

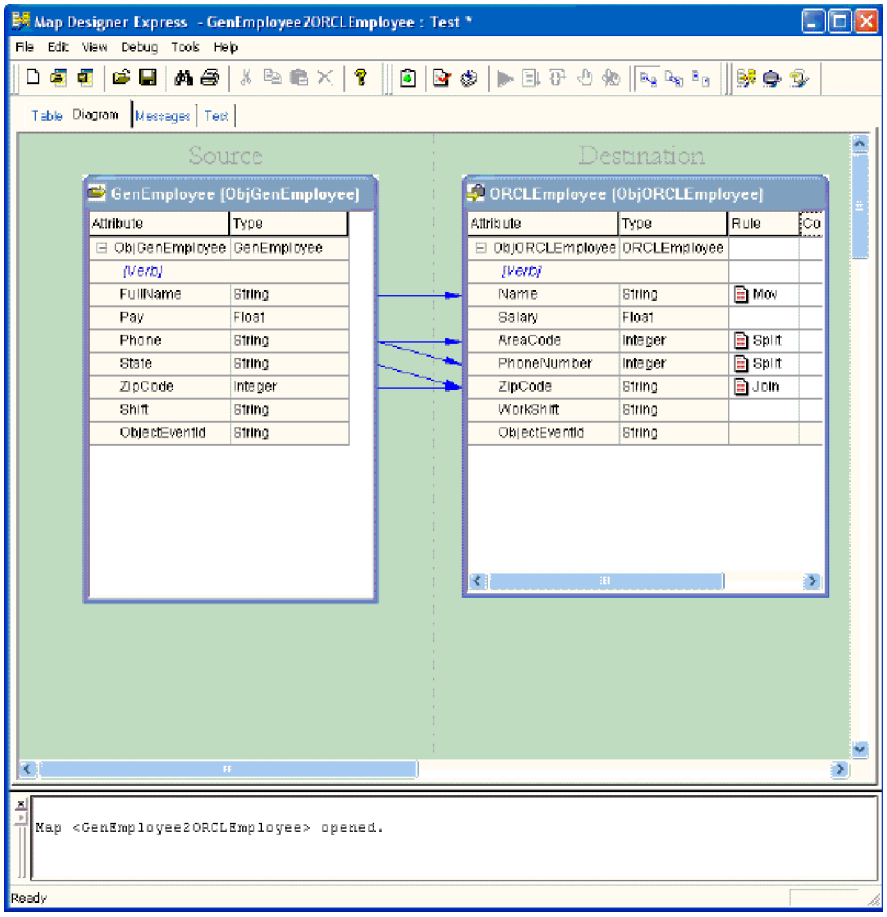


Figura 34. Mappa creata in modo automatico come risultato di un'inversione

Come mostrato in Figura 34, la trasformazione Sposta diventa una trasformazione Sposta ancora nella mappa invertita. Le trasformazioni Dividi e Aggiungi vengono invertite. Le trasformazioni Personalizza (Pay) e Imposta valore (Shift) non vanno utilizzate. Queste operazioni devono essere eseguite in modo manuale con Activity Editor. Le trasformazioni che non possono essere eseguite nella direzione invertita saranno elencate mediante avvisi nella finestra di output.

Per informazioni sull'utilizzo di Activity Editor, consultare Capitolo 5, "Personalizzazione di una mappa", a pagina 109.

Utilizzo di sinonimi per l'automazione

Per migliorare il processo di corrispondenza base, è possibile creare più sinonimi per un nome attributo. Ad esempio, è possibile associare un nome attributo non solo ad un nome corrispondente ma anche a diversi nomi equivalenti possibili.

Esempio: si supponga di avere *CR* come nome attributo sul lato sorgente. Potrebbe essere associato ai seguenti nomi di attributo sul lato destinazione:

- Difetto
- Modifica richiesta

- Numero errore
- Numero difetto
- CR

Questi sinonimi vengono aggiunti a livello progetto nella finestra Sinonimi di System Manager. E' possibile modificare le immissioni in questa finestra e aggiungere ulteriori stringhe separate da virgole per consentire l'automazione delle mappe. Si possono anche creare dei sinonimi *globali* che si applicano a tutti gli oggetti business presenti nel progetto.

Per informazioni sulla procedura di creazione dei sinonimi per l'automazione delle mappe in System Manager, consultare il manuale *System Implementation Guide*.

System Manager effettua una ricerca per tutti i sinonimi di un dato attributo ed esegue la mappatura automatica quando trova le corrispondenze. Ad esempio, *CR* dal lato sorgente corrisponde a *Difetto*, *Modifica richiesta*, *Numero errore* e *CR* se sono stati aggiunti come sinonimi nella finestra Sinonimi. La mappatura viene eseguita in modo automatico quando viene rilevata una di queste parole.

Ricerca di informazioni in una mappa

E' possibile utilizzare la funzione di ricerca di Map Designer Express per effettuare le seguenti ricerche:

- Ricercare il testo in un nome attributo o nel codice di trasformazione dell'attributo.
- Ricercare attributi non collegati.

Procedura per la ricerca di informazioni in una mappa

Effettuare le seguenti operazioni per ricercare informazioni in una mappa.

1. Avviare la ricerca in uno dei seguenti modi:
   - Dal menu Modifica selezionare Trova.
   - Utilizzare l'accesso rapido della tastiera `Ctrl+F`.
   - Nella barra degli strumenti Standard, fare clic sul pulsante Trova.

   Risultato: Map Designer Express visualizza il pannello di controllo Trova (consultare Figura 35).

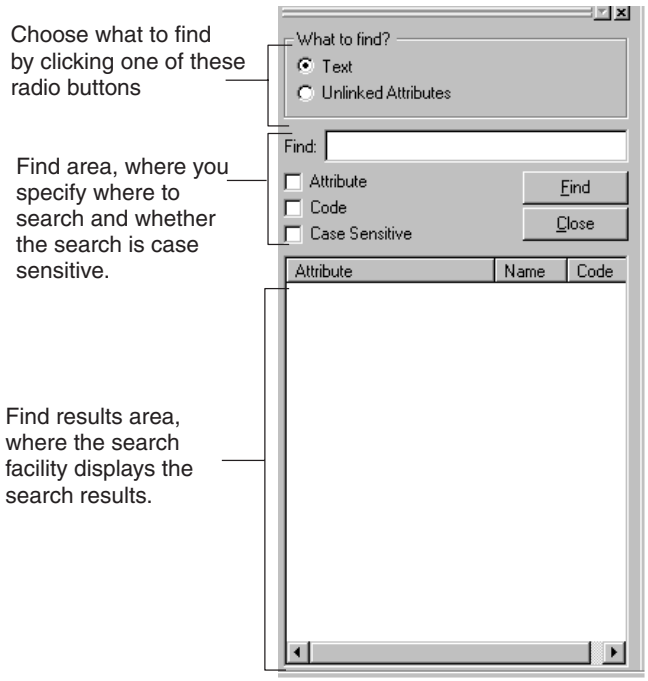


Figura 35. Pannello di controllo Trova

2. Nel pannello di controllo Trova, selezionare uno dei pulsanti di scelta dell'area Cosa cercare? per indicare quale tipo di ricerca si desidera eseguire:
   - Per ricercare il testo, effettuare quanto segue:
     a. Selezionare il pulsante di scelta Testo.
     b. Immettere il testo da ricercare nel campo Trova. E' possibile immettere più parole e spazi se necessario.
     c. Indicare l'ubicazione di ricerca del testo selezionando una o più opzioni nella area Trova:

        Attributo ricerca i nomi attributo per il testo specificato.

        Codice ricerca il codice di trasformazione degli attributi per il testo specificato. E' possibile selezionare Attributo o Codice oppure entrambe queste opzioni.

        L'opzione Maiuscole/minuscole rende sensibile al maiuscolo/minuscolo la ricerca del testo. Per trovare solo le istanze del testo che hanno le stesse lettere maiuscole/minuscole immesse, selezionare Maiuscole/minuscole.

        Limitazione: non è possibile eseguire la ricerca su commenti o tipi di dati.
     d. Fare clic su Trova per avviare la ricerca.
   - Per ricercare gli attributi non collegati, effettuare quanto segue:
     a. Selezionare il pulsante di scelta Attributi non collegati. Il pannello di controllo Trova disabilita i campi dell'area Trova.
     b. Fare clic su Trova per avviare la ricerca.

     Risultato: Map Designer Express visualizza i risultati della ricerca nella area Trova risultati. Fare clic su un qualsiasi nome attributo per selezionare automaticamente quell'attributo nella mappa.
3. Fare clic su Chiudi per chiudere il pannello di controllo Trova.

Ricerca e sostituzione del testo

Mediante l'utilizzo della funzione Trova e Sostituisci di Map Designer Express, è possibile ricercare il testo specificato nel codice Java personalizzato o nei commenti di una regola di trasformazione (o in entrambi) e sostituirlo con altro testo specificato.

Procedura per la ricerca e sostituzione del testo

Effettuare le seguenti operazioni per ricercare e sostituire il testo.

1. Avviare una ricerca e sostituzione in uno dei seguenti modi:
   - Dal menu Modifica, selezionare Sostituisci.
   - Utilizzare l'accesso rapido della tastiera `Ctrl+H`.

   Risultato: Map Designer Express visualizza la finestra Sostituisci.

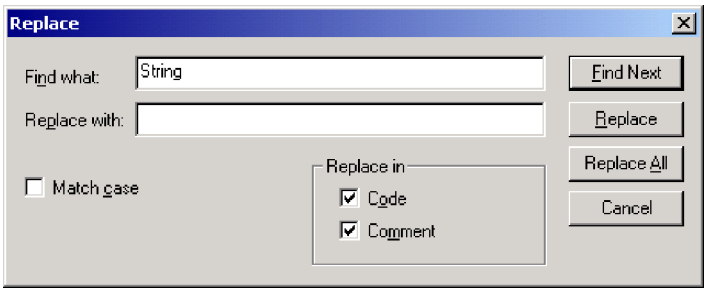


Figura 36. Finestra Sostituisci

2. Nella finestra Sostituisci, immettere il testo da ricercare nel campo Trova e il testo da sostituire nel campo Sostituisci. Se necessario, selezionare Maiuscole/minuscole.
3. Indicare l'ubicazione di sostituzione selezionando Codice, Commento oppure entrambi.
4. Fare clic su Trova successivo per avviare la ricerca.

   Risultato: è possibile che si verifichi una delle seguenti condizioni.
   - Se è stata specificata l'opzione Sostituisci in codice, quando viene trovato il testo nel codice Java personalizzato di una regola di trasformazione, l'applicazione Activity Editor viene visualizzata con il codice Java personalizzato in Modalità vista rapida.
   - Se è stata specificata l'opzione Sostituisci in commento, la vista Tabella viene attivata e il testo visualizzato nella colonna dei commenti della vista Tabella.
5. Fare clic su Sostituisci per sostituire la corrispondenza con il nuovo testo.

   Direttiva: è possibile sostituire tutte le corrispondenze uguali con un'unica operazione facendo clic su Sostituisci tutto.
6. Per continuare la ricerca e la sostituzione del testo specificato, istanza per istanza, ripetere i passi 4 e 5.

Stampa di una mappa

Map Designer Express consente la stampa di una mappa. Crea una rappresentazione tabellare della mappa, simile alla mappa che appare nella scheda Tabella. E' possibile stampare una mappa in uno dei seguenti modi:

- Dal menu File, selezionare Stampa per stampare la mappa corrente.

- Utilizzare l'accesso rapido della tastiera `Ctrl+P`.
- Nella barra degli strumenti Standard, fare clic sul pulsante Stampa.

Map Designer Express supporta anche le seguenti attività di stampa standard:

- Anteprima di stampa—selezionare Anteprima di stampa dal menu File per visualizzare la struttura della pagina per la mappa corrente.
- Impostazione stampante
  - Dal menu File selezionare Imposta stampante per visualizzare la finestra Imposta Stampante in cui è possibile configurare le opzioni come impostazione stampante, dimensione carta e orientamento.
  - Utilizzare l'accesso rapido della tastiera `Ctrl+Shift+P`.

Direttiva: quando Map Designer Express esegue la stampa o l'anteprima di stampa, esegue la copia della tabella di trasformazione attributi nella scheda Tabella. Prima di effettuare la stampa, è possibile regolare l'ampiezza delle singole colonne e l'altezza delle singole righe nella tabella di trasformazione attributi in modo da inserire l'intera mappa in una sola pagina oppure per personalizzare il risultato di stampa.

Eliminazione di oggetti

Questa sezione fornisce informazioni su come eliminare i seguenti oggetti:

- "Procedura per l'eliminazione delle fasi di trasformazione della mappa"
- "Procedura per l'eliminazione degli oggetti business" a pagina 81
- "Procedura per l'eliminazione delle mappe" a pagina 81

Procedura per l'eliminazione delle fasi di trasformazione della mappa

L'eliminazione di una fase di trasformazione della mappa è suddivisa in tre parti:

- Eliminazione del codice di trasformazione
- Eliminazione del commento
- Eliminazione della freccia del flusso di dati

Effettuare le seguenti operazioni per eliminare la procedura di trasformazione da una di queste schede mappe.

- Dalla scheda Tabella: selezionare la riga dell'attributo da eliminare facendo clic sulla colonna più a sinistra (la colonna alla sinistra dell'ordine di esecuzione) ed effettuando una delle seguenti operazioni:
  - Fare clic con il pulsante destro del mouse e selezionare Elimina riga dal menu di contesto.
  - Dal menu Modifica, selezionare Elimina selezione corrente.
  - Utilizzare l'accesso rapido della tastiera `Canc`.

  Risultato: Map Designer Express elimina automaticamente tutte le trasformazioni non complete quando si salva la mappa.
- Dalla scheda Diagramma: selezionare la freccia relativa al flusso di dati ed effettuare una delle seguenti operazioni:
  - Dal menu Modifica, selezionare Elimina selezione corrente.
  - Utilizzare l'accesso rapido della tastiera `Canc`.
  - Fare clic con il pulsante destro del mouse e selezionare Elimina dal menu di contesto dell'area di lavoro della mappa.

Risultato: viene visualizzata una finestra che richiede di eliminare la freccia del flusso di dati. Fare clic su Sì e Map Designer Express visualizza una seconda finestra di conferma con richiesta di eliminare il codice associato. Fare clic su Sì e tutte le tre voci vengono eliminate.

Procedura per l'eliminazione degli oggetti business

Effettuare le seguenti operazioni per eliminare un oggetto business da una mappa.

1. Visualizzare la finestra Elimina oggetto business in uno dei seguenti modi:
   - Dal menu Modifica, selezionare Elimina oggetto business.
   - Dalla scheda Tabella, effettuare una delle seguenti operazioni:
     - Fare clic con il pulsante destro del mouse nell'area vuota del pannello degli oggetti business e selezionare Elimina oggetto business dal menu di contesto.
     - Fare clic con il pulsante destro del mouse sull'oggetto business nel pannello degli oggetti business (fare clic sul nome nella cella) e selezionare Elimina <*BusObjName*> (dove *BusObjName* è il nome dell'oggetto business selezionato).

   Risultato: viene visualizzata la finestra Elimina oggetto business.

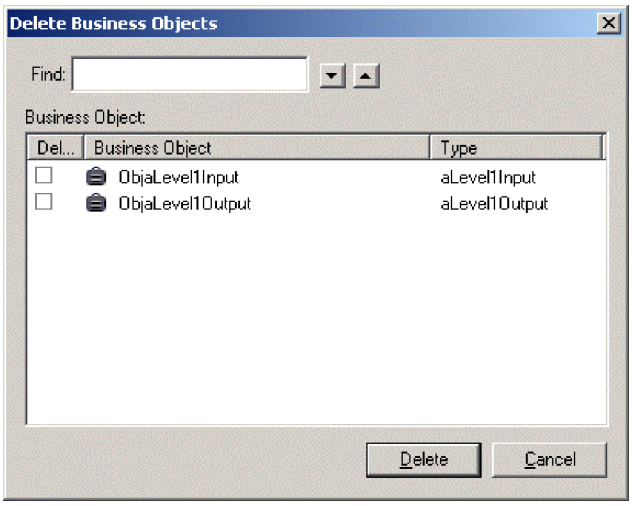


Figura 37. Finestra Elimina oggetto business

2. Utilizzando la finestra Elimina oggetto business è possibile specificare gli oggetti business che si intendono eliminare dalla mappa. La finestra Elimina oggetto business fornisce le seguenti funzioni:
   - Per eliminare un oggetto business, effettuare quanto segue:
     - Selezionare l'oggetto business dal relativo elenco.
     - Fare clic sul pulsante Elimina.
   - Per individuare un particolare oggetto business, immettere il relativo nome nel campo Trova. Le frecce rivolte verso l'alto e verso il basso scorrono l'elenco degli oggetti business.
   - Per chiudere la finestra, fare clic su Eseguito.

Procedura per l'eliminazione delle mappe

Effettuare le seguenti operazioni per eliminare una mappa da un progetto in System Manager:

1. Dal menu File, selezionare Elimina.

   Risultato: Map Designer Express visualizza la finestra Elimina mappa come Figura 38 illustra.

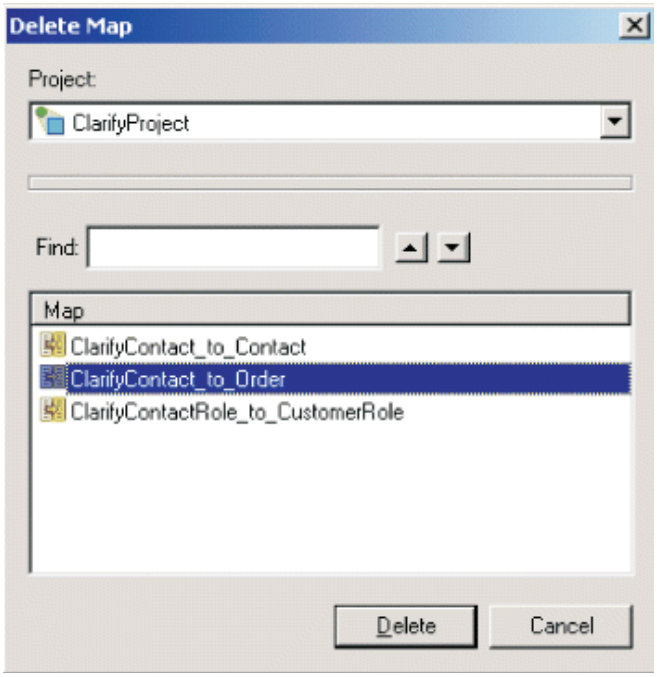


Figura 38. Finestra Elimina mappa

   Nota: Se una mappa è attualmente aperta, Map Designer Express la chiude prima di visualizzare la finestra Elimina mappa. E' possibile specificare se Map Designer Express deve chiudere una mappa attualmente aperta con l'opzione `Elimina mappa: chiudi mappa prima di eliminare`. Per impostazione predefinita, questa opzione è abilitata. Se l'opzione è disabilitata, Map Designer Express visualizza una richiesta di conferma se si seleziona la mappa attualmente aperta dalla finestra Elimina mappa. Si può modificare l'impostazione di quest'opzione dalla scheda Generale della finestra Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.

2. Immettere il nome del progetto.
3. Selezionare una o più mappe che si desidera eliminare.

   Dalla finestra Elimina mappa, è possibile:

   - Selezionare una singola mappa facendo clic sul nome della mappa nell'elenco.
   - Selezionare più mappe tenendo premuto il tasto `Ctrl` o `Maiusc` e facendo clic sui nomi delle mappe.
   - Individuare un particolare oggetto business immettendone il nome nel campo Trova. Le frecce rivolte verso l'alto e verso il basso scorrono l'elenco degli oggetti business.

4. Fare clic sul pulsante Elimina per eliminare le mappe.

   Risultato: Map Designer Express visualizza una casella di conferma per l'eliminazione.

Nota: E' possibile specificare se Map Designer Express deve confermare l'eliminazione di una mappa con l'opzione `Elimina mappa: visualizza sempre i messaggi di avviso`. Per impostazione predefinita, questa opzione è abilitata. Si può modificare l'impostazione di quest'opzione dalla scheda Generale della finestra Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.

Utilizzo dell'ordine di esecuzione

Per impostazione predefinita, l'esecuzione della mappa avviene secondo l'ordine in cui vengono visualizzati gli attributi di destinazione nella scheda Tabella. Vengono eseguiti solo gli attributi di destinazione che hanno trasformazioni. Spesso, l'ordine di esecuzione è l'ordine in cui vengono definiti gli attributi di destinazione nell'oggetto business di destinazione. Figura 39 mostra un ordine di esecuzione della mappa A-a-B in cui gli attributi di destinazione sono eseguiti nell'ordine in cui sono stati definiti.

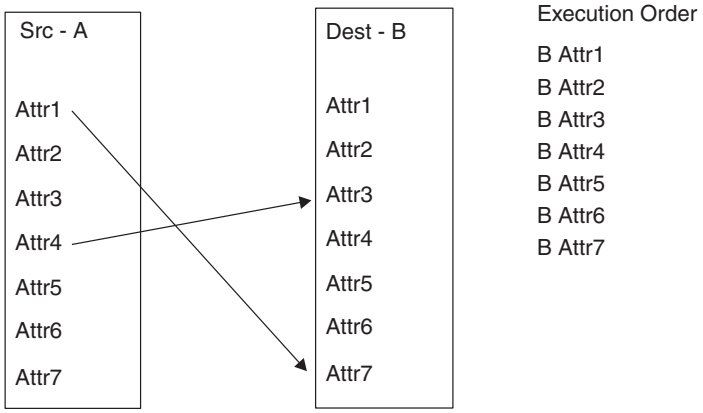


Figura 39. Ordine di esecuzione predefinito

Nota: Figura 39 suppone che tutti gli attributi di destinazione abbiano un codice di trasformazione.

Tuttavia, alcuni attributi potrebbero avere delle dipendenze nell'ordine di esecuzione. Per essere certi che il codice di trasformazione di determinati attributi sia eseguito prima del codice di trasformazioni di altri, è possibile specificare l'ordine della loro esecuzione. Si può modificare l'ordine di esecuzione per specificare il flusso di dati. Ad esempio, si supponga che nella mappa A-a-B `Attr7` debba essere eseguito immediatamente dopo `Attr3` (in altre parole, `Attr7` deve essere eseguito prima di `Attr4`). Figura 40 mostra come una specifica di sequenza nell'oggetto business di destinazione modifica la sequenza.

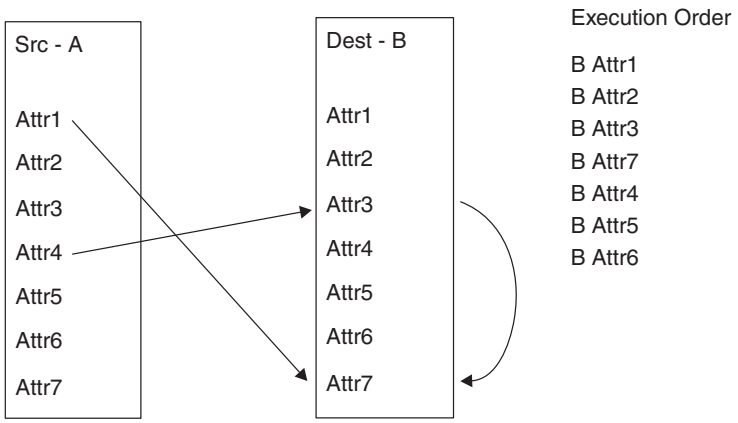


Figura 40. Modifica dell'ordine di esecuzione

E' possibile specificare una sequenza di esecuzione esplicita che sostituisca l'ordine predefinito nella scheda Tabella di Map Designer Express. Per specificare la sequenza delle trasformazioni tra due attributi di destinazione nella scheda Tabella, fare clic su Exec. Ordinare il campo dell'attributo di destinazione di cui si desidera modificare l'ordine di esecuzione e immettere il valore desiderato dell'ordine di esecuzione.

Nota: E' possibile specificare se Map Designer Express deve eseguire la rinumerazione dell'ordine di esecuzione per tutti gli attributi interessati da questa modifica con l'opzione `Definizione mappa: regola automaticamente l'ordine di esecuzione`. Per impostazione predefinita, questa opzione è disabilitata. Quando l'opzione è abilitata, Map Designer Express regola in modo automatico l'ordine di esecuzione degli altri attributi. Si può modificare l'impostazione di quest'opzione dalla scheda Generale della finestra Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.

Per impostazione predefinita, la scheda Tabella visualizza gli attributi nell'ordine in cui sono state definite le relative trasformazioni. Si può quindi scegliere di visualizzare questi attributi mappati per ordine di esecuzione, per nome degli attributi oppure ordinati in base ad una qualsiasi altra colonna della tabella di trasformazione attributi. Fare semplicemente clic sull'intestazione della colonna per ordinare gli attributi secondo quel valore di colonna.

Importante: Se si fa clic sull'intestazione riga della trasformazione e si trascina la trasformazione su una nuova posizione, viene modificato l'ordine in cui viene visualizzata la regola di trasformazione. Tuttavia, quest'operazione *non* influisce sull'ordine di esecuzione.

Creazione di mappe polimorfiche

La mappatura polimorfica consente di mappare un unico oggetto business sorgente a uno di tanti potenziali oggetti business di destinazione. Per effettuare questo tipo di mappatura, è necessario:

1. Creare una mappa separata (un oggetto sorgente e un oggetto di destinazione) per ogni possibile esito.
2. Creare una mappa polimorfica che abbia un unico oggetto business sorgente e molteplici oggetti di destinazione.

3. All'interno del primo attributo di ciascun oggetto business di destinazione, controllare alcune condizioni che indichino l'oggetto business di destinazione da riempire. Se la condizione è vera (true), eseguire la mappa appropriata per ottenere i risultati desiderati utilizzando il metodo `runMap()`.

Esempio: di seguito è riportato un codice di esempio del primo attributo in uno degli oggetti business di destinazione in una mappa polimorfica principale. In questo esempio, `ObjInput` è la variabile istanza per l'oggetto business sorgente, `ObjOutput1` è la variabile istanza per l'oggetto di output contenente questo codice e `InputToOutput1` è la mappa secondaria che esegue la mappatura reale da `ObjInput` a `ObjOutput1`. In questo caso, la condizione che impone l'esecuzione di questa mappatura o meno è basata sul valore dell'attributo `Attr1` all'interno dell'oggetto business sorgente. La condizione utilizzata ovviamente varierà.

```
BusObj[] rSrcBO = new BusObj[1];
BusObj[] rDstBO = new BusObj[1];

rSrcBO[0] = ObjInput;
String Attr1Val = ObjInput.getString("Attr1");

if (Attr1Val.equals("Poly1"))
   {
   try
      {
      rDstBO = DtpMapService.runMap("InputToOutput1",
         DtpMapService.CWMAPTYPE,rSrcBO,cwExecCtx);

      ObjOutput1.setContent(rDstBO[0]);
      }

   catch (MapFailureException e)
      {
      e.toString();
      e.printStackTrace();
      raiseException(e);
      }

   catch (MapNotFoundException e)
      {
         raiseException("MapNotFoundException",
            "runMap did not find map");
      }

   catch (Exception e)
      {
      e.printStackTrace();
      }
   }
```

Importazione ed esportazione di mappe da InterChange Server Express

Con il programma di utilità `repos_copy`, è possibile caricare e scaricare le definizioni di mappe specificate nell'archivio con l'opzione `-e`. Un *file archivio della mappa* è il file che il programma di utilità `repos_copy` crea quando estrae una definizione di mappa dall'archivio in un file `.jar`. Questo file contiene una definizione di mappa in un formato `.jar` definito da InterChange Server Express.

Consiglio: utilizzare l'estensione file `.jar` per il file archivio della mappa.

Esempio: Il seguente comando repos_copy scarica (esporta) la definizione di mappa ClCwCustomer (ClarifyBusOrg a Customer generico) dall'archivio di un InterChange Server Express denominato WebSphereICS in un file archivio della mappa:

```
repos_copy -eMap:ClCwCustomer+BusObj:Customer+BusObj:Clarify_Customer
-oNM_ClCwCustomer.jar -sWebSphereICS -pnull -uadmin
```

E' possibile creare un file archivio contenente tutti i file di definizione delle mappe, includendo:

- Le definizioni principali delle mappe
- Le definizioni di mappe secondarie
- I file per *entrambe* le direzioni, se applicabile.

Esempio: per copiare tutte le definizioni di mappe associate per la mappatura ClarifyBusOrg/Customer in un file archivio della mappa, utilizzare il seguente comando repos_copy:

```
repos_copy -eMap:ClCwCustomer+Map:CwClCustomer
-oNM_ClCwCustomer_and_CwClCustomer.jar -sWebSphereICS -pnull -uadmin
```

Se si riutilizza una mappa secondaria in diverse mappe, creare un file repos_copy separato per tale mappa anziché inserirla nel file di testo principalee.

Si può anche utilizzare il comando repos_copy per caricare (importare) una definizione di mappa nell'archivio da un file archivio della mappa.

Esempio: Il seguente comando repos_copy carica la definizione di mappa ClCwCustomer nell'archivio di un InterChange Server Express denominato WebSphereICS:

```
repos_copy -iNM_ClCwCustomer.jar -sWebSphereICS -uadmin -pnull
```

Tale comando repos_copy suppone che le definizioni di mappe ClCwCustomer e CwClCustomer *non* siano attualmente presenti nell'archivio. In caso contrario, questo comando non riesce a caricare queste nuove definizioni di mappe. E' possibile utilizzare una delle opzioni -a del comando repos_copy per scegliere come gestire le definizioni di mappa duplicate:

-ai	Ignora le definizioni di mappa duplicate durante il caricamento
-ar	Sovrascrive eventuali definizioni di mappa duplicate con la definizione di mappa presente nel file archivio della mappa.
-arp	Viene richiesto in modo interattivo all'utente se sovrascrivere le definizioni di mappa duplicate con la definizione di mappa presente nel file archivio della mappa.

Nota: In modalità di produzione, le mappe saranno automaticamente compilate.

Si può anche utilizzare il comando repos_copy per caricare e scaricare le definizioni di relazione nell'archivio. Per ulteriori informazioni, consultare "Caricamento e scaricamento delle relazioni" a pagina 327.

Capitolo 4. Compilazione e verifica delle mappe

Questo capitolo descrive come convalidare, compilare e testare le mappe con Map Designer Express.

Il capitolo include le seguenti attività:

Verifica del codice di trasformazione

Una volta terminata la scrittura del codice di trasformazione associato ad un attributo di destinazione, è possibile eseguire una limitata verifica sul codice. Eseguendo dei controlli man mano che si procede, si riduce il tempo relativo al debug richiesto al termine del processo di sviluppo della mappa. E' possibile controllare il codice dell'attributo utilizzando la tecnica della ricerca di delimitatori non corrispondenti.

Nota: Questa tecnica è utile anche quando si verifica un errore di compilazione la cui causa non viene immediatamente individuata nel messaggio di errore.

Rilevazione di delimitatori non corrispondenti

Map Designer Express una funzione per il controllo dei delimitatori non corrispondenti, utile per la risoluzione di uno degli errori più difficili da identificare in un programma. Questa funzione verifica la presenza di delimitatori non corrispondenti nel codice di trasformazione di un attributo. Map Designer Express verifica le seguenti coppie di token: ( ), [ ], { }, " " e ' '.

Passi per la rilevazione di delimitatori non corrispondenti

Per effettuare una verifica della sintassi su un codice di trasformazione di un attributo eseguire i seguenti passi:

1. Richiamare Activity Editor in modalità Java.

   Per informazioni su come visualizzare Activity Editor, consultare "Avvio di Activity Editor" a pagina 109.

2. Utilizzare l'opzione Controlla i delimitatori non corrispondenti in Activity Editor. Fare clic con il pulsante destro del mouse e selezionare Controlla i delimitatori non corrispondenti dal menu contestuale.

   Nota: Se è presente un'istanza di delimitatore non accoppiata, Activity Editor visualizza un messaggio nella finestra dell'output, fornendo il numero della riga in cui non è stato possibile risolvere l'errore. Il numero di riga potrebbe non essere la reale riga in cui manca il delimitatore.

3. Per andare all'origine del delimitatore spaiato, prendere nota del numero di riga visualizzato in basso nella finestra.

   Suggerimento: per andare alla riga, utilizzare l'opzione Vai alla riga dalmenu Modifica o dal menu contestuale di Activity Editor. Immettere un numero di riga per andare alla riga in cui si è verificato il problema.

Nota: Se il problema è causato da virgolette spaiate su un'estremità di una stringa letterale, la stringa non viene visualizzata in rosa come dovrebbe. Quando si aggiunge la virgoletta che mancava, l'intera stringa diventa rosa.

Convalida di una mappa

Il processo di convalida di Map Designer Express verifica l'accuratezza del flusso di dati della mappa eseguendo i seguenti controlli:

- Verifica che la mappa non abbia dei passi di trasformazione incompleti.
- Verifica che gli indici dei vettori di oggetti business siano in una sequenza corretta, a partire da zero (0).
- Invia un avviso se un passo di trasformazione viene associato all'attributo `ObjectEventId`.
- Convalida le trasformazioni:
  - Verifica che l'ordine di esecuzione sia corretto, cioè che sia univoco, un numero positivo e consecutivo.
  - Verifica che nessun attributo abbia delle dipendenze cicliche con un altro attributo. Se vengono rilevate delle trasformazioni cicliche, Map Designer Express visualizza le regole cicliche nella finestra di output.
  - Controlla le informazioni della trasformazione:

    Trasformazione Sposta—è implicato un solo attributo origine.

    Trasformazione Unisci—è implicato più di un attributo origine.

    Trasformazione Dividi—è implicato un solo attributo origine; l'indice di suddivisione è maggiore di o uguale a zero; il delimitatore di suddivisione non è vuoto.

    Trasformazione Imposta valore—non sono implicati attributi origine; è stato specificato un valore.

    Trasformazione Mappa secondaria—è implicato almeno un attributo origine; è specificato il nome della mappa secondaria.

    Trasformazione Riferimento incrociato—è implicato solo un attributo origine.

Map Designer Express convalida automaticamente una mappa durante il salvataggio. E' anche possibile scegliere di convalidare la mappa eseguendo una delle seguenti azioni:

- Dal menu File, selezionare Convalida mappa.
- Nella barra strumenti Designer fare clic sul pulsante Convalida.

A questo punto, se sono state specificate delle opzioni nella scheda Convalida della finestra di dialogo Preferenze, Map Designer Express invierà un avviso nel caso in cui una specifica condizione non sia mappata.

Per ulteriori informazioni sull'impostazione di dipendenze fra attributi, consultare "Utilizzo dell'ordine di esecuzione" a pagina 83.

Compilazione di una mappa

Durante la compilazione di una mappa, Map Designer Express genera un file `.class` dal file `.java` che conserva il codice Java delle trasformazioni della mappa. Genera il file `.java` dal codice della trasformazione memorizzato come parte della definizione della mappa nel progetto.

Importante: Per poter compilare una mappa, il compilatore Java (`javac`) deve essere presente sul sistema ed il suo percorso deve essere nella variabile di ambiente `PATH`. Per ulteriori informazioni, consultare "Requisiti per la configurazione di un ambiente di sviluppo" a pagina 11..

Passi per la compilazione di una mappa da Map Designer Express

Da Map Designer Express, è possibile avviare la compilazione di una mappa in diversi modi:

- Per compilare la mappa *corrente* , effettuare una delle seguenti operazioni:
  - Dal menu File, selezionare Compila.
  - Utilizzare il tasto funzione `F7`.
  - Nella barra strumenti Designer fare clic sul pulsante Compila.
- Per compilare la *mappa corrente e le mappe secondarie* che la mappa utilizza:
  - Dal menu File, selezionare Compila con mappe secondarie.
- Per compilare *tutte* le mappe o un sottoinsieme di mappe definito nel System Manager, effettuare una delle seguenti operazioni:
  - Dal menu File selezionare Compila tutto.
  - Utilizzare i tasti funzione `Ctrl+F7`.

  Per ulteriori informazioni, consultare "Compilazione di una serie di mappe" a pagina 90.

Per impostazione predefinita, Map Designer Express salva la mappa nel progetto prima di iniziare la compilazione e genera il codice Java nel file `.java` e nel file `.class`. Se è necessario un file di messaggi, Map Designer Express genera anche il file di messaggi.

Nota: E' possibile specificare se Map Designer Express deve salvare automaticamente una mappa nel progetto prima di compilarla con l'opzione `Compila mappa: salva mappa prima di compilare`. Per impostazione predefinita questa opzione è abilitata. E' possibile modificare l'impostazione dei questa opzione nella scheda Generale della finestra di dialogo Preferenze. Per ulteriori informazioni, consultare "Specifica delle Preferenze generali" a pagina 24.

Per compilare, Map Designer Express richiama il compilatore Java nel codice origine Java della mappa (file `.java`). Le azioni successive dipendono dalla riuscita della compilazione.

Passi per la compilazione di una mappa da System Manager

Anche System Manager fornisce diversi modi per compilare una mappa:

- Per compilare un'unica mappa, effettuare una delle seguenti operazioni:
  - Evidenziare la mappa desiderata e selezionare Compila dal menu Componente.

- Fare clic con il pulsante destro del mouse sulla mappa desiderata e selezionare Compila dal menu contestuale.

- Per compilare una mappa e le relative mappe secondarie:
  - Fare clic con il pulsante destro del mouse sulla mappa desiderata e selezionare Compila con mappe secondarie dal menu contestuale.
- Per compilare *tutte* le mappe definite in un progetto:
  - Evidenziare la cartella Mappe e dal menu Componente selezionare Compila tutto.

Nota: Per selezionare la cartella di mappe del progetto di cui si desidera compilare tutte le mappe, è necessario fare clic con il pulsante destro del mouse sulla cartella di mappe e selezionare Compila tutto dal menu contestuale.

Compilazione di una mappa riuscita

Quando la mappa viene compilata con esito positivo, Map Designer Express esegue le seguenti operazioni:

- Compila il codice Java in un file `.java`.
- Visualizza il seguente messaggio nella finestra di output presente nella parte inferiore di ciascuna scheda Mappa per indicare che non ci sono stati errori durante la compilazione:

  ```
  Compilazione riuscita.
  ```

Compilazione di una mappa non riuscita

Se si verifica un errore durante la compilazione, Map Designer Express genera dei messaggi di errore e li visualizza nella finestra di output presente in basso nello schermo. A meno che non sia già aperta una finestra di output, Map Designer Express ne apre una in basso nella scheda Mappa per visualizzare i messaggi della compilazione.

Quando si verifica un errore di compilazione, la finestra di output visualizza il messaggio di errore con il nome dell'attributo che ha il problema ed il numero di riga in azzurro. Fare clic sul collegamento ipertestuale per andare all'area con il problema nella vista Java di Activity Editor.

Suggerimento: è possibile eliminare i messaggi dalla finestra di output selezionando Cancella output dal menu Vista.

Alcuni errori sono semplici da rilevare, altri no.

Compilazione di una serie di mappe

Utilizzando l'opzione Compila tutto nel menu File è possibile compilare tutte le mappe di System Manager o un sottoinsieme di mappe.

Passi per la compilazione di una serie di mappe

Per compilare una serie di mappe, eseguire i seguenti passi:

1. Dal menu File selezionare Compila tutto.

   Risultato: Map Designer Express visualizza la finestra Compila tutte le mappe.
2. Selezionare il progetto per la compilazione delle mappe.
3. Selezionare le mappe da compilare.

Linea guida: la selezione di qualsiasi casella di controllo principale contrassegnerà automaticamente tutte le relative caselle do controllo secondarie. Quindi, quando si seleziona un progetto, vengono selezionate tutte le mappe di quel progetto. Per selezionare solo un sottoinsieme di mappe, deselezionare le appropriate caselle di controllo Compila.

La Figura 41 illustra la finestra Compila tutte le mappe.

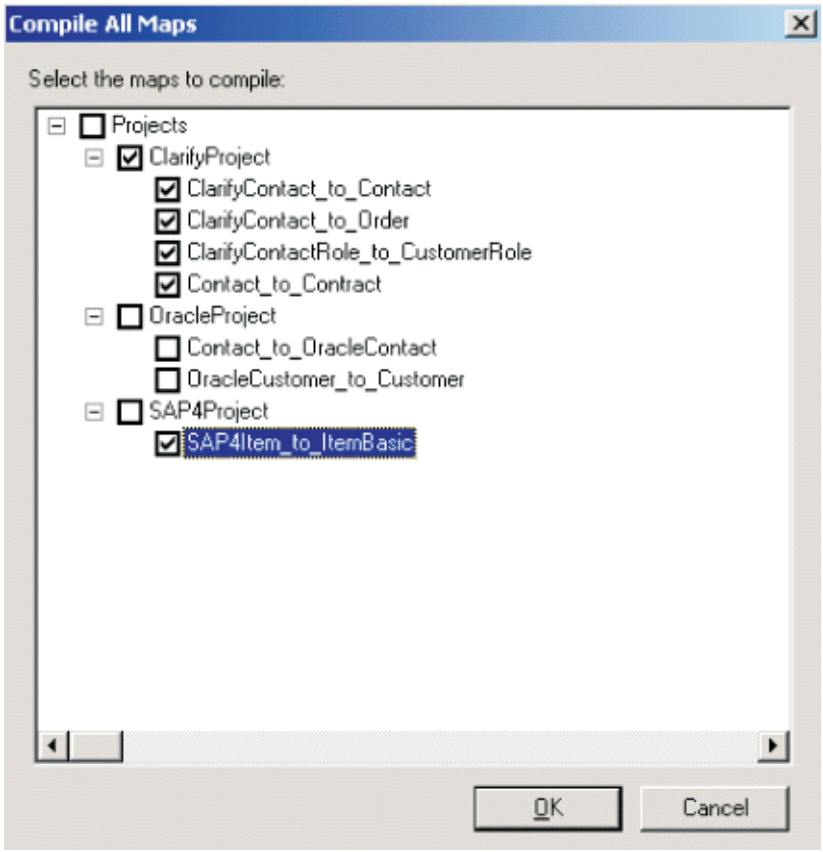


Figura 41. Finestra Compila tutte le mappe

Risultato: Map Designer Express visualizza la riuscita o la non riuscita di ciascuna compilazione di mappa nella finestra di output. Si potrebbe voler ingrandire la dimensione della finestra di output prima di iniziare il processo di compilazione, in modo da poter visualizzare ulteriori messaggi di stato della compilazione.

Test delle mappe

E' possibile testare i passi di trasformazione di una mappa fornendo dei dati semplici per l'oggetto business origine ed eseguendo un test della mappa. Un'*esecuzione test* è un'esecuzione della mappa che non implica un evento inviato da un connettore o da una chiamata inviata da un client di accesso; la mappa si esegue all'interno di Map Designer Express. Map Designer Express fornisce una scheda separata, la scheda Test nella finestra di Map Designer Express per testare le mappe e visualizzare i risultati del test.

Nota: Quando una mappa viene selezionata dall'ambiente di test per un ulteriore processo di debug, l'ambiente di test avvierà Map Designer Express, fornendogli gli oggetti business di input per la mappa da testare.

Questa sezione descrive come configurare ed eseguire una sessione di test, utilizzando questi passi principali:

- "Passi per la preparazione all'esecuzione del test"
- "Creazione dei dati del test"
- "Impostazione dei punti di interruzione (punti di interruzione)" a pagina 95
- "Esecuzione della mappa di test" a pagina 97
- "Visualizzazione dei risultati dell'esecuzione del test" a pagina 99
- "Passi per la modifica della mappa e riesecuzione" a pagina 99

Suggerimento: una strategia di verifica alternativa, che non viene descritta in dettaglio, è di impostare dei punti di interruzione nella mappa e di inviare un evento trigger dal connettore, che causa l'esecuzione della mappa.

Passi per la preparazione all'esecuzione del test

Prima di effettuare il test, eseguire i seguenti passi:

1. Aprire dal progetto la mappa su cui si deve eseguire il debug.
2. Se la mappa *non* è stata compilata dall'ultima modifica, compilarla selezionando Compila dal menu File. Per ulteriori informazioni, consultare "Compilazione di una mappa" a pagina 89.
3. Se la scheda Test di Map Designer Express *non* è al momento visualizzata nella finestra della scheda, selezionare la scheda Test.

Creazione dei dati del test

Ogni volta che si testa una mappa, è necessario caricare i dati nell'oggetto business origine. Per farlo, utilizzare il pannello Dati di test origine nella scheda Test (vedere la Figura 42). Il pannello Dati di test origine consente di specificare le seguenti informazioni sul test:

- Il contesto di chiamata—indica il contesto di esecuzione della mappa.
- L'oggetto business generico—fornisce i dati del test per l'oggetto business generico quando si testa il contesto di chiamata `SERVICE_CALL_RESPONSE` di una relazione d'identità.
- I dati del test—i dati per gli attributi dell'oggetto business origine.

Importante: Il contesto di chiamata e l'oggetto business generico sono richiesti *solo* per testare le relazioni all'interno delle mappe. Per ulteriori informazioni, consultare "Test di mappe che contengono delle relazioni" a pagina 100.

Primo test della mappa

Quando si testa la mappa per la prima volta, è necessario inserire manualmente i valori degli attributi nel pannello Dati di test origine.

Le seguenti sezioni forniscono le informazioni su come immettere questi dati:

- "Linee guida per la creazione dei dati del test dell'oggetto business origine" a pagina 92
- "Passi per la creazione dei dati del test di un oggetto business secondario" a pagina 93

Linee guida per la creazione dei dati del test dell'oggetto business origine: Per creare i dati dell'oggetto business origine la prima volta, seguire queste regole:

- Per impostare l'istruzione, selezionarla dalla casella combinata delle istruzioni nella relativa riga.
- Per assegnare un valore ad un attributo origine, immetterlo nella colonna Valore dell'attributo. *Non* è necessario fornire i valori per tutti gli attributi.
- Per assegnare un valore ad un attributo di relazione, specificare il valore appropriato nella colonna Valore ed accertarsi di specificare il contesto di chiamata corretto. Per ulteriori informazioni, consultare "Test di mappe che contengono delle relazioni" a pagina 100.
- Per assegnare dei valori ad un oggetto business secondario, fare clic con il pulsante destro del mouse sull'oggetto secondario e selezionare Aggiungi istanza dal menu contestuale. Per ulteriori informazioni, consultare "Passi per la creazione dei dati del test di un oggetto business secondario" a pagina 93.
- Per assegnare i valori predefiniti agli attributi origine, selezionare l'oggetto business origine e selezionare Ripristina dal menu contestuale.
- Se si stanno testando le relazioni, assicurarsi di impostare gli `ObjectEventId` dell'oggetto principale di origine e tutti gli oggetti secondari che partecipano alle relazioni.
- Per salvare i valori immessi per future esecuzioni di test, creare un file oggetto business (`.bo`) selezionando l'oggetto business di origine ed eseguendo una delle seguenti azioni:
  - Fare clic sul pulsante Salva nel pannello Dati di test origine.
  - Selezionare Salva dal menu contestuale. Quando richiesto, immettere il nome file in cui verranno memorizzati questi valori.

    Risultato: la volta successiva che verrà testata la mappa, è possibile fare clic sul pulsante Carica da e gli attributi verranno compilati automaticamente dal file dell'oggetto business.

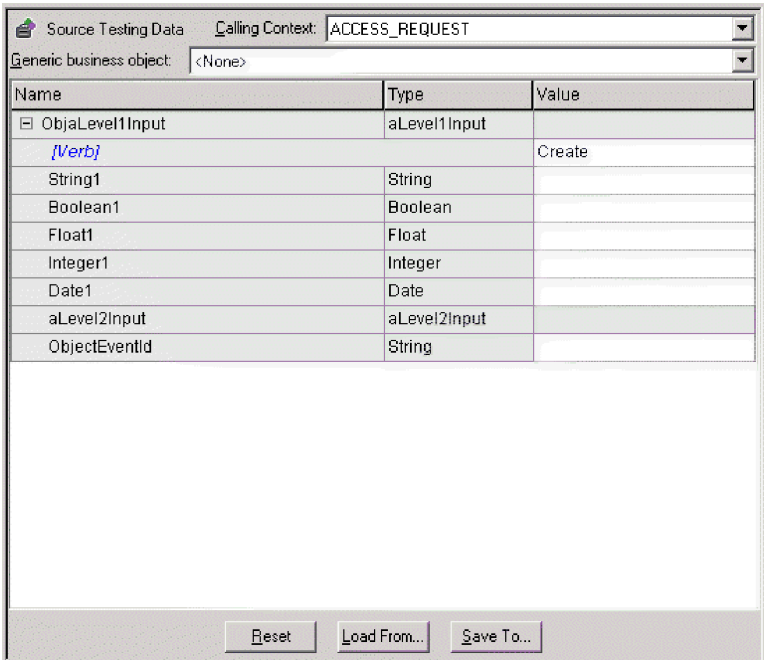


Figura 42. Pannello Dati di test origine della scheda Test

Passi per la creazione dei dati del test di un oggetto business secondario: Se l'oggetto business origine contiene degli oggetti business secondari e si desidera

specificare i dati di test degli attributi secondari, è necessario prima creare un'istanza per ogni oggetto secondario necessario. Per procedere in tal modo, eseguire i seguenti passi:

1. Fare clic con il pulsante destro del mouse sul nome dell'oggetto business secondario e selezionare Aggiungi istanza dal menu contestuale. Quando si espande l'oggetto, viene visualizzata l'istanza che Map Designer Express ha creato.

   Linea guida: la prima istanza aggiunta ha un numero indice pari a zero. E' possibile avere quante istanze si desidera (purché l'attributo secondario abbia la cardinalità multipla).
2. Fare clic sul simbolo più (+) accanto al numero di indice dell'istanza per espandere l'oggetto business secondario.

   Risultato: quando si espande l'oggetto vengono visualizzati gli attributi secondari di questa istanza.
3. Per creare i dati per l'istanza oggetto business secondario, seguire queste linee guida:
   - Per impostare l'istruzione, per l'oggetto business secondario, selezionarla dalla casella combinata dell'istruzione nella relativa riga.
   - Per specificare un valore di un attributo secondario, selezionarlo ed immettere il valore nella colonna Valore.
   - Se il nome dell'attributo è seguito da `(N)`, l'attributo contiene un oggetto business secondario a cardinalità multipla ed è possibile aggiungere altre istanze.

     Per aggiungere un oggetto business secondario alla fine del vettore, fare clic con il pulsante destro del mouse sull'ultimo indice e selezionare Aggiungi istanza dal menu contestuale.
   - Modificare i valori di tutte le istanze desiderate. Aggiungere e rimuovere le istanze nel modo indicato di seguito:
     – Per aggiungere un'istanza, fare clic con il pulsante destro del mouse sul nome istanza secondaria e selezionare Aggiungi istanza.
     – Per eliminare un'istanza, fare clic con il pulsante destro del mouse sul nome dell'istanza secondaria che si desidera eliminare e selezionare Rimuovi istanza.
     – Per eliminare *tutte* le istanze, fare clic con il pulsante destro del mouse sul nome istanza secondaria e selezionare Rimuovi tutte le istanze. Questa opzione è abilitata solo se l'oggetto business secondario ha una cardinalità multipla.

Test della mappa in esecuzioni successive

Per le esecuzioni di test successive, Map Designer Express riutilizza i dati di test specificati precedentemente. E' possibile eseguire qualsiasi azione su quei dati fra quelle elencate di seguito:

- Lasciare tutti i dati di test così come sono.
- Modificare i valori di qualsiasi singolo attributo cambiando le voci appropriate della colonna Valori.

  Suggerimento: se si modificato un dati, ricordarsi di salvare di nuovo qualsiasi file oggetto business (`.bo`).
- Caricare una serie di valori da un file oggetto business (`.bo`).

  Per caricare i valori attributo da un file oggetto business, selezionare l'oggetto business origine ed eseguire una delle seguenti azioni:
  – Fare clic sul pulsante Carica da nel pannello Dati di test origine.

– Selezionare Carica da dal menu contestuale.

Quando richiesto, immettere il nome del file dell'oggetto business da caricare.

- Riportare tutti i valori di destinazione origine alle loro impostazioni predefinite selezionando l'oggetto business origine e selezionando l'opzione Ripristina dal menu contestuale.

Impostazione dei punti di interruzione (punti di interruzione)

Quando si imposta un punto di interruzione, l'esecuzione della mappa viene sospesa subito prima della trasformazione dell'attributo di destinazione su cui è stato impostato il punto di interruzione. L'uso di punti di interruzione consente di entrare nell'esecuzione della mappa e di verificare la sequenza ed i risultati delle singole operazioni. E' possibile impostare tutti i punti di interruzione desiderati.

Linea guida: i punti di interruzione non fanno parte della definizione della mappa. I punti di interruzione vengono impostati sulla mappa dopo aver aperto la mappa in Map Designer Express e quando viene eseguito il debug della mappa (con Debug > Esegui test o Debug > Avanzato > Collega). I punti di interruzione non hanno effetto sulla mappa quando non ne è stato eseguito il debug da Map Designer Express.

Nota: E' possibile impostare un punto di interruzione solo su un attributo di destinazione che ha definita una trasformazione.

Passi per l'impostazione di punti di interruzione

Per impostare un punto di interruzione, eseguire i passi riportati di seguito.

1. Utilizzare uno dei seguenti metodi:
   - Fare clic con il pulsante destro del mouse su un attributo di destinazione nella finestra Dati di test destinazione e selezionare Imposta punto di interruzione dal menu contestuale. Se l'attributo origine di destinazione non è stato ancora espanso, è possibile farlo mediante uno dei seguenti comandi:
     – Fare clic sul simbolo più (+) accanto all'oggetto business di destinazione.
     – Selezionare l'oggetto business di destinazione e selezionare Espandi dal menu contestuale.

     Nota: Il menu contestuale dell'oggetto business di destinazione fornisce anche un'opzione Comprimi.
   - Selezionare Attiva/disattiva punto d'interruzione dal menu Debug.
   - Utilizzare il tasto funzione della tastiera `F9`.
   - Nella barra strumenti Designer, fare clic sul pulsante Attiva/disattiva punto di interruzione.

   Nota: L'opzione Attiva/disattiva punto di interruzione alterna l'attivazione della definizione di un punto di interruzione. Se il punto di interruzione *non* è al momento impostato, Attiva/disattiva punto di interruzione lo imposta. Se al momento il punto di interruzione è impostato, Attiva/disattiva punto di interruzione lo rimuove.

   Risultato: Map Designer Express visualizza un pallino scuro accanto all'attributo di destinazione su cui è impostato il punto di interruzione, come mostrato nella Figura 43 a pagina 96.

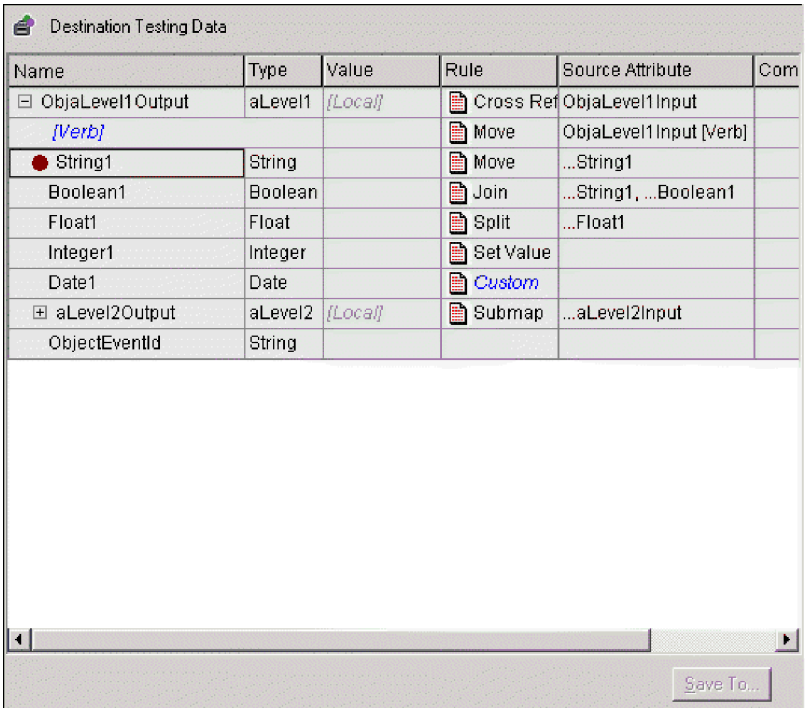


Figura 43. Punto di interruzione impostato

Una volta impostato il punto di interruzione, l'esecuzione dell'istanza della mappa viene sospesa in questo punto di interruzione ed è possibile visualizzare lo stato corrente della mappa. A meno che non si specifichi almeno un punto di interruzione, la mappa viene eseguita e finisce con il messaggio:

```
Esecuzione test terminata
```

Regole: è necessario fornire sempre i valori dei dati origine associati agli attributi di destinazione in cui vengono impostati i punti di interruzione. Altrimenti la regola di trasformazione verrà eseguita normalmente come anche i punti di interruzione, ma il valore di destinazione di solito sarà vuoto, a seconda della regola di trasformazione definita. Per ulteriori informazioni, consultare "Creazione dei dati del test" a pagina 92.

Per visualizzare tutti i punti di interruzione della mappa selezionare Punti di interruzione dal menu Debug.

Risultato: Map Designer Express visualizza la finestra di dialogo Punti di interruzione (vedere la Figura 44).

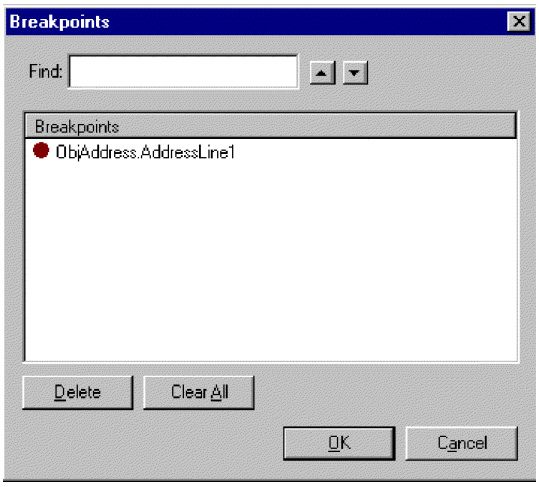


Figura 44. Finestra di dialogo Punti di interruzione della scheda Test

2. Dalla finestra di dialogo Punti di interruzione è possibile eseguire una delle seguenti azioni:
   - Individuare un attributo di destinazione sul quale è stato impostato un punto di interruzione—fare doppio clic sul nome del punto di interruzione.

     Suggerimento: per individuare un particolare punto di interruzione, immetterne il nome nel campo Trova. Scorrere l'elenco di oggetti business con le frecce verso l'alto e verso il basso. Nella finestra Dati di test destinazione, Map Designer Express evidenzia l'attributo di destinazione.
   - Rimuovere un punto di interruzione—nell'area Punti di interruzione, selezionare il punto di interruzione da rimuovere e fare clic sul pulsante Elimina.

     E' possibile rimuovere un punto di interruzione anche eseguendo una delle seguenti azioni:
     - Fare clic con il pulsante destro del mouse su un attributo di destinazione nella finestra Dati di test destinazione e selezionare Cancella punto di interruzione dal menu contestuale.
     - Utilizzare uno dei comandi dell'opzione Attiva/disattiva punto di interruzione su un punto di interruzione esistente. Per ulteriori informazioni, consultare "Impostazione dei punti di interruzione (punti di interruzione)" a pagina 95.
   - Cancellare tutti i punti di interruzione visualizzati nell'area Punti di interruzione—fare clic sul pulsante Cancella tutti.

     E' possibile cancellare tutti i punti di interruzione anche eseguendo una delle seguenti azioni:
     - Dal menu Debug, selezionare Cancella tutti i punti di interruzione.
     - Nella barra strumenti Designer fare clic sul pulsante Cancella tutti i punti di interruzione.

Esecuzione della mappa di test

Una volta immessi i dati di test di origine ed impostati i punti di interruzione desiderati, si è pronti per testare la mappa. L'esecuzione di un test della mappa implica:

1. "Passi per l'avvio dell'esecuzione del test" a pagina 98

2. “Passi per l’elaborazione dei punti di interruzione” (se sono stati impostati dei punti di interruzione)

Passi per l’avvio dell’esecuzione del test

Per avviare l’esecuzione del test, eseguire i seguenti passi:

1. Eseguire una delle seguenti azioni:
   - Dal menu Debug, selezionare Esegui test.
   - Nella barra strumenti Designer fare clic sul pulsante Esegui test.

   Risultato: viene visualizzata la finestra di dialogo Connetti a InterChange Server Express, che consente di collegarsi al server per il test.
2. Nella finestra di dialogo, immettere il nome server, il nome utente e la password.
3. Specificare se si desidera distribuire la mappa e gli oggetti business dipendenti per l’esecuzione del test.

   Linea guida: la distribuzione al server di un gruppo minimo di oggetti business per il test ridurrà i tempi di inizializzazione del debug.

Risultato: viene avviata l’esecuzione della mappa. Map Designer Express visualizza il seguente messaggio nella finestra di output:

```
Avvio esecuzione test in corso...
```

Passi per l’elaborazione dei punti di interruzione

L’esecuzione della mappa viene sospesa quando si incontra un attributo di destinazione in cui è stato impostato un punto di interruzione. Quando si raggiunge un punto di interruzione, Map Designer Express esegue le seguenti azioni:

1. Evidenzia l’attributo di destinazione sul quale è stato impostato il punto di interruzione e visualizza un pallino scuro con accanto una freccia gialla.
2. Visualizza il seguente messaggio nella finestra di output:

   ```
   Esecuzione test interrotta all'attributo AttrName (prossima trasformazione > "Regola").
   ```

   Suggerimento: con l’esecuzione della mappa sospesa, è possibile esaminare i valori degli attributi di destinazione che sono stati elaborati fino a quel momento osservando la colonna Valore della finestra Dati di test destinazione.
3. Elabora il punto di interruzione e continua l ’esecuzione della mappa, quando si esegue una delle seguenti azioni:
   - Si passa al successivo punto di interruzione o al termine della mappa, a seconda di cosa viene prima.

     Per continuare l’esecuzione della mappa, eseguire una delle seguenti azioni:
     - Dal menu Debug, selezionare Continua.
     - Utilizzare il tasto funzione della tastiera F8.
     - Nella barra strumenti Designer, fare clic sul pulsante Continua.
   - Eseguire l’attributo di destinazione ed interrompere prima dell’esecuzione dell’attributo successivo.

     Per continuare di un altro passo l’esecuzione della mappa, eseguire una delle seguenti azioni:
     - Dal menu Debug, selezionare Ignora.

       Suggerimento: selezionare questa opzione per osservare l’attributo di esecuzione del codice per attributo.
     - Utilizzare il tasto funzione F10 della tastiera.
     - Nella barra strumenti Designer, fare clic sul pulsante Ignora.

Risultato: quando l'esecuzione del test termina senza errori di runtime, Map Designer Express visualizza il seguente messaggio nella finestra di output:

```
Esecuzione test terminata.
```

Visualizzazione dei risultati dell'esecuzione del test

I risultati dell'esecuzione del test vengono visualizzati nell'oggetto business di destinazione, che si trova nella finestra Dati di test destinazione. I valori risultanti dalle trasformazioni della mappa sono visibili nella colonna Valori di questa tabella. E' possibile visualizzare i risultati dell'esecuzione del test tramite:

- "Osservazione del processo"
- "Visualizzazione dei risultati dopo l'esecuzione"

Osservazione del processo

Durante un'esecuzione di un test che contiene dati e punti di interruzione, è possibile osservare l'oggetto business di destinazione mentre viene compilato con i valori. I valori vengono visualizzati nella colonna Valori della finestra Dati di test destinazione appena vengono elaborati. Quando l'esecuzione della mappa viene sospesa su un punto di interruzione, tutti gli attributi di destinazione che *precedono* quell'attributo nell'ordine di esecuzione, hanno i valori visualizzati.

Per vedere le trasformazioni mentre si verificano:

- Impostare un punto di interruzione sul secondo attributo di destinazione ed entrare nell'esecuzione della mappa con l'opzione Ignora. La mappa sarà disponibile in sola lettura.

Visualizzazione dei risultati dopo l'esecuzione

Per visualizzare i risultati dell'esecuzione del test quando la mappa è stata già eseguita, esaminare l'oggetto business di destinazione contenuto nella finestra Dati di test destinazione.

Per salvare i risultati del test:

- Evidenziare l'oggetto business di destinazione e selezionare Salva in dal menu contestuale.

  Risultato: Map Designer Express salva i valori degli attributi di destinazione in un file oggetto business (`.bo`).

Passi per la modifica della mappa e riesecuzione

Durante il test della mappa si potrebbe rilevare la necessità di modificare la mappa. Per modificare la mappa e continuare il test, eseguire i seguenti passi:

1. Andare alla scheda Tabella o Diagramma per visualizzare le trasformazioni della mappa.
2. Apportare le modifiche per correggere gli errori.
3. Ricompilare la mappa.
4. Continuare il processo di test tornando alla scheda Test.
5. Iniziare una nuova esecuzione di test.

Importante:

1. Assicurarsi di completare l'esecuzione del test, con esito positivo o negativo, prima di tentare di ricompilare la mappa.
2. Dopo aver modificato la mappa, distribuirla al server perché l modifiche si riflettano nel server.

Debug avanzato

Oltre alle mappe di debug memorizzate nei progetti locali, è possibile effettuare direttamente il debug di una mappa che risiede sul server. Per farlo, eseguire i seguenti passi:

1. Selezionare Debug > Avanzato > Collega.

   Risultato: viene visualizzata la finestra di dialogo Connetti a InterChange Server Express.

2. Immettere il nome server, il nome utente, la password e fare clic su Connetti.

   Risultato: Map Designer Express visualizza un elenco delle nuove mappe presenti su quel server.

3. Selezionare la mappa alla quale ci si desidera collegare.

   Risultato: la mappa viene aperta in Map Designer Express in modalità di sola lettura.

4. Impostare dei punti di interruzione nella mappa perché il server sospenda l'esecuzione della mappa ad una determinata regola di trasformazione.

   Risultato: quando si incontra un punto di interruzione sul server, è possibile ignorarlo o continuare l'esecuzione della mappa, come sempre. I valori dell'oggetto business risultanti verranno visualizzati nella finestra Dati di test destinazione.

5. Interrompere la sessione di debug in qualsiasi momento utilizzando Debug > Avanzato > Scollega.

   Risultato: Map Designer Express chiuderà la mappa.

Test di mappe che contengono delle relazioni

Quando si testa una mappa che contiene una trasformazione di relazione, oltre ai dati del test è necessario fornire le seguenti informazioni:

- Il contesto di chiamata

  Parte del contesto di esecuzione della mappa include un contesto di chiamata. Molti dei metodi di relazione nell'API di mappatura utilizzano questo contesto di chiamata per stabilire quale azione intraprendere durante la mappatura. Per questo motivo, se si sta testando un attributo di relazione in una mappa, di solito è necessario specificare l'appropriato contesto di chiamata per la trasformazione.

- La definizione dell'oggetto business generico

  Quando si testa il contesto di chiamata `SERVICE_CALL_RESPONSE` di una relazione d'identità, è necessario specificare l'oggetto business generico della mappa, in modo che l'esecuzione del test possa individuare il valore chiave generico nella relazione.

Nota: Per ulteriori informazioni sul richiamo di mappe all'interno di una collaborazione, consultare il manuale *Guida allo sviluppo della collaborazione*.

Specificare queste informazioni nella finestra Dati di test origine della scheda Test.

Suggerimento: se la larghezza della finestra Dati di test origine non è sufficiente a consentire una visualizzazione completa delle opzioni di menu della casella combinata Contesto di chiamata, è possibile espandere le dimensioni dell'area posizionando il cursore sul margine destro finché non viene visualizzato il simbolo <-||-> e trascinare il margine verso destra.

Se si stanno testando le relazioni, selezionare l'appropriato oggetto generico dall'elenco degli oggetti business, selezionare Contesto di chiamata ed impostare gli attributi `ObjectEventId` degli oggetti principali e secondari che corrispondono a quelli già impostati nella finestra dei dati del test. Il contesto di chiamata che è necessario fornire e se si deve specificare un oggetto business generico, dipendono dal tipo di relazione che si sta testando. Questa sezione fornisce le informazioni sul:

- "Test di una relazione d'identità"
- "Test di una relazione di ricerca" a pagina 104

Test di una relazione d'identità

Per testare la mappatura point-to-point (dall'Applicazione 1 all'Applicazione 2) per una relazione d'identità si utilizzano tre mappe:

- Una mappa in entrata da un oggetto business specifico per l'Applicazione 1 ad un oggetto business generico—`App1_to_Generic`
- Una mappa in uscita dall'oggetto business generico all'oggetto business specifico per l'Applicazione 2—`Generic_to_App2`
- Una mappa in entrata da un oggetto business specifico per l'Applicazione 2 ad un oggetto business generico—`App2_to_Generic`

Esempio: la Figura 45 mostra un esempio di comunicazione point-to-point di dati cliente fra un'applicazione Clarify ed un'applicazione SAP. Se ciascuna applicazione utilizza un valore chiave univoco per identificare i clienti, i tre oggetti business possono essere messi in relazione con una relazione d'identità. Pertanto ogni mappa include una regola di trasformazione riferimento incrociato. Quando ognuna di queste mappe viene eseguita, i metodi di relazione accedono al contesto di chiamata per stabilire quale azione intraprendere.

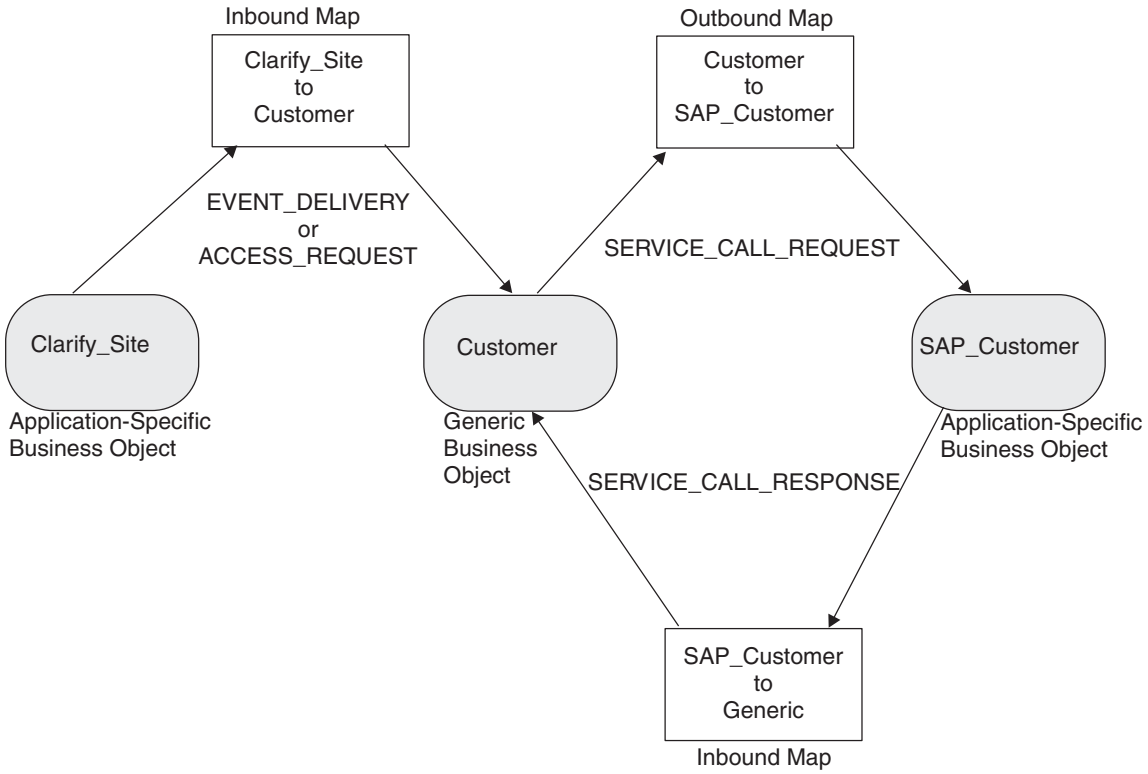


Figura 45. Mappe coinvolte nel test point-to-point di una relazione d'identità

Per testare l'istruzione Crea è necessario verificare che un nuovo valore chiave specifico per applicazione nell'Applicazione 1 (applicazione Clarify nella Figura 45) comporti l'aggiunta di un nuovo valore chiave generico per l'oggetto business generico *e* di un nuovo valore chiave specifico per applicazione nell'Applicazione 2 (applicazione SAP nella Figura 45). Quindi il test implica tre passi:

1. Test della mappa in entrata, `App1_to_Generic`, per inviare un nuovo valore chiave dall'Applicazione 1 e verificare che sia generato un nuovo valore chiave per l'oggetto business generico. Seguire i passi della Tabella 21.

Tabella 21. Test della mappa App1-to-Generic per una relazione d'identità

Per configurare l'esecuzione test	**Per verificare l'esecuzione test**
1. Impostare il contesto di chiamata su `EVENT_DELIVERY` o `ACCESS_REQUEST` selezionando l'appropriato contesto di chiamata dalla casella combinata Contesto di chiamata. 2. Immettere il valore specifico per applicazione nella chiave dell'oggetto business origine. Questo valore è univoco per l'attributo chiave nell'Applicazione 1. 3. Eseguire il test.	4. Leggere il valore chiave generico risultante nell'oggetto business di destinazione, che è stato aggiunto alla tabella delle relazioni per la relazione d'identità `App1/Generic`. 5. Salvare i dati dell'oggetto business di destinazione in un file `.bo` (ad es. `App1_to_Generic.bo`) selezionando l'oggetto business di destinazione e selezionando Salva in dal menu contestuale.

2. Test della mappa in uscita, `Generic_to_App2`, per verificare che il nuovo valore chiave generico venga inviato all'Applicazione 2.

   Per testare una relazione d'identità nella mappa in uscita `Generic_to_App2`, è necessario fornire il valore chiave generico nei dati del test di origine. Si potrebbe voler eseguire una delle due operazioni riportate di seguito *ma sono entrambe sbagliate*:

- Mettere un numero arbitrario nell'attributo primario dell'oggetto business generico ed eseguire la mappa.
- Creare il record direttamente nella tabella delle relazioni.

In entrambi i casi, Map Designer Express genera l'eccezione `RelationshipRuntimeException` o `NullPointerException`. L'errore si verifica perché il valore chiave generico deve essere presente nel sistema, perché `SERVICE_CALL_REQUEST` funzioni correttamente, e la tabella delle relazioni *non* è l'unico punto in cui viene memorizzato il valore chiave generico.

La soluzione corretta è di eseguire prima una mappa in entrata `EVENT_DELIVERY` (o `ACCESS_REQUEST`) che utilizza la stessa relazione d'identità (come descritto al passo 1). Seguire i passi contenuti in Tabella 22 per testare la mappa in uscita `Generic_to_App2`.

Tabella 22. Test della mappa generic-to-app2 per una relazione d'identità

Per configurare l'esecuzione test	Per verificare l'esecuzione test
1. Impostare il contesto di chiamata su `SERVICE_CALL_REQUEST` selezionando questo contesto di chiamata dalla relativa casella combinata. 2. Caricare l'oggetto business generico con i risultati del test provenienti dal passo precedente (ad es. `App1_to_Generic.bo`). 3. Eseguire il test.	4. Leggere il risultante valore chiave specifico per applicazione nell'oggetto business di destinazione, che è vuoto perché l'Applicazione 2 non ha ancora generato il suo valore chiave. 5. Salvare i dati dell'oggetto business di destinazione in un file `.bo` (ad es. `Generic_to_App2.bo`) selezionando l'oggetto business di destinazione e selezionando Salva in dal menu contestuale.

3. Test della mappa in entrata, `app2_to_generic`, per verificare che il nuovo valore chiave proveniente dall'Applicazione 2 venga associato al nuovo valore chiave generico.

   Quando il contesto di chiamata `SERVICE_CALL_RESPONSE`, una relazione d'identità deve creare un riferimento incrociato fra 'ID contenuto nell'oggetto business specifico per applicazione e l'ID contenuto nell'oggetto business generico. Pertanto per questo test è necessario specificare la definizione dell'oggetto business generico. Seguire i passi della Tabella 23.

Tabella 23. Test della mappa App2_to_Generic per una relazione d'identità

Per configurare l'esecuzione test	Per verificare l'esecuzione test
1. Impostare il contesto di chiamata su SERVICE_CALL_RESPONSE selezionando questo contesto di chiamata dalla relativa casella combinata. 2. Impostare l'oggetto business generico selezionando il nome dell'appropriato oggetto business generico dalla relativa casella combinata. Map Designer Express aggiunge l'oggetto business generico specificato alla finestra Dati di test origine. 3. Caricare l'oggetto business specifico per applicazione con i risultati del test provenienti dal passo precedente (ad es. Generic_to_App2.bo). 4. Nell'oggetto business specifico per l'applicazione, immettere il valore specifico per applicazione nella chiave dell'oggetto business. 5. Nell'oggetto business generico, immettere il valore chiave generico associato alla chiave Applicazione 1. Questo valore deve essere lo stesso valore chiave generato per l'oggetto business generico nel test EVENT_DELIVERY/ACCESS_REQUEST (passo 1). 6. Eseguire il test.	7. Leggere il valore chiave generico risultante nell'oggetto business di destinazione, che deve essere lo stesso valore immesso nell'oggetto business generico. 8. E' possibile utilizzare il gestore relazioni per verificare che per questa relazione d'identità i corretti valori chiave specifici per applicazione siano stati associati a questo valore chiave generico.

Il test di altre istruzioni implica dei passi simili. Per informazioni più dettagliate sulle azioni dei metodi di relazione relative alla relazione d'identità, consultare Capitolo 8, "Implementazione delle relazioni", a pagina 271.

Test di una relazione di ricerca

Per testare la mappatura point-to-point (dall'Applicazione 1 all'Applicazione 2) per una relazione di ricerca si utilizzano due mappe:

- Dall'oggetto business specifico per applicazione dell'Applicazione 1 ad un oggetto business generico—App1_to_Generic
- Dall'oggetto business generico all'oggetto business specifico per applicazione dell'Applicazione 2—Generic_to_App2

Esempio: la Figura 46 mostra un esempio di comunicazione point-to-point di dati cliente fra un'applicazione Clarify ed un'applicazione SAP. Se ciascuna applicazione utilizza un codice statico speciale per identificare gli stati geografici, questi tre oggetti business possono essere messi in relazione con una relazione di ricerca. Pertanto ogni mappa include le trasformazioni personalizzate che effettuano le ricerche statiche. Per ulteriori informazioni, consultare l'esempio di attività ″Ricerca statica″ in "Esempio 3: utilizzo della ricerca statica per la conversione" a pagina 164. Quando ognuna di queste mappe viene eseguita, i metodi di relazione accedono al contesto di chiamata per stabilire quale azione intraprendere.

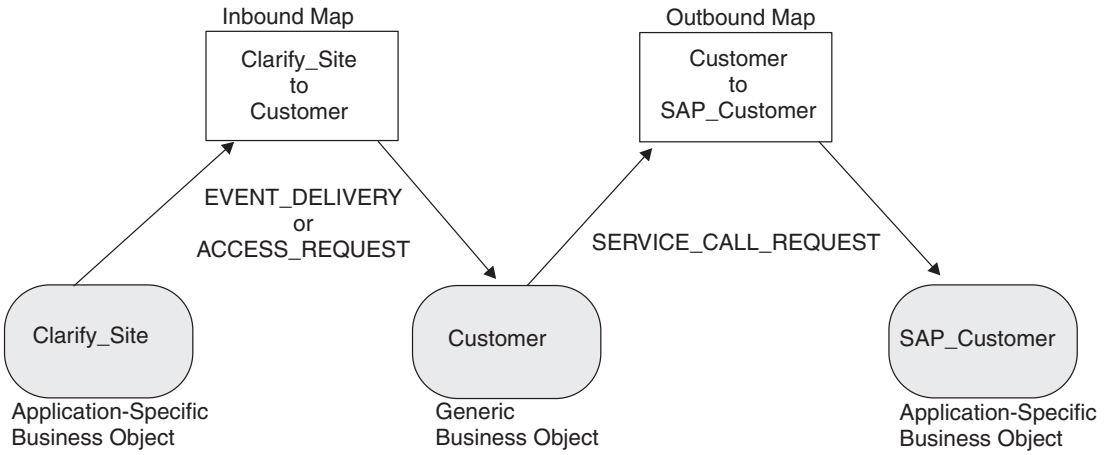


Figura 46. Mappe coinvolte nel test point-to-point di una relazione di ricerca

Per testare l'istruzione Crea è necessario verificare che un valore esistente di ricerca specifico per applicazione nell'Applicazione 1 (applicazione Clarify nella Figura 46) comporti l'aggiunta di un valore ricerca generico per l'oggetto business generico *e* di un valore ricerca specifico per applicazione nell'Applicazione 2 (applicazione SAP nella Figura 46). Quindi il test implica due passi:

1. Test della mappa in entrata, `App1_to_Generic`, per inviare un valore ricerca esistente dall'Applicazione 1 e verificare che il valore ricerca generico associato venga richiamato per l'oggetto business generico. Seguire i passi della Tabella 24.

Tabella 24. Test della mappa App1-to-Generic per una relazione di ricerca

Per configurare l'esecuzione test	**Per verificare l'esecuzione test**
1. Impostare il contesto di chiamata su `EVENT_DELIVERY` o `ACCESS_REQUEST` selezionando l'appropriato contesto di chiamata dalla casella combinata Contesto di chiamata. 2. Immettere il valore specifico per applicazione nel campo di ricerca dell'oggetto business origine. Questo è un valore esistente di ricerca i cui dati sono già caricati nella tabella delle relazioni `App1/Generico`. 3. Eseguire il test.	4. Leggere il valore ricerca generico risultante nell'oggetto business di destinazione, che è stato richiamato alla tabella delle relazioni per la relazione di ricerca `App1/Generico`. 5. Salvare i dati dell'oggetto business di destinazione in un file `.bo` (ad es. `App1_to_Generic.bo`) evidenziando l'oggetto business di destinazione e selezionando Salva in dal menu contestuale.

2. Test della mappa in uscita, `Generic_to_App2`, per inviare il valore ricerca generico e verificare che il valore ricerca associato venga richiamato per l'Applicazione 2. Seguire i passi riportati nella Tabella 25.

Tabella 25. Test della mappa Generic-to-App2 Map per una relazione di ricerca

Per configurare l'esecuzione test	Per verificare l'esecuzione test
1. Impostare il contesto di chiamata su `SERVICE_CALL_REQUEST` selezionando questo contesto di chiamata dalla relativa casella combinata. 2. Caricare l'oggetto business generico con i risultati del test provenienti dal passo precedente (ad es. `App1_to_Generic.bo`). 3. Eseguire il test.	4. Leggere il risultante valore chiave specifico per applicazione nell'oggetto business di destinazione, che contiene il valore di ricerca dell'Applicazione 2. 5. Salvare i dati dell'oggetto business di destinazione in un file `.bo` (ad es. `Generic_to_App2.bo`) evidenziando l'oggetto business di destinazione e selezionando Salva in dal menu contestuale.

Nota: Una relazione di ricerca può essere testata per il contesto di chiamata `SERVICE_CALL_RESPONSE`. Tuttavia, in genere questo caso è richiesto solo se la mappa sta facendo qualcosa che richiede i dati di ricerca. I metodi di relazione per una relazione di ricerca nell'API di mappatura non scrivono mai i dati in una tabella di relazioni.

Debug delle mappe

In questa sezione vengono fornite le seguenti informazioni sul debug di una mappa:

- "Risoluzione degli errori runtime"
- "Suggerimenti per il debug" a pagina 107

Per informazioni su come testare le relazioni, consultare "Test di mappe che contengono delle relazioni" a pagina 100.

Risoluzione degli errori runtime

Anche se la mappa viene compilata con esito positivo, si può verificare un errore di runtime durante l'esecuzione della mappa nel Debugger.

Importante: quando si risolvono errori di runtime, verificare che *non* vi siano eventi in sospeso o non riusciti correlati all'oggetto business dipendente *prima* di iniziare il debug della mappa.

Esempio 1: si dispone di una mappa in uscita con l'oggetto business generico da una parte ed un oggetto business specifico per applicazione dall'altra. Si supponga che questa mappa contenga una relazione d'identità.

1. Andare alla scheda Test e selezionare il contesto di chiamata `SERVICE_CALL_REQUEST`.
2. Selezionare l'istruzione ″Aggiorna.″
3. Eseguire il test.

   Risultato: viene visualizzato un messaggio di errore simile a quello quello riportato di seguito:

   ```
   Eccezione al passo 17,
   attributo <nome attributo>,
   java.lang.nullpointerexception
   ```

Questa eccezione si è verificata perché la mappa sta tentando di aggiornare una voce nel repository che non è stata creata. Idealmente, è necessario verificare che la sequenza dei passi sia corretta. Cercare nel database delle voci di relazione relative alla mappa in questione. Poi trarre le conclusioni tenendo presente se è pronta o meno per SERVICE_CALL_REQUEST.

Esempio 2: si dispone della seguente riga per il codice di mappatura di `Customer.CustomerId`:

```
_cw_CpBTBSourceValue = ObjSAP_CustomerMaster.get("CustomerIdd");
```

Chiaramente contiene un errore di battitura (una lettera in più, *d*, nel nome dell'attributo). Sfortunatamente il programma di compilazione non rileva questo errore perché è contenuto in una costante a stringa. Non vi è modo per il programma di compilazione di verificare quale dovrebbe essere un valore "corretto" per la costante. Tuttavia, quando si esegue la mappa viene visualizzata la seguente finestra di dialogo di errore di InterChange Server Express:

```
Errore ICS: Eccezione al passo 3, attributo CustomerId, Numero msg
di errore - 11030, Error11030 L'attributo CustomerIdd non esiste nell'oggetto
business SAP_CustomerMaster.
```

Quando si riceve questo errore di runtime, lasciare la scheda Test e correggere la mappa.

Suggerimenti per il debug

Questa sezione fornisce i seguenti suggerimenti per semplificare il debug di una mappa:

- "Utilizzo della registrazione dei messaggi"
- "Scrittura di codice di mappatura sicuro" a pagina 108

Importante: quando si esegue il debug di una mappa, verificare che *non* vi siano eventi in sospeso o non riusciti correlati all'oggetto business dipendente *prima* di iniziare il debug della mappa.

Utilizzo della registrazione dei messaggi

Utilizzare il metodo `logInfo()` per tenere traccia dell'esecuzione della mappa. Prende una `Stringa` come argomento, che viene inviata nel logo di InterChange Server Express. E' necessario inserirla in Activity Editor per l'attributo la cui esecuzione deve essere registrata. Per essere sicuri che la mappa secondaria venga eseguita, creare una regola di trasformazione personalizzata ed utilizzare il blocco funzione ″Registra informazioni″ per personalizzare l'attività o scrivere direttamente il codice.

Esempio: il codice può essere semplice come il seguente:

```
logInfo("in submap");
```

Posizionarlo nella prima riga del codice del primo attributo dell'oggetto di destinazione nella mappa secondaria.

Esempio: se si deve tenere traccia del valore dell'attributo specifico `SAP.CustomerName`, utilizzare:

```
logInfo(ObjSAP_CustomerMaster.getString("CustomerName"));
```

Si potrebbe non voler sempre vedere questo messaggio. In questo caso, modificare la proprietà `DataValidationLevel` della mappa.

Per impostare `DataValidationLevel`, selezionare l'opzione Proprietà mappa dal menu Modifica di Map Designer Express e modificare 0 in 1 o in un numero più grande. Le impostazioni sono le seguenti:

0	Nessuna convalida dati

1	Livello di convalida dati IBM
2 o maggiore	Convalida dati definita dall'utente

Per essere sicuri che il messaggio `logInfo` non venga visualizzato, impostare `DataValidationLevel` su 1. Nel codice, prima di richiamare il metodo `logInfo()`, verificare il livello di convalida dati. Questo è il codice:

```
if (dataValidationLevel > 1)
   logInfo(ObjSAP_CustomerMaster.getString("CustomerName"));
```

Questo assicura che `logInfo` venga eseguito solo se il livello di convalida dati è impostato su un numero maggiore di 1. Se si decide di visualizzare il messaggio, cambiare a 2 l'impostazione del livello di convalida dati nelle proprietà della mappa.

Scrittura di codice di mappatura sicuro

Se si personalizza la regola di trasformazione in Activity Editor o si scrive il proprio codice di mappatura, *non* vi è garanzia che funzionerà correttamente durante il runtime. Per essere sicuri che la mappa continui l'esecuzione quando si verifica un errore e se ne riceve la notifica, utilizzare il blocco funzione ″Rileva errore″ in Activity Editor o seguire la metodologia Java per la gestione delle eccezioni.

Esempio: collocare il codice nel blocco `try`, ad esempio:

```
try
   {
   BusObj temp = new BusObj("SAP_Order");
// resto del codice
   }
```

Quindi utilizzare un blocco `catch` per rilevare qualsiasi eccezione possa verificarsi quando viene eseguito il codice:

```
catch (Exception e)
   {
   logInfo(e.toString());
   }
```

Il metodo `logInfo()` può essere utilizzato per inviare i messaggi di errore generati dal sistema al log di InterChange Server Express.

Capitolo 5. Personalizzazione di una mappa

Questo capitolo fornisce due metodi per generare del codice Java: utilizzando Activity Editor per definire graficamente le regole di trasformazione e scrivendo il codice Java direttamente.

Il capitolo descrive i seguenti argomenti:

Panoramica di Activity Editor

Utilizzando Activity Editor, è possibile specificare graficamente il flusso di attività di una specifica regola di trasformazione, senza conoscere la programmazione o il codice Java. Per ogni regola di trasformazione in Map Designer Express, è possibile visualizzare un'attività e le relative attività secondarie. E' possibile visualizzare graficamente il codice di trasformazione degli attributi associati, modificarlo e creare, tramite lo strumento, il corrispondente codice Java.

Avviare Activity Editor direttamente da Map Designer Express (consultare "Avvio di Activity Editor" a pagina 109). All'avvio, Activity Editor comunica con System Manager per individuare la serie di attività consentite. Dopo aver progettato l'attività di una particolare regola di trasformazione, salvarne le modifiche in Activity Editor; vengono comunicate a Map Designer Express.

Questa sezione descrive i seguenti argomenti per introdurre Activity Editor:

Avvio di Activity Editor

Activity Editor si avvia dalla colonna della regola di trasformazione nella scheda Tabella o Diagramma di Map Designer Express. Per farlo, eseguire i passi riportati di seguito:

1. Selezionare l'attributo che si desidera utilizzare.
2. Effettuare una delle seguenti operazioni:

- Fare doppio clic sulla corrispondente cella dell'attributo nella colonna della regola di trasformazione.
- Fare clic sull'icona bitmap nella cella corrispondente della colonna della regola di trasformazione.

Risultato: Map Designer Express risponde a queste azioni in base a quanto segue:

- Se il codice è ancora in modalità aggiornamento automatico

  Il codice di trasformazione è in modalità aggiornamento automatico se è stato generato da Map Designer Express e non è stato personalizzato in alcun modo dall'utente. Quando si personalizza il codice in aggiornamento automatico, Activity Editor visualizza una richiesta di conferma che informa che salvando il codice viene disattivata la modalità aggiornamento automatico. Per il codice non in modalità aggiornamento automatico, Map Designer Express visualizza la regola di trasformazione in carattere corsivo blu nella relativa colonna.

  Se il codice di trasformazione *non* è in modalità aggiornamento automatico (cioè se è stato modificato il codice generato automaticamente), Map Designer Express apre Activity Editor nella vista Java, quando si fa doppio clic sulla cella della regola di trasformazione dell'attributo oppure si fa clic sull'icona della regola di mappatura.

- Al tipo di trasformazione definito

  Il codice di trasformazione in modalità aggiornamento automatico viene generato da una delle trasformazioni standard che Map Designer Express fornisce nella casella combinata della colonna della regola di trasformazione. Quando si fa doppio clic sulla cella della regola di trasformazione dell'attributo o si fa clic sull'icona della regola di mappatura, il tipo di trasformazione determina cosa visualizzerà Map Designer Express:

  – Per la Trasformazione personalizzata, Map Designer Express apre Activity Editor sul codice di trasformazione.
  – Per tutte le altre trasformazioni standard (Imposta valore, Unisci, Dividi, Mappa secondaria e Riferimento incrociato), Map Designer Express visualizza la finestra di dialogo della trasformazione. Nella finestra di dialogo, fare clic sul pulsante Visualizza codice per aprire Activity Editor in una nuova finestra con il nome dell'attributo nella barra del titolo. E' possibile aprire più istanze di Activity Editor contemporaneamente.

Layout di Activity Editor

Activity Editor dispone di due viste principali: vista Grafica e vista Java. A seconda del tipo di attività, in qualsiasi momento è visibile solo una vista alla volta. Quindi, se Map Designer Express richiama Activity Editor per visualizzare un'attività grafica, Activity Editor si avvierà con la vista Grafica. Se si sceglie di convertire questa attività grafica in codice Java code, verrà visualizzata la vista Java al posto di quella grafica.

Limitazioni: una volta passata al codice Java, l'attività non verrà riconvertita di nuovo nel tipo grafico.

Entrambe le viste hanno gli elementi della finestra comuni nelle loro modalità Progettazione e Vista rapida, come descritto in Tabella 26.

Tabella 26. Elementi comuni della finestra

Elemento finestra	Descrizione
Barra del titolo	Contiene il nome dell'applicazione (Activity Editor), la relativa icone ed il nome dell'attività principale.
Menu	Contiene i menu principali (solo in modalità Progettazione).
Barra strumenti	Contiene le barre strumenti con gli accessi rapidi a varie funzioni e strumenti (solo modalità Progettazione).
Area di visualizzazione documento	Visualizza la rappresentazione della definizione dell'attività. E' organizzata con l'aspetto di un quaderno di esercitazione.
Barra di stato	Visualizza le informazioni sullo stato ed alcuni comodi accessi rapidi.

Utilizzo della vista Grafica

Se Map Designer Express apre Activity Editor con una definizione di attività di natura grafica, Activity Editor visualizzerà la definizione dell'attività nella vista Grafica in una delle due modalità disponibili: modalità Progettazione e modalità Vista rapida.

- **Modalità Progettazione:** In modalità Progettazione, Activity Editor assomiglia ad una normale applicazione; oltre alla principale finestra di modifica, dispone di una barra menu, di barre strumenti e delle finestre Libreria, Contenuto e Proprietà, che supportano quanto necessita alle modifiche durante la fase di progettazione della definizione dell'attività.

  La Figura 47 mostra una vista Grafica in modalità Progettazione.

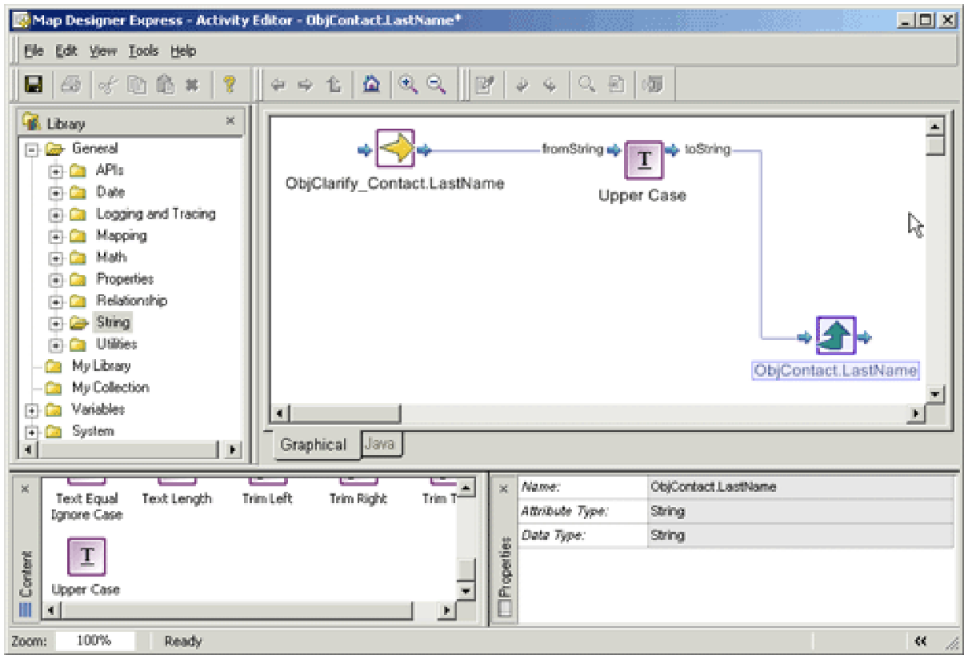


Figura 47. Vista Grafica in modalità Progettazione

La vista Grafica ha quattro finestre principali: Quaderno attività, Libreria, Contenuto e Proprietà.

– Finestra Quaderno attività--Questa finestra è l'area di modifica dell'attività principale e di solito viene detta Area di disegno di modifica. E' nota anche come area di disegno attività o area di disegno grafica. Questa è l'area in cui si trascinano i blocchi delle funzioni.
– Finestra Libraria--Questa finestra contiene una vista ad albero dei blocchi funzione disponibili e facoltativamente, i gruppi denominati. I blocchi funzione vengono sistemati in cartelle in base al loro scopo (consultare "Identificazione dei blocchi funzione supportati" a pagina 123) ed è possibile espanderli per mostrare gli effettivi blocchi funzione. E' anche possibile visualizzare i blocchi funzione come icone nella finestra Contenuto.

  Inoltre, la finestra Libreria contiene le seguenti cartelle:

  - Sistema--Questa cartella contiene gli elementi del sistema che possono essere aggiunti all'area di disegno. Gli elementi di sistema includono commenti, descrizioni, etichette, tag attività e costanti.
  - Libreria personale--Questa cartella consente di personalizzare la finestra Libreria. Contiene gli eventuali blocchi funzione definiti dall'utente che sono stati specificati nella vista Impostazioni attività in System Manager. Questa cartella contiene anche i blocchi funzione dei servizi Web che sono stati esportati da System Manager.
  - Raccolta personale--Questa cartella consente di creare una raccolta dei componenti che vengono utilizzati più spesso. In questa cartella è possibile collocare i normali blocchi funzione oppure è possibile creare il proprio gruppo di componenti riutilizzabili. Per ulteriori informazioni, consultare "Passi per la definizione di blocchi di gruppi di attività" a pagina 122.
  - Variabili--Questa cartella contiene le variabili globali accessibili all'attività corrente. Di solito contiene le variabili oggetto business della porta, tutti gli altri oggetti business e le variabili definite nell'attività, e la variabile globale `cwExecCtx`.

– Finestra Contenuto--Questa finestra contiene un ampio elenco di icone relative ai blocchi funzioni disponibili nella cartella al momento selezionata nella finestra Libreria. E' possibile selezionare un blocco funzione per visualizzarne la descrizione e le proprietà nella finestra Proprietà oppure trascinare un blocco funzione nell'area di disegno di modifica per creare parte del flusso di attività.
– Finestra Proprietà--Questa finestra visualizza con un layout a griglia le proprietà del blocco funzione al momento selezionato. Alcune proprietà sono modificabili; altre sono in sola lettura.

• **Modalità Vista rapida:** Nella modalità Vista rapida, Activity Editor visualizza solo l'area di disegno principale; tutte le altre finestre di supporto (Libreria, Contenuto e Proprietà), la barra menu e le barre strumenti sono nascoste.

  La Figura 48 mostra la vista Grafica in modalità Vista rapida.

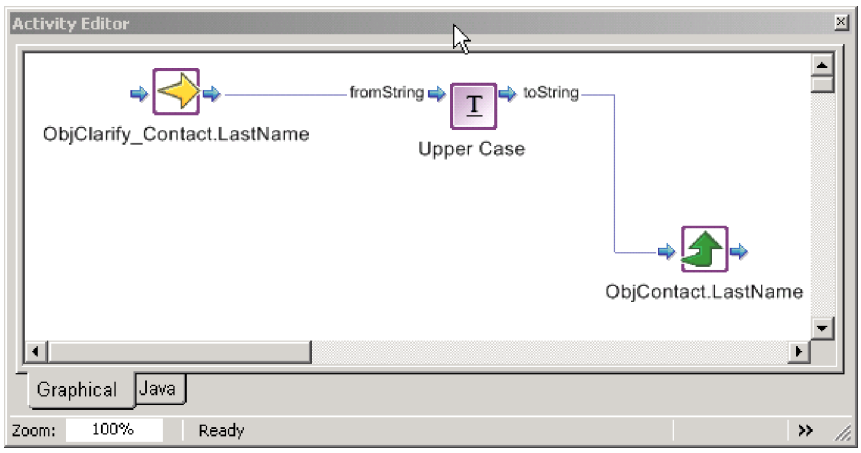


Figura 48. Vista Grafica in modalità Vista rapida

Inizialmente, quando viene aperta una definizione di attività di natura grafica, Activity Editor visualizza in una finestra a schede la vista di livello superiore della definizione. All'interno della finestra a scheda si trova l'*area di disegno*. Per informazioni sull'utilizzo delle definizioni di attività dell'area di disegno, consultare "Operazioni con le definizioni di attività" a pagina 118.

Utilizzo della vista Java

Se Map Designer Express apre Activity Editor con una definizione di attività che contiene solo codice Java personalizzato, Activity Editor visualizza la definizione dell'attività nella vista Java. Come per la vista Grafica, Activity Editor è disponibile nella vista Java in due modalità: modalità Progettazione e modalità Vista rapida.

- **Modalità Progettazione:** In modalità Progettazione, la vista Java di Activity Editor contiene WordPad Java per visualizzare e modificare il codice Java personalizzato, per fornire la definizione all'attività. WordPad è contenuto in un'are di una finestra a schede. Oltre alle normali opzioni di modifica di WordPad (Taglia, Copia, Incolla, Elimina, Seleziona tutto, Annulla, Ripeti), WordPad Java fornisce l'evidenziazione della sintassi per il linguaggio di programmazione Java.

  Per impostazione predefinita, i commenti sono verdi, le stringhe di lettere sono rosa e le parole chiave sono blu.

  Suggerimento: è possibile personalizzare gli schemi di evidenziazione della sintassi nella finestra di dialogo Preferenze.

  La Figura 49 mostra la vista Java in modalità Progettazione.

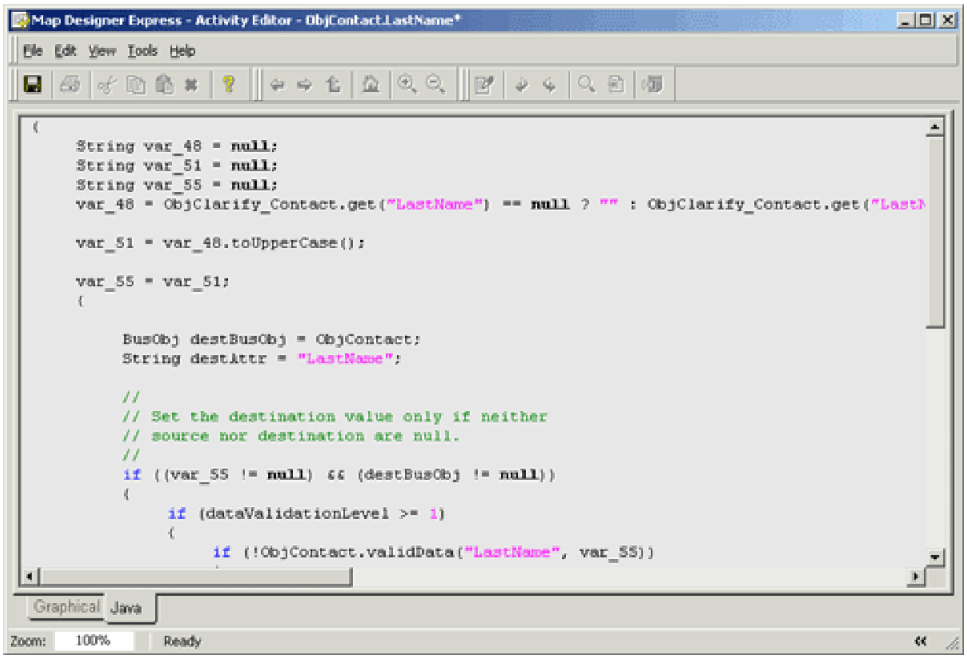


Figura 49. Vista Java in modalità Progettazione

- **Modalità Vista rapida:** In modalità Vista rapida, la vista Java visualizza soltanto WordPad. La Figura 50 mostra la vista Java in modalità Vista rapida.

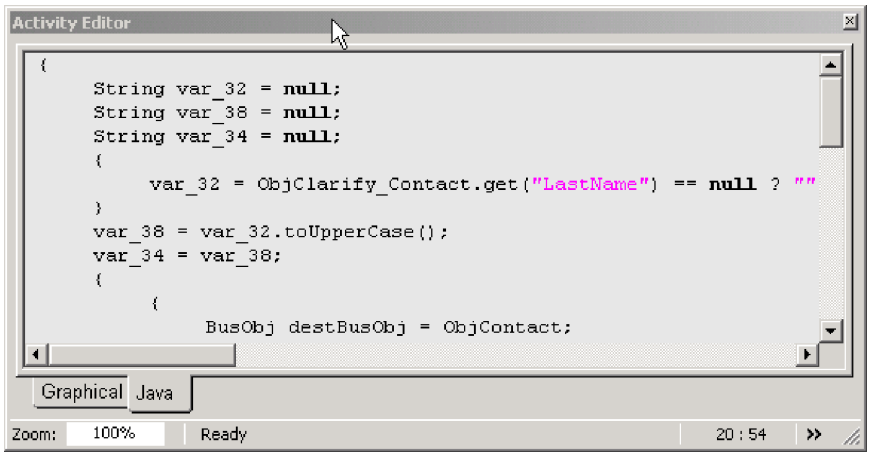


Figura 50. Vista Java in modalità Vista rapida

Suggerimento: per passare dalla modalità Vista rapida alla modalità Progettazione fare clic sul pulsante >> nella barra di stato. Se non si riesce a vedere il pulsante >>, modificare orizzontalmente le dimensioni della finestra Vista rapida finché non appare il pulsante.

Nota: Inizialmente la vista Java sarà solo in modalità di sola lettura. Per inserire del codice Java personalizzato fare clic sul pulsante Modifica codice presente nella barra strumenti oppure selezionare Modifica codice dal menu Strumenti.

Utilizzo delle funzioni di Activity Editor

E' possibile accedere alle funzioni di Activity Editor tramite:

- i menu a discesa
- il menu contestuale
- i pulsanti della barra strumenti
- gli accessi rapidi della tastiera

Menu a discesa di Activity Editor ed accessi rapidi della tastiera

Activity Editor fornisce i seguenti menu a discesa:

- menu File
- menu Modifica
- menu Visualizza
- menu Strumenti
- menu Guida

Le sezioni riportate di seguito descrivono le opzioni di ciascun menu ed i relativi accessi rapidi associati.

Funzioni del menu File: Il menu a discesa File di Activity Editor fornisce le seguenti opzioni:

- Salva [Ctrl+S]--Salva l'attività in Map Designer Express.
- Imposta stampante [Ctrl+Maiusc+P]--Apre la finestra di dialogo Imposta stampante per specificare le opzioni di stampa.
- Anteprima di stampa--Cambia Activity Editor alla modalità di anteprima di stampa.
- Stampa [Ctrl+P]--Apre la finestra di dialogo Stampa per stampare l'attività corrente.
- Chiudi--Chiude Activity Editor.

Funzioni del menu Modifica: Il menu a discesa Modifica di Activity Editor fornisce le seguenti opzioni:

- Annulla [Ctrl+A]--Annulla l'ultima modifica apportata e ripristina la versione precedente.
- Ripeti [Ctrl+Y]--Ripristina una modifica che era stata precedentemente rimossa con il comando Annulla.
- Taglia [Ctrl+X]--Elimina la voce selezionata e la copia in Appunti.
- Copia [Ctrl+C]--Copia la voce selezionata in Appunti.
- Incolla [Ctrl+P]--Incolla nella posizione del cursore l'oggetto contenuto in Appunti, se è compatibile.
- Elimina [Canc]--Elimina la voce selezionata.
- Seleziona tutto [Ctrl+A]--Seleziona tutte le voci.
- Cerca [Crtl+F]--Trova il testo specifico nell'area di modifica.
- Sostituisci [Ctrl+H]--Sostituisce del testo specifico con un testo diverso nell'area di modifica.
- Vai alla riga [Ctrl+G]--Sposta il cursore ad una riga specifica.

Funzioni del menu Visualizza: Il menu a discesa Visualizza di Activity Editor fornisce le seguenti opzioni:

- Modalità Progettazione--Alterna le modalità Progettazione e Vista rapida. (Viene abilitata solo una modalità alla volta).
- Modalità Vista rapida--Alterna le modalità Vista rapida e Progettazione. (Viene abilitata solo una modalità alla volta).
- Vai a--Fornisce le seguenti opzioni:
  - Indietro [Alt+Freccia sinistra]--Si sposta indietro nella cronologia di esplorazione nella vista Grafica.
  - Avanti [Alt+Freccia destra]--Si sposta avanti nella cronologia di esplorazione nella vista Grafica.
  - Livello superiore--Visualizza il diagramma superiore di un livello.
  - Inizio [Alt+Home]--Va al diagramma principale nella vista Grafica.
- Zoom avanti [Ctrl++]--Ingrandisce il contenuto in Activity Editor.
- Zoom indietro [Ctrl+-]--Riduce il contenuto in Activity Editor.
- Zoom [Crtl+M]--Apre la finestra di dialogo Zoom per specificare un determinato livello di zoom.
- Finestra Libreria--Attiva e disattiva la finestra Libreria.
- Finestra Contenuto--Attiva e disattiva la finestra Contenuto.
- Finestra Proprietà--Attiva e disattiva la finestra Proprietà.
- Barre degli strumenti--Apre un menu secondario per la visualizzazione delle barre strumenti (Standard, Grafici e Java) che alternativamente si attivano e si disattivano.
- Barra di stato--Attiva e disattiva la barra di stato.
- Preferenze... {Ctrl+U]--Apre la finestra di dialogo Preferenze per specificare il funzionamento predefinito di Activity Editor.

Funzioni del menu Strumenti: Il menu a discesa Strumenti di Activity Editor fornisce le seguenti opzioni:

- Converti [Ctrl+T]--Converte l'attività corrente in codice Java ed apre la vista Java.
- Modifica codice--Consente di modificare il codice in Java.
- Controlla delimitatori non corrispondenti--Verifica la presenza di delimitatori spaiati nel codice Java.
- Generatore di espressioni--Apre il programma di utilità Generatore di espressioni.

Funzioni del menu Guida: Il menu a discesa Guida di Activity Editor fornisce le seguenti opzioni:

- Argomenti della Guida [F1]--Apre gli argomenti della Guida sensibili al contesto
- Documentazione--Apre la documentazione InterChange Server Express .

Menu contestuale

Activity Editor fornisce anche un menu contestuale per eseguire molte attività nell'area di disegno. Per accedere al menu contestuale, fare clic con il pulsante destro del mouse sull'area di disegno. Il menu contestuale fornisce le seguenti opzioni:

- Nuova costante--Crea un nuovo contenitore Costante nell'area di disegno.
- Aggiungi etichetta--Crea un nuovo componente etichetta nell'area di disegno.
- Aggiungi descrizione--Crea un nuovo componente descrizione nell'area di disegno.

- Aggiungi commento--Crea un nuovo componente commento nell'area di disegno.
- Aggiungi attività--Crea un nuovo componente promemoria nell'attività.
- Aggiungi a Raccolta personale--Crea un nuovo componente al gruppo da riutilizzare nella finestra Libreria.

Barre degli strumenti di Activity Editor

Activity Editor fornisce tre barre degli strumenti per le comuni attività da eseguire:

- barra degli strumenti Standard
- barra degli strumenti Grafici
- barra degli strumenti Java

Le funzioni dei pulsanti della barra degli strumenti sono le stesse delle corrispondenti voci dei menu.

Suggerimento: per identificare la funzione di ogni pulsante della barra degli strumenti, passare con il cursore del mouse su ciascun pulsante.

Barra degli strumenti Standard: la Figura 51 mostra la barra degli strumenti Standard.

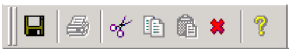


Figura 51. Barra degli strumenti Standard di Activity Editor

La Tabella 27 indica la funzione di ciascun pulsante della barra degli strumenti Standard (da sinistra a destra) ed il comando di menu corrispondente.

Tabella 27. Funzioni dei pulsanti della barra degli strumenti Standard

Funzione	Comando di menu corrispondente
Salva attività	File > Salva
Stampa attività	File > Stampa
Taglia	Modifica > Taglia
Copia	Modifica > Copia
Incolla	Modifica > Incolla
Elimina	Modifica > Elimina
Guida	Guida > Argomenti della Guida

barra degli strumenti Grafici: la Figura 52 mostra la barra strumenti Grafici.

Figura 52. Barra degli strumenti Grafici di Activity Editor

La Tabella 28 indica la funzione di ciascun pulsante della barra degli strumenti Grafici (da sinistra a destra) ed il comando di menu corrispondente.

Tabella 28. Funzioni dei pulsanti della barra degli strumenti Grafici

Funzione	Comando di menu corrispondente
Indietro	Visualizza > Vai a > Indietro
Avanti	Visualizza > Vai a > Avanti
Livello superiore	Visualizza > Vai a > Livello superiore

Tabella 28. Funzioni dei pulsanti della barra degli strumenti Grafici (Continua)

Funzione	Comando di menu corrispondente
Inizio	Visualizza > Vai a > Inizio
Zoom avanti	Visualizza > Zoom avanti
Zoom indietro	Visualizza > Zoom indietro

La Figura 53 mostra la barra degli strumenti Java.

Figura 53. Barra degli strumenti Java di Activity

La Tabella 29 indica la funzione di ciascun pulsante della barra degli strumenti Java (da sinistra a destra) ed il comando di menu corrispondente.

Tabella 29. Pulsanti della barra degli strumenti Java

Funzione	Comando di menu corrispondente
Modifica codice	Strumenti > Modifica codice
Annulla	Modifica > Annulla
Ripeti	Modifica > Ripeti
Cerca testo	Modifica > Cerca
Vai alla riga	Modifica > Vai alla riga
Generatore di espressioni	Strumenti > Generatore di espressioni

Elementi della barra di stato: Activity Editor fornisce anche una barra di stato, come illustrato nella Figura 54.

Figura 54. Barra di stato di Activity Editor

La Tabella 30 indica la funzione di ciascun elemento della barra di stato, da sinistra a destra.

Tabella 30. Funzioni degli elementi della barra di stato

Elemento	Funzione
Zoom: 100%	Modificare la casella per specificare una percentuale di zoom
Pronto	Messaggio dello stato
10.9	Pannello di esplorazione che indica la posizione corrente della *I-bar* nell'editor Java
>> (visualizzato nella modalità Vista rapida)	Alterna fra le modalità Vista rapida e Progettazione
<< (visualizzato nella modalità Progettazione)	

Operazioni con le definizioni di attività

Activity Editor viene utilizzato per definire e modificare le definizioni di attività per le regole di trasformazione. Queste operazioni vengono effettuate nell'area di disegno tramite i relativi componenti: blocchi funzione, collegamenti di connessione e l'icona Nuova costante.

Utilizzo dei blocchi funzione

Una definizione di attività viene generata con i *blocchi funzione*, che rappresentano discrete parti della definizione di un'attività, ad esempio una costante, una variabile o un metodo di programmazione. Molti dei blocchi funzione in Activity Editor corrispondono a dei singoli metodi nella Mappatura API.

E' possibile collocare i blocchi funzione nell'area di disegno trascinandoli dalle finestre Libreria o Contenuto. Una volta collocato un blocco funzione nell'area di disegno, è possibile spostarlo facendo clic per selezionarlo e trascinandolo nella posizione desiderata.

I blocchi funzione possono avere degli input, degli output o entrambi. Gli input e gli output di ciascun blocco funzione sono predefiniti ed accettano solo il tipo di valore specificato. Quando si trascina il blocco funzione nell'area di disegno, le relative porte di input e output vengono rappresentate da frecce. Queste porte servono da punti di connessione per i collegamenti fra il blocco funzione e gli altri componenti. Per impostazione predefinita, il nome di ogni input e output viene visualizzato accanto alla relativa porta di connessione (per nascondere i nomi utilizzare l'opzione Visualizza > Preferenze).

Per una descrizione dei blocchi funzione supportati nei contesti Map Designer Express e Relationship Designer Express, consultare "Identificazione dei blocchi funzione supportati" a pagina 123.

Nota: Oltre ai blocchi funzione standard forniti da Activity Editor, è possibile esportare i servizi Web da System Manager in Activity Editor. Il processo di esportazione converte ciascun metodo contenuto nel servizio Web in un blocco funzione, che è poi possibile utilizzare per le definizioni di attività allo stesso modo degli altri blocchi funzione. Per ulteriori informazioni, consultare "Esportazione di servizi Web in Activity Editor" a pagina 166.

E' possibile anche importare la propria libreria Java, da utilizzare come blocchi funzione in Activity Editor. L'importazione delle librerie Jar personalizzate nelle impostazioni delle attività abilita qualsiasi metodo pubblico contenuto nella libreria Jar ad essere utilizzato come blocco funzione in Activity Editor. Per ulteriori informazioni, consultare "Importazione di pacchetti Java e di altro codice personalizzato" a pagina 172.

Suggerimento: Utilizzo dei blocchi funzione direttamente in Map Designer Express

Se si desidera utilizzare solo un blocco funzione standard in una trasformazione personalizzata, è possibile configurare il blocco funzione nella finestra di dialogo Preferenze in modo da poter essere utilizzato direttamente in Map Designer Express. Quindi, dopo aver selezionato gli attributi di origine e di destinazione per la trasformazione personalizzata, è possibile selezionare il blocco funzione configurato nella casella combinata della regola di trasformazione in Personalizza di Map Designer Express.

Passi per utilizzare i blocchi funzione direttamente in Map Designer Express:
Per impostare l'utilizzo diretto dei blocchi funzione in Map Designer Express, effettuare le seguenti operazioni:

1. Avviare Map Designer Express. Per informazioni sull'avvio di Map Designer Express, consultare "Avvio di Map Designer Express" a pagina 16

2. Dal menu Visualizza, selezionare Preferenze oppure utilizzare l'accesso rapido `Ctrl+U` dalla tastiera.

   Risultato: viene visualizzata la finestra di dialogo Preferenze.
3. Nella finestra di dialogo Preferenze selezionare la scheda Mappatura personalizzata.
4. Dall'elenco di blocchi funzione standard selezionare i blocchi funzione da utilizzare direttamente in Map Designer Express.

   La Figura 55 mostra la scheda Mappatura personalizzata con i blocchi funzione selezionati.

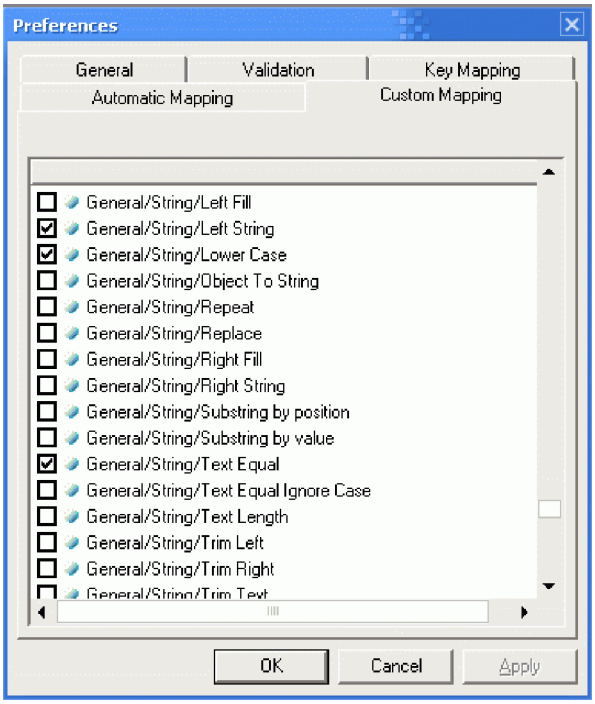


Figura 55. Finestra Preferenze con visualizzata la scheda Mappatura personalizzata

5. Fare clic su OK.

Risultato: i blocchi funzione configurati saranno disponibili per essere utilizzati direttamente nella casella combinata della trasformazione in Personalizza di Map Designer Express.

Utilizzo dei collegamenti di connessione

I blocchi funzione sono collegati da dei *collegamenti di connessione*. I collegamenti di connessione definiscono il flusso di attività fra i vari componenti della definizione di attività. Collegano la porta dell'output di un blocco funzione alla porta di input di un altro blocco funzione.

Nota: Le porte in uscita possono collegarsi a più collegamenti di connessione ma le porte in entrata possono collegarsi ad uno solo.

Suggerimento: quando si effettua un trascinamento per collegare due blocchi funzione, Activity Editor utilizza l'opzione impostata nella scheda Convalida della

finestra di dialogo Preferenze per determinare se è necessario effettuare la convalida e verifica se il tipo di parametro ″Da″ è lo stesso del tipo di parametro ″A″.

- Per impostazione predefinita questa preferenza è impostata su ″Avvertenza″, vale a dire che quando si crea un collegamento fra due parametri di tipo diverso, Activity Editor visualizzerà un messaggio che informa che ciò potrebbe portare ad un errore di compilazione.
- Impostando l'opzione su ″Ignora″, Activity Editor non effettua alcuna convalida.
- Impostando l'opzione su ″Errore″, Activity Editor non consente la creazione di collegamenti fra tipi diversi.

Esempio: per specificare che l'output del blocco funzione A deve andare all'input del blocco funzione B, procedere come segue:

1. Fare clic e tenere premuto il pulsante sinistro del mouse sulla porta in uscita del blocco funzione A.
2. Tenendo sempre premuto il pulsante sinistro del mouse, spostare il cursore alla porta in entrata del blocco funzione B.
3. Rilasciare il pulsante sinistro del mouse.

Risultato: il collegamento di connessione viene collocato fra la porta in uscita del blocco funzione A e la porta in entrata del blocco funzione B. Graficamente, il collegamento di connessione viene visualizzato come una riga angolata a destra fra i componenti. Se la porta in entrata del blocco funzione B è già collegata ad un altro collegamento di connessione, la nuova connessione sostituirà quella esistente.

Utilizzo di tag etichetta, descrizione, commento e attività

La cartella Sistema (ubicata nelle finestre Libreria e Contenuto) contiene i blocchi funzione per aggiungere *etichette*, *descrizioni*, *commenti*, e *attività* alle definizioni di attività. Queste tag facilitano l'identificazione di ciascuna attività e attività secondaria oppure servono da promemoria per qualcosa che deve essere eseguito. Questi blocchi funzione vengono trascinati nell'area di disegno, come qualsiasi altro blocco funzione. Non esistono, tuttavia, porte di input e di output.

Per creare una nuova tag, fare clic al centro della tag. I cursore cambia in un I-beam e si può immettere il testo. Le tag mandano a capo il testo automaticamente, quando le righe sono troppo lunghe. Se si desidera iniziare una riga nuova, premere Invio.

Per modificare le dimensioni di una tag, fare clic con il pulsante sinistro del mouse nell'angolo inferiore destro e tenendo premuto trascinare la tag alla dimensione desiderata.

La Figura 56 mostra come modificare una tag etichetta ed inserire più righe di testo.

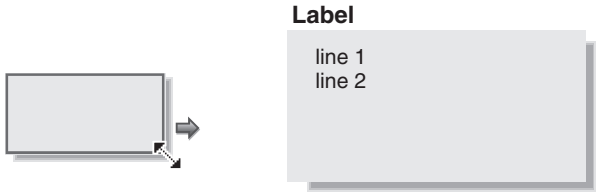


Figura 56. Modifica dimensione di un'etichetta ed immissione di più righe di testo

Limitazione: ognuno di questi componenti di modifica ha una dimensione minima ed i componenti non possono essere resi più piccoli di una determinata dimensione.

Per spostare la tag all'interno dell'area di disegno, fare clic sul margine del componente e trascinarlo.

Utilizzo del blocco funzione Nuova costante

Activity Editor dispone di un *blocco funzione Nuova costante* che si può trascinare dell'area di disegno per definire un valore di costante da impostare ed utilizzare come input per altri blocchi funzione. Il blocco funzione Nuova costante si trova nella cartella Sistema nella finestre Libreria e Contenuto. Activity Editor visualizza una casella di modifica testo sopra l'icona del blocco funzione per poter inserire il valore della costante. Per rivedere questo valore, fare doppio clic sull'icona Costante ed immettere il nuovo valore. Le costanti contengono una porta di uscita.

Nota: La costante è l'unico componente della definizione di attività che accetta soltanto una riga per il valore. Ciò accade perché la costante viene convertita in una stringa di codice Java ed il sistema non può convertire un valore di costante su più righe. Se è richiesto input su più righe, utilizzare il valore ″\n″ per effettuare una separazione delle righe nella costante.

Esempio: il valore ″line1\nline2″ indicherà al sistema di creare l'output del testo su due righe.

Passi per la definizione di blocchi di gruppi di attività

Una volta definito un flusso di attività con una serie di blocchi funzione nell'area di disegno, è possibile selezionarlo e salvarlo come gruppo di attività denominato, per poterlo poi riutilizzare in un'altra definizione di attività. Il gruppo di attività salvato viene rappresentato da un'icona. La seguente procedura descrive le operazioni da effettuare.

Prima di iniziare: è necessario abilitare ″Mostra funzioni secondarie nella finestra Libreria″ nella finestra di dialogo Preferenze per visualizzare il gruppo aggiunto.

Effettuare le seguenti operazioni:

1. Selezionare i componenti dell'attività che si desidera raggruppare nell'area di disegno. Per selezionare più componenti, tenere premuto il tasto `Ctrl` e fare clic su ciascun componente.
2. Fare clic con il pulsante destro del mouse sull'area di disegno per aprire il menu contestuale. Quindi selezionare Aggiungi a raccolta personale. In alternativa, fare clic con il pulsante destro del mouse sul componente e selezionare Aggiungi a Raccolta personale.

   Risultato: viene visualizzata la finestra di dialogo Aggiungi a Raccolta personale.
3. Nella finestra di dialogo Aggiungi a Raccolta personale, immettere un nome ed una descrizione del blocco gruppo di attività e selezionare un'icona che lo rappresenti. Fare clic su OK.

   Risultato: il blocco gruppo di attività viene aggiunto alla cartella Raccolta personale nelle finestre Libreria e Contenuto. E' possibile trascinare l'icona nell'area di disegno di qualsiasi definizione di attività.

Nota: Qualsiasi parametro di input o output non connesso quando viene salvato il gruppo utente, verrà visualizzato come input/output del gruppo di attività.

Esempio: la Figura 57 mostra un'attività in cui i componenti grafici racchiusi nella casella vengono salvati come un gruppo di attività.

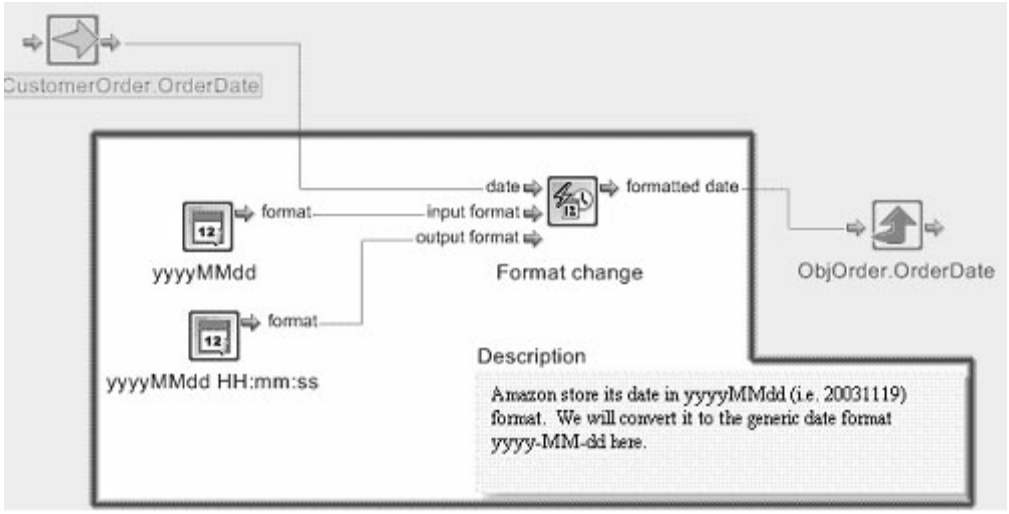


Figura 57. Componenti grafici salvati come un'unica attività

Quando questo gruppo di attività viene riutilizzato, disporrà di un'icona, come illustrato nella Figura 58.

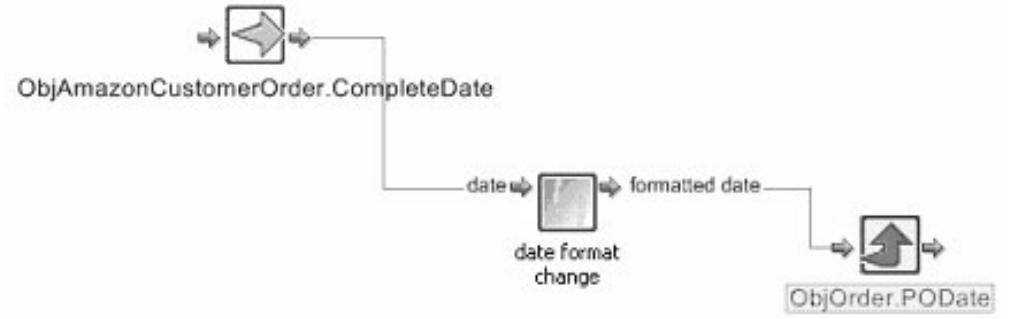


Figura 58. Gruppo di attività riutilizzato rappresentato come icona

Suggerimento: facendo doppio clic su questa icona gruppo verranno visualizzati i dettagli della definizione del gruppo.

Identificazione dei blocchi funzione supportati

I blocchi funzione supportati, nel contesto Map Designer Express, sono organizzati in categorie, descritte nella tabella riportata di seguito. Queste categorie corrispondono a cartella nelle finestre Libreria e Contenuto.

Tabella 31. Organizzazione dei blocchi funzione

Cartella del blocco funzione	Descrizione	Per ulteriori informazioni
Generale/API/Oggetto business	Blocchi funzione da utilizzare con gli oggetti business.	Tabella 32 a pagina 125
Generale/API/Oggetto business/Vettore	Blocchi funzione da utilizzare con i vettori Java nella classe BusObj.	Tabella 33 a pagina 129
Generale/API/Oggetto business/Costanti	Blocchi funzione da utilizzare con le costanti Java nella classe BusObj.	Tabella 34 a pagina 129
Generale/API/Vettore oggetto business	Blocchi funzione da utilizzare con i vettori dell'oggetto business.	Tabella 35 a pagina 129
Generale/API/Connessione database	Blocchi funzione per la creazione e la gestione di una connessione a database.	Tabella 36 a pagina 131
Generale/API/Relazione identità	Blocchi funzione da utilizzare con le relazioni d'identità.	Tabella 37 a pagina 132
Generale/API/Mappe	Blocchi funzione per effettuare query ed impostare valori runtime necessari per l'esecuzione di una mappa.	Tabella 38 a pagina 134
Generale/API\Mappe/ Costanti	Costanti dei blocchi funzione.	Tabella 39 a pagina 134
Generale/API/Mappe/ Eccezione	Blocchi funzione per la creazione di nuovi oggetti eccezione in una mappa.	Tabella 40 a pagina 135
Generale/API/Partecipante	Blocchi funzione per l'impostazione ed il richiamo di valori dei partecipanti nelle relazioni d'identità.	Tabella 41 a pagina 136
Generale/API/Partecipante/ Vettore	Blocchi funzione per la creazione e l'utilizzo di vettori di partecipanti.	Tabella 42 a pagina 138
Generale/API/Partecipante/ Costanti	Costanti dei blocchi funzione da utilizzare coi partecipanti.	Tabella 43 a pagina 138
Generale/API/Relazione	Blocchi funzione per la manipolazione di istanze runtime delle relazioni.	Tabella 44 a pagina 138
Generale/Data	Blocchi funzione da utilizzare con le date.	Tabella 45 a pagina 141
Generale/Data/Formati	Blocchi funzione per specificare formati di data diversi.	Tabella 46 a pagina 142
Generale/Registro e traccia	Blocchi funzione per la gestione dei messaggi log e di traccia.	Tabella 47 a pagina 142
Generale/Registro e traccia/Errore log	Blocchi funzione per la formattazione dei messaggi di errore.	Tabella 48 a pagina 143

Tabella 31. Organizzazione dei blocchi funzione (Continua)

Cartella del blocco funzione	Descrizione	Per ulteriori informazioni
Generale/Registro e traccia/Informativo log	Blocchi funzione per la formattazione dei messaggi informativi.	Tabella 49 a pagina 143
Generale/Registro e traccia/Avviso log	Blocchi funzione per la formattazione dei messaggi di avvertenza.	Tabella 50 a pagina 144
Generale/Registro e traccia/Traccia	Blocchi funzione per la formattazione dei messaggi di traccia.	Tabella 51 a pagina 145
Generale/Mappatura	Blocchi funzione per l'esecuzione di mappe in un contesto specifico.	Tabella 52 a pagina 145
Generale/Matematica	Blocchi funzione per la attività matematiche di base.	Tabella 53 a pagina 146
Generale/Proprietà	Blocchi funzione per il richiamo dei valori delle proprietà di configurazione.	Tabella 54 a pagina 148
Generale/Relazione	Blocchi funzione per la gestione e le query delle relazioni d'identità.	Tabella 55 a pagina 148
Generale/Stringa	Blocchi funzione per la manipolazione degli oggetti Stringa.	Tabella 56 a pagina 148
Generale/Utilità	Blocchi funzione per inviare e raccogliere eccezioni, per il loop, lo spostamento di attributi e l'impostazione di condizioni.	Tabella 57 a pagina 150
Generale\Utilità/Vettore	Blocchi funzione da utilizzare con gli oggetti Vettore.	Tabella 58 a pagina 151

Le seguenti tabelle descrivono i blocchi funzione di ciascuna categoria, includo i valori accettabili per i relativi input ed output.

Tabella 32. Generale/API/Oggetto business

Nome	Descrizione	Input e output con valori accettabili
Copia	Copia tutti i valori attributo dall'oggetto business di input. API: BusObj.copy()	Input: • copia in--BusObj • copia da--BusObj
Duplicato	Crea un oggetto business esattamente uguale all'originale. API: BusObj.duplicate()	Input:originale--BusObj Output: duplicato--BusObj
Chiavi uguali	Confronta i valori dell'oggetto business 1 con quelli dell'oggetto business 2, per stabilire se sono uguali. API: BusObj.equalKeys()	Input: • oggetto business 1--BusObj • oggetto business 2--BusObj Output: valori chiave uguali?-- boolean

Tabella 32. Generale/API/Oggetto business (Continua)

Nome	Descrizione	Input e output con valori accettabili
Uguale	Confronta i valori dell'oggetto business 1 con quelli dell'oggetto business 2, inclusi gli oggetti business secondari, per stabilire se sono uguali. API: BusObj.equals()	Input: • oggetto business 1--BusObj • oggetto business 2--BusObj Output: uguale?-- boolean
Esiste	Verifica l'esistenza di un attributo dell'oggetto business con un nome specificato. API: BusObj.exists()	Input: • oggetto business--BusObj • attributo--String Output: esiste?-- boolean
Ottieni valore booleano	Richiama il valore di un singolo attributo da un oggetto business, come booleano. API: BusObj.getBoolean()	Input: • oggetto business--BusObj • attributo--String Output: valore-- boolean
Ottieni oggetto business	Richiama il valore di un singolo da un oggetto business, come BusObj. API: BusObj.getBusObj()	Input: • oggetto business--BusObj • attributo--String Output: valore--BusObj
Ottieni vettore oggetto business	Richiama il valore di un singolo attributo da un oggetto business, come vettore oggetto business (BusObj). API: BusObj.getBusObjArray()	Input: • oggetto business--BusObj • attributo--String Output: valore--BusObjArray
Ottieni tipo di oggetto business	Richiama il nome della definizione oggetto business su cui è stato basato questo oggetto business. API: BusObj.getType()	Input: oggetto business--BusObj Output: tipo--String
Ottieni virgolette	Richiama il valore di un attributo singolo da un oggetto business, come un doppio (double). API: BusObj.getDouble()	Input: • oggetto business--BusObj • attributo--String Output: valore--double
Ottieni mobile	Richiama il valore di un attributo singolo da un oggetto business, come mobile (float). API: BusObj.getFloat()	Input: • oggetto business--BusObj • attributo--String Output: valore--float
Ottieni numero intero	Richiama il valore di un singolo attributo da un oggetto business, come numero intero (int). API: BusObj.getInt()	Input: • oggetto business--BusObj • attributo--String Output: valore--int
Ottieni lungo	Richiama il valore di un attributo singolo da un oggetto business, come lungo (long). API: BusObj.getLong()	Input: • oggetto business--BusObj • attributo--String Output: valore--long

Tabella 32. Generale/API/Oggetto business (Continua)

Nome	Descrizione	Input e output con valori accettabili
Ottieni testo lungo	Richiama il valore di un attributo singolo da un oggetto business, come testo lungo (text long). API: BusObj.getLongText()	Input: • oggetto business--BusObj • attributo--String Output: valore--String
Ottieni oggetto	Richiama il valore di un singolo attributo da un oggetto business, come oggetto. L'attributo può essere specificato come nome attributo o come posizione dell'attributo. API: BusObj.get()	Input: • oggetto business--BusObj • attributo--String, int Output: valore--Object
Ottieni stringa	Richiama il valore di un attributo singolo da un oggetto business, come stringa. API: BusObj.getString()	Input: • oggetto business--BusObj • attributo--String Output: valore--String
Ottieni verbo	Richiama l'istruzione (verb) dell'oggetto business. API: BusObj.getVerb()	Input: oggetto business--BusObj Output: verbo--String
E' vuoto	Rileva se il valore di un attributo è impostato su una stringa di lunghezza zero. API: BusObj.isBlank()	Input: • oggetto business--BusObj • attributo--String Output: vuoto?--boolean
E' oggetto business	Verifica se il valore è un oggetto business (BusObj).	Input: valore--Object Output: risultato--boolean
E' chiave	Rileva se l'attributo di un oggetto business è definito come attributo chiave. API: BusObj.isKey()	Input: • oggetto business--BusObj • attributo--String Output: chiave?--boolean
E' nullo	Rileva se il valore di un attributo dell'oggetto business è null. API: BusObj.isNull()	Input: • oggetto business--BusObj • attributo--String Output: null?--boolean
Richiesto	Rileva se l'attributo di un oggetto business è definito come attributo richiesto. API: BusObj.isRequired()	Input: • oggetto business--BusObj • attributo--String Output: richiesto?--boolean
Itera secondari	Scorre il vettore oggetto business secondario.	Input: • oggetto business--BusObj • attributo--String • indice corrente--int • elemento corrente--BusObj

Tabella 32. Generale/API/Oggetto business (Continua)

Nome	Descrizione	Input e output con valori accettabili
Chiave a stringa	Richiama i valori degli attributi di chiave primaria dell'oggetto business come stringa. API: BusObj.keysToString()	Input: oggetto business--BusObj Output: stringa chiave--String
Nuovo oggetto business	Crea una nuova istanza dell'oggetto business (BusObj) del tipo specificato. API: Collaboration.BusObj()	Input: tipo--String Output: oggetto business--BusObj
Imposta contenuto	Imposta il contenuto dell'oggetto business per un altro oggetto business. I due oggetti business disporranno del contenuto insieme. Le modifiche apportate ad uno dei due oggetti business verranno riflesse nell'altro. API: BusObj.setContent()	Input: • oggetto business--BusObj • contenuto--BusObj
Imposta valori predefiniti attributo	Imposta tutti gli attributi sui valori predefiniti. API: BusObj.setDefaultAttrValues()	Input: oggetto business--BusObj
Imposta chiavi	Imposta i valori ″a″ degli attributi chiave dell'oggetto business sui valori degli attributi chiave nell'oggetto business ″da″. API: BusObj.setKeys()	Input: • da oggetto business--BusObj • a oggetto business--BusObj
Imposta valore con Crea	Imposta l'attributo dell'oggetto business su un valore specificato per un tipo di dati particolare, creando un oggetto per il valore, se non esiste già. API: BusObj.setWithCreate()	Input: • oggetto business--BusObj • attributo--String • valore--BusObj, BusObjArray, Object
Imposta istruzione	Imposta l'istruzione (verb) di un oggetto business. API: BusObj.setVerb()	Input: • oggetto business--BusObj • verbo--String
Imposta verbo con Crea	Imposta l'istruzione (verb) di un oggetto business secondario, creando l'oggetto business secondario, se non esiste già. API: BusObj.setVerbWithCreate()	Input: • oggetto business--BusObj • attributo--String • verbo--String
Imposta valore	Imposta un attributo dell'oggetto business su un valore specificato per un tipo di dati particolare. API: BusObj.set()	Input: • oggetto business--BusObj • attributo--String • valore--boolean, double, float, int, long, Object, String, BusObj
Shallow uguale	Confronta i valori dell'oggetto business 1 con quelli dell'oggetto business 2, esclusi gli oggetti business secondari, per stabilire se sono uguali. API: BusObj.equalsShallow()	Input: • oggetto business 1--BusObj • oggetto business 2--BusObj Output: uguale?--boolean
In una stringa	Richiama i valori di tutti gli attributi di un oggetto business come stringa. API: BusObj.toString()	Input: oggetto business--BusObj Output: stringa--String

Tabella 32. Generale/API/Oggetto business (Continua)

Nome	Descrizione	Input e output con valori accettabili
Dati validi	Verifica se il valore specificato è di tipo valido per un determinato attributo. API: BusObj.validData()	Input: • oggetto business--BusObj • attributo--String • valore--Object, BusObj, BusObjArray, String, long, int, double, float, boolean Output: valido?--boolean

Tabella 33. Generale/API/Oggetto business/Vettore

Nome	Descrizione	Input e output con valori accettabili
Ottieni BusObj in	Richiama l'elemento nell'indice specificato nel vettore oggetto business.	Input: • vettore--BusObj[] • indice--int Output: oggetto business--BusObj
Nuovo vettore oggetto business	Crea un nuovo vettore oggetto business.	Input: dimensione--int Output: vettore--BusObj[]
Imposta BusObj a	Imposta l'elemento sull'indice specificato nel vettore oggetto business.	Input: • vettore--BusObj[] • indice--int • oggetto business--BusObj
Dimensione	Richiama la dimensione del vettore oggetto business	Input: vettore--BusObj[] Output: dimensione--int

Tabella 34. Generale/API/Oggetto business/Costanti

Nome	Descrizione	Input e output con valori accettabili
Verb: Create	Istruzione (verb) oggetto business ″Crea″.	Output: Crea--String
Verb: Delete	Istruzione (verb) oggetto business ″Elimina″.	Output: Elimina--String
Verb: Retrieve	Istruzione (verb) oggetto business ″Richiama″.	Output: Richiama--String
Verb: Update	Istruzione (verb) oggetto business ″Aggiorna″.	Output: Aggiorna--String

Tabella 35. Generale/API/Vettore oggetto business

Nome	Descrizione	Input e output con valori accettabili
Aggiungi elemento	Aggiunge un oggetto business a questo oggetto business API: BusObjArray.addElement()	Input: • vettore oggetto business--BusObjArray • elemento--BusObj
Duplicato	Crea un vettore oggetto business esattamente uguale all'originale. API: BusObjArray.duplicate()	Input: originale--BusObjArray Output: duplicato--BusObjArray
Uguale	Confronta i valori del vettore oggetto business 1 con quelli del vettore oggetto business 2, per stabilire se sono uguali. API: BusObjArray.equals()	Input: • vettore 1--BusObjArray • vettore 2--BusObjArray Output: uguale?-- boolean

Tabella 35. Generale/API/Vettore oggetto business (Continua)

Nome	Descrizione	Input e output con valori accettabili
Ottieni elemento	Richiama un singolo oggetto business specificandone la posizione nel vettore oggetto business. API: BusObjArray.elementAt()	Input: • vettore oggetto business--BusObjArray • indice--int Output: elemento--BusObj
Ottieni elementi	Richiama il contenuto del vettore oggetto business. API: BusObjArray.getElements()	Input: vettore oggetto business--BusObjArray Output: elemento--BusObj[]
Ottieni ultimo indice	Richiama l'ultimo indice disponibile da un vettore oggetto business. API: BusObjArray.getLast Index()	Input: vettore oggetto business--BusObjArray Output: ultimo indice--int
Valore corretto per Vettore oggetto business	Verifica se il valore è un vettore oggetto business (BusObjArray).	Input: valore--Object Output: risultato--boolean
Valore massimo attributo	Richiama i valori massimi dell'attributo specificato tra tutti gli elementi nel vettore oggetto business. API: BusObjArray.max()	Input: • vettore oggetto business--BusObjArray • attributo--String Output: massimo--String
Valore minimo attributo	Richiama il valore minimo dell'attributo specificato tra tutti gli elementi nel vettore oggetto business. API: BusObjArray.min()	Input: • vettore oggetto business--BusObjArray • attributo--String Output: minimo--String
Rimuovi tutti gli elementi	Rimuove tutti gli elementi dal vettore oggetto business. API: BusObjArray.removeAllElements()	Input: vettore oggetto business--BusObjArray
Rimuovi Elemento	Rimuove un elemento oggetto business da un vettore oggetto business. API: BusObjArray.removeElement()	Input: • vettore oggetto business--BusObjArray • elemento--BusObj
Rimuovi elemento a	Rimuove un elemento di una particolare posizione in questo vettore oggetto business. API: BusObjArray.removeElementAt()	Input: • vettore oggetto business--BusObjArray • indice--int
Imposta elemento a	Imposta il valore di un oggetto business nel vettore oggetto business. API: BusObjArray.setElementAt()	Input: • vettore oggetto business--BusObjArray • indice--int • elemento--BusObj
Dimensione	Ottiene il numero di elementi del vettore oggetto business. API: BusObjArray.size()	Input: vettore oggetto business--BusObjArray Output: dimensione--int
Somma	Aggiunge i valori dell'attributo specificato per tutti gli oggetti business in questo vettore oggetto business. API: BusObjArray.sum()	Input: • vettore oggetto business--BusObjArray • attributo--String Output: somma--double

Tabella 35. Generale/API/Vettore oggetto business (Continua)

Nome	Descrizione	Input e output con valori accettabili
Cambia	Inverte le posizioni dei due oggetti business nel vettore oggetto business. API: BusObjArray.swap()	Input: • vettore oggetto business--BusObjArray • indice 1--int • indice 2--int
Alla stringa	Richiama i valori del vettore oggetto business come singola stringa. API: BusObjArray.to String()	Input: vettore oggetto business--BusObjArray Output: stringa--String

Tabella 36. Generale/API/Connessione database

Nome	Descrizione	Input e output con valori accettabili
Inizia transazione	Inizia una transazione esplicita per la connessione corrente. API: CwDBConnection.beginTransaction()	Input: connessione database--CwDBConnection
Convalida	Esegue il commit della transazione attiva associata con la connessione corrente. API: CwDBConnection.commit()	Input: connessione database--CwDBConnection
Esegui SQL preparata	Esegue una query SQL preparata specificandone la sintassi. API: CwDBConnection.executePreparedSQL()	Input: • connessione database--CwDBConnection • query--String Output: uguale?-- boolean
Esegui SQL preparata con parametro	Esegue una query SQL preparata specificandone la sintassi con in parametri specificati. API:CwDBConnection.executePreparedSQL()	Input: • connessione database--CwDBConnection • query--String • parametri--java.util.Vector
Esegui SQL	Esegue una query SQL statica specificandone la sintassi. API: CwDBConnection.executeSQL()	Input: • connessione database--CwDBConnection • query--String
Esegui SQL con parametro	Esegue una query SQL statica specificandone la sintassi con i parametri specificati. API: CwDBConnection.executeSQL()	Input: • connessione database--CwDBConnection • query--String • parametri--java.util.Vector
Esegui procedura memorizzata	Esegue una procedura memorizzata SQL specificandone il nome ed il vettore di parametri. API: CwDBConnection.executeStored Procedure()	Input: • connessione database--CwDBConnection • query--String • parametri--java.util.Vector
Ottieni connessione database	Stabilisce una connessione ad un database e restituisce un oggetto CwDBConnection(). API: BaseDLM.getDBConnection() o BaseCollaboration.getDBConnection()	Input: nome pool connessione--String Output: connessione database--CwDBConnection

Tabella 36. Generale/API/Connessione database (Continua)

Nome	Descrizione	Input e output con valori accettabili
Ottieni connessione database con transazione	Stabilisce una connessione ad un database e restituisce un oggetto CwDBConnection(). API: BaseDLM.getDBConnection() o BaseCollaboration.getDBConnection()	Input: • nome pool connessione--String • transazione implicita--boolean Output: connessione database--CwDBConnection
Ottieni riga successiva	Richiama la riga successiva dal risultato della query. API: CwDBConnection.nextRow()	Input: connessione database--CwDBConnection Output: riga--java.util.Vector
Ottieni conteggio di aggiornamento	Richiama il numero di righe interessate dall'ultima operazione di scrittura nel database. API: CwDBConnection.getUpdateCount()	Input: connessione database--CwDBConnection Output: conteggio--int
Più righe presenti	Determina se nei risultati della query sono presenti altre righe da elaborare. API: CwDBConnection.hasMoreRows()	Input: connessione database--CwDBConnection Output: altre righe?--boolean
In transazione	Determina se nella connessione corrente è attiva una transazione. API: CwDBConnection.inTransaction()	Input: connessione database--CwDBConnection Output: in transazione?--boolean
E' attiva	Determina se la connessione corrente è attiva. API: CwDBConnection.isActive()	Input: connessione database--CwDBConnection Output: è attivo?--boolean
Rilascio	Rilascia la connessione corrente, restituendola al relativo pool di connessione. API: CwDBConnection.release()	Input: connessione database--CwDBConnection
Esegui Rollback	Esegue il rollback della transazione attiva associata alla connessione corrente. API: CwDBConnection.rollback()	Input: connessione database--CwDBConnection

Tabella 37. Generale/API/Relazione identità

Nome	Descrizione	Input e output con valori accettabili
Aggiungi elementi secondari personali	Aggiunge le istanze secondarie specificate ad una relazione principale/secondario di una relazione d'identità. API: IdentityRelationship.addMyChildren()	Input: • mappa--BaseDLM • parentChildRelDefName--String • parentParticipantDefName--String • parentBusObj--BusObj • childParticipantDefName--String • childBusObjList--BusObj,BusObjArray
Elimina tutti gli elementi secondari personali	Rimuove tutte le istanze secondarie da una relazione principale/secondario per una relazione d'identità che appartiene all'elemento principale specificato. API: IdentityRelationship.deleteMyChildren()	Input: • mappa--BaseDLM • parentChildRelDefName--String • parentParticipantDefName--String • parentBusObj--BusObj • childParticipantDefName--String

Tabella 37. Generale/API/Relazione identità (Continua)

Nome	Descrizione	Input e output con valori accettabili
Elimina elementi secondari personali	Rimuove le istanze secondarie specificate da una relazione principale/secondario per una relazione che appartiene all'elemento principale specificato. API: IdentityRelationship.deleteMyChildren()	Input: • mappa--BaseDLM • parentChildRelDefName--String • parentParticipantDefName--String • parentBusObj--BusObj • childParticipantDefName--String • childBusObjList--BusObj,BusObjArray
Riferimento incrociato chiave esterna	Esegue una ricerca nella tabella delle relazioni nel database relazioni in base alla chiave esterna dell'oggetto business origine, aggiungendo una nuova istanza di relazione nella tabella relazioni esterne, se la chiave esterna non esiste già. API: IdentityRelationship.foreignKeyXref()	Input: • mappa--BaseDLM • RelDefName--String • appParticipantDefName--String • genParticipantDefName--String • appSpecificBusObj--BusObj • appForeignAttr--String • genericBusObj--BusObj • genForeignAttr--String
Ricerca chiave esterna	Esegue una ricerca in una tabella relazioni esterne in base alla chiave esterna dell'oggetto business origine, non riuscendo a trovare un'istanza relazione se la chiave esterna non esiste nella tabella di relazioni esterne. API: IdentityRelationship.foreignKeyLookup()	Input: • mappa--BaseDLM • relDefName--String • appParticipantDefName--String • appSpecificBusObj--BusObj • appForeignAttr--String • genericBusObj--BusObj • genForeignAttr--String
Conserva verbo secondario	Imposta il verbo dell'oggetto business secondario basato sul contesto di esecuzione della mappa e sul verbo dell'oggetto business principale. API: IdentityRelationship.maintainChildVerb()	Input: • mappa--BaseDLM • relDefName--String • appSpecificParticipantName--String • genericParticipantName--String • appSpecificObj--BusObj • appSpecificChildObj--String • genericObj--BusObj • genericChildObj--String • to_Retrieve--boolean • Is_Composite--boolean
Conserva relazione composta	Mantiene una relazione d'identità composta dalla mappa principale. API: IdentityRelationship.maintain CompositeRelationship()	Input: • mappa--BaseDLM • relDefName--String • participantDefName--String • appSpecificBusObj--BusObj • genericBusObjList--BusObj, BusObjArray

Tabella 37. Generale/API/Relazione identità (Continua)

Nome	Descrizione	Input e output con valori accettabili
Conserva relazione identità semplice	Mantiene una relazione d'identità semplice da una mappa principale o secondaria. API: IdentityRelationship.maintain Simple Identity Relationship()	Input: • mappa--BaseDLM • relDefName--String • participantDefName--String • appSpecificBusObj--BusObj • genericBusObj--BusObj
Aggiorna elementi secondari personali	Aggiunge ed elimina istanze secondarie in una relazione principale/secondario specificata di una relazione d'identità, quando necessario. API: IdentityRelationship.updateMyChildren()	Input: • mappa--BaseDLM • parentChildRelDefName--String • parentParticipantDef--String • parentBusObj--BusObj • childParticipantDef--String • childAttrName--String • childIdentityRelDefName--String • childIdentityParticipantDefName--String

Tabella 38. Generale/API/Mappe

Nome	Descrizione	Input e output con valori accettabili
Ottieni nome adattatore	Richiama il nome adattatore associato all'istanza della mappa corrente. API: MapExeContext.getConnName()	Input: mappa--BaseDLM Output: nome adattatore--String
Ottieni contesto chiamata	Richiama il contesto chiamata associato all'istanza della mappa corrente. API: MapExeContext.getInitiator()	Input: mappa--BaseDLM Output: contesto chiamata--String
Ottieni oggetto business di richiesta originale	Richiama l'oggetto business di richiesta originale associato all'istanza della mappa corrente. API: MapExeContext.getOriginalRequestBO()	Input: mappa--BaseDLM Output: oggetto business originale--BusObj

Tabella 39. Generale/API/Mappe/Costanti

Nome	Descrizione	Input e output con valori accettabili
Contesto chiamata: ACCESS _REQUEST	Un client di accesso ha inviato una richiesta di accesso da un'applicazione esterna a InterChange Server Express. API: MapExeContext.ACCESS_REQUEST	Output: ACCESS_REQUEST--String
Contesto chiamata: ACCESS _RESPONSE	L'oggetto business origine viene inviato di nuovo al client di accesso originario in risposta ad una richiesta di distribuzione sottoscrizione. API: MapExeContext.ACCESS_RESPONSE	Output: ACCESS_RESPONSE--String
Contesto chiamata: EVENT _DELIVERY	Un connettore ha inviato un evento dall'applicazione a InterChange Server Express (flusso attivato da evento). API: MapExeContext.EVENT_DELIVERY	Output: EVENT_DELIVERY--String

Tabella 39. Generale/API/Mappe/Costanti (Continua)

Nome	Descrizione	Input e output con valori accettabili
Contesto chiamata: SERVICE_CALL _FAILURE	Una richiesta di chiamata di servizio della collaborazione non è riuscita. Potrebbe essere necessario eseguire un'azione correttiva. API: MapExeContext.SERVICE_CALL_FAILURE	Output: SERVICE_CALL_FAILURE --String
Contesto chiamata: SERVICE_CALL _REQUEST	Una collaborazione sta inviando un oggetto business all'applicazione mediante una richiesta di chiamata di servizio. API: MapExeContext.SERVICE_CALL_REQUEST	Output: SERVICE_CALL_REQUEST --String
Contesto chiamata: SERVICE_CALL _RESPONSE	Un oggetto business è stato ricevuto dall'applicazione come risultato di una risposta con esito positivo ad una richiesta di chiamata di servizio della collaborazione. API: MapExeContext.SERVICE_CALL _RESPONSE	Output: SERVICE_CALL_RESPONSE --String

Tabella 40. Generale/API/Mappe/Eccezione

Nome	Descrizione	Input e output con valori accettabili
Eccezione mappa	Promuove un'eccezione runtime della mappa. API: raiseException()	Input: • mappa--BaseDLM • tipo di eccezione--String • messaggio--String
Eccezione mappa 1	Promuove un'eccezione runtime della mappa. API: raiseException()	Input: • mappa--BaseDLM • tipo di eccezione--String • messaggio--String • parametro 1--String
Eccezione mappa 2	Promuove un'eccezione runtime della mappa. API: raiseException()	Input: • mappa--BaseDLM • tipo di eccezione--String • messaggio--String • parametro 1--String • parametro 2--String
Eccezione mappa 3	Promuove un'eccezione runtime della mappa. API: raiseException()	Input: • mappa--BaseDLM • tipo di eccezione--String • messaggio--String • parametro 1--String • parametro 2--String • parametro 3--String

Tabella 40. Generale/API/Mappe/Eccezione (Continua)

Nome	Descrizione	Input e output con valori accettabili
Eccezione mappa 4	Promuove un'eccezione runtime della mappa. API: raiseException()	Input: • mappa--BaseDLM • tipo di eccezione--String • messaggio--String • parametro 1--String • parametro 2--String • parametro 3--String • parametro 5--String
Eccezione mappa 5	Promuove un'eccezione runtime della mappa. API: raiseException()	Input: • mappa--BaseDLM • tipo di eccezione--String • messaggio--String • parametro 1--String • parametro 2--String • parametro 3--String • parametro 5--String • parametro 5--String
Eccezione RunTimeEntity della mappa eccezione	Promuove un'eccezione runtime della mappa. API: raiseException()	Input: • mappa--BaseDLM • eccezione--RunTimeEntityException

Tabella 41. Generale/API/Partecipante

Nome	Descrizione	Input e output con valori accettabili
Ottieni dati booleani	Richiama i dati associati all'oggetto Partecipante. API: Participant.getBoolean()	Input: partecipante--Server.RelationshipServices.Participant Output: dati--boolean
Ottieni dati oggetto business	Richiama i dati associati all'oggetto Partecipante. API: Participant.getBusObj()	Input: partecipante--Server.RelationshipServices.Participant Output: dati--BusObj
Ottieni dati doppi	Richiama i dati associati all'oggetto Partecipante. API: Participant.getDouble()	Input: partecipante--Server.RelationshipServices.Participant Output: dati--double
Ottieni dati mobili	Richiama i dati associati all'oggetto Partecipante. API: Participant.getFloat()	Input: partecipante--Server.RelationshipServices.Participant Output: dati--mobili
Ottieni ID istanza	Richiama l'ID istanza di relazione della relazione alla quale l'istanza partecipante sta partecipando. API: Participant.getInstanceId()	Input: partecipante--Server.RelationshipServices.Participant Output: ID istanza--int
Ottieni dati Int	Richiama i dati associati all'oggetto Partecipante. API: Participant.getInt()	Input: partecipante--Server.RelationshipServices.Participant Output: dati--int

Tabella 41. Generale/API/Partecipante (Continua)

Nome	Descrizione	Input e output con valori accettabili
Ottieni dati lunghi	Richiama i dati associati all'oggetto Partecipante. API: Participant.getLong()	Input: partecipante--Server.RelationshipServices.Participant Output: dati--long
Ottieni nome partecipante	Richiama il nome definizione del partecipante dal quale viene creata l'istanza del partecipante. API: Participant.getParticipantDefinition()	Input: partecipante--Server.RelationshipServices.Participant Output: nome--String
Ottieni nome relazione	Richiama il nome di definizione della relazione alla quale sta partecipando l'istanza del partecipante. API: Participant.getRelationshipDefinition()	Input: partecipante--Server.RelationshipServices.Participant Output: nome--String
Ottieni dati Stringa	Richiama i dati associati all'oggetto Partecipante. API: Participant.getString()	Input: partecipante--Server.RelationshipServices.Participant Output: dati--String
Nuovo partecipante	Crea una nuova istanza partecipante senza istanze di relazione. API: Participant()	Input: • relDefName--String • partDefName--String • partData--BusObj, String, long, int, double, float, boolean Output: partecipante--Server.RelationshipServices.Participant
Nuovo partecipante in relazione	Crea una nuova istanza partecipante da aggiungere ad un partecipante esistente in un'istanza relazione. API: RelationshipServices.Participant()	Input: • relDefName--String • partDefName--String • instanceId--int • partData--BusObj, String, long, int, double, float, boolean Output: participante--Server.RelationshipServices.Participant
Imposta dati	Imposta i dati associati all'istanza del partecipante. API: Participant.set()	Input: • participante--Server.RelationshipServices.Participant • partData--BusObj, String, long, int, double, float, boolean
Imposta ID istanza	Imposta l'ID istanza della relazione a cui sta partecipante il partecipante. API: Participant.setInstanceId()	Input: • participante--Server.RelationshipServices.Participant • id--int
Imposta la definizione del partecipante	Imposta il nome definizione del participante dal quale viene creata l'istanza del partecipante. API: Participant.setParticipantDefinition()	Input: • participante--Server.RelationshipServices.Participant • partDefName--String
Imposta definizione relazione	Imposta la definizione di relazione nella quale l'istanza partecipante sta partecipando. API: Participant.setRelationshipDefinition()	Input: • participante--Server.RelationshipServices.Participant • relDefName--String

Tabella 42. Generale/API/Partecipante/Vettore

Nome	Descrizione	Input e output con valori accettabili
Ottieni partecipante in	Ottieni l'elemento all'indice specificato nel vettore partecipante.	Input: • vettore--Server.RelationshipServices.Participant[] • indice--int Output: participante--Server.RelationshipServices.Participant
Nuovo vettore partecipante	Crea un nuovo vettore partecipante con la dimensione specificata.	Input: dimensione--int Output: vettore--Server.RelationshipServices.Participant[]
Imposta partecipante a	Imposta l'elemento nell'indice specificato nel vettore partecipante.	Input: • vettore--Server.RelationshipServices.Participant[] • indice--int • participante--Server.RelationshipServices.Participant
Dimensione	Ottieni la dimensione del vettore partecipante.	Input: vettore--Server.RelationshipServices.Participant[] Output: dimensione--int

Tabella 43. Generale/API/Partecipante/Costante

Nome	Descrizione	Input e output con valori accettabili
Partecipante: INVALID _INSTANCE_ID	La costante Partecipante che indica l'ID partecipante non è valida. API: Participant.INVALID_INSTANCE_ID	Output: INVALID_INSTANCE_ID--int

Tabella 44. Generale/API/Relazione

Nome	Descrizione	Input e output con valori accettabili
Aggiungi partecipante	Aggiunge un oggetto partecipante esistente all'istanza di relazione. API: Relationship.addParticipant()	Input: partecipante--Server.RelationshipServices.Participant Output: ID istanza risultato--int
Aggiungi dati partecipante	Aggiunge un nuovo partecipante ad un'istanza di relazione esistente. API: Relationship.addParticipant()	Input: • relDefName--String • partDefName--String • instanceId--int • partData--BusObj, String, long, int, double, float, boolean Output: ID istanza risultato--int

Tabella 44. Generale/API/Relazione (Continua)

Nome	Descrizione	Input e output con valori accettabili
Aggiungi dati partecipante a nuova relazione	Aggiunge un partecipante ad una nuova istanza di relazione. API: Relationship.addParticipant()	Input: • relDefName--String • partDefName--String • partData--BusObj, String, long, int, double, float, boolean Output: ID istanza risultato--int
Crea relazione	Crea una nuova istanza di relazione. API: Relationship.create()	Input: • relDefName--String • partDefName--String • partData--BusObj, String, long, int, double, float, boolean Output: ID istanza--int
Crea relazione con partecipante	Crea una nuova istanza di relazione. API: Relationship.create()	Input: partecipante--Server.RelationshipServices.Participant Output: ID istanza--int
Disattiva partecipante	Disattiva un partecipante da una o più istanze di relazione. API: Relationship.deactivate Participant()	Input: partecipante--Server.RelationshipServices.Participant
Disattiva partecipante in base ai dati	Disattiva un partecipante da una o più istanze di relazione. API: Relationship.deactivate Participant()	Input: • relDefName--String • partDefName--String • partData--BusObj, String, long, int, double, float, boolean
Disattiva partecipante in base all'istanza	Disattiva un partecipante da una specifica istanza di relazione. API: Relationship.deactivate ParticipantByInstance()	Input: • relDefName--String • partDefName--String • instanceId--int
Disattiva partecipante in base ai dati di istanza	Disattiva un partecipante da una specifica istanza di relazione coni dati associati al partecipante. API: Relationship.deactivate ParticipantByInstance()	Input: • relDefName--String • partDefName--String • instanceId--int • partData--BusObj, String, long, int, double, float, boolean
Elimina partecipante	Rimuove un'istanza di partecipante da una o più istanze di relazione. API: Relationship.deleteParticipant()	Input: partecipante--Server.RelationshipServices.Participant
Elimina partecipante in base all'istanza	Rimuove un partecipante da una specifica istanza di relazione. API: Relationship.deleteParticipanByInstancet()	Input: • relDefName--String • partDefName--String • instanceId--int

Tabella 44. Generale/API/Relazione (Continua)

Nome	Descrizione	Input e output con valori accettabili
Elimina partecipante in base ai dati dell'istanza	Rimuove un partecipante da una specifica istanza di relazione con in dati associati al partecipante. API: Relationship.deleteParticipanByInstancet()	Input: • relDefName--String • partDefName--String • instanceId--int • partData--BusObj, String, long, int, double, float, boolean
Elimina partecipante con dati	Rimuove un'istanza di partecipante da una o più istanze di relazione. API: Relationship.deleteParticipant()	Input: • relDefName--String • partDefName--String • partData--BusObj, String, long, int, double, float, boolean
Ottieni ID istanza successiva	Restituisce l'ID istanza relazione della successiva relazione disponibile, in base al nome della definizione della relazione. API: Relationship.getNewID()	Input: relDefName--String Output: ID--int
Ottieni istanze	Richiama zero o più ID di istanze di relazione che contengono i partecipanti forniti. API: Relationship.retrieveInstances()	Input: • relDefName--String • partDefName--String,String[] • partData--BusObj, String, long, int, double, float, boolean Output: ID istanza--int
Ottieni istanze per partecipante	Richiama zero o più ID di istanze di relazione che contengono un dato partecipante. API: Relationship.retrieveInstances()	Input: • relDefName--String • partData--BusObj, String, long, int, double, float, boolean Output: ID istanza--int
Ottieni partecipanti	Richiama zero o più partecipanti da un'istanza di relazione. API: Relationship.retrieveParticipants()	Input: • relDefName--String • partDefName--String, String[] • instanceId--int Output: istanze partecipante--Server.RelationshipServices.Participant[]
Ottieni partecipanti con ID	Richiama zero o più partecipanti da un'istanza di relazione. API: Relationship.retrieveParticipants()	Input: • relDefName--String • instanceId--int Output: istanze partecipante--Server.RelationshipServices.Participant[]
Aggiorna partecipante	Aggiorna un partecipante in una o più istanze di relazione. API: Relationship.updateParticipant()	Input: • relDefName--String • partDefName--String • partData--BusObj

Tabella 45. Generale/Data

Nome	Descrizione	Input e output con valori accettabili
Aggiungi giorno	Aggiunge ulteriori giorni alla data Da.	Input: • dalla data--String • formato data--String • giorni da aggiungere--int Output: alla data-- String
Aggiungi mese	Aggiunge ulteriori mesi alla data Da.	Input: • dalla data--String • formato data--String • mese da aggiungere--int Output: alla data-- String
Aggiungi anno	Aggiunge ulteriori anni alla data Da.	Input: • dalla data--String • formato data--String • anni da aggiungere--int Output: alla data-- String
Data successiva	Confronta due date e stabilisce se la Data 1 è successiva alla Data 2.	Input: • Data 1--String • formato Data 1--String • Data 2--String • formato Data 2--String Output: la Data 1 è successiva alla Data 2?--boolean
Data precedente	Confronta due date e stabilisce se la Data 1 è precedente alla Data 2.	Input: • Data 1--String • formato Data 1--String • Data 2--String • formato Data 2--String Output: la Data 1 è precedente alla Data 2?--boolean
Data uguale a	Confronta due date e stabilisce se sono uguali.	Input: • Data 1--String • formato Data 1--String • Data 2--String • formato Data 2--String Output: sono uguali?-- boolean
Modifica formato	Modifica il formato di una data.	Input: • data--String • formato input--String • formato output--String Output: data formattata--String

Tabella 45. Generale/Data (Continua)

Nome	Descrizione	Input e output con valori accettabili
Ottieni giorno	Restituisce il giorno numerico del mese in base al formato della data.	Input: • Data--String • Formato--String Output: Giorno--int
Ottieni mese	Restituisce il mese numerico dell'anno in base al formato della data.	Input: • Data--String • Formato--String Output: Mese--int
Ottieni anno	Restituisce l'anno numerico in base al formato della data.	Input: • Data--String • Formato--String Output: Anno--int
Ottieni giorno mese anno	Fornita una data di input, estrae rispettivamente le parti Anno/Mese/Giorno dalla data di input.	Input: • Data--String • Formato--String Output: • Anno--int • Mese--int • Giorno--int
Ora	Ottieni data odierna.	Input: formato--String Output: adesso--String

Tabella 46. Generale/Data/Formati

Nome	Descrizione	Input e output con valori accettabili
gg-MM-aaaa	Formato data aaaa-MM-gg **Esempio:** 2003-02-25	Output: formato--String
ggMMaaaa	Formato data aaaaMMgg **Esempio:** 20030225	Output: formato--String
ggMMaaaa HH:mm:ss	Formato data aaaaMMgg HH:mm:ss **Esempio:** 20030225 12:36:40	Output: formato--String

Tabella 47. Generale/Registro e traccia

Nome	Descrizione	Input e output con valori accettabili
Errore log	Invia il messaggio di errore specificato al file di log di InterChange Server Express.	Input: messaggio--String, byte, short, int, long, float, double
ID errore log	Invia il messaggio di errore associato all'ID specificato al file di log di InterChange Server Express.	Input: ID--String, byte, short, int, long, float, double
Informativo log	Invia il messaggio informativo specificato al file di log di InterChange Server Express.	Input: messaggio--String, byte, short, int, long, float, double

Tabella 47. Generale/Registro e traccia (Continua)

Nome	Descrizione	Input e output con valori accettabili
ID informativo log	Invia il messaggio informativo associato all'ID specificato al file di log di InterChange Server Express.	Input: ID--String, byte, short, int, long, float, double
Avviso log	Invia il messaggio di avvertenza specificato al file di log di InterChange Server Express	Input: messaggio--String, byte, short, int, long, float, double
ID avviso log	Invia il messaggio di avvertenza associato all'ID specificato al file di log di InterChange Server Express.	Input: ID--String, byte, short, int, long, float, double
Traccia	Invia il messaggio di traccia specificato al file di log di InterChange Server Express.	Input: messaggio--String, byte, short, int, long, float, double

Tabella 48. Generale/Registro e traccia/Errore log

Nome	Descrizione	Input e output con valori accettabili
ID errore log 1	Formatta il messaggio di errore associato all'ID specificato con il parametro e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro--String, byte, short, int, long, float, double
ID errore log 2	Formatta il messaggio di errore associato all'ID specificato con i parametri e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double
ID errore log 3	Formatta il messaggio di errore associato all'ID specificato con i parametri e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double • parametro 3--String, byte, short, int, long, float, double

Tabella 49. Generale/Registro e traccia/Informativo log

Nome	Descrizione	Input e output con valori accettabili
ID informativo log 1	Formatta il messaggio informativo associato all'ID specificato con il parametro e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro--String, byte, short, int, long, float, double

Tabella 49. Generale/Registro e traccia/Informativo log (Continua)

Nome	Descrizione	Input e output con valori accettabili
ID informativo log 2	Formatta il messaggio informativo associato all'ID specificato con i parametri e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double
ID informativo log 3	Formatta il messaggio informativo associato all'ID specificato con i parametri e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double • parametro 3--String, byte, short, int, long, float, double

Tabella 50. Generale/Registro e traccia/Avviso log

Nome	Descrizione	Input e output con valori accettabili
ID avviso log 1	Formatta il messaggio di avviso associato all'ID specificato con il parametro e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro--String, byte, short, int, long, float, double
ID avviso log 2	Formatta il messaggio di avviso associato all'ID specificato con i parametri e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double
ID avviso log 3	Formatta il messaggio di avviso associato all'ID specificato con i parametri e lo invia al file di log di InterChange Server Express.	Input: • ID--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double • parametro 3--String, byte, short, int, long, float, double

Tabella 51. Generale/Registro e traccia/Traccia

Nome	Descrizione	Input e output con valori accettabili
ID traccia 1	Formatta il messaggio di traccia associato all'ID specificato con il parametro e lo visualizza se la traccia viene impostata al livello specificato o ad un livello superiore.	Input: • ID--String, byte, short, int, long, float, double • livello--String, byte, short, int, long, float, double • parametro--String, byte, short, int, long, float, double
ID traccia 2	Formatta il messaggio di traccia associato all'ID specificato con i parametri e lo visualizza se la traccia viene impostata al livello specificato o ad un livello superiore.	Input: • ID--String, byte, short, int, long, float, double • livello--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double
ID traccia 3	Formatta il messaggio di traccia associato all'ID specificato con i parametri e lo visualizza se la traccia viene impostata al livello specificato o ad un livello superiore.	Input: • ID--String, byte, short, int, long, float, double • livello--String, byte, short, int, long, float, double • parametro 1--String, byte, short, int, long, float, double • parametro 2--String, byte, short, int, long, float, double • parametro 3--String, byte, short, int, long, float, double
Traccia a livello	Visualizza il messaggio di traccia se viene impostato al livello specificato o ad un livello superiore.	Input: • messaggio--String, byte, short, int, long, float, double • livello--String, byte, short, int, long, float, double

Tabella 52. Generale/Mappatura

Nome	Descrizione	Input e output con valori accettabili
Esegui mappa	Esegue la mappa specificata con il contesto chiamata specificato.	Input: • Nome mappa--String • Oggetti business di origine--BusObj, BusObj[] Output: Risultati mappa--BusObj, BusObj[]
Esegui mappa con contesto	Esegue la mappa specificata con il contesto di chiamata specificato.	Input: • Nome mappa--String • Oggetti business di origine--BusObj, BusObj[] • contesto chiamata--String Output: Risultati mappa--BusObj, BusObj[]

Tabella 53. Generale/Matematica

Nome	Descrizione	Input e output con valori accettabili
Valore assoluto	a=abs(b) API: Math.abs()	Input: b--byte, short, int, long, float, double Output: a--byte, short, int, long, float, double
Limite superiore	Restituisce il numero intero successivo più alto, superiore o uguale all'espressione numerica specificata.	Input: numero--String, float, double Output: limite superiore--int
Dividi	a=b/c	Input: • b--byte, short, int, long, float, double • c--byte, short, int, long, float, double Output: a--byte, short, int, long, float, double
Uguale	Il valore 1 è uguale al valore 2?	Input: • valore 1--String, byte, short, int, long, float, double • valore 2--String, byte, short, int, long, float, double Output: sono uguali?--boolean
Limite inferiore	Restituisce il numero intero successivo più basso, superiore o uguale all'espressione numerica specificata.	Input: numero--String, float, double Output: limite inferiore--int
Maggiore di	Il valore 1 è maggiore del valore 2?	Input: • valore 1--byte, short, int, long, float, double • valore 2--byte, short, int, long, float, double Output: risultato--boolean
Maggior di o uguale a	Il valore 1 è superiore o uguale al valore 2?	Input: • valore 1--byte, short, int, long, float, double • valore 2--byte, short, int, long, float, double Output: risultato--boolean
Minore di	risultato=valore 1 è minore del valore 2?	Input: • valore 1--byte, short, int, long, float, double • valore 2--byte, short, int, long, float, double Output: risultato--boolean
Minore di o uguale a	Il valore 1 è inferiore o uguale al valore 2?	Input: • valore 1--byte, short, int, long, float, double • valore 2--byte, short, int, long, float, double Output: risultato--boolean

Tabella 53. Generale/Matematica (Continua)

Nome	Descrizione	Input e output con valori accettabili
Massimo	a=max(b, c) API: Math.max()	Input: • b--byte, short, int, long, float, double • c--byte, short, int, long, float, double Output: a--byte, short, int, long, float, double
Minimo	a=min(b, c) API: Math.min()	Input: • b--byte, short, int, long, float, double • c--byte, short, int, long, float, double Output: a--byte, short, int, long, float, double
Meno	a=b-c	Input: • b--byte, short, int, long, float, double • c--byte, short, int, long, float, double Output: a--byte, short, int, long, float, double
Moltiplica	a=b*c	Input: • b--byte, short, int, long, float, double • c--byte, short, int, long, float, double Output: a--byte, short, int, long, float, double
Non uguale	risultato=il valore 1 non è uguale al valore 2?	Input: • valore 1--String, byte, short, int, long, float, double • valore 2--String, byte, short, int, long, float, double Output: non sono uguali?--boolean
Non un numero	Restituisce true se l'input non è un numero.	Input: input--String Output: non è un numero--boolean
Da numero a stringa	Converte un'espressione numerica in un'espressione di caratteri.	Input: numero--String, short, int, long, float, double Output: stringa--String
Più	a=b+c	Input: • b--byte, short, int, long, float, double • c--byte, short, int, long, float, double Output: a--byte, short, int, long, float, double
Arrotondamento	Arrotonda un'espressione numerica al precedente numero intero, se <5; altrimenti viene arrotondato al numero intero successivo.	Input: numero--String, float, double Output: numero arrotondato--int
Da stringa a numero	Converte un'espressione di caratteri in un'espressione numerica. API: Math.type()	Input: stringa--String Output: String, short, int, long, float, double

Tabella 54. Generale/Proprietà

Nome	Descrizione	Input e output con valori accettabili
Ottieni proprietà	Richiama il valore della proprietà di configurazione specificato.	Input: nome proprietà--String Output: valore proprietà--String

Tabella 55. Generale/Relazione

Nome	Descrizione	Input e output con valori accettabili
Conserva relazione identità semplice	Mantiene una relazione d'identità con l'API di relazione maintainSimpleIdentityRelationship().	Input: • nome relazione--String • nome partecipante--String • Oggetto business generico--String • Oggetto business specifico per applicazione--String • contesto chiamata--String
Ricerca statica	Cerca un valore statico nella relazione.	Input: • nome relazione--String • nome partecipante--String • in entrata?--boolean • valore origine--String Output: valore ricerca--String

Tabella 56. Generale/Stringa

Nome	Descrizione	Input e output con valori accettabili
Aggiungi testo	Accoda ″in stringa 2″ alla fine della stringa ″in stringa 1.″	Input: • in stringa 1--String • in stringa 2--String Output: risultato--String
Se	Restituisce il primo valore se la condizione è true, il secondo valore se la condizione è false.	Input: • condizione--boolean, Boolean • valore 1--String • valore 2--String Output: risultato--String
E' vuoto	Restituisce il secondo valore se il primo valore è vuoto.	Input: • valore 1--String • valore 2--String Output: risultato--String
E' NULL	Restituisce il secondo valore se il primo valore è null.	Input: • valore 1--String • valore 2--String Output: risultato--String

Tabella 56. Generale/Stringa (Continua)

Nome	Descrizione	Input e output con valori accettabili
Compila a sinistra	Restituisce una stringa della lunghezza specificata; compila a sinistra con il valore indicato.	Input: • stringa--String • compila stringa--String • lunghezza--int Output: stringa compilata--String
Stringa sinistra	Restituisce la parte sinistra della stringa per il numero di posizioni specificato.	Input: • stringa--String • lunghezza--int Output: stringa sinistra--String
Minuscolo	Cambia tutti i caratteri in lettere minuscole	Input: daString--String Output: aString--String
Da oggetto a stringa	Ottiene una rappresentazione a stringa dell'oggetto.	Input: oggetto--Object Output: stringa--String
Ripeti	Restituisce una stringa di caratteri che contiene un'espressione di caratteri specificata ripetuta per un numero di volte specificato.	Input: • stringa ripetitiva--String • conteggio ripetizione--int Output: risultato--String
Sostituisci	Sostituisce la parte di una stringa con i dati del valore indicato.	Input: • input--String • vecchia stringa--String • nuova stringa--String Output: stringa sostituita--String
Compila a destra	Restituisce una stringa della lunghezza specificata; compila a destra con il valore indicato.	Input: • stringa--String • compila stringa--String • lunghezza--int Output: stringa compilata--String
Stringa destra	Restituisce la parte destra della stringa per il numero di posizioni specificato.	Input: • stringa--String • lunghezza--int Output: stringa destra--String
Stringa secondaria in base alla posizione	Restituisce una parte della stringa in base ai parametri di inizio e fine.	Input: • stringa--String • posizione iniziale--int • posizione finale--int Output: stringa secondaria--String

Tabella 56. Generale/Stringa (Continua)

Nome	Descrizione	Input e output con valori accettabili
Stringa secondaria in base al valore	Restituisce una parte della stringa in base ai parametri di inizio e fine. La stringa secondaria non include il valore di inizio e quello di fine.	Input: • stringa--String • valore iniziale--int • valore finale--int Output: stringa secondaria--String
Testo uguale	Confronta le stringhe ″inString1″ e ″inString2″ e stabilisce se sono uguali.	Input: • inString1--String • inString2--String Output: sono uguali?--boolean
Testo uguale ignora maiuscolo/minuscolo	Confronta le stringhe ″inString1″ e ″inString2″ in base al lessico, ignorando maiuscolo e minuscolo.	Input: • inString1--String • inString2--String Output: sono uguali?--boolean
Lunghezza testo	Rileva il numero totale di caratteri contenuti in una stringa	Input: str--String Output: lunghezza---byte, short, int, long, float, double
Taglia a sinistra	Taglia il numero specificato di caratteri dalla parte laterale sinistra della stringa.	Input: • input--String • taglia lunghezza--int Output: risultato--String
Taglia a destra	Taglia il numero specificato di caratteri dalla parte laterale destra della stringa.	Input: • input--String • taglia lunghezza--int Output: risultato--String
Taglia testo	Taglia gli spazi vuoti prima e dopo la stringa	Input: in stringa--String Output: stringa tagliata--String
Maiuscolo	Cambia tutti caratteri in lettere maiuscole	Input: daString--String Output: aString--String

Tabella 57. Generale/Utilità

Nome	Descrizione	Input e output con valori accettabili
Errore	raccoglie tutte le eccezioni verificatesi nell'attività corrente e nelle relative attività secondarie. (Fare doppio clic sull'icona blocco funzione per definire l'attività secondaria).	Input: • Nome errore--String • Messaggio di errore--String
Tipo di errore	Raccoglie il tipo di eccezione specificato verificatasi nell'attività corrente e nelle relative attività secondarie. (Fare doppio clic sull'icona blocco funzione per definire l'attività secondaria).	Input: • tipo errore--String • Messaggio di errore--String

Tabella 57. Generale/Utilità (Continua)

Nome	Descrizione	Input e output con valori accettabili
Condizione	Se la condizione (″Condition″) è true, esegue l'attività secondaria definita in ″True Action″; altrimenti, esegue l'attività secondaria definita in ″False Action.″ (Fare doppio clic sull'icona blocco funzione per definire l'attività secondaria).	Input: Condizione--boolean
Loop	Ripete l'attività secondaria finché ″Condizione″ è false. (Fare doppio clic sull'icona blocco funzione per definire l'attività secondaria).	Input: Condizione--boolean
Sposta attributo in secondario	Sposta il valore da ″da attributo″ a ″ad attributo″.	Input: • elemento principale origine--BusObj • attributo BO secondario origine--string • da attributo--String • principale di destinazione--BusObj • attributo BO secondario di destinazione--String • ad attributo--String
Errore	Invia una nuova eccezione Java con il messaggio fornito.	Input: messaggio--String
Tipo di errore	Invia l'eccezione Java specificata con il messaggio fornito.	Input: • tipo errore--String • messaggio--String

Tabella 58. Generale/Utilità/Vettore

Nome	Descrizione	Input e output con valori accettabili
Aggiungi elemento	Aggiunge l'elemento specificato al termine del vettore, aumentando di uno la sua dimensione.	Input: vettore--java.util.Vector Output: elemento--Object
Ottieni elemento	Richiama l'elemento nell'indice specificato nell'oggetto Vettore.	Input: • vettore--java.util.Vector • indice--int Output: elemento--Object
Itera vettore	Ripete l'oggetto vettore.	Input: • vettore--java.util.Vector • indice corrente--int • elemento corrente--Object
Nuovo vettore	Crea un nuovo oggetto vettore.	Output: vettore--java.util.Vector
Dimensione	Richiama il numero di elementi in questo vettore.	Input: vettore--java.util.Vector Output: dimensione--int
In vettore	Richiama la rappresentazione vettore contenente tutti gli elementi di questo vettore.	Input: vettore--java.util.Vector Output: vettore--Object[]

Esempio 1: passi per modificare un valore in lettere maiuscole

L'esempio che segue illustra i passi per utilizzare Activity Editor per modificare il valore dell'attributo origine in lettere maiuscole ed assegnare la modifica all'attributo di destinazione.

Effettuare le seguenti operazioni:

1. Dalla scheda Diagramma, trascinare l'attributo origine nell'attributo di destinazione per creare una regola di trasformazione personalizzata. Poi fare clic sull'icona della regola di trasformazione personalizzata per aprire Activity Editor.

   Risultato: si apre Activity Editor.

   Esempio: la Figura 59 mostra la trasformazione personalizzata utilizzata in questo esempio. L'attributo origine è ObjClarify_contact.LastName e quello di destinazione è ObjContact.LastName.

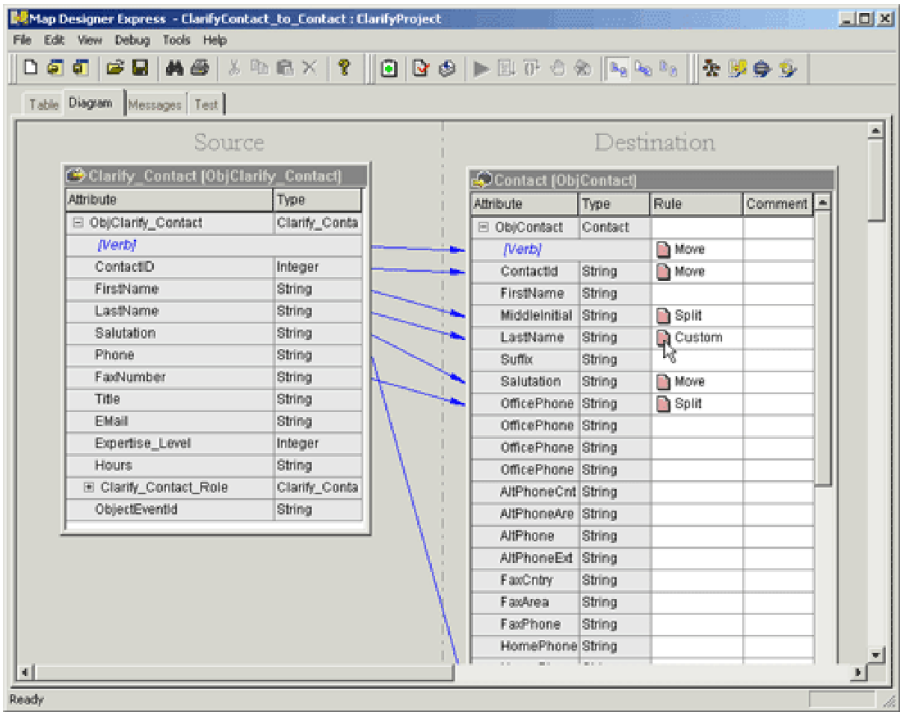


Figura 59. Regola della trasformazione personalizzata

   Per ulteriori informazioni sulla trasformazione personalizzata ed altre trasformazioni, consultare Capitolo 2, "Creazione di mappe", a pagina 15.

2. Selezionare una categoria nella finestra Libreria (in alto a sinistra) per visualizzare i blocchi funzione disponibili nella categoria, come icone, nella finestra Contenuto (in basso a sinistra).

   La Figura 60 mostra i blocchi funzione disponibili per la categoria "Stringa"; gli attributi origine e di destinazione di questo esempio vengono visualizzati come icone nell'area di disegno.

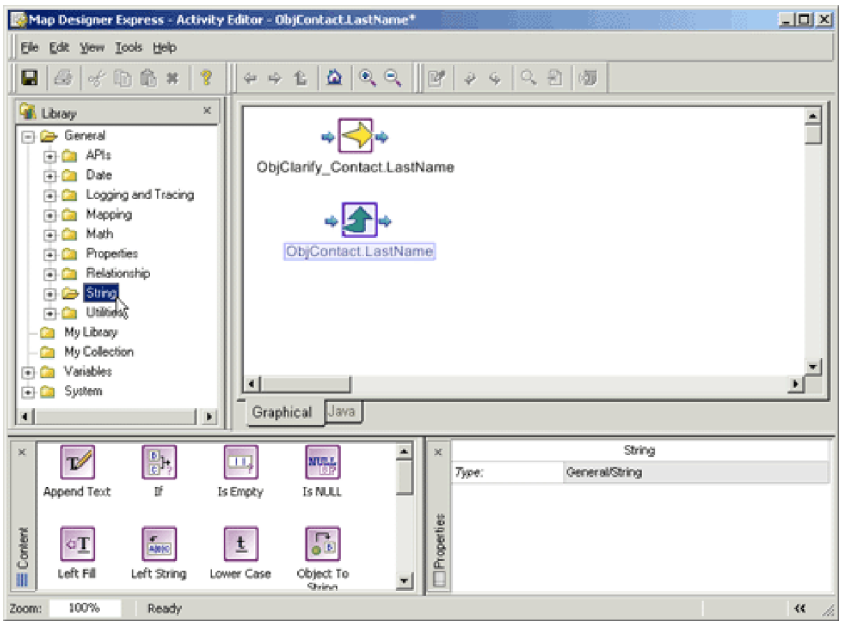


Figura 60. Blocchi funzione nella categoria Stringa ed icone degli attributi di origine e di destinazione

3. Per utilizzare uno dei blocchi funzione dell'attività, trascinare il blocco funzione dalla struttura ad albero nella finestra Libreria e rilasciarlo nell'area di disegno; in alternativa, trascinare l'icona dalla finestra Contenuto e rilasciarla nell'area di disegno.

   Esempio: nell'esempio che segue, si deve cambiare l'attributo origine in lettere maiuscole, quindi si trascina il blocco funzione Maiuscolo nella categoria Stringa dalla finestra Contenuto all'area di disegno. Quest'azione viene visualizzata nella Figura 61.

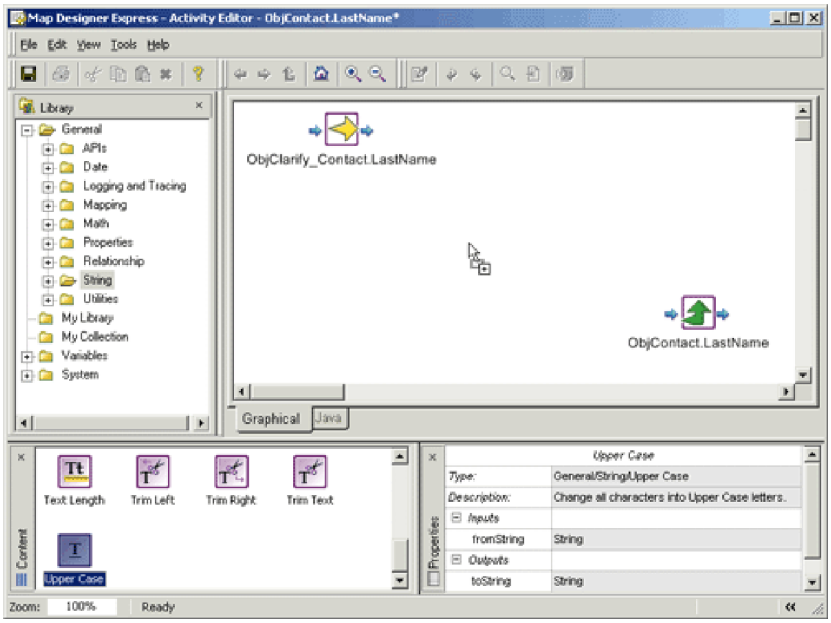


Figura 61. Trascinamento del blocco funzione Maiuscolo

4. Dopo aver trascinato un blocco funzione nell'area di disegno, è possibile spostarlo nell'area selezionando l'icona del blocco funzione e trascinandola nella posizione desiderata. Quando il blocco funzione è nella posizione desiderata, è pronto per collegare gli input e gli output del blocco funzione per definire il flusso di esecuzione.

   Esempio: nell'esempio che segue si desidera convertire il valore dell'attributo di ObjClarify_Contact.LastName in lettere maiuscole. E' possibile eseguire questa operazione collegando l'output dell'icona di ObjClarify_Contact.LastName all'input del blocco funzione Maiuscolo. Per farlo, portare il cursore del mouse sull'output dell'icona della porta ObjClarify_Contact.LastName.

   Risultato: l'aspetto dell'icona cambia in una freccia, ad indicare che è possibile avviare un collegamento in quel punto, come mostrato nella Figura 62.

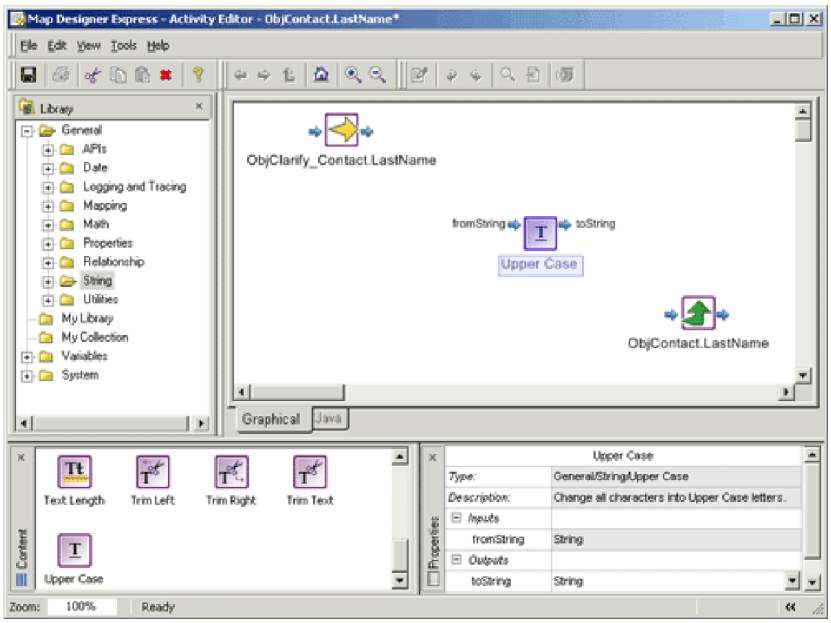


Figura 62. Cursore come freccia alla porta di output di ObjClarify_Contact.LastName

5. Quando l'icona del mouse diventa una freccia, tenere premuto il pulsante del mouse e portare il mouse sull'input del blocco funzione Maiuscolo e rilasciare il pulsante del mouse. Verrà disegnato un collegamento di connessione per collegare l'input e gli output.

   Per indicare che il risultato del blocco funzione Maiuscolo deve essere assegnato all'attributo di destinazione (nell'esempio ObjContact.LastName), ripetere gli stessi passi per effettuare il trascinamento dall'output del blocco funzione Maiuscolo all'input dell'icona porta ObjContact.LastName. La Figura 63 mostra i collegamenti di connessione.

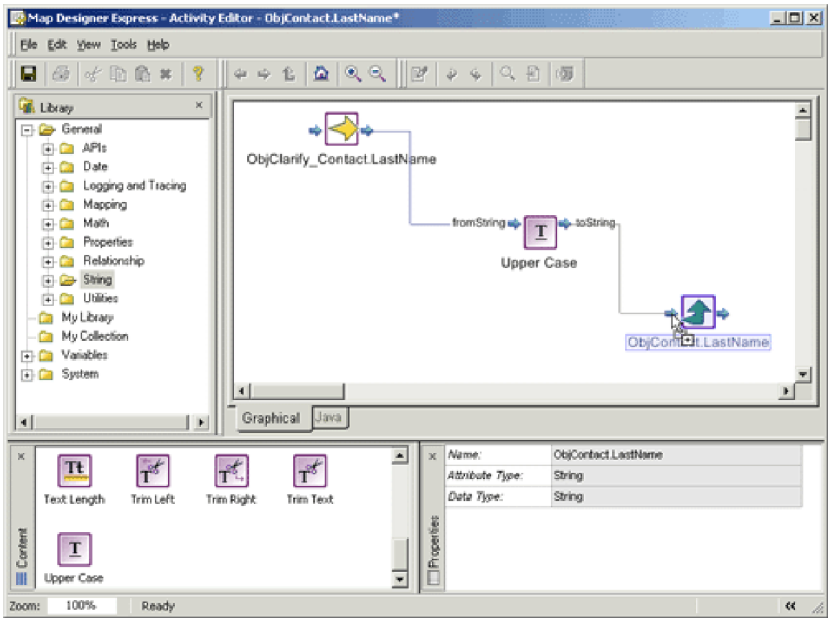


Figura 63. Blocco funzione Maiuscolo con collegamenti di connessione

Risultato: è stata definita un'attività che prenderà il valore dell'attributo origine, lo renderà maiuscolo ed imposterà il valore trasformato in maiuscolo sull'attributo di destinazione.

6. Salvare l'attività selezionando Sul progetto o Sul file dal menu secondario File > Salva o facendo clic sul pulsante Salva mappa sul progetto o Salva mappa sul file nella barra strumenti Standard.
7. Per vedere un esempio di codice Java che verrà generato da questa attività, fare clic sulla scheda Java.

   Risultato: la scheda Java verrà attivata con l'esempio di codice Java, come visualizzato nella Figura 64.

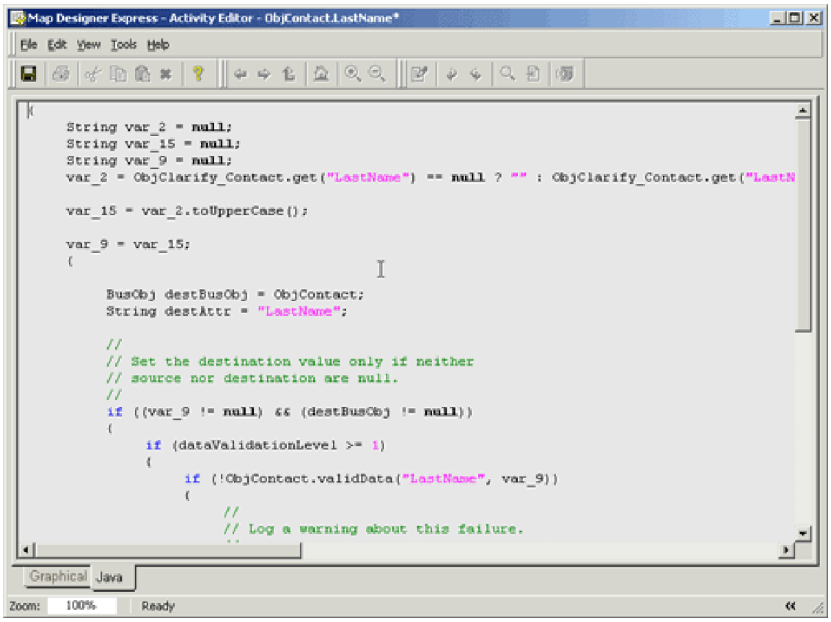


Figura 64. Scheda Java con codice

Esempio 2: passi per modificare un formato di data

L'esempio che segue illustra i passi per utilizzare Activity Editor per modificare la data del valore attributo origine in un formato diverso ed assegnarlo all'attributo di destinazione.

Effettuare le seguenti operazioni:

1. Dalla scheda Diagramma, trascinare l'attributo origine nell'attributo di destinazione per creare una regola di trasformazione personalizzata. Poi fare clic sull'icona della regola di trasformazione personalizzata per aprire Activity Editor.

   Risultato: si apre Activity Editor.

   Esempio: la Figura 65 mostra la trasformazione personalizzata utilizzata in questo esempio. L'attributo origine è ObjClarify_QuoteSchedule.PriceProgExpireDate e quello di destinazione è ObjARInvoice.GLPostingDate.

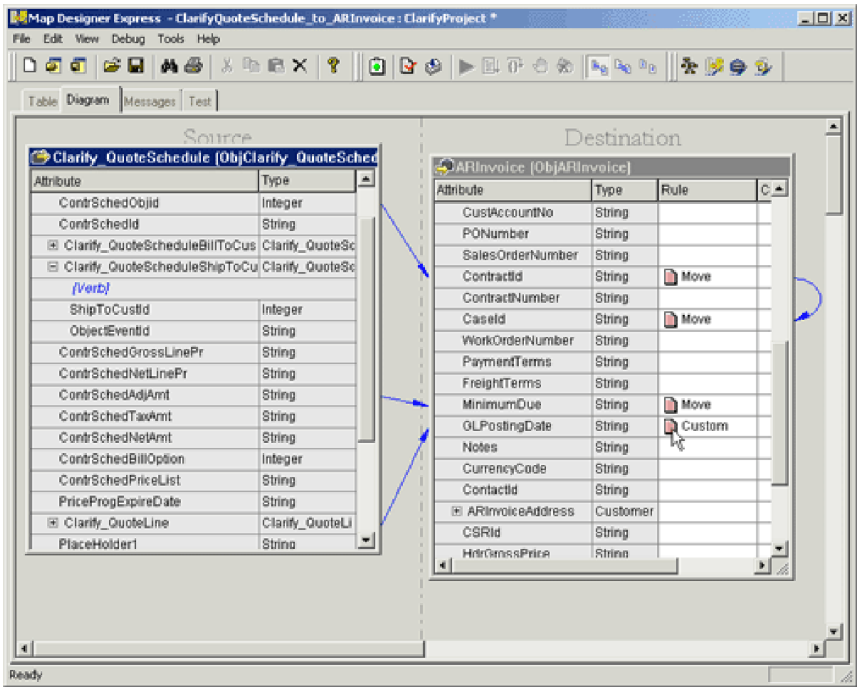


Figura 65. Regola della trasformazione personalizzata

   Per ulteriori informazioni sulla trasformazione personalizzata ed altre trasformazioni, consultare Capitolo 2, "Creazione di mappe", a pagina 15.

2. Selezionare una categoria nella finestra Libreria (in alto a sinistra) per visualizzare i blocchi funzione disponibili in quella categoria, come icone, nella finestra Contenuto (in basso a sinistra).

   La Figura 66 mostra i blocchi funzione disponibili per la categoria "Data"; gli attributi origine e di destinazione di questo esempio vengono visualizzati come icone nell'area di disegno.

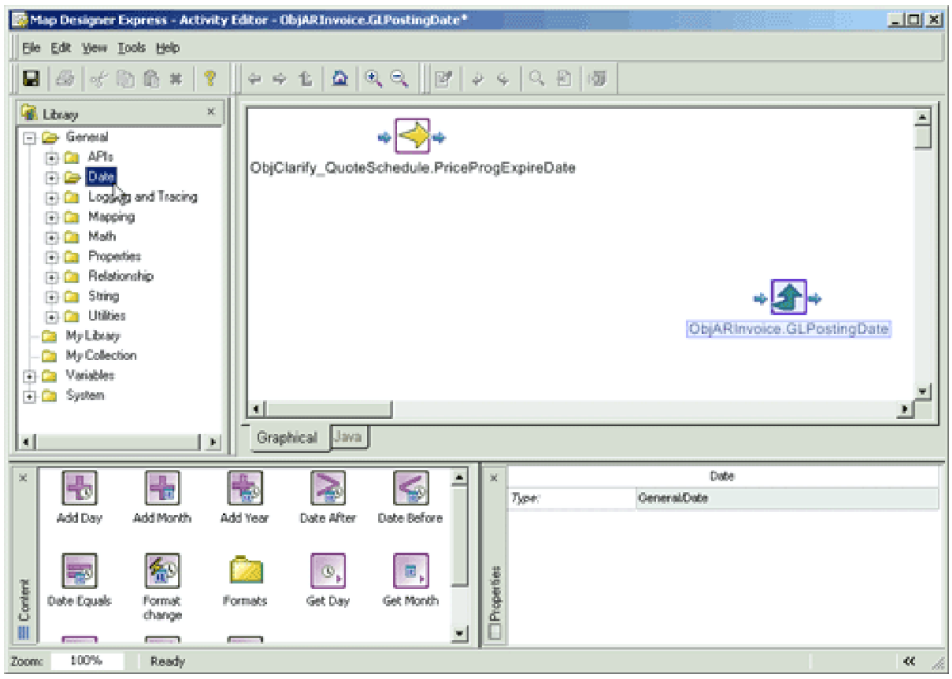


Figura 66. Blocchi funzione nella categoria Data ed icone degli attributi di origine e di destinazione

3. Per utilizzare uno dei blocchi funzione dell'attività, trascinare il blocco funzione dalla struttura ad albero nella finestra Libreria e rilasciarlo nell'area di disegno; in alternativa, trascinare l'icona dalla finestra Contenuto e rilasciarla nell'area di disegno dell'attività.

   Esempio: si desidera modificare il formato della data dell'attributo origine da "aaaaMMgg" a "aaaa.MM.gg G 'alle' HH:mm:ss z" ed assegnarlo all'attributo di destinazione; quindi si trascina il blocco funzione Modifica formato nella categoria Data dalla finestra Contenuto all'area di disegno, come illustrato nella Figura 67.

   Nota: Una data formattata con "aaaaMMgg" ha questo aspetto: "20030227"; una data formattata con "aaaa.MM.gg G 'alle' HH:mm:ss z" diventa "2003.02.27 AD alle 00:00:00 PDT".

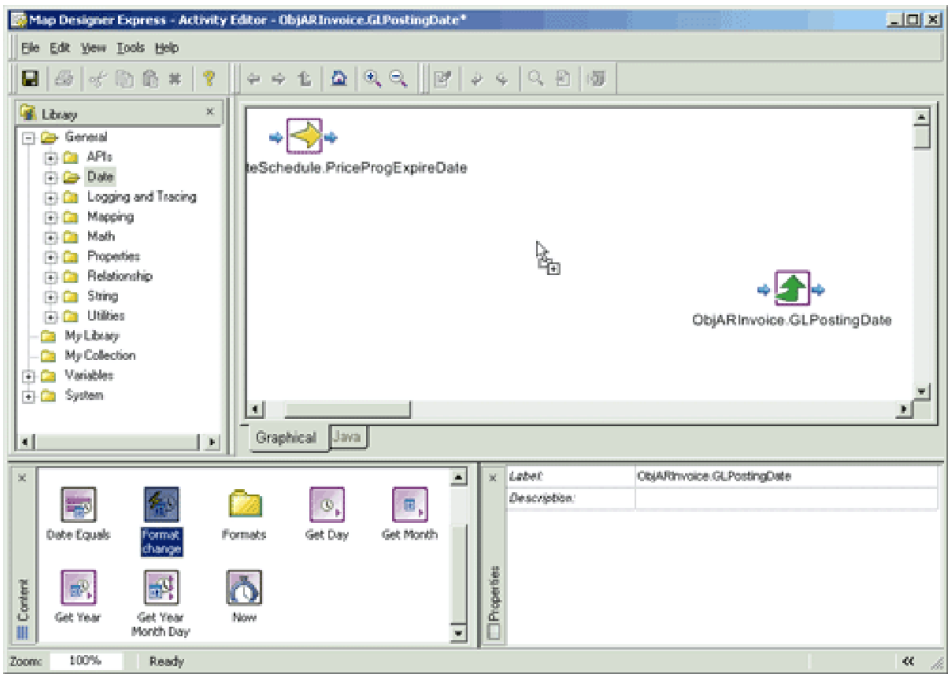


Figura 67. Trascinamento del blocco funzione Modifica formato

4. Dopo aver trascinato un blocco funzione nell'area di disegno dell'attività, è possibile spostarlo nell'area selezionando l'icona del blocco funzione e trascinandola nella posizione desiderata. Quando il blocco funzione è nella posizione desiderata, è pronto per collegare gli input e gli output del blocco funzione per definire il flusso di esecuzione.

   Esempio: si desidera modificare il formato della data dell'attributo origine ObjClarify_QuoteSchedule.PriceProgExpireDate. E' possibile eseguire questa operazione collegando l'output dell'icona della porta ObjClarify_QuoteSchedule.PriceProgExpireDate all'input della data del blocco funzione Modifica formato. Per farlo, portare il cursore del mouse sull'output dell'icona della porta ObjClarify_QuoteSchedule.PriceProgExpireDate.

   Risultato: l'aspetto dell'icona cambia in una freccia, ad indicare che è possibile avviare un collegamento in quel punto, come mostrato nella Figura 68.

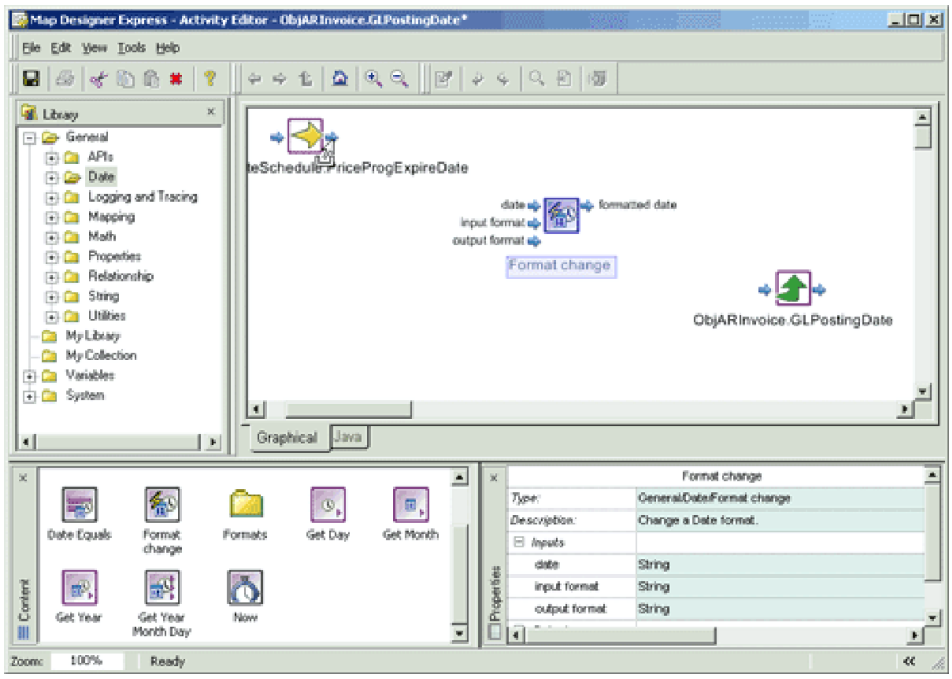


Figura 68. Cursore come freccia alla porta di output di ObjClarify_QuoteSchedule.PriceProgExpireDate

5. Quando l'icona del mouse diventa una freccia, tenere premuto il pulsante del mouse e portare il mouse sull'input della data del blocco funzione Modifica formato e rilasciare il pulsante del mouse. Verrà disegnato un collegamento di connessione per collegare l'input e gli output.

   Per indicare che il risultato del blocco funzione Modifica formato deve essere assegnato all'attributo di destinazione ObjARInvoice.GLPostingDate, ripetere gli stessi passi per effettuare il trascinamento dall'output del blocco funzione Modifica formato all'input dell'icona porta ObjARInvoice.GLPostingDate. La Figura 69 mostra i collegamenti di connessione.

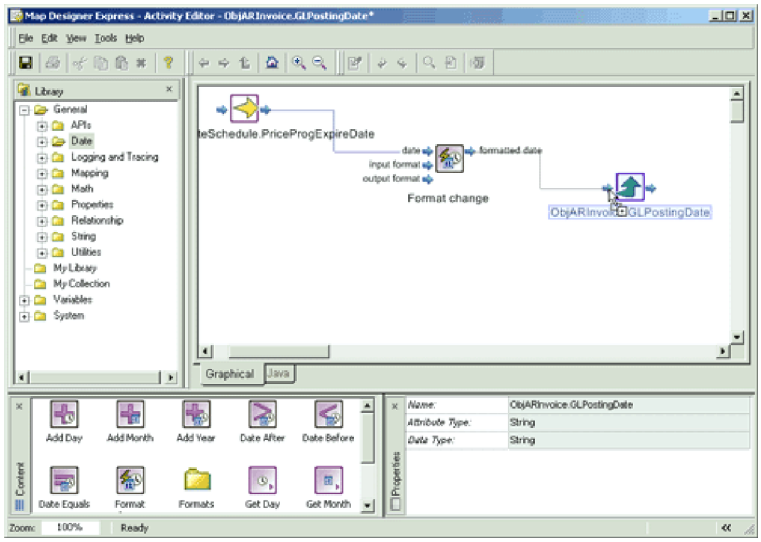


Figura 69. Blocco funzione Modifica formato con collegamenti di connessione

Risultato: ora il blocco funzione Modifica formato è stato istruito a prendere l'input dell'attributo ObjClarify_QuoteSchedule.PriceProgExpireDate, cambiarne il formato della data ed assegnare il risultato all'attributo ObjARInvoice.GLPostingDate. Tuttavia, è necessario anche far sapere al blocco funzione Modifica formato quale sia il formato originale della data e quale sia il formato risultante desiderato.

6. Nell'esempio, se l'attributo origine ObjClarify_QuoteSchedule.PriceProgExpireDate è in formata data aaaMMgg (cioè, 20030227), è possibile utilizzare il blocco funzione Formato data predefinito aaaaMMgg. Trascinare il blocco funzione aaaaMMgg nell'area di disegno dell'attività e collegare il formato dell'output del blocco funzione aaaaMMgg al formato dell'input del blocco funzione Modifica formato.

   Risultato: Questo specifica che il formato dell'input della data è in formato aaaaMMgg, come illustrato nella Figura 70.

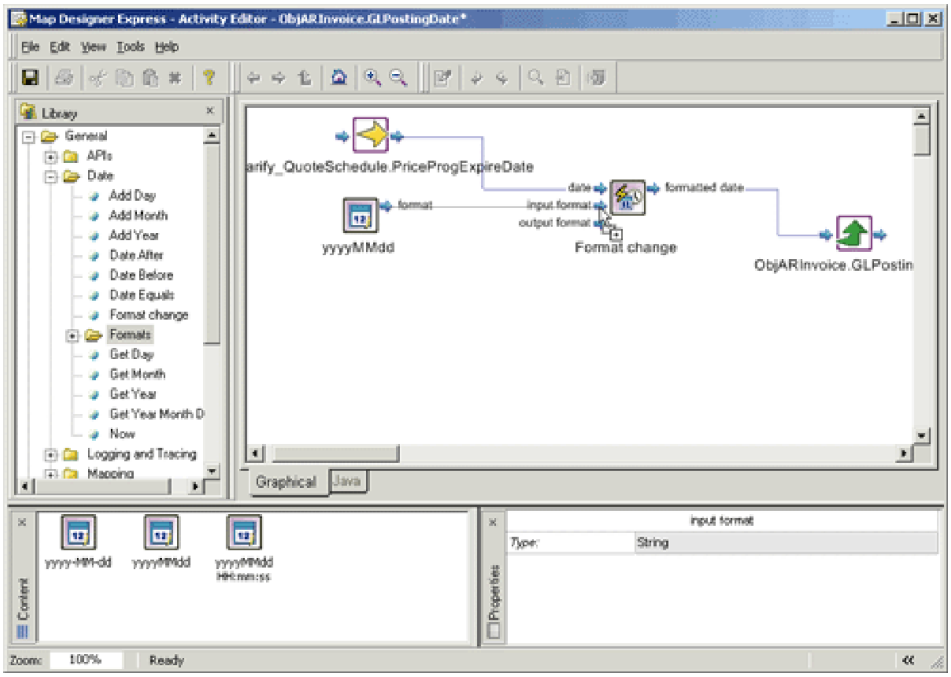


Figura 70. Formato della data di input

7. Activity Editor fornisce tre formati data predefiniti: aaaaMMGG HH:mm:ss, aaaaMMGG e aaaa-MM-gg. Se il formato data desiderato non è uno dei tre formati predefiniti, è possibile specificare il formato data desiderato utilizzando una *Costante*.

   Esempio: si desidera che il blocco funzione Modifica formato cambi il formato data in aaaa.MM.gg G 'alle' Hh"mm"ss z. Questo non è uno dei formati predefiniti quindi è necessario creare un componente Nuova costante nell'area di disegno attività trascinando l'*icona Nuova costante* (che si trova nella categoria Sistema) dalla finestra Contenuto all'area di disegno. La Figura 71 mostra il risultato di questa azione.

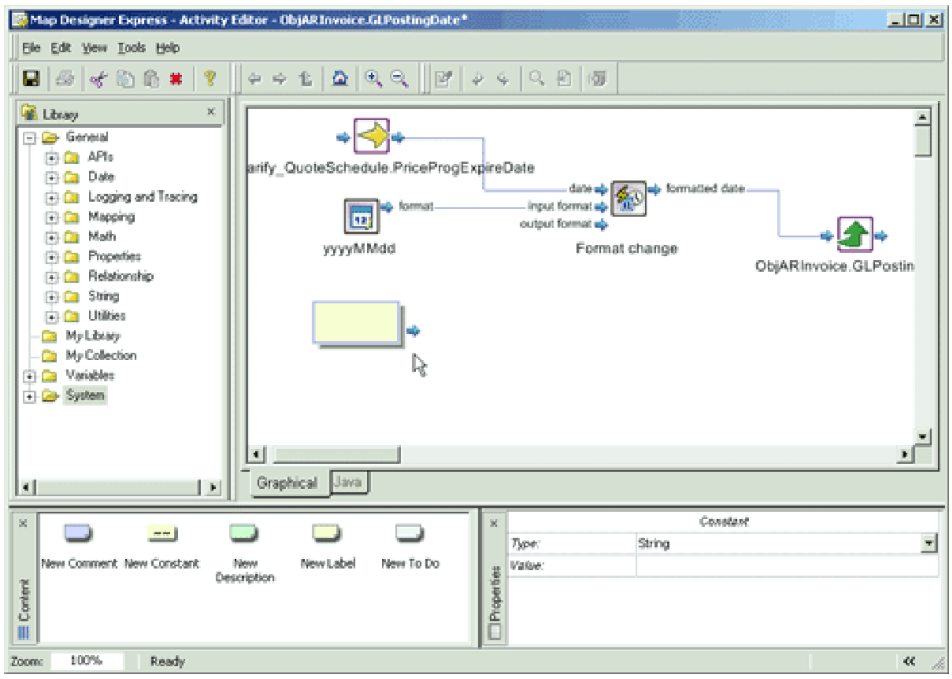


Figura 71. Icona Nuova costante trascinata nell'area di disegno

8. Per specificare una costante con il valore aaaa.MM.gg G 'alle' Hh"mm"ss z, fare clic sull'area modificabile del componente Costante nell'area di disegno dell'attività ed inserire il testo `aaaa.MM.gg G 'alle' Hh"mm"ss z`. Per impostazione predefinita, qualsiasi componente Costante è di tipo Stringa (visualizzato nella finestra Proprietà quando viene selezionato il componente Costante). Tuttavia, è possibile cambiare il tipo di Costante selezionando la Costante e utilizzando la casella combinata della finestra Proprietà. La Figura 72 mostra l'icona Nuova costante con il valore del testo immesso.

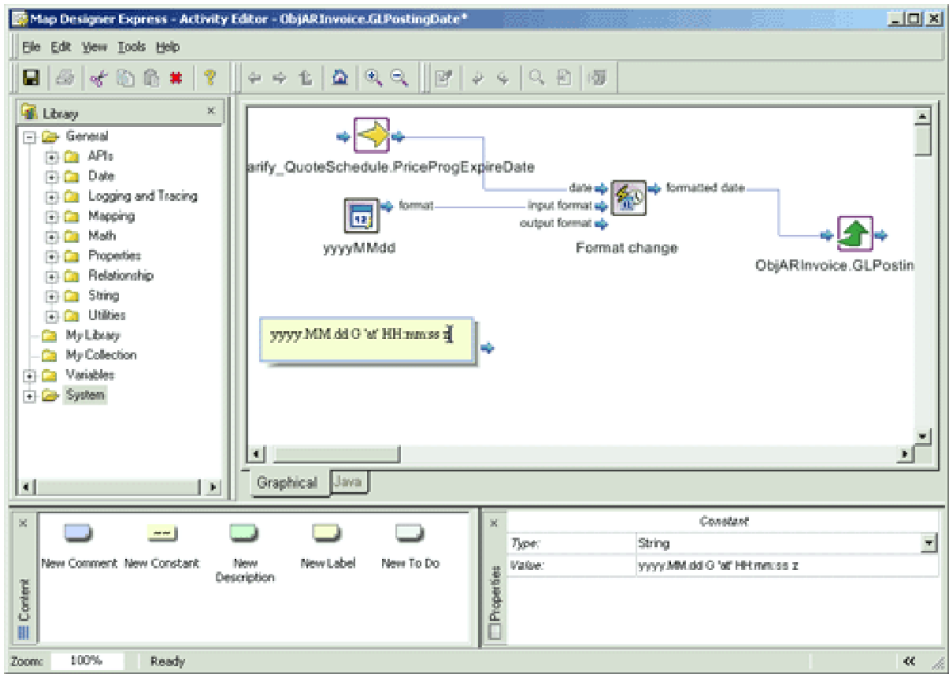


Figura 72. Nuova costante con testo immesso

9. Per continuare a specificare che si desidera il formato dell'output del blocco funzione Modifica formato come aaaa.MM.gg G 'alle' Hh"mm"ss z, si definisce un collegamento di connessione fra il componente Costante ed il formato output del blocco funzione Modifica formato.

   Risultato: è terminata la definizione dell'attività che modificherà il formato della data dell'attributo origine nel nuovo formato di data e lo assegnerà all'attributo di destinazione.

10. Per aggiungere un commento o una descrizione come promemoria relativo all'attività, è possibile aggiungere un componente *Descrizione* all'attività ed inserire una descrizione.

    Suggerimento: utilizzare il menu contestuale nell'area di disegno di modifica e selezionare Aggiungi descrizione oppure trascinare l'*icona Nuova descrizione* contenuta nella cartella Sistema della finestra Contenuto e rilasciarla nell'area di disegno. La Figura 73 mostra come aggiungere il componente Descrizione utilizzando il menu contestuale.

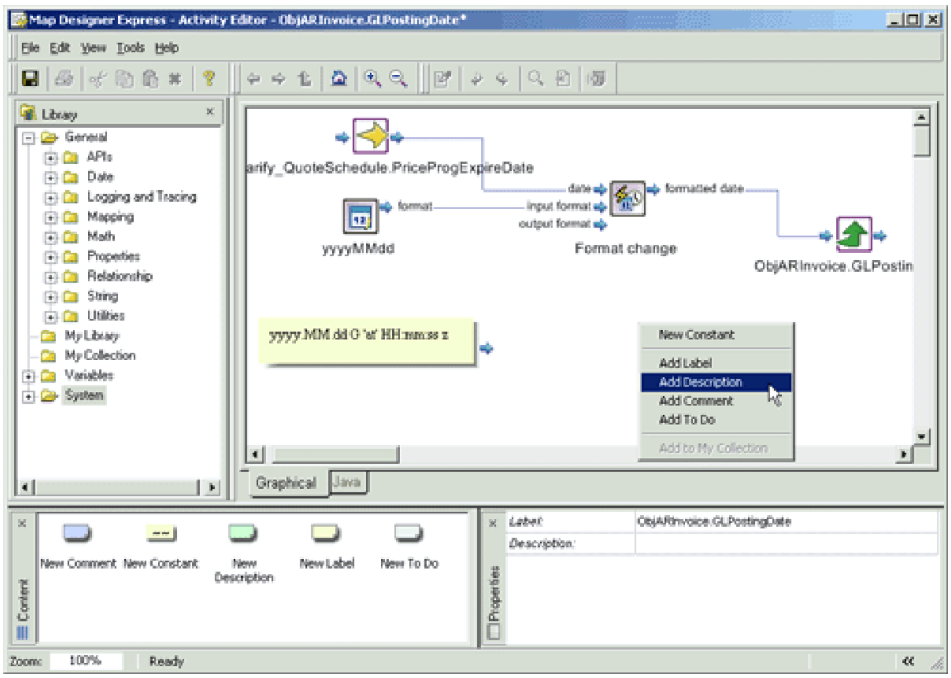


Figura 73. Aggiunta di una descrizione tramite il menu contestuale

Risultato: il componente Descrizione verrà creato nell'area di disegno.

11. Immettere la descrizione nel componente Descrizione facendo clic sull'area modificabile del componente e digitando direttamente nel componente. E' possibile modificare le dimensioni della Descrizione facendo clic sull'angolo destro inferiore del componente Descrizione e spostando il cursore. La Figura 74 mostra l'aggiunta della descrizione.

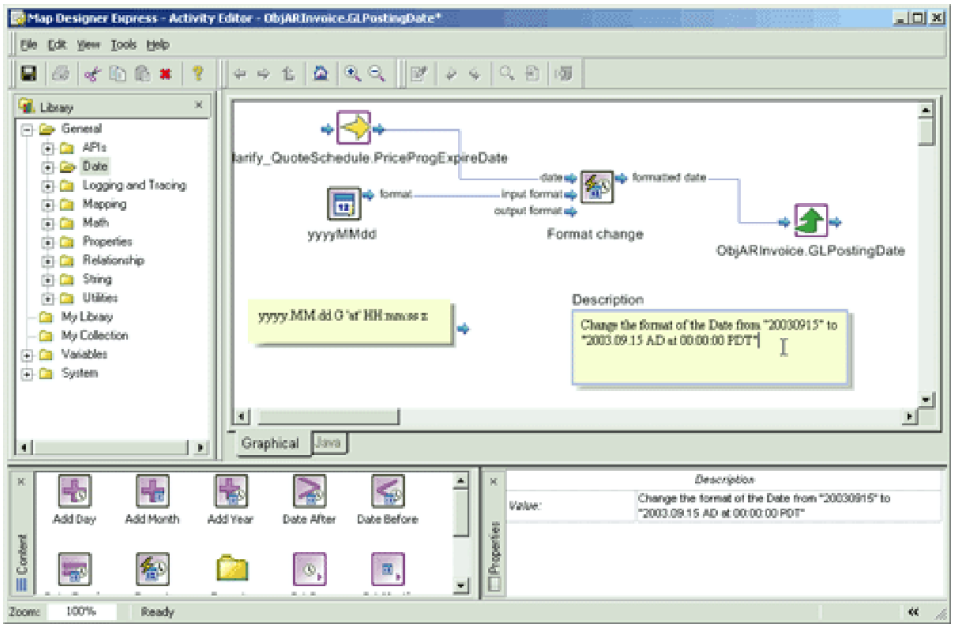


Figura 74. Aggiunta della descrizione

12. Salvare l'attività selezionando Sul progetto o Sul file dal menu secondario File > Salva o facendo clic sul pulsante Salva mappa sul progetto o Salva mappa sul file nella barra strumenti Standard. La Figura 75 mostra il salvataggio dell'attività.

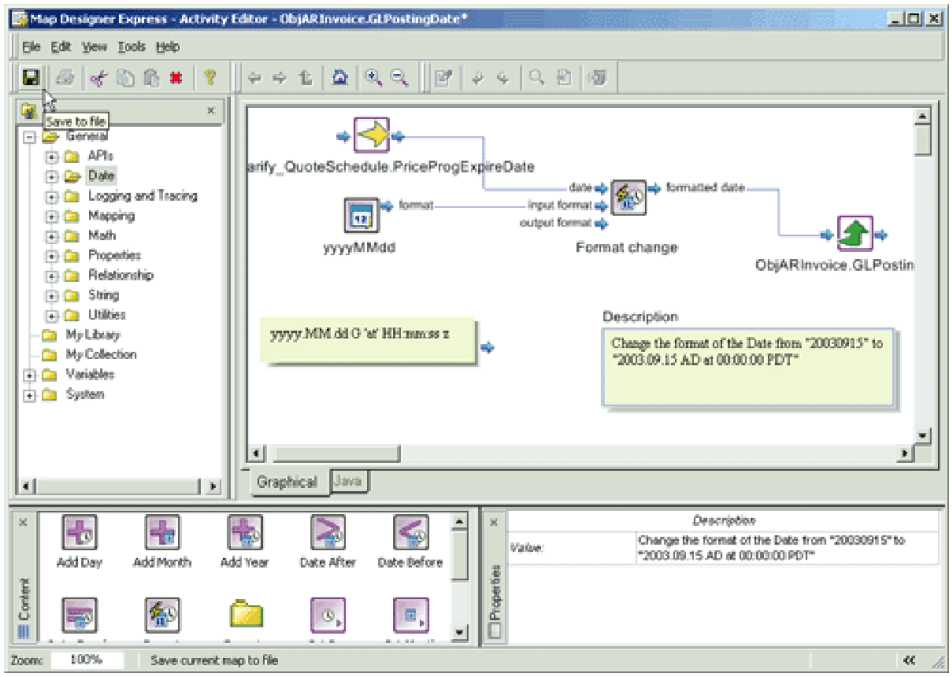


Figura 75. Salvataggio dell'attività

Esempio 3: utilizzo della ricerca statica per la conversione

Il seguente esempio illustra illustra il blocco funzione di relazione *Ricerca statica* in Activity Editor.

In InterChange Server Express, una relazione di ricerca statica di solito è composta da due o più tabelle di relazione. Ad esempio, si consideri un sistema composto da tre applicazioni finali, come illustrato nella Figura 76.

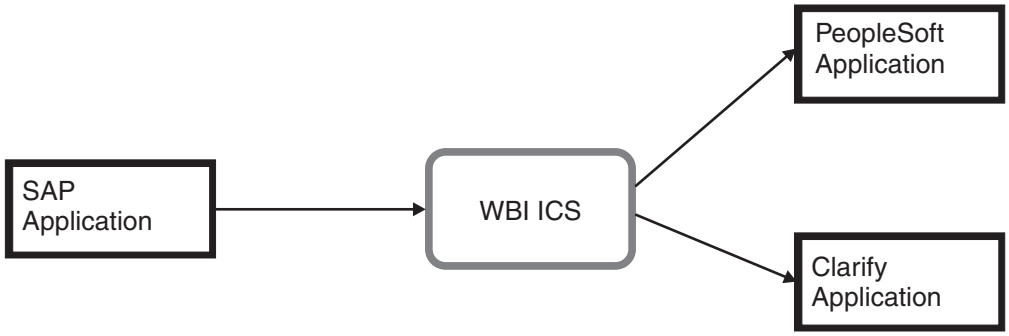


Figura 76. Relazione Ricerca statica con tre applicazioni finali

Ognuna di queste tre applicazioni ha una rappresentazione diversa per le informazioni ″Stato″, come illustrato nella Tabella 59.

Tabella 59. Rappresentazione specifica per applicazione delle informazioni sullo stato

	Applicazione SAP	**Applicazione PeopleSoft**	**Applicazione Clarify**
California	CA	01	Stato1
Washington	WA	02	Stato2
Hawaii	HI	03	Stato3
Delaware	DE	04	Stato4

Quando le informazioni sullo stato vengono inviate al sistema WebSphere Business Integration Server Express dall'applicazione SAP, il codice SAP dello stato specificato viene inviato a InterChange Server Express. Ma quando InterChange Server Express deve passare queste informazioni ad altre applicazioni, l'informazione sullo stato deve essere convertita in un formato comprensibile per l'applicazione di destinazione. Per poter fare ciò, il sistema ha bisogno di una rappresentazione generica dell'informazione ″Stato″. Con la rappresentazione generica il sistema può elaborare la logica di business in maniera generale ed unificata; la rappresentazione generica verrà convertita nel formato specifico per l'applicazione solo quando necessario.

Quindi, nell'esempio precedente, verrebbe creata una relazione di ricerca statica per eseguire questa conversione di ″Stato″, con i dati specifici per l'applicazione nella forma di partecipanti gestiti da WebSphere Business Integration Server Express. Con questa impostazione, viene utilizzato un ID generico per rappresentare le informazioni dello stato nel sistema WebSphere Business Integration Server Express. La Tabella 60 mostra questa rappresentazione.

Tabella 60. Rappresentazione generica delle informazioni sullo stato

	ID generico	**Applicazione SAP**	**Applicazione PeopleSoft**	**Applicazione Clarify**
California	1	CA	01	Stato1
Washington	2	WA	02	Stato2

Tabella 60. Rappresentazione generica delle informazioni sullo stato (Continua)

	ID generico	Applicazione SAP	Applicazione PeopleSoft	Applicazione Clarify
Hawaii	3	HI	03	Stato3
Delaware	4	DE	04	Stato4

I dati specifici per l'applicazione vengono convertiti in un ID generico appena entra nel sistema InterChange Server Express, e l'ID generico viene convertito in dati specifici per l'applicazione appena esce dal sistema. Questa conversione di dati viene illustrata nella Figura 77.

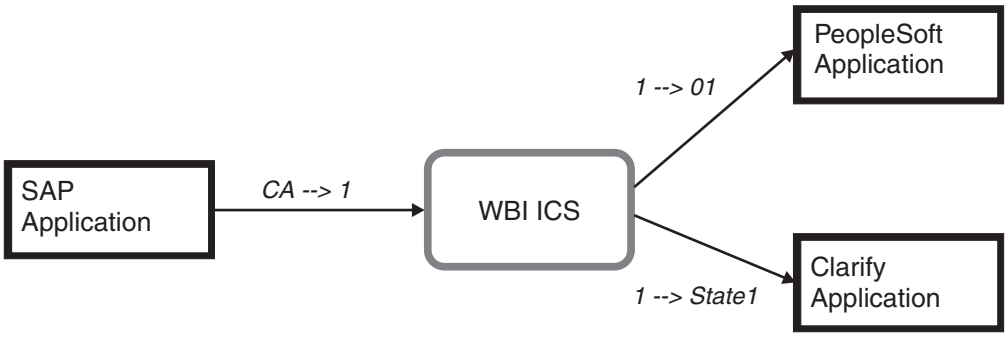


Figura 77. Conversione dati da specifici per applicazione a generici e a specifici per applicazione

La conversione ID di solito viene eseguita in mappe che convertono gli oggetti business specifici per applicazione in oggetti business generici o viceversa. Ad esempio nella mappa SAP-a-Generico, si eseguirebbe una ricerca statica per i dati "CA" e lo si convertirebbe in una rappresentazione generica comprensibile per InterChange Server Express, "1". E nella mappa Generico-a-Clarify, verrebbe eseguita una ricerca statica del dato generico "1", che verrebbe convertito in "State1". In entrambe le mappe è richiesto un'unica ricerca statica.

La Figura 78 mostra come utilizzare il blocco funzione Ricerca statica per convertire i dati specifici per SAP in dati generici di stato InterChange Server Express per l'elaborazione in InterChange Server Express.

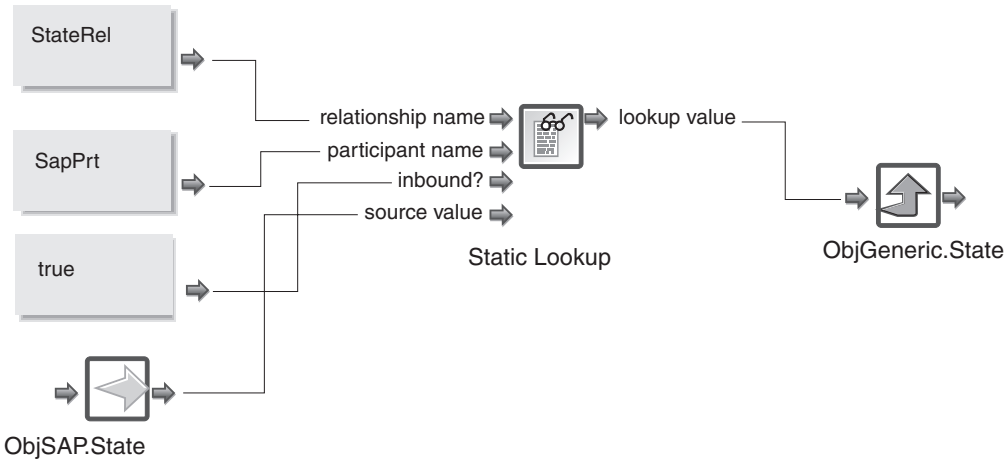


Figura 78. Utilizzo del blocco funzione Ricerca statica per convertire i dati di stato specifici per SAP in dati di stato generici di InterChange Server Express

Allo stesso modo, il blocco funzione Ricerca statica viene utilizzato per convertire i dati di stato generici di InterChange Server Express in dati di stato specifici per Clarify nella mappa Generico-a-Clarify. Questo viene illustrato nella Figura 79.

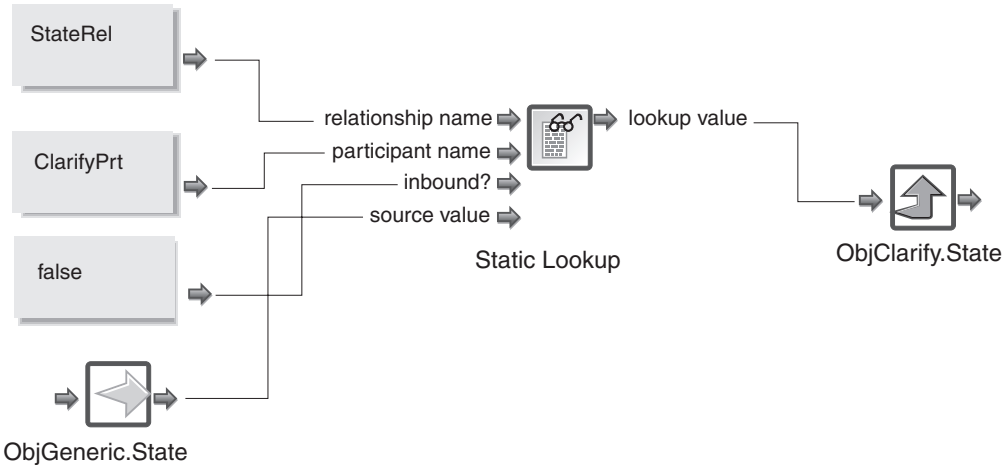


Figura 79. Utilizzo del blocco funzione Ricerca statica per convertire dati generici di stato InterChange Server Express in dati di stato specifici per Clarify

Di solito, in una relazione di ricerca statica si convertono dati specifici per applicazione in dati generici oppure dei dati generici in dati specifici per applicazione. In questi scenari, viene utilizzato un solo blocco funzione Ricerca statica. Ma in casi particolari, in cui si desidera cercare direttamente una coppia nome-valore, sono richiesti due blocchi funzione di ricerca statica.

Per ulteriori informazioni sulla definizione e l'utilizzo delle relazioni statiche, consultare Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249.

Esportazione di servizi Web in Activity Editor

Un *servizio Web* fa parte di un progetto InterChange Component Library nel System Manager, esattamente come gli oggetti business e le mappe. Dopo aver registrato, testato e verificato il servizio Web, i suoi servizi e metodi possono essere esportati come blocchi funzione in Activity Editor ed essere utilizzati nelle mappe, come gli altri blocchi funzione.

Per informazioni sulla registrazione, il test, la verifica e l'esportazione di un servizio Web da System Manager in Activity Editor, consultare il manuale *Guida all'implementazione del sistema*.

Utilizzo dei servizi Web in Activity Editor

Dopo aver esportato un servizio Web da System Manager, è necessario riavviare Activity Editor. Quando si apre Activity Editor, il servizio Web esportato viene aggiunto come categoria in Libreria personale. Ha la stessa funzionalità delle altre categorie contenute in Libreria personale.

La Figura 80 mostra la categoria dei servizi Web ed i blocchi funzione in Activity Editor dopo l'esportazione da System Manager.

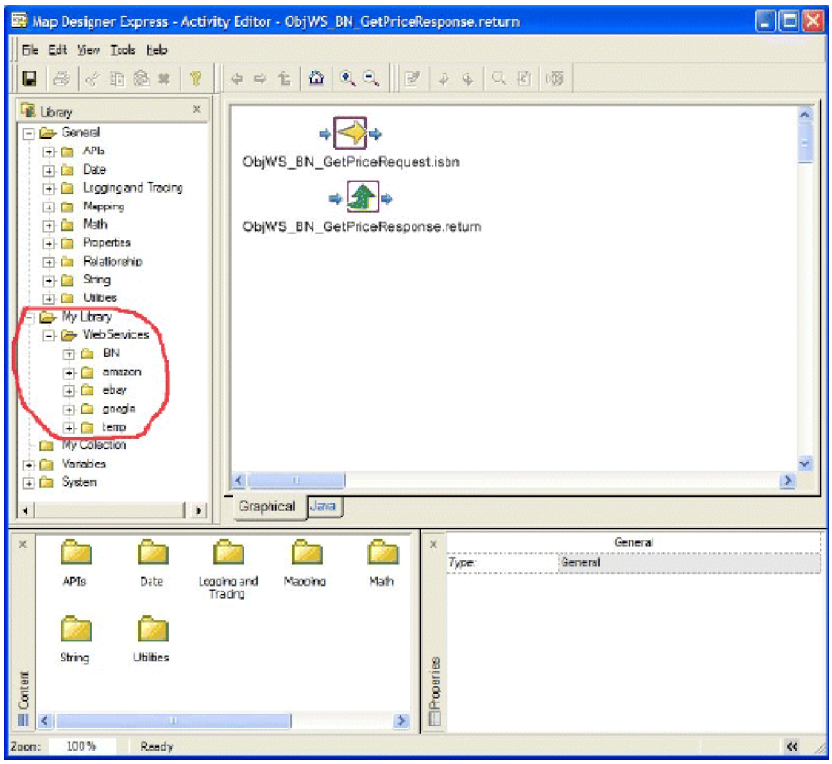


Figura 80. Categoria dei servizi Web in Activity Editor

Come per gli altri blocchi funzione in Activity Editor, per utilizzare qualsiasi blocco funzione dei servizi Web basta trascinarlo e rilasciarlo e collegare gli input e gli output. Per informazioni sull'utilizzo di Activity Editor, consultare "Operazioni con le definizioni di attività" a pagina 118.

Esempio di utilizzo di un servizio Web in una mappa

L'esempio che segue illustra come richiamare un servizio Web utilizzando Activity Editor per modificare il codice avviamento postale di un attributo origine per la temperatura di una città ed assegnare la modifica all'attributo di destinazione.

Effettuare le seguenti operazioni:

1. Dalla scheda Diagramma di Map Designer Express, creare una trasformazione personalizzata trascinando l'attributo ObjTemperatrueInput.zipcode dell'oggetto business origine nell'attributo dell'oggetto business di destinazione ObjTemperatureOutput.currentTemperature. Fare clic sull'icona della regola di trasformazione Personalizzata per avviare Activity Editor.

   La Figura 81 mostra la trasformazione personalizzata.

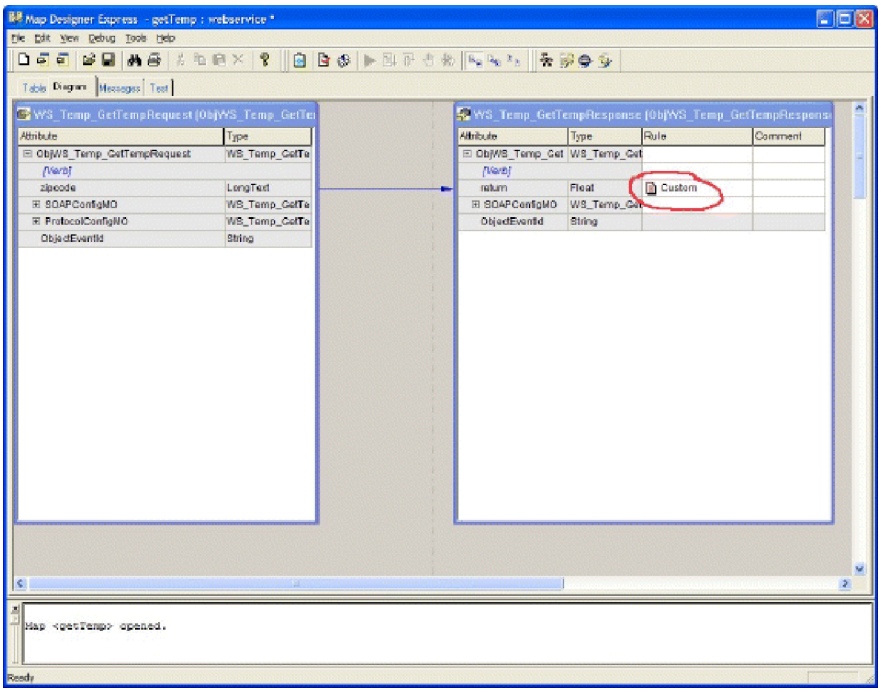


Figura 81. Creazione della trasformazione personalizzata

2. Selezionare la categoria dei servizi Web nella finestra Libreria per visualizzare i blocchi funzione disponibili in quella categoria, come icone, nella finestra Contenuto.
3. Trascinare il blocco funzione dei servizi Web getTemp dalla finestra Contenuto all'area di disegno per la modifica.
4. Collegare la porta output dell'icona per l'attributo dell'oggetto business di origine ObjTemperatureInput.zipcode alla porta input "codiceavviamentopostale" del blocco funzione getTemp; quindi collegare la porta output "risultato" del blocco funzione getTemp alla porta input dell'icona dell'attributo dell'oggetto business di destinazione ObjTemperatureOutput.currentTemperature.

   La Figura 82 mostra gli input e gli output collegati del blocco funzione getTemp.

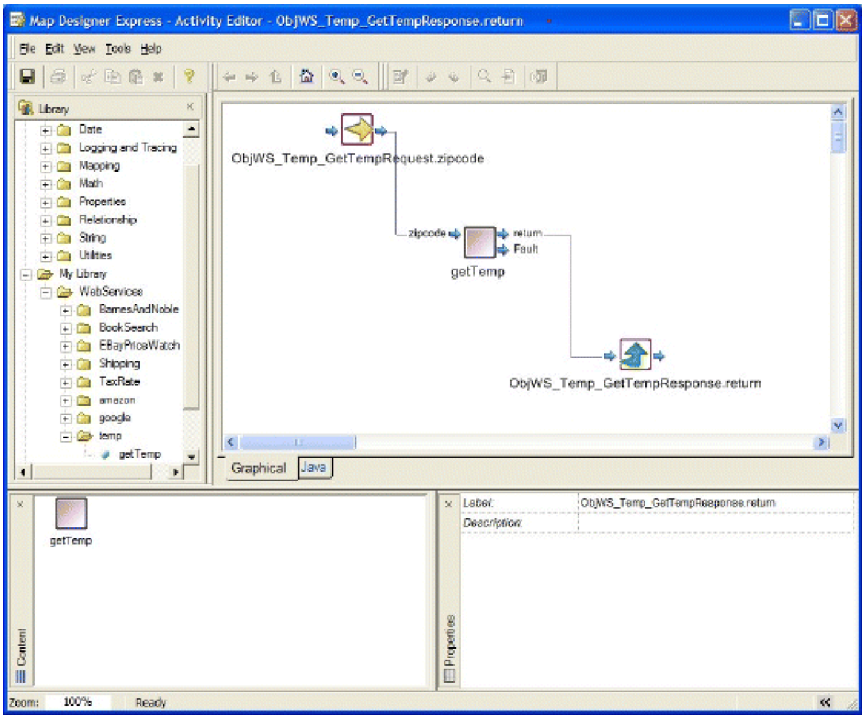


Figura 82. Collegamento di input e output

5. Salvare la maschera dell'attività e la mappa.
6. Andare alla vista Test in Map Designer Express. Immettere un codice di avviamento postale valido nel campo CAP origine. Fare clic su Esegui debug mappa. E' possibile scegliere di distribuire la mappa e gli oggetti business al server, se non è stato già fatto.

   Risultato: al termine dell'esecuzione del test, si vedrà la temperatura corrente per il codice di avviamento postale nell'oggetto business di destinazione.

   La Figura 83 mostra come il codice di avviamento postale 94010 dell'attributo dell'oggetto business di origine è stato trasformato in 59 gradi per l'attributo dell'oggetto business di destinazione.

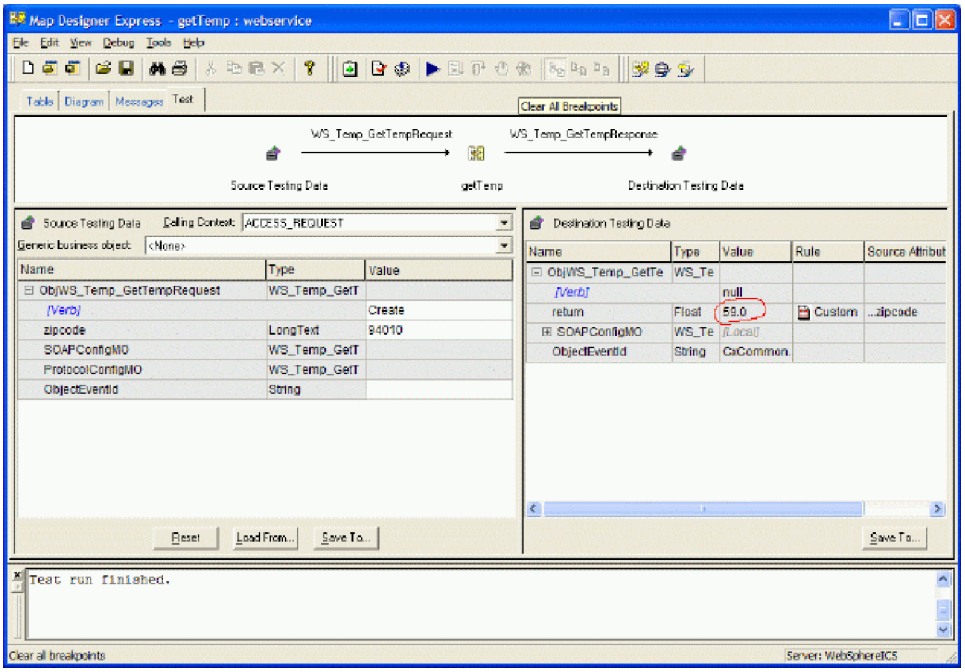


Figura 83. Risultati della vista Test

Utilizzo della funzione bidirezionale in Activity Editor

WebSphere Business Integration Server Express supporta i linguaggi bidirezionali. Questo supporto è in un formato standard bidirezionale di tipo Windows (logico da sinistra a destra). Per questo supporto, anche tutte le mappe supportano i linguaggi bidirezionali. Tuttavia, l'immissione di dati in una mappa può provenire da:

Un adattatore che supporta i linguaggi bidirezionali. Per stabilire se l'adattatore supporta i linguaggi bidirezionali, consultare la guida per l'utente del proprio adattatore.

Un componente che non supporta i linguaggi bidirezionali, un adattatore che non li supporta o i dati importati da un origine esterna dove il supporto bidirezionale è sconosciuto.

Le incongruenze del formato bidirezionale causano la restituzione di risultati non corretti da parte dei confronti eseguiti all'interno di una mappa. Questi tipi di errori possono essere evitati nel modo seguente:

- Accettando input solo da origini che applicano lo stesso formato bidirezionale del sistema WebSphere Business Integration Server Express, come gli adattatori già abilitati a questo supporto.
- Abilitando i connettori per questa mappa per applicare il formato bidirezionale corretto (consultare "Abilitazione dei connettori per linguaggi bidirezionali" nel manuale *Guida allo sviluppo della collaborazione*).
- Utilizzando le API nella classe CwBidiEngine per trasformare tutti i dati in un formato bidirezionale congruente (consultare Capitolo 12, "Classe CwBidiEngine", a pagina 379).

InterChange Server Express abilita automaticamente la funzione BiDi tramite i seguenti dieci connettori: `Email`, `JDBC`, `JMS`, `JText`, `Lotus Domino`, `MQ Series`,

PS, SAP, servizi Web e XML. Quindi, quando i dati in formato BiDi Windows utilizzano questi connettori abilitati in un servizio Web, non è necessaria una configurazione particolare.

Nel caso in cui un servizio Web operi con dati BiDi non in formato BiDi Windows, sono possibili due risultati:

- La connessione a quel servizio potrebbe all'improvviso non riuscire.
- I dati BiDi che sono in un formato diverso da CWBF (Common Windows Bidirectional Format) danno dei risultati imprevedibili durante l'elaborazione dei dati, poiché i dati vengono confrontati con i dati in formato CWBF. In altre parole, delle informazioni identiche vengono gestite in formati BiDi diversi. Per correggere questa potenziale situazione è necessario eseguire i passi riportati di seguito per la distribuzione del servizio Web all'interno di Activity Editor.

Passi per la distribuzione di API Bidi in un servizio Web

Per distribuire BiDi in un servizio Web eseguire i seguenti passi:

1. Registrare il servizio Web.

   Per informazioni sulla registrazione, il test, la verifica e l'esportazione di un servizio Web, consultare il manuale *Guida all'implementazione del sistema*.
2. Testare il servizio Web utilizzando i dati BiDi per stabilire lo standard del formato BiDi.
3. Esportare il servizio Web in Activity Editor.
4. Progettare il flusso di dati utilizzando Activity Editor. La Figura 84 mostra un esempio di processo di progettazione BiDi.

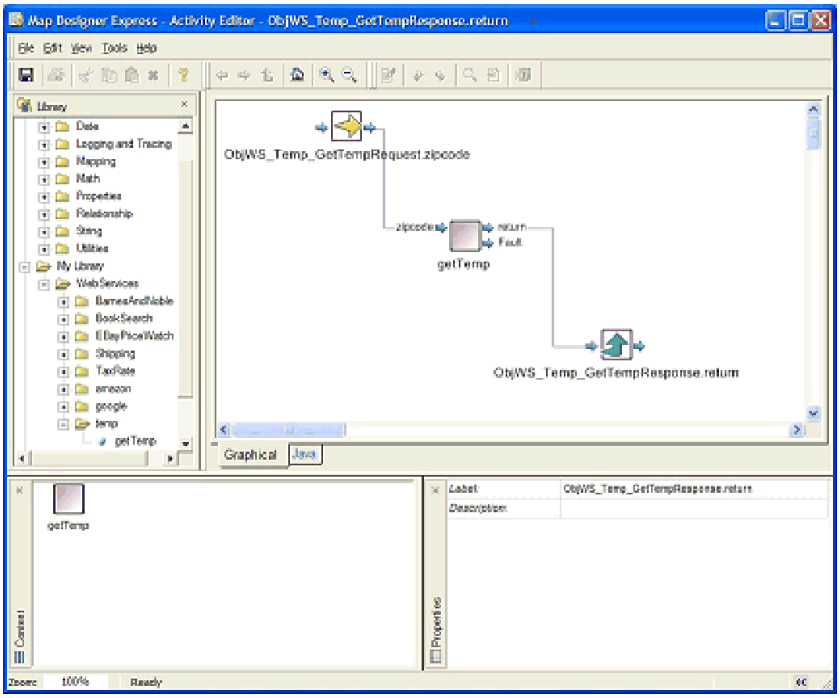


Figura 84. Finestra di Activity Editor con elementi BiDi

5. Aggiungere delle chiamate all'API BiDi all'interno del codice generato subito prima che i dati vengano inviati al servizio Web e subito dopo aver ricevuto la risposta, per conservare la coerenza dei dati BiDi.

Nota: Per ulteriori informazioni sulle trasformazioni BiDi del contenuto istanza BO utilizzate nell'API BiDi, consultare Capitolo 12, "Classe CwBidiEngine", a pagina 379.

Importazione di pacchetti Java e di altro codice personalizzato

Map Designer Express fornisce due modi per importare dei pacchetti Java ed altro codice personalizzato:

Segue una descrizione di ciascun metodo.

Importazione di librerie Jar come blocchi funzione di attività

Oltre ad utilizzare i blocchi funzione standard forniti da Activity Editor, Map Designer Express consente di importare la propria libreria Java da utilizzare come blocchi funzione in Activity Editor. L'importazione di librerie Jar personalizzate nelle impostazioni delle attività abilita l'utilizzo di qualsiasi metodo pubblico contenuto nella libreria Jar come blocchi funzione in Activity Editor.

Passi per importare le librerie Jar come blocchi funzione di attività

Prima di iniziare: è necessario esportare le proprie classi Java in un file `.jar` .

Per importare una libreria Jar in Activity Editor, eseguire i seguenti passi:

1. In System Manager, aprire la vista Impostazioni attività facendo clic su Finestra > Mostra vista > Altro e selezionando Impostazione attività dalla categoria System Manager.
2. Fare clic con il pulsante destro del mouse su Librerie BuildBlock e selezionare Aggiungi libreria. La Figura 85 mostra la vista Impostazioni attività per l'aggiunta di una libreria Jar personalizzata.

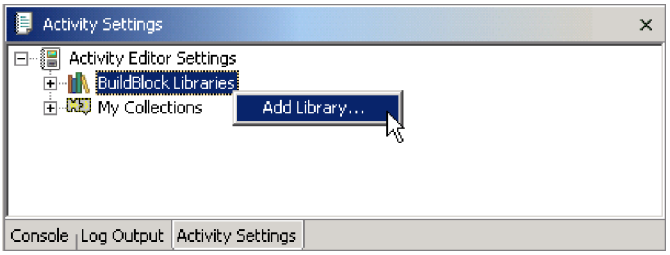


Figura 85. Vista Impostazioni attività

3. Nella finestra di dialogo Apri file, andare ai file `.jar` personalizzati e selezionare Apri.

   System Manager tenterà di importare il file `.jar personalizzato` da utilizzare come blocchi funzione in Activity Editor. Se il file viene importato correttamente, il relativo nome verrà visualizzato nelle Librerie BuildBlock nella vista Impostazioni attività.

   Suggerimento: dopo aver importato i propri file `.jar` personalizzati nella finestra Impostazioni attività, quando si compileranno le proprie mappe e la maschera di collaborazione in System Manager, il file `.jar` personalizzato verrà automaticamente incluso nel `CLASSPATH` di compilazione. Per preparare InterChange Server Express alla compilazione, verificare che il relativo

CLASSPATH includa il file custom.jar. Per informazioni sulla configurazione di InterChange Server Express per l'importazione dei file .jar personalizzati, consultare "Importazione di classi di terzi in InterChange Server Express" a pagina 176.

4. Riavviare Map Designer Express.

   Regola: dopo qualsiasi modifica alle impostazioni delle vista Impostazioni attività, è necessario riavviare Map Designer Express perché le modifiche divengano effettive in Activity Editor.

Risultato: quando si apre Activity Editor, la libreria Jar personalizzata verrà visualizzata nella finestra Libreria nella Libreria personale in Activity Editor. Per impostazione predefinita, i blocchi funzione personalizzati disponibili vengono elencati secondo la struttura del pacchetto. E' possibile utilizzarli in un'attività nello stesso modo dei blocchi funzione standard.

Personalizzazione delle impostazioni di visualizzazione delle librerie Jar personalizzate

E' possibile personalizzare le impostazioni di visualizzazione dei blocchi funzione importati in Activity Editor, come il nome e l'icona, modificando le proprietà della libreria Jar personalizzata. Per farlo, eseguire i passi riportati di seguito:

- Visualizzare la finestra Proprietà relativa alla libreria Jar personalizzata facendo clic con il pulsante destro del mouse sulla libreria Jar personalizzata elencata nelle Librerie BuildBlock nella vista Impostazioni attività in System Manager.

  Risultato: quando viene aperta la finestra Proprietà relativa alla libreria Jar personalizzata, elencherà i blocchi funzione disponibili in quella libreria personalizzata in una struttura ad albero sul lato destro della finestra di dialogo. I blocchi funzione disponibili sono elencati come nodi secondari nella classe Java e nel pacchetto che li contiene.

Per il pacchetto e le classi Java, è possibile personalizzare il nome visualizzato della voce e se Activity Editor deve visualizzare il pacchetto/classe Java nella struttura ad albero della Libreria personale modificando la casella di controllo "Nascondi livello nella vista ad albero." Se questa opzione è abilitata, Activity Editor non visualizzerà questa voce nella struttura secondaria della Libreria personale. Questa opzione di solito è utile quando i metodi Java nella libreria Jar personalizzata sono in una classe Java che si trova in un pacchetto con molti livelli, ed abilitando questa opzione si può organizzare meglio la struttura secondaria della Libreria personale in Activity Editor.

La Figura 86 a pagina 174 mostra la finestra di dialogo per la personalizzazione della visualizzazione della libreria Jar.

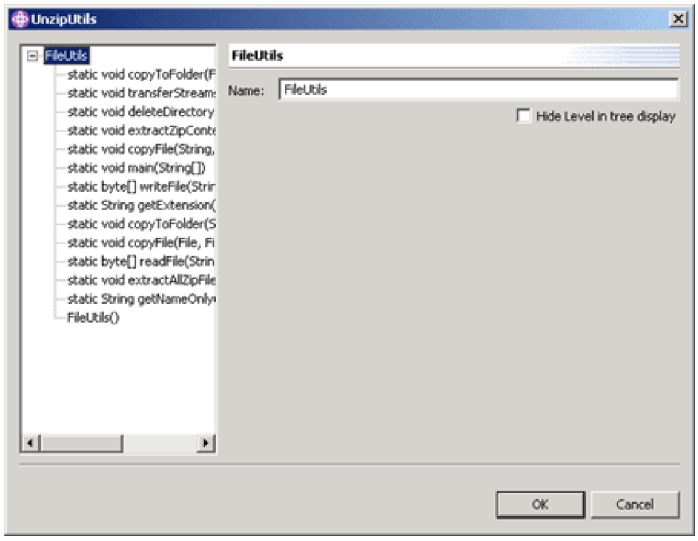


Figura 86. Finestra di dialogo Proprietà per la personalizzazione della visualizzazione della libreria Jar

Per i metodi Java utilizzati come blocchi funzione in Activity Editor, è possibile specificare il nome visualizzato del blocco funzione, la descrizione, l'icona ed il nome visualizzato del parametro nella finestra Proprietà. Quando si importa un'icona per il blocco funzione, l'icona selezionata verrà copiata nella cartella Impostazioni attività e sarà disponibile per altri blocchi funzione nello stesso pacchetto.

Raccomandazione: se si sceglie di importare un'icona da utilizzare per il proprio blocco funzione, l'icona deve essere di 32 pixel per 32 pixel, come dimensione e deve essere in formato .bmp. La profondità colore dell'icona può essere fino a 24 bit.

La Figura 87 a pagina 175 mostra la finestra di dialogo Proprietà per la personalizzazione della visualizzazione dei blocchi funzione della libreria Jar.

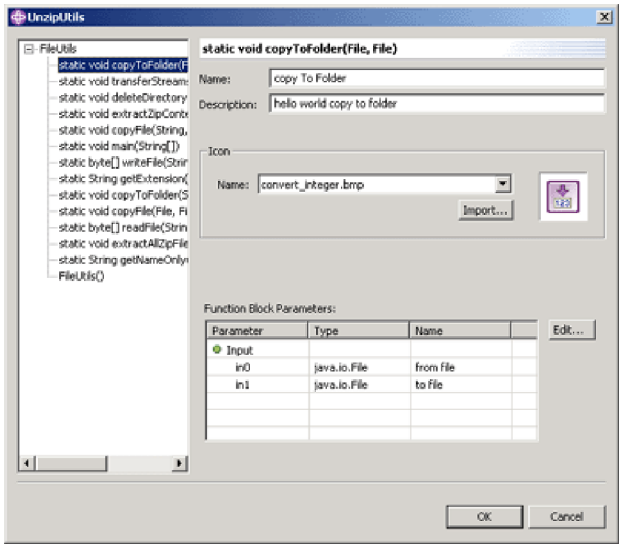


Figura 87. Finestra di dialogo Proprietà per la personalizzazione della visualizzazione dei blocchi funzione della libreria Jar

Regola: dopo qualsiasi modifica alle impostazioni delle vista Impostazioni attività, è necessario riavviare Map Designer Express perché le modifiche divengano effettive in Activity Editor.

Nota: E' possibile importare ed esportare le impostazioni attività utilizzando l'Importazione/Esportazione guidata del Workbench Eclipse. Per ulteriori informazioni, consultare la documentazione relativa al Workbench Eclipse.

Importazione mediante la finestra di dialogo Proprietà mappa

Map Designer Express importa automaticamente i pacchetti Java necessari per eseguire una mappa. Se si scrive del codice di trasformazione che utilizza dei metodi provenienti da un altro pacchetto Java o se si utilizza il pacchetto Utilità mappa (`MapUtils`), è necessario importare il pacchetto nella mappa. E' possibile anche scrivere il proprio codice Java personalizzato fuori da Map Designer Express ed importarlo in una mappa per utilizzarlo nel codice di trasformazione.

Nota: Per lo sviluppo di mappe deve essere installata la versione di JDK (Java Development Kit) appropriata al rilascio di IBM WebSphere Business Integration Server Express.

Attenzione: Map Designer Express non è in grado di eseguire il debug o di verificare la logica di nessun codice importato in una mappa. Di conseguenza, lo sviluppatore di mappe è responsabile di tutti gli errori ed eccezioni causati dal codice importato.

Passi per importare i pacchetti Java o altro codice personalizzato

Per importare un pacchetto Java o un altro codice personalizzato:

1. Visualizzare la scheda Generale della finestra di dialogo Proprietà mappa nel seguente modo:

   Dal menu Modifica, selezionare Proprietà mappa. Per informazioni su altri modi di visualizzare la finestra di dialogo Proprietà mappa, consultare "Specifica delle informazioni sulla proprietà della mappa" a pagina 62. Viene visualizzata la finestra di dialogo Proprietà mappa (consultare Figura 92 a pagina 199).
2. Inserire le istruzioni di `importazione` nel `blocco dichiarazione file mappa`. E' anche possibile immettere altre istruzioni Java nel `blocco dichiarazione locale mappa`.

   Nota: Quando si compila la mappa, il compilatore Java cerca i pacchetti importati nella directory definita dalla variabile di ambiente `CLASSPATH`. Se si importa un pacchetto in una mappa e si distribuisce la mappa ad un altro sistema InterChange Server Express prima di compilarla, accertarsi di consegnare con la mappa anche il pacchetto importato.

   Per i requisiti di compilazione di una mappa con System Manager, consultare "Compilazione di una mappa" a pagina 89.

   Tutte le istruzioni si eseguono all'inizio della mappa, *prima* di eseguire qualsiasi passo di trasformazione.
3. Fare clic su OK per chiudere la finestra di dialogo Proprietà mappa.

Passi per l'importazione delle utilità della mappa

Per poter utilizzare il pacchetto Utilità mappa è necessario importarlo nella mappa eseguendo questi passi:

1. Verificare che il file `CollabUtils.jar` sia nella directory <ProductDir>\lib.
2. Verificare che il file `start_server.bat` (o `CWSharedEnv.sh`) contenga un riferimento al file `CollabUtils.jar`.

   Nota: I passi 1 e 2 sono necessari per la compilazione del server.
3. Modificare la preferenza Classpath del compilatore per la compilazione delle mappe e le maschere di collaborazione provenienti da System Manager:
   a. In System Manager, selezionare Finestra > Preferenze per aprire la finestra di dialogo Preferenze.
   b. Espandere le Preferenze di System Manager e selezionare Compilatore.
   c. Nella pagina di preferenze Compilatore, fare clic su Nuovo e selezionare il file `.jar` da includere nel `CLASSPATH` di compilazione per le mappe e le maschere di collaborazione.
4. Da Map Designer Express, aprire la mappa che deve utilizzare il pacchetto Utilità mappa.
5. Visualizzare la finestra di dialogo Proprietà mappa.
6. Nella finestra di dialogo Proprietà mappa, digitare la seguente istruzione `import` nel `Blocco dichiarazione file mappa`:

   ```
   import com.crossworlds.MapUtils.*;
   ```
7. Fare clic su OK per chiudere la finestra di dialogo Proprietà mappa.

Importazione di classi di terzi in InterChange Server Express

Se le classi importate sono in un pacchetto di terzi, piuttosto che nel JDK, per poter configurare la compilazione del server, è necessario aggiungerle al percorso delle classi importate nella variabile `JCLASSES`.

Raccomandazione: è necessario utilizzare qualche tipo di meccanismo per differenziare le classi standard da quelle personalizzate in `JCLASSES`.

Esempio: è possibile creare una nuova variabile per contenere solo le classi personalizzate ed aggiungere questa nuova variabile a `JCLASSES`, eseguendo i seguenti passi:

1. Creare una nuova proprietà della mappa, ad esempio una denominata `DEPENDENCIES`.
2. Collocare il file `CwMacroUtils.jar` nella sua directory.

   Esempio: Creare una directory `dependencies` sotto la directory del prodotto e collocarvi il file jar.
3. Aggiungere la directory `dependencies` al file utilizzato per avviare InterChange Server Express (per impostazione predefinita, `start_server.bat` o `CWSharedEnv.sh`), che si trova nella directory `bin` sotto la directory del prodotto. Ad esempio, aggiungere la seguente voce per Linux:

   ```
   set DEPENDENCIES=$ProductDir/
   dependencies/CwMacroUtils.jar
   ```

   Aggiungere la seguente voce per Windows:

   ```
   set DEPENDENCIES="%ProductDir%"\dependencies\
   CwMacroUtils.jar
   ```
4. Aggiungere `DEPENDENCIES` alla voce `JCLASSES`:

   Per Linux, aggiungere:

```
set JCLASSES=$JCLASSES:ExistingJarFiles:
$DEPENDENCIES
```

Per Windows, aggiungere:

```
set JCLASSES=ExistingJarFiles
;%DEPENDENCIES%
```

5. In ogni mappa che utilizza le classi includere il *PackageName.ClassName* specificato nel file `CwMacroUtils.jar`.
6. Riavviare InterChange Server Express per rendere i metodi disponibili per le mappe.

Nota: Verificare di aver modificato la preferenza Classpath del compilatore per la compilazione delle mappe e delle maschere di collaborazione provenienti da System Manager. Per procedere in tal modo, eseguire i seguenti passi:

1. In System Manager, selezionare Finestra > Preferenze per aprire la finestra di dialogo Preferenze.
2. Espandere le Preferenze di System Manager e selezionare Compilatore.
3. Nella pagina di preferenze Compilatore, fare clic su Nuovo e selezionare il file `.jar` da includere nel `CLASSPATH` di compilazione per le mappe e le maschere di collaborazione.

Linee guida: quando si importa una classe personalizzata, è possibile che si riceva un errore che indica che il software non ha trovato la classe personalizzata. Se questo si verifica, controllare quanto segue:

- Verificare che la classe personalizzata faccia parte di un pacchetto. E' buona norma nella programmazione collocare le classi personalizzate in un pacchetto. Verificare che il codice della classe personalizzata includa una corretta istruzione pacchetto e che venga collocata all'inizio del file origine, prima di qualsiasi dichiarazione di classe o di interfaccia.
- Verificare che l'istruzione `import` sia corretta nel codice di definizione della mappa. L'istruzione `import` deve fare riferimento al nome pacchetto corretto; può inoltre specificare il nome della classe personalizzata o far riferimento a tutte le classi contenute nel pacchetto.

  Esempio: se il nome del pacchetto è `COM.acme.graphics` e la classe personalizzata è `Rectangle`, è possibile importare l'intero pacchetto:

  ```
  import COM.acme.graphics.*;
  ```

  Oppure, è possibile importare solo la classe personalizzata Rectangle:

  ```
  import COM.acme.graphics.Rectangle;
  ```

- Accertarsi di aver aggiornato la variabile di ambiente `CLASSPATH` in modo da includere il percorso per il pacchetto contenente la classe personalizzata oppure quello per la classe stessa, se non è contenuta in un pacchetto.

Esempio: quando si importa una classe personalizzata, è possibile creare una cartella denominata `%ProductDir%\lib\com\<ProductDir>\pacchetto`, dove `pacchetto` è il nome del pacchetto. Quindi, collocare il file della classe personalizzata nella cartella appena creata. Infine, nella variabile `CLASSPATH` nel file `start_server.bat`, includere il percorso `%ProductDir%\lib`.

Utilizzo delle variabili

Una variabile è un segnaposto per un valore nel codice Java. Questa sezione fornisce le seguenti informazioni sull'utilizzo delle variabili nel codice di trasformazione:

Utilizzo delle variabili e degli attributi generati dell’oggetto business

Questa sezione fornisce le informazioni sulla creazione di variabili di oggetti business per gli oggetti business origine e di destinazione. Quando si aggiunge un oggetto business alla mappa, Map Designer Express genera automaticamente quanto segue:

- Un nome istanza

  Il nome istanza che Map Designer Express genera è una variabile locale dichiarata al sistema che si può utilizzare per questo oggetto business nel codice di mappatura. Viene assegnato un prefisso con le lettere `Obj`, seguite dal nome della definizione dell’oggetto business.

  Esempio: se si aggiunge `Customer` alla mappa, il suo nome istanza è `ObjCustomer`. Map Designer Express genera un nome istanza sia per gli oggetti business origine che per quelli di destinazione.

  Quando si scrive del codice in Activity Editor, si utilizza il nome istanza per far riferimento all’oggetto business e ai suoi attributi.
- Un indice per l’oggetto business in un vettore oggetto business (se l’oggetto business è a cardinalità multipla)

  L’indice dell’oggetto business rappresenta l’ordine dell’oggetto business origine o di destinazione. Il numero di indice dei primi oggetti business origine e di destinazione in una mappa è zero. Gli altri oggetti business prendono il successivo numero disponibile, ad esempio 1, 2, 3, e così via.

  Quando viene eseguita la mappa, il numero di indice rappresenta la posizione dell’oggetto business nel vettore che viene passato nella mappa (oggetti business origine) oppure restituito dalla mappa (oggetti business di destinazione).

Map Designer Express visualizza queste informazioni nelle seguenti ubicazioni:

- Nella scheda Oggetti business della finestra di dialogo Proprietà mappa

  Fare clic con il pulsante destro del mouse sulla barra del titolo della finestra oggetto business e selezionare Proprietà dal menu contestuale. Viene visualizzata la finestra di dialogo Proprietà mappa con la scheda Oggetti business che mostra l’oggetto business selezionato evidenziato nell’elenco. Questa scheda visualizza sia il nome dell’istanza che il relativo indice all’interno del vettore oggetto business (se l’oggetto business è a cardinalità multipla).
- Nella scheda Tabella—nel pannello dell’oggetto business
- Nella scheda Diagramma—nella barra del titolo della finestra dell’oggetto business nel seguente formato:

  La barra del titolo visualizza il nome istanza dell’oggetto business.

Nota: E’ possibile specificare se Map Designer Express deve visualizzare i nomi delle variabili per gli oggetti business origine e di destinazione con l’opzione `Definizione mappa: mostra nome istanza oggetto business`. Per impostazione predefinita, questa opzione è abilitata e Map Designer Express visualizza questi nomi variabile (`Obj`*`BusObj`*) in entrambe le schede Tabella e Diagramma. Quando l’opzione è disabilitata, Map Designer Express visualizza solo i nomi degli oggetti business di origine e di destinazione. E’ possibile modificare l’impostazione di questa opzione nella scheda Generale

della finestra di dialogo Preferenze. Per ulteriori informazioni, consultare “Specifica delle Preferenze generali” a pagina 24.

Passi per la modifica delle variabili dell’oggetto business

E’ possibile modificare queste variabili dell’oggetto business dalla scheda Oggetti business della finestra di dialogo Proprietà mappa (vedere la Figura 88).

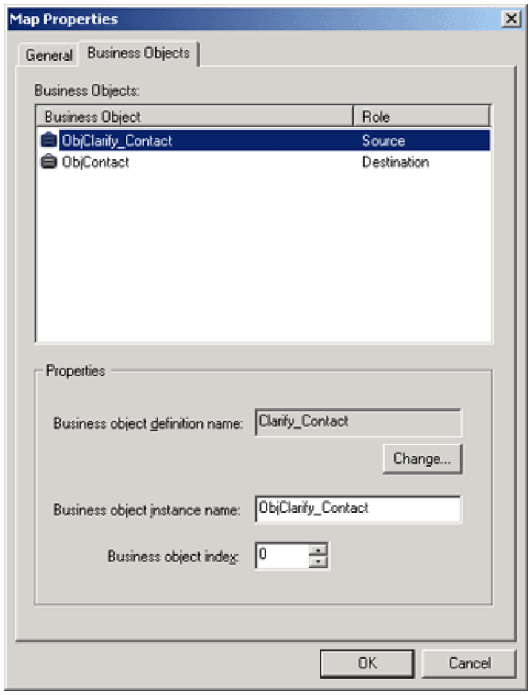


Figura 88. Scheda Oggetti business della finestra di dialogo Proprietà mappa

Per modificare il tipo di oggetto business dell’oggetto business origine o di destinazione nella mappa nella finestra di dialogo Proprietà mappa, eseguire i seguenti passi:

1. Aprire la mappa.
2. Visualizzare la scheda Oggetti business della finestra di dialogo Proprietà mappa in uno dei seguenti modi:
   - Dal menu Modifica, selezionare Proprietà mappa.
   - Dalla scheda Diagramma, fare clic con il pulsante destro del mouse sulla finestra dell’oggetto business e selezionare Proprietà dal menu contestuale.

     Risultato: viene visualizzata la scheda Generale della finestra Proprietà mappa. Fare clic sulla scheda Oggetti business.

   Per informazioni su altri metodi per visualizzare la finestra di dialogo Proprietà mappa, consultare “Specifica delle informazioni sulla proprietà della mappa” a pagina 62..
3. Selezionare il tipo di oggetto business che si desidera modificare.
4. Fare clic sul pulsante Modifica sotto Tipo oggetto business.
5. Selezionare il nuovo tipo.
6. Fare clic su OK per chiudere la finestra di dialogo Seleziona oggetto business.
7. Fare clic su OK per chiudere la finestra Proprietà mappa.

Nota: Le regole di trasformazione non valide verranno eliminate.

Riferimento agli attributi degli oggetti business

Utilizzare le variabili oggetto business che Map Designer Express genera per fare riferimento agli oggetti business e ad i loro attributi, nel seguente modo:

- Per fare riferimento agli attributi in un oggetto business origine o di destinazione, utilizzare il nome oggetto business seguito dal nome attributo, con un punto (.) come separatore:

  Esempio:

  `Obj`*`BusObjName.AttrName`*

- Per fare riferimento agli attributi in un oggetto business secondari, utilizzare il nome oggetto business secondario ed il nome attributo dell'elemento secondario.

  Esempio: il seguente esempio imposta il valore dell'attributo `OrigHireDate` in `ObjPsft_Employee` sull'attributo `HireDate` di `EmployeeHR_Misc`, elemento secondario di `ObjEmployee`:

  ```
  ObjPsft_Employee.set("OrigHireDate",
     ObjEmployee.getString("EmployeeHR_Misc.HireDate"));
  ```

- Se l'oggetto business secondario ha una cardinalità `n` (cioè possono esserci più istanze dell'elemento secondario associato all'elemento principale), è necessario fornire un numero indice per l'oggetto business secondario.

  Esempio: il seguente esempio imposta il valore dell'attributo `TimeZone` di `Address`, che è un elemento secondario a cardinalità multipla di `ObjCustomer`:

  ```
  ObjCustomer.set("Address[0].TimeZone",
  ObjSAP_Customer.getString("TimeZone"));
  ```

Creazione di variabili temporanee

Map Designer Express consente la creazione di variabili temporanee che possono essere accedute durante i passi della trasformazione in tutta la mappa; le variabili temporanee sono globali per la mappa. Ad esempio, è possibile calcolare un valore in un passo della trasformazione, memorizzarlo in una variabile temporanea e creare un riferimento alla variabile in un altro passo della trasformazione. Questo è utile specialmente se un determinato calcolo viene eseguito ripetutamente; è possibile eseguire il calcolo una volta, memorizzare il risultato in una variabile temporanea e richiamare il valore quando necessario (ad esempio con una trasformazione Sposta).

Passi per la creazione delle variabili temporanee degli oggetti business

Le variabili temporanee sono definite in un oggetto business temporaneo. Per creare una variabile oggetto business temporanea, eseguire i seguenti passi:

1. Selezionare Aggiungi oggetti business dal menu Modifica.

   Risultato: viene visualizzata la scheda Generale della finestra di dialogo Aggiungi oggetto business.

   Per informazioni su altri metodi per visualizzare la finestra di dialogo Aggiungi oggetto business, consultare "Passi per la specifica di oggetti business dal dialogo Aggiungi oggetto business" a pagina 37.

2. Fare clic sulla scheda Temporaneo. Qui vengono definite le variabili temporanee. La Figura 89 mostra la scheda Temporaneo della finestra di dialogo Aggiungi oggetto business. Nel campo Nome viene visualizzato il nome dell'oggetto business temporaneo, che Map Designer Express ha generato. Il primo nome generato è `ObjTemporary`. Questo campo è in sola lettura.

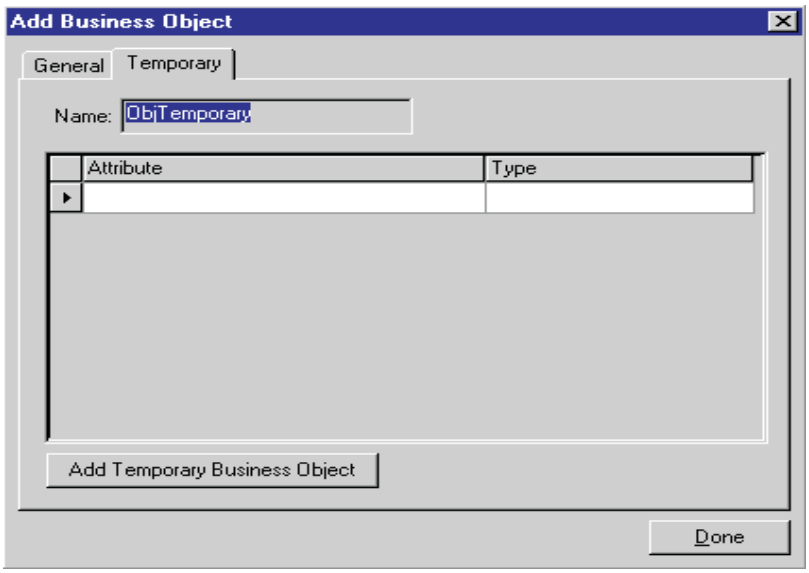


Figura 89. Scheda Temporaneo della finestra di dialogo Aggiungi oggetto Business

3. Fare clic nel campo Attributo.

   Risultato: viene visualizzata una nuova riga nella tabella delle variabili. Immettere il nome della variabile temporanea.

   Nota: *Non* creare due variabili temporanee con lo stesso nome.
4. Fare clic sul campo Tipo e selezionare il tipo di dati della variabile temporanea dall'elenco a discesa.

   Nota: Per essere compatibile con lo schema del tipo di InterChange Server Express, tutte le variabili temporanee hanno come tipo interno Stringa. Il tipo di dati specificato nella finestra di dialogo Aggiungi oggetto business riguarderà solo l'avvio della variabile. Se si desidera scrivere del codice Java personalizzato per assegnare dei valori alla variabile temporanea, il valore deve prima essere convertito in una stringa.
5. Ripetere i passi 3 e 4 per ognuna delle variabili temporanee necessarie nella mappa.
6. Fare clic sul pulsante Aggiungi oggetto business temporaneo.
7. E' possibile definire un altro oggetto business temporaneo oppure fare clic su Ok per terminare.

Passi per l'utilizzo delle variabili oggetto business temporanee nei passi di trasformazione

Utilizzare la variabile temporanea in un passo di trasformazione nel seguente modo:

- Nella scheda Diagramma:
  1. Fare clic sulla colonna di intestazione della riga (estrema sinistra) dell'attributo temporaneo.
  2. Copiare il valore della variabile in un attributo tenendo premuto il tasto `Ctrl` e trascinando la variabile nell'attributo.
- Nella scheda Java di Activity Editor Java, utilizzare il nome variabile nel passo di trasformazione di un attributo.

Importante: Poiché una variabile temporanea è una variabile globale, è necessario inizializzare esplicitamente una variabile temporanea su `null` quando si utilizza l'opzione Riutilizzo istanza mappa. In caso contrario, il valore della variabile temporanea proveniente da una esecuzione precedente dell'istanza mappa può essere utilizzato in maniera non corretta come valore ella variabile temporanea nelle esecuzioni successive della stessa mappa. Quando *non* si utilizza l'opzione Riutilizzo istanza mappa, il sistema InterChange Server Express inizializza automaticamente le variabili temporanee fra i richiami separati della mappa.

Risultato: quando Map Designer Express crea l'oggetto business temporaneo, questo oggetti viene visualizzato nelle schede Tabella e Diagramma con gli altri oggetti business della mappa, nel seguente modo:

- Nella scheda Tabella:
  - La finestra dell'oggetto business aggiunge una nuova area per l'oggetto business temporaneo. Fare clic con il pulsante destro del mouse sul nome dell'oggetto business temporaneo per aprire il menu contestuale con le opzioni di modifica ed eliminare l'oggetto business.
  - L'oggetto business temporaneo ed i relativi attributi vengono visualizzati nelle caselle combinate delle colonne `Attributo origine` e `Attributo dest.` nella tabella di trasformazione dell'attributo.
- Nella scheda Diagramma lo spazio di lavoro della mappa aggiunge a una nuova finestra oggetto business per l'oggetto business temporaneo.

  Tale finestra ha molte delle stesse caratteristiche della finestra dell'oggetto business. Le variabili create vengono visualizzate nella tabella delle variabili, come per gli attributi in un oggetto business. Questa finestra oggetto business fornisce una colonna Regola e una Commento, dove è possibile aggiungere rispettivamente il codice di trasformazione della variabile temporanea ed il commento.

  E' possibile fare clic con il pulsante destro del mouse sulla barra del titolo della finestra dell'oggetto business temporaneo per richiamare il menu contestuale che fornisce le opzioni di modifica ed eliminare l'oggetto business, insieme alle sue proprietà. Specificare un valore per la variabile in uno dei seguenti modi:
  - Per immettere il codice che restituisce il valore della variabile, fare doppio clic nella colonna Regola, selezionare la regola di trasformazione appropriata e fare clic su Modifica codice per inserire il codice in Activity Editor.
  - Per copiare un valore da un attributo oggetto business ad una variabile, tenere premuto il tasto `Ctrl` e trascinare l'attributo nel nome della variabile. E' anche possibile dividere ed unire gli attributi in una variabile.

Nota: Un oggetto business temporaneo viene visualizzato anche nella scheda Oggetti business della finestra di dialogo Proprietà mappa .

Dichiarazione delle variabili

Suggerimenti: tenere presente questi suggerimenti quando si dichiarano le variabili:

- Quando si crea una variabile locale per l'attributo corrente (non visibile a tutti gli altri attributi), dichiararla all'inizio del passo di trasformazione dell'attributo corrente (in Activity Editor).
- Quando si crea una variabile globale per la mappa corrente (visibile a tutto gli attributi), dichiararla nella sezione `Blocco dichiarazione locale mappa` della scheda Generale nella finestra di dialogo Proprietà mappa. Quando si scrive del

codice per assegnare dei valori a queste variabili, farlo all'inizio di Activity Editor nel primo attributo dell'oggetto di destinazione (come specificato dall'ordine di esecuzione).

Ulteriori metodi di trasformazione attributi

E' possibile eseguire le trasformazioni attributo in modo interattivamente nei seguenti modi:

- Utilizzando *solo* Map Designer Express—creare una delle trasformazioni standard.

  La Tabella 14 a pagina 40 elenca le trasformazioni standard per le quali Map Designer Express può generare del codice.
- Utilizzando una combinazione di Map Designer Express e Activity Editor per modificare e migliorare il codice—creare una trasformazione personalizzata.

  E' possibile creare delle trasformazioni personalizzate in uno dei seguenti modi:
  - Creando una trasformazione standard ed aprendo Activity Editor per modificare il codice generato. Dopo aver personalizzato una trasformazione standard, Map Designer Express visualizza il tipo di trasformazione in carattere corsivo blu nella colonna della regola di trasformazione.
  - Creando una trasformazione personalizzata ed aprendo Activity Editor per definirla. Dopo aver creato una trasformazione personalizzata, Map Designer Express visualizza la parola chiave `Personalizzato` con caratteri neri nella colonna della regola di trasformazione. Per ulteriori informazioni, consultare "Creazione di una trasformazione personalizzata" a pagina 52.

Questa sezione descrive come implementare i seguenti tipi di trasformazioni personalizzate:

- "Logica basata sul contenuto"
- "Formattazione della data" a pagina 188
- "Utilizzo del generatore di espressioni per le trasformazioni delle stringhe" a pagina 191

Logica basata sul contenuto

```
Customer.CustomerStatus = 'Inactive'
if SAP_CustomerMaster.DeleteInd = 'X'.
```

Altrimenti, `CustomerStatus = 'Active'`.

E' possibile creare una trasformazione Personalizzata e scrivere l'intera parte di codice per implementare da soli la logica basata sul contenuto. Tuttavia, una procedura migliore è quella di iniziare creando una trasformazione Sposta fra `DeleteInd` (nell'oggetto `SAP_CustomerMaster`) e `CustomerStatus` (consultare "Copia di un attributo di origine ad un attributo di destinazione" a pagina 42 per la procedura di spostamento di attributi).

Come risultato, Map Designer Express genera la trasformazione di spostamento, come illustrato nel codice semplice riportato di seguito:

```
{
   Object _cw_CpBTBSourceValue = null;

   //
   // RICHIAMA ORIGINE
   // ---------------
   //
```

```
   // Richiama il valore origine dall'oggetto business origine e
   // lo colloca in una variabile locale per la protezione del codice.
   //
   _cw_CpBTBSourceValue = ObjSAP_CustomerMaster.get("DeleteInd");

   //
   // IMPOSTA DESTINAZIONE
   // ---------------
   //
   // Colloca il valore origine nell'attributo dell'oggetto business
   // di destinazione.
   //
   {
      Object _cw_SetSrcVal = _cw_CpBTBSourceValue;
      BusObj _cw_SetDestBusObj = ObjCustomer;
      String _cw_SetDestAttr = "CustomerStatus";

      //
      // Imposta il valore della destinazione solo se né
      // l'origine né la destinazione sono null.
      //
      if ((_cw_SetSrcVal != null) && (_cw_SetDestBusObj != null))
      {
         if (dataValidationLevel >= 1)
         {
            if (!_cw_SetDestBusObj.validData(_cw_SetDestAttr, _cw_SetSrcVal))
            {
               String warningMessage =
"Rilevati dati non validi durante il tentativo di impostazione del valore
attributo \'_cw_SetDestAttr\' di BusObj \'_cw_SetDestBusObj\' mentre era in esecuzione
la mappa \'" + getName() + "\'.  Il valore non valido era \'" + _cw_SetSrcVal
+ "\'.";
               //
               // Registra un avviso su questo errore.
               //
               logWarning(warningMessage);
               if (failOnInvalidData)
               {
                  //
                  // Interrompere l'esecuzione della mappa con un messaggio di avviso.
                  //
                  throw new MapFailureException(warningMessage);
               }
            }
         }
// SEZIONE DA AGGIORNARE CON LA LOGICA

      if (_cw_SetSrcVal != null)
         {
         if (_cw_SetSrcVal instanceof BusObj)
            {
            //
            // Poiché i BusObj non sono immutabili, è necessario effettuare
            // una copia dell'oggetto origine prima di collocarlo
            // realmente nell'attributo do destinazione.
            //
            _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
               ((BusObj)_cw_SetSrcVal).duplicate());
            }
         else if (_cw_SetSrcVal instanceof BusObjArray)
            {
            //
            // Poiché i BusObj non sono immutabili, è necessario effettuare
            // una copia dell'oggetto origine prima di collocarlo
            // realmente nell'attributo do destinazione.
            //
            _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
```

```
                ((BusObjArray)_cw_SetSrcVal).duplicate());
            }
         else
            {
            //
            // Poiché la nostra versione di tipi di dati semplici è immutabile in
            // Java (stringhe incluse), non è necessario effettuare una copia
            // del valore origine qui.
            //
            _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, _cw_SetSrcVal);
            }
         }
      }
   }
}

// CODICE MODIFICATO

   if (_cw_SetSrcVal != null)
      {
      if (((String)_cw_SetSrcVal).equals("X"))
         _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, "Inactive");
      else
         _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, "Active");
      }

      }
   }
}
```

Selezionare Salva dal menu File di Activity Editor per salvare le modifiche.

Locica basata su istruzione (verb)

`Customer.CustomerStatus = 'Inactive' if Verb = 'Create'`. Altrimenti, `CustomerStatus = 'Active'`.

Seguire la procedura descritta in "Logica basata sul contenuto" a pagina 183 ma utilizzare la seguente istruzione condizionale per sostituire la sezione del codice generato:

```
// CODICE MODIFICATO
if (_cw_SetSrcVal != null)
   {
   if (ObjSAP_CustomerMaster.getVerb() .equalsIgnoreCase("Create"))
      _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, "Inactive");
   else
      _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,"Active");
```

Fornire dei valori predefiniti se mancano i dati origine

Mappare `SAP_CustomerMaster.State` in `Customer.CustomerAddress.State`. Se manca lo stato di SAP impostare come valore predefinito `CA`.

Notare che prima che l'attributo di destinazione venga impostato sul valore origine, il codice verifica che l'attributo origine non sia uguale a null.

Esempio: in questo esempio, se mancano i dati origine, l'attributo di destinazione viene impostato su un valore predefinito.

Iniziare spostando `SAP_CustomerMaster.State` in `Customer.CustomerAddress.State`. Modificare l'istruzione della condizione come segue:

```
// CODICE MODIFICATO
if (_cw_SetSrcVal != null)
   _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, _cw_SetSrcVal);
```

```
else
   _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, "CA");
_cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, "CA");
```

Suggerimenti: Se si è iniziato il codice copiando l'attributo origine nell'attributo di destinazione, la riga di codice che imposta il valore predefinito deve essere digitata due volte. Questo perché il codice generato quando l'attributo origine viene spostato nell'attributo di destinazione verifica due volte che l'attributo origine non sia null. Quindi, immettere il valore predefinito.

Logica basata su contesto di chiamata

Quando è necessario verificare il valore del contesto di chiamata, utilizzare la variabile inclusa `strInitiator`, che è di tipo `Stringa`.

Esempio: per verificare se il contesto di chiamata è `EVENT_DELIVERY`, utilizzare la seguente istruzione:

```
if (strInitiator.equals(MapExeContext.EVENT_DELIVERY))
//resto del codice
```

Forzare una mappa all'errore se mancano i dati origine

Mappare `SAP_CustomerMaster.State` in `Customer.CustomerAddress.State`. Se manca lo stato di SAP, interrompere l'esecuzione della mappa.

Iniziare spostando `SAP_CustomerMaster.State` in `Customer.CustomerAddress.State`. Quindi aggiungere un'altra istruzione `if()` per verificare se l'attributo Stato per SAP è uguale a null. Il codice generato è simile a questo:

```
{
   Object _cw_CpBTBSourceValue = null;

   //
   // RICHIAMA ORIGINE
   // ---------------
   //
   // Richiama il valore origine dall'oggetto business origine e
   // lo colloca in una variabile locale per la protezione del codice.
   //
   _cw_CpBTBSourceValue = ObjSAP_CustomerMaster.get("State");

   //
   // IMPOSTA DESTINAZIONE
   // ---------------
   //
   // Colloca il valore origine nell'attributo dell'oggetto business
   // di destinazione.
   //
   {
      Object _cw_SetSrcVal = _cw_CpBTBSourceValue;
      BusObj _cw_SetDestBusObj = ObjCustomer;
      String _cw_SetDestAttr = "CustomerAddress.State";
   // Nuovo codice
   if (_cw_SetSrcVal == null)
      {
      String errorMessage = "Mancano i dati nell'attributo stato";
      logError(errorMessage);
      //
      // Interrompere l'esecuzione della mappa con un messaggio di avviso.
      //
      throw new MapFailureException(errorMessage);
      }
   // Fine del nuovo codice
```

```
   //
   // Imposta il valore della destinazione solo se né
   // l'origine né la destinazione sono null.
   //
   else if ((_cw_SetSrcVal != null) && (_cw_SetDestBusObj != null))
      {
      if (dataValidationLevel >= 1)
         {
         if (!ObjCustomer.validData("CustomerAddress.State", _cw_SetSrcVal))
            {
            String warningMessage =
"Rilevati dati non validi durante il tentativo di impostazione del valore
attributo \"CustomerAddress.State\" di BusObj \'ObjCustomer\'
durante l'esecuzione della mappa \'" + getName() + "\'.  Il valore non valido
era \'" + _cw_SetSrcVal + "\'.";

            //
            // Registra un avviso su questo errore.
            //
            logWarning(warningMessage);
            if (failOnInvalidData)
               {
               //
               // Interrompere l'esecuzione della mappa con un messaggio di avviso.
               //
               throw new MapFailureException(warningMessage);
               }
            }
         }

      if (_cw_SetSrcVal != null)
         _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr, _cw_SetSrcVal);

      }
   }
}
```

Se la mappa viene eseguita e l'attributo Stato non viene impostato, viene prodotto un messaggio di errore sia in una finestra di dialogo che nella schermata del server. Il messaggio sul server è simile a questo:

```
[1999/09/22 17:58:51.008] [Server]
Sub_SaCwCustomerMaster: Errore Mancano
i dati nell'attributo stato
```

La mappa interrompe l'esecuzione.

Suggerimento: Utilizzare il seguente codice per visualizzare l'effettivo messaggio di errore generato dal sistema in caso di errore nella mappa:

```
try
   {
   // codice dell'utente
   }
catch (Exception e)
   {
  throw new MapFailureException(e.toString());
   }
```

Registrazione di messaggi da un file di messaggi

Notare in che modo l'esempio in "Forzare una mappa all'errore se mancano i dati origine" a pagina 186 utilizza il metodo `logError()` per visualizzare il messaggio di errore sullo schermo. E' possibile utilizzare i messaggi forniti nel file di

messaggi generici, CWMapMessages.txt, memorizzato in \DLMs\messages. Per ulteriori informazioni sui file di messaggi, consultare Capitolo 27, “File di messaggi”, a pagina 517.

CWMapMessages.txt ha il seguente formato:

```
# errore mancano dati critici
10
Mancano i dati dell'attributo {1}. Interrotta l'esecuzione della mappa.
# Altro errore
11
Altro messaggio di errore
```

Notare che invece di {1} è possibile utilizzare la parola Stato in modo da corrispondere direttamente all’errore che deve essere visualizzato. Se implementato, tuttavia, il messaggio non è più generico. Se anche un altro attributo ha dei dati critici, è necessario un altro messaggio nel file di messaggi per quel particolare attributo.

Nota: *Non* modificare il file CwMapMessages.txt. Se si devono scrivere dei messaggi propri, immetterli nella scheda Messaggi di Map Designer Express, che crea un file di messaggi di errore specifico per la mappa denominato *mapName_locale*.txt (dove *mapName* è lo stesso nome della mappa), ad esempio mapName_en_US.txt. Il server salva questo file di messaggi nella directory \DLMs\messages dopo la distribuzione.

Per visualizzare il messaggio #10, utilizzare il seguente codice:

```
logWarning(10, "State");
```

La parola State sostituisce {1} nel messaggio di testo.

Il seguente messaggio viene visualizzato nel file di log di InterChange Server Express:

```
[1999/09/23 10:17:43.648] [Server]
Sub_SaCwCustomerMaster:
Avviso 10: mancano i dati dell'attributo Stato.
Interrotta l'esecuzione della mappa.
```

Formattazione della data

L’API di mappatura fornisce i metodi per la formattazione della data nella classe DtpDate. La Tabella 61 riassume i metodi di formattazione della data.

Tabella 61. Metodi di formattazione data della classe DtpDate

Formattazione data	Metodo DtpDate
Richiamo del nome del mese da una data	getMonth(), getShortMonth()
Richiamo del valore del mese da una data	getIntMonth(), getNumericMonth()
Richiamo del giorno del mese	getDayOfMonth(), getIntDay()
Richiamo del giorno della settimana	getDayOfWeek(), getIntDayOfWeek()
Richiamo dell’anno da una data	getYear(), getIntYear()
Richiamo del valore dell’ora	getHours()
Richiamo del valore dei minuti da una data	getMinutes(), getIntMinutes()
Richiamo del valore dei secondi da una data	getSeconds(), getIntSeconds()
Richiamo del numero di millisecondi nella data	getMSSince1970()
Richiamo della prima data di un elenco	getMinDate(), getMinDateBO()

Tabella 61. Metodi di formattazione data della classe DtpDate (Continua)

Formattazione data	Metodo DtpDate
Richiamo della data più recente da un elenco	`getMaxDate()`, `getMaxDateBO()`
Analisi della data secondo un formato specificato	`DtpDate()`
Richiamo della data in un formato specificato o in quello predefinito	`toString()`
Riformattazione di una data nel formato data generico IBM	`getCWDate()`
Aggiunta di giorni alla data	`addDays()`
Aggiunta di giorni della settimana alla data	`addWeekdays()`
Aggiunta di anni alla data	`addYears()`
Calcolo di giorni fra delle date	`calcDays()`
Calcolo di giorni della settimana fra delle date	`calcWeekdays()`
Confronto di date	`after()`, `before()`
Utilizzo nomi completi dei mesi	`get12MonthNames()`, `set12MonthNames()`, `set12MonthNamesToDefault()`
Utilizzo dei nomi brevi dei mesi	`get12ShortMonthNames()`, `set12ShortMonthNames()`, `set12ShortMonthNamesToDefault()`
Utilizzo dei nomi dei giorni della settimana	`get7DayNames()`, `set7DayNames()`, `set7DayNamesToDefault()`

Suggerimenti: Cogliere sempre `DtpDateException`. Questo garantisce che se il formato della data non è valido, viene inviato un messaggio al log di InterChange Server Express.

Utilizzo del formato data generico

L'IBM utilizza il seguente formato di data per i suoi oggetti business generici:

AAAAMMGG OOMMSS

Questo formato viene denominato il *formato data generico*.

Passi per la conversione al formato data generico: Per convertire una data specifica per l'applicazione in questo formato generico utilizzare il metodo `getCWDate()` della classe `DtpDate`.

Esempio: per mappare l'attributo data SAP in una data generica, eseguire una copia dell'attributo origine (la stringa della data SAP) nell'attributo di destinazione (la stringa della data generica):

1. Creare un oggetto `DtpDate` per contenere la data generica.

   Copiare l'attributo origine (la stringa della data SAP) in un nuovo oggetto `DtpDate` (la stringa della data generica) analizzando la stringa della data SAP (*AAAAMMGG*) con il costruttore `DtpDate()`.
2. Convertire la stringa della data SAP nel formato data generico (*AAAAMMGG OOMMSS*) con il metodo `getCWDate()`.
3. Creare l'attributo di destinazione ed inizializzarne il valore sul formato data generico con il metodo `setWithCreate()`.

L'ultima sezione del codice deve essere simile a questa:

```
// CODICE MODIFICATO
if (_cw_SetSrcVal != null)
   {
   try
      {
      DtpDate myDate = new DtpDate((String)_cw_SetSrcVal, "YMD");
      _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
      myDate.getCWDate());
      }
   catch (DtpDateException de)
      {
      logError(5501);
      logInfo(de.getMessage());
      }
   }
```

Esempio: per mappare l'attributo data Clarify in una data generica, eseguire una copia dell'attributo origine (la stringa della data Clarify) nell'attributo di destinazione (la stringa della data generica):

1. Creare un oggetto `DtpDate` per contenere la data generica.

   Copiare l'attributo origine (la stringa della data Clarify) in un nuovo oggetto `DtpDate` (la stringa della data generica) analizzando la stringa della data Clarify (*MM/GG/AAAA OO:MM:SS*) con il costruttore `DtpDate()`.
2. Convertire la stringa della data Clarify nel formato data generico (*AAAAMMGG OOMMSS*) con il metodo `getCWDate()`.
3. Creare l'attributo di destinazione ed inizializzarne il valore sul formato data generico con il metodo `setWithCreate()`.

Il seguente codice converte la data Clarify nella data generica:

```
if (_cw_SetSrcVal != null)
   {
   try
      {
      DtpDate myDate = new DtpDate((String)_cw_SetSrcVal,
         "M/D/Y h:m:s");
      _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
         myDate.getCWDate());
      }
   catch (DtpDateException de)
      {
      logError(5501);
      logInfo(de.getMessage());
      }
   }
```

Per mappare la data generica nel formato Clarify, utilizzare il seguente codice:

```
if (_cw_SetSrcVal != null)
   {
   try
      {
      DtpDate myDate = new DtpDate((String)_cw_SetSrcVal, "YMD hms");
      _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
      myDate.toString("M/D/Y h:m:s", true));
      }
   catch (DtpDateException de)
      {
      logError(5501);
      logInfo(de.getMessage());
      }
}
```

Passi per la conversione dal formato data generico: Per convertire dal formato data generico al formato data SAP, eseguire i passi riportati di seguito:

1. Creare un oggetto `DtpDate` per contenere la data SAP.

   Eseguire una copia dell'attributo origine (la stringa della data generica) in un nuovo oggetto `DtpDate` (la stringa della data SAP) analizzando la stringa della data generica (*AAAAMMGG OOMMSS*) con il costruttore `DtpDate()`.
2. Convertire il formato della data generica IBM nella stringa della data SAP (*AAAAMMGG*) con il metodo `toString()`.
3. Creare l'attributo di destinazione ed inizializzarne il valore sul formato data SAP con il metodo `setWithCreate()`.

Il seguente frammento di codice mostra la conversione data generica-a-SAP:

```
// CODICE MODIFICATO
if (_cw_SetSrcVal != null)
   {
   try
      {
      DtpDate myDate = new DtpDate((String)_cw_SetSrcVal, "YMD hms");
      _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
      myDate.toString("YMD"));
      }
   catch (DtpDateException de)
      {
      logError(5501);
      logInfo(de.getMessage());
      }
   }
```

Richiamo della data corrente

Per creare un oggetto `DtpDate` inizializzato con la data corrente del sistema, utilizzare il costruttore `DtpDate()` senza alcun parametro:

```
DtpDate()
```

Esempio: per mappare la data corrente nel formato generico, utilizzare il seguente codice:

```
//richiamo data corrente
DtpDate myDate = new DtpDate();
// formattazione della data al formato dell'oggetto di destinazione e mappatura
_cw_SetDestBusObj.set(_cw_SetDestAttr, myDate.getCWDate());
```

Per mappare la data corrente nel formato specifico per l'applicazione, utilizzare il seguente codice:

```
//richiamo data corrente
DtpDate myDate = new DtpDate();
// formattazione della data al formato dell'oggetto di destinazione e mappatura
_cw_SetDestBusObj.set(_cw_SetDestAttr, myDate.toString(
   "formato applicazione");
```

Utilizzo del generatore di espressioni per le trasformazioni delle stringhe

Quando si scrive il codice per la trasformazione, potrebbe essere necessario costruire delle complesse espressioni Java per fare riferimento ad un particolare attributo, manipolare una stringa o richiamare un metodo API. E' possibile immettere queste espressioni manualmente in Activity Editor oppure utilizzare il generatore di espressioni (Expression Builder) per costruire l'espressione interattivamente. Il generatore di espressioni è un programma di utilità disponibile in Activity Editor.

Suggerimento: in alternativa è possibile utilizzare la vista Grafica di Activity Editor.

Per visualizzare il Generatore di espressioni, posizionare il cursore in Activity Editor nel punto in cui si desidera inserire l'espressione ed eseguire una delle seguenti operazioni:

- Dal menu Strumenti, selezionare Generatore di espressioni.
- Nella barra strumenti Java fare clic sul pulsante Generatore di espressioni.
- Fare clic con il pulsante destro del mouse in qualunque punto nella finestra di Activity Editor e selezionare Generatore di espressioni dal menu contestuale.

La Figura 90 identifica i componenti principali del generatore di espressioni.

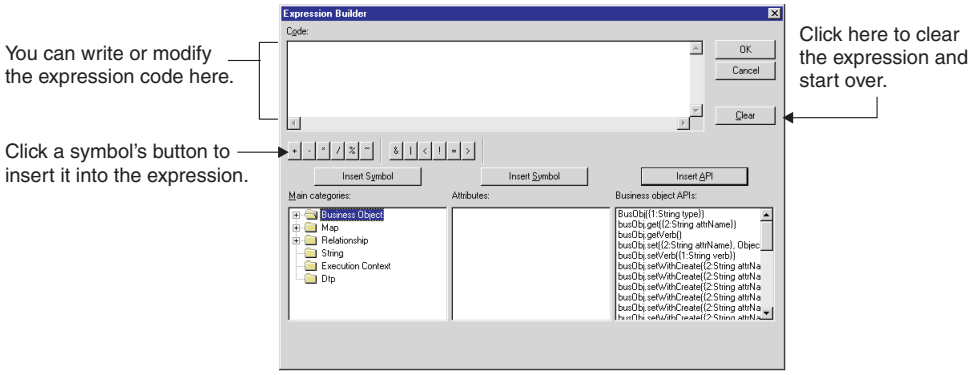


Figura 90. Finestra di dialogo Generatore di espressioni

Nota: Nell'elenco di API, il numero 1 si riferisce all'oggetto selezionato dalla finestra a sinistra ed il numero 2 si riferisce all'oggetto selezionato dalla finestra centrale, se vi è presente un oggetto. Quando si inserisce l'API, delle coppie di parentesi racchiudono le variabili (<<*variabile*>>). E' necessario sostituire le parentesi e la variabile con un valore.

Questa sezione descrive come utilizzare il generatore di espressioni per eseguire i seguenti tipi di trasformazioni Stringa:

- "Passi per la conversione di testo in maiuscolo"
- "Utilizzo dei metodi di manipolazione della stringa" a pagina 193

Passi per la conversione di testo in maiuscolo

Spostare `SAP_CustomerMaster.CustomerName` in `Customer.AccountOpenedBy` e convertire tutte le lettere in testo maiuscolo.

Per utilizzare le funzioni di manipolazione di stringa `toUpperCase()` eseguire i seguenti passi:

1. Eseguire una trasformazione Sposta da `SAP_CustomerMaster.CustomerNamea` `Customer.AccountOpenedBy`.
2. Aprire la finestra di Activity Editor facendo doppio clic nella colonna Regola trasformazione associata a `Customer.AccountOpenedBy`. In Activity Editor, scorrere fino al punto in cui la destinazione è impostata sui dati origine. Cambiarla in modo che contenga solo il seguente codice:

```
if (_cw_SetSrcVal != null)
   {
   //
   // Poiché la nostra versione di tipi di dati semplici è immutabile in
```

```
// Java (stringhe incluse), non è necessario effettuare una copia
// del valore origine qui.
//
_cw_SetDestBusObj.setWithCreate(
_cw_SetDestAttr, _cw_SetSrcVal);
}
```

La variabile `_cw_SetSrcVal` contiene `SAP.CustomerName` e `_cw_SetDestAttr` contiene `AccountOpenedBy`.

3. Convertire `CustomerName` in lettere maiuscole prima di copiarlo in `AccountOpenedB`.

   Se si conosce il metodo da utilizzare, aggiungerlo alla riga di codice riportata prima; altrimenti, è possibile utilizzare il Generatore di espressioni per facilitare la ricerca del metodo corretto. Procedere nel modo seguente:

   a. Copiare (`Ctrl+C`) `_cw_SetSrcVal`.
   b. Evidenziare `_cw_SetSrcVal` (questo codice verrà sostituito con il codice generato dal Generatore di espressioni) e selezionare Generatore di espressioni dal menu Strumenti.
   c. Selezionare Stringa nell'elenco Categorie principali.
   d. Selezionare il metodo `toUpperCase()` dall'elenco di API Stringa.
   e. Fare clic su Inserisci API.
   f. Quando questo metodo viene visualizzato nella finestra superiore, sostituire la parola "string," che è un segnaposto, con la parola `_cw_SetSrcVal` copiata in precedenza.
   g. Fare clic su OK.

      Risultato: il richiamo del metodo viene inserita nel codice.
   h. Selezionare l'opzione Salva dal menu File per salvare il codice.

Utilizzo dei metodi di manipolazione della stringa

La seguente trasformazione di stringa utilizza i metodi di manipolazione (come `length()` e `substring()`) per spostare `SAP_CustomerMaster.AddressLine1` in `Customer.CustomerAddress.AddressLine1` e `AddressLine2`:

- Se la lunghezza di `AddressLine1` è minore di 10 caratteri, spostare `AddressLine1` di SAP in `CustomerAddress.AddressLine1` e *non* mappare `AddressLine2`.
- Se la lunghezza è maggiore o uguale a 10 caratteri, mappare tutto ciò che segue la virgola in `AddressLine1` di SAP in `CustomerAddress.AddressLine2`. Se non c'è una virgola, mappare 9 caratteri con `AddressLine1` ed il resto con `AddressLine2`.

Passi per l'esecuzione della trasformazione della stringa: Per eseguire questa trasformazione della stringa, seguire questa procedura:

1. Eseguire una trasformazione Sposta di `AddressLine1` in `CustomerAddress.AddressLine1`.
2. Aprire la finestra Activity Editor di `AddressLine1` e scorrere fino al punto in cui la destinazione è impostata sui dati origine. Cambiarla in modo che contenga il seguente codice:

```
if (_cw_SetSrcVal != null)
   //
   // Poiché la nostra versione di tipi di dati semplici è immutabile in
   // Java (stringhe incluse), non è necessario effettuare una copia
   // del valore origine qui.
   //
   {
   _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
      _cw_SetSrcVal);
   }
```

La variabile _cw_SetSrcVal contiene SAP.AddressLine1 e _cw_SetDestAttr contiene CustomerAddress.AddressLine1.

3. Verificare la lunghezza di SAP.AddressLine1 e mappare l'intera stringa in CustomerAddress.AddressLine1 o mappare la stringa secondaria di SAP.AddressLine1 che precede la virgola.

   E' possibile modificare da soli il codice oppure utilizzare il Generatore di espressioni. In qualsiasi caso, utilizzare i metodi length(), substring(int, int), substring(int) ed i metodi indexOf(int) della classe Stringa.

4. Modificare il codice come segue:

```
if (_cw_SetSrcVal != null)
   {
   // Prima verificare la lunghezza di _cw_SetSrcVal
   if (((String)_cw_SetSrcVal).length() < 10)
      {
      // se è minore di 10, mapparla con AddressLine1
      _cw_SetDestBusObj.setWithCreate(
         _cw_SetDestAttr, _cw_SetSrcVal);
      }
   else
      {
      // se la lunghezza non è minore di 10, cercare la virgola
      int index = ((String)_cw_SetSrcVal).indexOf(",");

      // se si individua la virgola, prendere una stringa secondaria di _cw_SetSrcVal fino
      // alla virgola e mapparla con AddressLine1
      if (index != -1)
         _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
            ((String)_Cw_SetSrcVal).substring(0, index));
         // se non si trova la virgola, prendere i primi 9 caratteri di
         // _cw_SetSrcVal e mapparli con AddressLine1
      else
         _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
            ((String)_Cw_SetSrcVal).substring(0, 9));
      }
   }
```

5. Spostare SAP.AddressLine1 into CustomerAddress.AddressLine2. Modificare la parte finale del codice in modo che contenga quanto segue:

```
if (_cw_SetSrcVal != null)
   {
   // Prima verificare la lunghezza di _cw_SetSrcVal
   if (((String)_cw_SetSrcVal).length() >= 10)
      {
      // se la lunghezza non è minore di 10, cercare la virgola
      int index = ((String)_cw_SetSrcVal).indexOf(",");
      // se si individua la virgola, prendere una stringa secondaria di _cw_SetSrcVal dopo
      // la virgola e mapparla con AddressLine2

      if (index != -1)
         _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
            ((String)_Cw_SetSrcVal).substring(index + 1));
         // se non si trova la virgola, prendere tutti i caratteri di
         // _cw_SetSrcVal iniziando dal decimo e mapparli con
         // AddressLine1

      else
         _cw_SetDestBusObj.setWithCreate(_cw_SetDestAttr,
               ((String)_Cw_SetSrcVal).substring(9));
      }
   }
```

Convalida dei dati origine

Se si deve scrivere il codice, verificare che i dati origine non siano null o vuoti. Per verificare se l'attributo origine è null, utilizzare una delle seguenti istruzioni:

- `if (ObjSource.isNull("Attr"))`— restituisce `true` se è vuoto
- `if (ObjSource.get("Attr") == null)` — restituisce `true` se *attr* è vuoto

Per verificare se l'attributo origine è una stringa vuota, procedere come segue:

- `if (ObjSource.isBlank("Attr"))` — restituisce true se è vuoto
- `if (ObjSource.getString("Attr").length() == 0)` — restituisce true se *attr* di tipo Stringa è vuoto

Riutilizzo delle istanze di mappe

Di solito, il sistema di sviluppo mappe crea un'istanza di una mappa per elaborare ogni trasformazione di dati fra gli oggetti business di origine e di destinazione. Quando l'istanza completa la gestione della trasformazione, il sistema libera le sue risorse. Per ridurre l'utilizzo di memoria, il sistema IBM ricicla un'istanza di una mappa mettendola nella cache e riutilizzandola quando viene creata un'istanza dello stesso tipo di mappa in un momento successivo. Quando il sistema IBM può riciclare un'istanza di mappa esistente, può evitare il sovraccarico della creazione di un'istanza di mappa, migliorando quindi le prestazioni generali del sistema e l'utilizzo della memoria.

Limitazioni: il sistema di sviluppo di mappe pone automaticamente nella cache un'istanza di mappa; un'istanza di mappa usa l'opzione Riutilizzo istanza mappe per impostazione predefinita. Tuttavia, l'opzione Riutilizzo istanza mappe impone sulla mappa i seguenti requisiti di programmazione:

- Evitare l'utilizzo di variabili globali nel codice della mappa.

  Una variabile globale è una variabile che viene dichiarata nell'area `Blocco dichiarazione locale mappa` della scheda Generale nella finestra di dialogo Proprietà mappa.
- Se la mappa richiede le variabili globali, evitare di inizializzarle al momento della dichiarazione. Verificare, invece, che le variabili globali siano sempre inizializzate su un nodo di mappa, preferibilmente il primo nodo (attributo) di trasformazione in una mappa.

Attenzione: Una mappa che contiene delle variabili globali *non* inizializzate al primo nodo di trasformazione, non possono essere riciclate in sicurezza poiché i valori delle variabili nella mappa della cache persistono, quando viene riutilizzata l'istanza. Quando un'istanza di mappa nella cache viene riutilizzata ed inizia l'esecuzione, ciascuna variabile globale contiene il valore rimasto dal termine del precedente utilizzo dell'istanza di mappa.

Se non è possibile definire una propria mappa, in modo che soddisfi le limitazioni precedenti, è *necessario* disabilitare per questa mappa l'opzione Riutilizzo istanza mappe. Per disabilitare questa opzione, rimuovere il segno di spunta dalla casella `Riutilizzo istanza mappe`, che viene visualizzata nella finestra Proprietà mappa relativa alla mappa in System Manager. Questa finestra consente inoltre di specificare la dimensione del pool di istanze mappa.

Nota: La distribuzione della mappa sul server non aggiorna l'istanza runtime. E' possibile aggiornare le proprietà della mappa dinamicamente dalla vista di gestione componenti del server, facendo clic con il pulsante destro del

mouse sulla mappa e selezionando le proprietà dal menu contestuale. Le modifiche verranno aggiornate automaticamente sul server.

Gestione delle eccezioni

Un'*eccezione* rappresenta una ricorrenza che, se non gestita esplicitamente nella mappa, ne interrompe l'esecuzione. Durante l'esecuzione di una mappa, possono verificarsi delle eccezioni runtime. Quando si definisce una regola di trasformazione personalizzata, è possibile utilizzare il blocco funzione ″Errore″ per catturare qualsiasi eccezione runtime. Quando si rileva una particolare eccezione, è possibile determinare come gestirla.

Eccezioni delle relazioni

Quando in una mappa si utilizzano delle relazioni, possono verificarsi diverse eccezioni. Tutte queste eccezioni sono delle classi secondarie di `RelationshipRuntimeException`. Se non si è interessati al tipo di eccezione ma semplicemente a rilevarle tutte, è possibile rilevare solo `RelationshipRuntimeException`. Altrimenti, è possibile rilevare qualsiasi eccezione fra quelle riportate di seguito, per i casi specifici:

- `RelationshipRuntimeDataAccessException`—inviata se si verifica un problema durante l'accesso al database delle relazioni. E' possibile rilevare questa eccezione in qualsiasi chiamata di metodo dalla classe `Relazione` o `Partecipante`.
- `RelationshipRuntimeDuplicateIdentityEntryException`—inviata se si tenta di aggiungere un partecipante ad una relazione d'identità con lo stesso ID istanza di relazione di un'istanza di relazione esistente. E' possibile rilevare questa eccezione nelle chiamate di metodo `addMyChildren()` e `create()`.
- `RelationshipRuntimeUserErrorException`—è un'eccezione astratta. Viene inviata solo se si verifica un `RelationshipRuntimeMetaDataErrorException` o `RelationshipRuntimeGeneralUserErrorException`. E' possibile rilevare questa eccezione in qualsiasi chiamata di metodo dalla classe `Relazione` o `Partecipante` durante lo sviluppo della mappa. Una volta eseguito il debug della mappa, è possibile rimuovere i gestori di questa eccezione.
- `RelationshipRuntimeMetaDataErrorException`—inviata se si verifica un errore durante la manipolazione dei metadati associati alle istanze partecipante, come il nome della relazione o il nome della definizione del partecipante. E' possibile rilevare questa eccezione in qualsiasi chiamata di metodo che aggiunge, modifica o elimina le istanze partecipante.
- `RelationshipRuntimeGeneralUserErrorException`—inviata se è presente un errore nei dati runtime forniti con una chiamata di metodo della classe `Relazione` o `Partecipante`.

  Esempio: si verifica l'eccezione se si passa un oggetto business di tipo errato al metodo `create()`.

La Figura 91 illustra la gerarchia delle eccezioni runtime di relazione. Qualsiasi eccezione rilevata automaticamente, individua quelle inferiori nella gerarchia. Tuttavia, se viene inviata un'eccezione inferiore nella gerarchia, non è possibile sapere esattamente quale sia a meno che non la si rilevi in modo specifico.

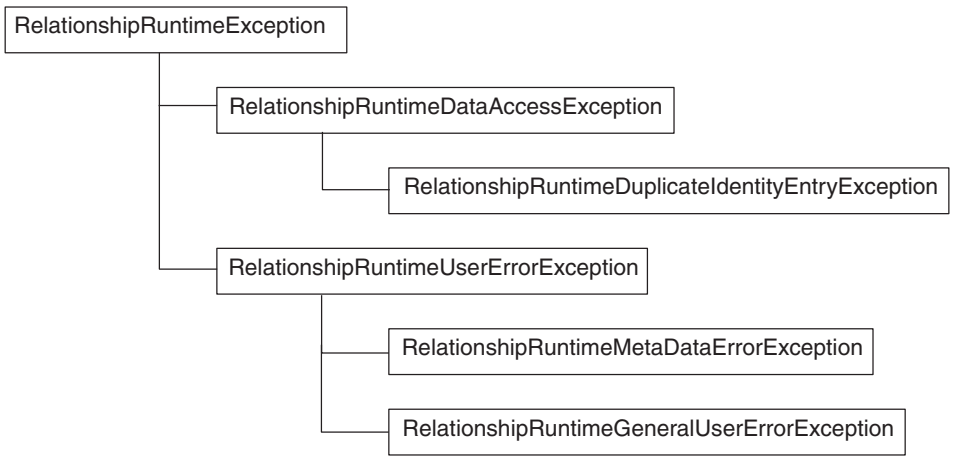


Figura 91. Eccezioni runtime di relazione

Esempio: se si rileva `RelationshipRuntimeUserErrorException`, si rileva automaticamente `RelationshipRuntimeMetaDataErrorException` e `RelationshipRuntimeGeneralUserErrorException`. Tuttavia, non è facile sapere esattamente quale di queste è stata inviata realmente, a meno che non si testi l'eccezione con l'operatore `istanza di`. L'eccezione che si sceglie di rilevare dipende da quanto deve essere specifica la gestione dell'eccezione.

Esempio: gestione di ID istanza di relazione duplicati

Quando si aggiunge un partecipante ad una relazione d'identità utilizzando il metodo `addMyChildren()` o `create()`, si verifica un `RelationshipRuntimeDuplicateIdentityEntryException` se si passa un ID istanza di relazione che esiste già. Questa eccezione fornisce anche un metodo, `getInstanceId()`, per richiamare l'ID istanza di relazione che ha causato la duplicazione.

Esempio: il seguente esempio mostra come è possibile rilevare questa eccezione e richiamare l'ID.

```
int instanceId;

try
   {
   instanceId = Relationship.create(myParticipantObj);
   }
catch (RelationshipRuntimeDuplicateIdentityEntryException rrdiee)
   {
   /*
   ** Esiste già un'istanza di relazione con questa
   ** voce, prendere l'ID istanza dall'eccezione
   */
   instanceId = rrdiee.getInstanceId();
   }
```

Creazione di livelli personalizzati di convalida dati

Quando i valori vengono mappati da un oggetto business ad un altro in base al codice di trasformazione, possono risultare dei dati non corretti. La funzione di convalida dei dati verifica ogni operazione di una mappa e registra un errore quando i dati contenuti nell'oggetto business in entrata non possono essere trasformati in dati nell'oggetto in uscita secondo determinate regole.

Esempio: si supponga che una mappa trasformi un valore di stringa dell'oggetto business origine in un valore intero nell'oggetto business di destinazione. Questo tipo di conversione funziona correttamente quando un valore stringa in entrata rappresenta un numero intero (ad esempio "1234" rappresenta il numero intero 1234). Tuttavia, la conversione non avviene correttamente se il valore stringa non rappresenta un numero intero (ad esempio "ABCD" potrebbe indicare dei dati non validi).

Questa sezione fornisce le seguenti informazioni sull'utilizzo dei livelli di convalida dei dati in una mappa:

- "Codifica del livello di convalida dati"
- "Passi per il test del livello di convalida dati" a pagina 199

Codifica del livello di convalida dati

Il sistema di sviluppo mappe definisce i livelli di convalida dati 0 e 1; i livelli 2 e successivi sono disponibili per essere definiti dall'utente. La Tabella 62 riassume i livelli di convalida dati:

Tabella 62. Livelli di convalida dati

Livello	Descrizione
0	Valore predefinito; nessuna convalida dati
1	Verifiche del tipo di dati definiti dall'IBM
2 e successivi	Verifiche di convalida definite dall'utente

Per creare un livello di convalida personalizzato, fare doppio clic su un attributo per visualizzarne il codice nella finestra di Activity Editor. Osservare questa istruzione `if`, che introduce il codice che invia un'eccezione quando vengono rilevati dei dati non validi:

```
if (dataValidationLevel >= 1)
```

Aggiungere la propria istruzione `if` per specificare l'azione che si desidera associare al livello di convalida dati. Le regole di convalida dati che vengono associate ad un livello possono essere di qualsiasi tipo appropriato necessario per la propria logica di business e le proprie applicazioni. Ad esempio, una regola del livello 2 di un particolare attributo potrebbe verificare un determinato valore e, se non presente, impostare l'attributo su un valore predefinito.

Esempio: il seguente esempio di codice utilizza il livello 2 di convalida dati per verificare che un numero di carta di credito sia valido:

```
/*
** Al livello 2 di convalida dati, verificare i numeri della carta di credito risultanti
** da errori di immissione dati, ecc.
*/
if (dataValidationLevel >= 2)
   {
   if (!CustUtility.validateCreditCard(ObjPurchaseReq.getString
         ("creditCardNumber")))
      {
      logWarning("Invalid credit card number sent through.");
      if (failOnInvalidData)
         {
         throw new MapFailureException(
            "Numero di carta di credito non valido.");
         }
      }
   }
```

Passi per il test del livello di convalida dati

In una esecuzione di test della mappa, è possibile impostare il livello di convalida per vedere se funziona correttamente. Per testare il livello di convalida dati di un'istanza mappa, eseguire i seguenti passi:

1. Visualizzare la finestra di dialogo Proprietà mappa, procedendo come segue:

   Dal menu Modifica, selezionare Proprietà mappa. Per informazioni su altri modi di visualizzare la finestra di dialogo Proprietà mappa, consultare "Specifica delle informazioni sulla proprietà della mappa" a pagina 62.

   Risultato: viene visualizzata la scheda Generale della finestra di dialogo Proprietà.

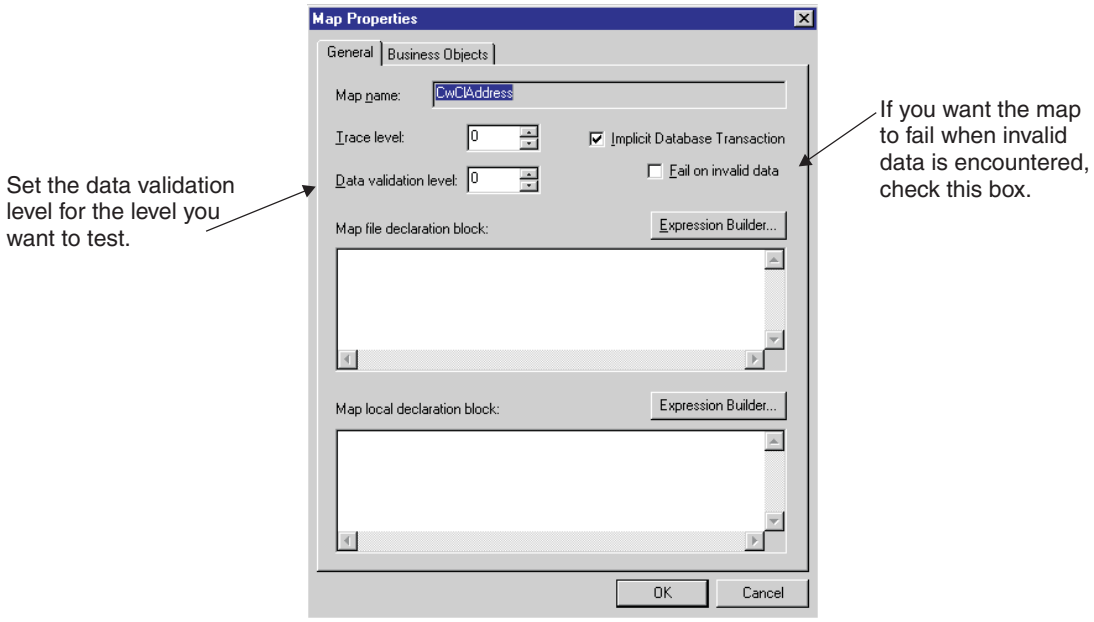


Figura 92. Scheda Generale della finestra di dialogo Proprietà mappa

2. Impostare il livello di convalida dati che si desidera testare. Facoltativamente, impostare la mappa in modo che non riesca, se i dati non sono validi.
3. Arrestare e riavviare la mappa perché il nuovo livello di convalida dati diventi effettivo.
   - Se è stato modificato il codice della mappa, ricompilare la mappa. La ricompilazione riavvia automaticamente la mappa.
   - Se il codice della mappa *non* è stato modificato, non è necessario ricompilare. Tuttavia è necessario arrestare e riavviare esplicitamente la mappa, prima che il nuovo livello di convalida dati divenga effettivo. Utilizzare il menu Componente del Service Manager per arrestare e riavviare una mappa.

Nota: E' anche possibile impostare il livello di convalida dati e se ha esito negativo quando sono presenti dei dati non validi dalla finestra Proprietà mappa e dalla vista di gestione componenti del server.

Contesti di esecuzione delle mappe

Ciascuna istanza di mappa viene eseguita in un *contesto di esecuzione* specifico che viene impostato dall'unità di controllo del connettore. L'API di mappatura rappresenta il contesto di esecuzione della mappa con un'istanza della classe `MapExeContext`.

Per ogni mappa generata da Map Designer Express, il contesto di esecuzione della mappa è accessibile mediante una variabile definita dal sistema, denominata `cwExecCtx`. E' possibile far riferimento a questa variabile nella cartella Variabili in Activity Editor o quando si richiama un metodo API di mappatura che richiede il contesto di esecuzione, incluso quelli contenuti nella Tabella 63.

Tabella 63. Metodi API di mappatura che richiedono il contesto di esecuzione della mappa

Scopo	Metodo API di mappatura	Per ulteriori informazioni
Richiamo di una mappa secondaria	runMap()	"Trasformazione con una mappa secondaria" a pagina 47
Conservazione di una relazione	• addMyChildren() • deleteMyChildren() • updateMyChildren() • maintainChildVerb() • maintainSimpleIdentityRelationship() • maintainCompositeRelationship() • foreignKeyXref() • foreignKeyLookup()	Capitolo 8, "Implementazione delle relazioni", a pagina 271

La Tabella 64 mostra due informazioni molto spesso necessarie, provenienti dal contesto di esecuzione della mappa.

Tabella 64. Metodi MapExeContext per le informazioni sul contesto

Informazioni del contesto	Metodo MapExeContext
Il contesto di chiamata	`getInitiator()`, `setInitiator()`
L'oggetto business di richiesta originale	`getOriginalRequestBO()`

Questa sezione descrive entrambe le informazioni del contesto di esecuzione della mappa.

Nota: Inoltre, è possibile utilizzare i metodi `getConnName()` e `setConnName()` di `MapExeContext` per accedere al nome del connettore dal contesto di esecuzione della mappa.

Contesti di chiamata

Il *contesto di chiamata* indica lo scopo dell'esecuzione della mappa corrente. Quando si trasformano gli attributi di relazione, di solito è necessario intraprendere delle azioni in base al contesto di chiamata della mappa. La Tabella 65 elenca le costanti valide per i contesti di chiamata.

Tabella 65. Contesti di chiamata

Costante contesto di chiamata	Descrizione
EVENT_DELIVERY	Gli oggetti business origine mappati sono degli eventi provenienti da un'applicazione, inviati da un connettore a InterChange Server Express in risposta ad una richiesta di sottoscrizione (flusso attivato da evento).
ACCESS_REQUEST	Gli oggetti business origine mappati sono delle chiamate provenienti da un'applicazione, inviate da un client di accesso a InterChange Server Express (flusso attivato da chiamata).
ACCESS_RESPONSE	Gli oggetti business origine mappati vengono inviati indietro al client di accesso in risposta risposta ad una richiesta di consegna di sottoscrizione.
SERVICE_CALL_REQUEST	Gli oggetti business origine mappati vengono inviati da InterChange Server Express ad un'applicazione, tramite un connettore.
SERVICE_CALL_RESPONSE	Gli oggetti business origine mappati vengono inviati indietro a InterChange Server Express da un'applicazione in risposta ad una richiesta di chiamata di servizio riuscita.
SERVICE_CALL_FAILURE	Gli oggetti business origine mappati vengono inviati indietro a InterChange Server Express da un'applicazione dopo una richiesta di chiamata di servizio non riuscita.

E' possibile fare riferimento a questi contesti di chiamata come costanti nell'oggetto MapExeContext disponibile in ogni mappa creata da Map Designer Express.

Esempio: si fa riferimento al contesto di chiamata SERVICE_CALL_REQUEST come MapExeContext.SERVICE_CALL_REQUEST.

La Figura 93 illustra quando si verifica ciascun contesto di chiamata in un flusso attivato da un evento. Il flusso attivato da un evento viene avviato quando un connettore invia un evento ad una collaborazione in InterChange Server Express.

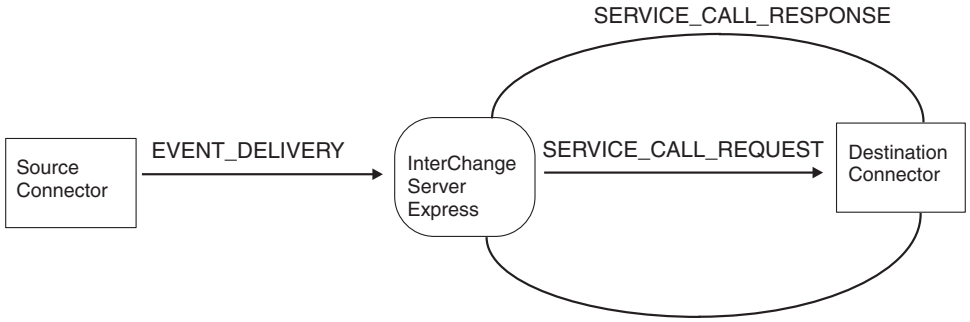


Figura 93. Contesti di chiamata in un flusso attivato da un evento

Come mostra la Figura 93, qualsiasi richiesta di mappatura proveniente da un connettore per InterChange Server Express (cioè una mappa da un oggetto business specifico per applicazione ad un oggetto business generico) ha un contesto di chiamata EVENT_DELIVERY. Qualsiasi richiesta di mappatura proveniente da InterChange Server Express per un connettore (cioè, una mappa da un oggetto business generico ad un oggetto business specifico per applicazione) ha un

contesto di chiamata SERVICE_CALL_REQUEST. Le richieste di mappatura inviate connettori in risposta ad una richiesta di chiamata di servizio di collaborazione può avere i contesti SERVICE_CALL_RESPONSE o SERVICE_CALL_FAILURE.

La Figura 94 illustra quando si verifica ciascun contesto di chiamata in un flusso attivato da chiamata. Un flusso attivato da chiamata viene avviato quando un client di accesso invia una chiamata diretta Server Access Interface ad una collaborazione in InterChange Server Express.

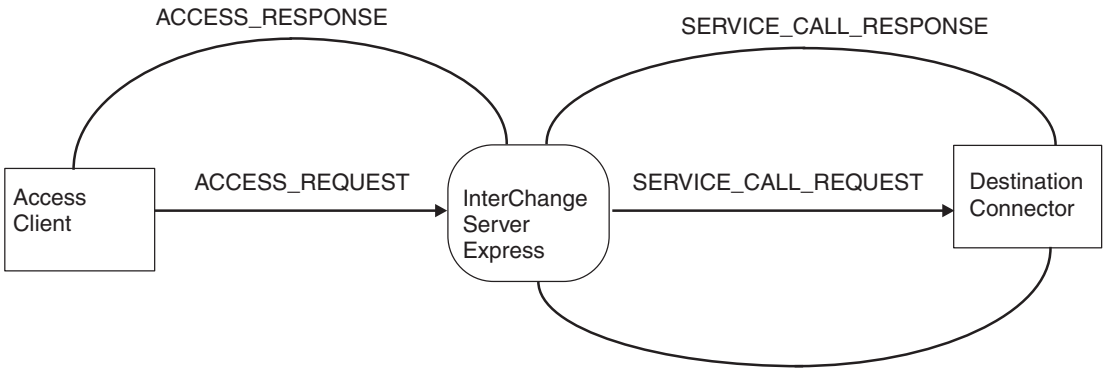


Figura 94. Contesti di chiamata in un flusso attivato da chiamata

Come mostra la Figura 94, qualsiasi richiesta di mappatura proveniente da un client di accesso a InterChange Server Express (cioè, una mappa da un oggetto business specifico per applicazione ad un oggetto business generico) ha un contesto di chiamata ACCESS_REQUEST. Qualsiasi richiesta di mappatura proveniente da InterChange Server Express ad un client di accesso (cioè, una mappa da un oggetto business generico ad un oggetto business specifico per applicazione) ha un contesto di chiamata ACCESS_RESPONSE.

Oggetti business della richiesta originale

Un'altra parte importante del contesto della mappa è l'*oggetto business della richiesta originale*. Questo oggetto business è quello che ha avviato l'esecuzione della mappa. La Tabella 66 mostra i contesti di chiamata e l'oggetto business della richiesta originale associato.

Tabella 66. Contesti di chiamata e gli oggetti business della richiesta originale associati

Contesto di chiamata	Oggetto business della richiesta originale	Oggetto business della richiesta originale da esempio
EVENT_DELIVERY, ACCESS_REQUEST	Oggetto business specifico per applicazione proveniente dall'applicazione	Specifico AppA
SERVICE_CALL_REQUEST, SERVICE_CALL_FAILURE	Oggetto business generico inviato da InterChange Server Express	Generico
SERVICE_CALL_RESPONSE	Oggetto business generico inviato dal SERVICE_CALL_REQUEST	Generico
ACCESS_RESPONSE	Oggetto business specifico per applicazione proveniente inizialmente dalla richiesta di accesso	Specifico AppA

Ad esempio, l'*oggetto business generico* è l'oggetto business della richiesta originale per le mappe che vanno in esecuzione con un contesto di chiamata

SERVICE_CALL_RESPONSE, SERVICE_CALL_FAILURE o SERVICE_CALL_REQUEST. Queste mappe utilizzano l'oggetto business generico per memorizzare gli ID istanza di relazione per gli attributi di relazione che si stanno trasformando. Il possesso degli ID istanza di relazione è necessario alla mappa per cercare l'istanza di relazione ed inserire i dati partecipante rilevanti relativi gli oggetti appena creati o aggiornati.

Esempio: il seguente esempio illustra come questo potrebbe funzionare in uno scenario di sincronizzazione di un cliente. Si supponga di utilizzare il sistema per mantenere i dati sincronizzati fra l'Applicazione A e l'Applicazione B. Entrambe le applicazioni memorizzano i dati del cliente, e gli attributi dell'ID cliente vengonno gestiti tramite una relazione. Per lo scopo di questo esempio, vengono omessi di dettagli sulle collaborazioni ed i connettori implicati.

Quando viene aggiunto un nuovo cliente all'Applicazione A:

1. Una mappa trasforma un oggetto business specifico AppA in un oggetto business generico con un contesto di chiamata EVENT_DELIVERY.

   Quando si trasforma l'attributo ID cliente, la mappa crea una nuova istanza di relazione nella tabella di relazioni ID cliente ed inserisce il nuovo ID istanza di relazione nell'attributo ID cliente dell'oggetto business generico.
2. Una mappa trasforma l'oggetto business generico in un oggetto business specifico AppA con un contesto di chiamata SERVICE_CALL_REQUEST.

   Non si verificano cambiamenti nella tabelle di relazione. L'Applicazione B aggiunge con esito positivo il nuovo cliente all'applicazione.
3. Una mappa trasforma l'oggetto business specifico AppB in un oggetto business generico con un contesto di chiamata SERVICE_CALL_RESPONSE. Il contesto dell'esecuzione di questa mappa include l'oggetto business generico generato nel passo 1.

   Il motivo di questa esecuzione è di inserire i dati del nuovo partecipante per l'istanza di relazione creata al passo 1. In questo caso, il dato del nuovo partecipante è l'ID cliente del nuovo cliente aggiunto all'applicazione B.

La Figura 95 illustra quando avviene l'esecuzione di ogni passo della mappa per un flusso attivato da chiamata che aggiunge con esito positivo un nuovo ID cliente all'Applicazione B.

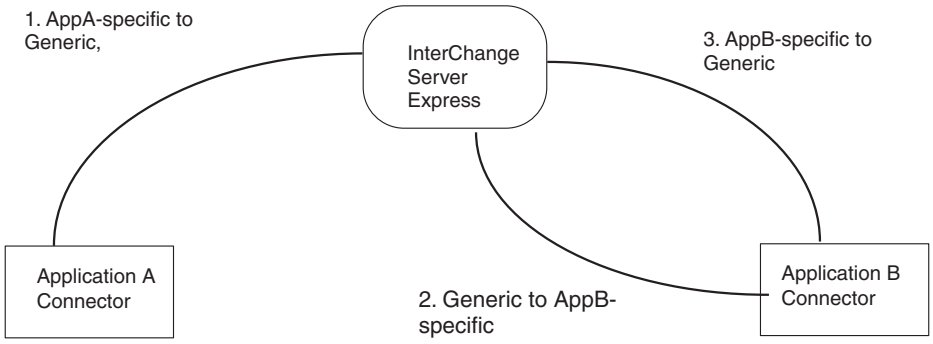


Figura 95. Esempio di contesti di chiamata

Mappatura di oggetti business secondari

Quando l'oggetto business origine contiene degli oggetti business secondari, la loro mappatura dipende dalla cardinalità dell'elemento secondario negli oggetti business origine e di destinazione. Questa sezione fornisce le informazioni su come mappare gli oggetti business secondari nei seguenti casi:

- "Mappatura di origine e destinazione a cardinalità singola"
- "Mappatura dell'origine a cardinalità singola con la destinazione a cardinalità multipla"
- "Mappatura di origine e di destinazione a cardinalità multipla"

Mappatura di origine e destinazione a cardinalità singola

Per mappare un oggetto business origine secondario a cardinalità singola ad un oggetto business, a cardinalità singola, di destinazione:

- Fare clic sul simbolo (+) a sinistra del nome dell'oggetto secondario di origine per espanderlo e seguire i semplici passi di trasformazione (consultare "Specifica delle trasformazioni degli attributi standard" a pagina 40 e "Ulteriori metodi di trasformazione attributi" a pagina 183) per mappare gli attributi di origine a quelli di destinazione. Per fare questo è possibile creare una mappa secondaria ma non è necessario.
- E' buona norma impostare un verbo per l'oggetto secondario de destinazione. Per fare ciò, seguire le istruzioni contenute in "Impostazione del verbo dell'oggetto business di destinazione" a pagina 39. Impostare l'istruzione (verb) sull'istruzione dell'oggetto principale di origine (a meno che gli oggetti non abbiano dei particolari requisiti di relazione).

Esempio: questo tipo di mappatura è necessario per la seguente mappa:

```
SAP_CustomerMaster
```

a

```
Customer.CustomerAddress
```

Mappatura dell'origine a cardinalità singola con la destinazione a cardinalità multipla

Se l'oggetto business secondario di origine è a cardinalità 1, contiene dei dati da mappare solo ad una singola istanza dell'oggetto business secondario di destinazione. Per mappare un oggetto di origine secondario a cardinalità singola ad un oggetto di destinazione secondario a cardinalità multipla, seguire la procedura descritta in "Mappatura di origine e destinazione a cardinalità singola" a pagina 204. Non è necessario creare una mappa secondaria.

Esempio: questo tipo di mappatura è necessario per la seguente mappa:

```
SAP_CustomerMaster.SAP_CustCreditCentralData[1]
```

a

```
Customer.CustomerInformation.CustomerCreditData[n]
```

Mappatura di origine e di destinazione a cardinalità multipla

Per mappare un oggetto business di origine secondario a cardinalità multipla ad un oggetto business di destinazione a cardinalità multipla è necessario creare una mappa secondaria a cardinalità multipla. Per una introduzione alle mappe secondarie, consultare "Trasformazione con una mappa secondaria" a pagina 47.

Esempio: questo tipo di mappatura è necessario per la seguente mappa:
`SAP_CustBankData[n]`

a

`Customer.CustomerInformation.CustomerBankData[n]`

Quando si trasformano più oggetti business origine dello stesso tipo, eseguire le attività secondarie descritte nella Tabella 67.

Tabella 67. Creazione di una mappa secondaria a cardinalità multipla

Attività secondaria	Procedura associata (vedere . . .)
1. Creare una mappa secondaria a cardinalità multipla, che esegue le trasformazioni degli attributi di un oggetto origine ad un oggetto business di destinazione.	"Passi per la creazione di mappe secondarie a cardinalità multipla"
2. Richiamare questa mappa secondaria dalla mappa principale, nell'attributo cardinalità multiple dell'oggetto business di destinazione. Il richiamo della mappa secondaria è contenuto in un loop `for` che esegue un loop per ogni oggetto business contenuto nell'oggetto a cardinalità multipla in modo che ogni oggetto business viene passato alla mappa secondaria.	"Passi per il richiamo della mappa secondaria a cardinalità multipla" a pagina 206

Passi per la creazione di mappe secondarie a cardinalità multipla

Per creare una mappa secondaria che mappa gli oggetti secondari a cardinalità multipla dall'oggetto origine a quello di destinazione, si crea una mappa con un singolo oggetto business di origine ed un singolo oggetto business di destinazione. Questa mappa contiene le trasformazioni degli attributi contenuti nell'oggetto origine nei corrispondenti attributi contenuti nell'oggetto di destinazione.

Per creare una mappa secondaria a cardinalità multipla, eseguire i seguenti passi da Map Designer Express:

1. Chiudere la mappa principale ed avviare una nuova mappa.
2. Nella scheda Diagramma, trascinare l'oggetto business secondario di origine nella metà di sinistra dello spazio di lavoro della mappa e quello di destinazione nella metà di destra.

   Per mappare l'oggetto business `SAP_CustBankData` all'oggetto business `Customer.CustomerInformation.CustomerBankData` trascinare `SAP_CustBankData` nella metà di sinistra dello spazio di lavoro e `CustomerBankData` nella metà di destra.
3. Salvare la mappa secondaria.

   Raccomandazione: iniziare i nomi delle mappe secondarie con il prefisso "`Sub_`". Ad esempio: `Sub_SaCwCustBankData`.
4. Impostare l'istruzione dell'oggetto di destinazione come descritto in "Impostazione del verbo dell'oggetto business di destinazione" a pagina 39.
5. Mappare i singoli attributi come descritto in "Specifica delle trasformazioni degli attributi standard" a pagina 40 and "Ulteriori metodi di trasformazione attributi" a pagina 183.
6. Compilare la mappa secondaria selezionando Compila dal menu File.

   Risultato: se tutto è corretto, viene visualizzato il seguente messaggio:

   ```
   Convalida mappa riuscita.
   Compilazione mappa riuscita.
   ```

Suggerimenti: Se si dimentica di compilare la mappa secondaria, non sarà possibile visualizzarla nella finestra di dialogo mappa secondaria. Viene visualizzato il messaggio `Non sono disponibili mappe secondarie`.

Passi per il richiamo della mappa secondaria a cardinalità multipla

Per richiamare una mappa secondaria a cardinalità multipla, eseguire il metodo runMap() dalla mappa principale, nell'attributo cardinalità multipla dell'oggetto business di destinazione. Il richiamo della mappa secondaria è contenuto in un loop `for` che esegue un loop per ogni oggetto business contenuto nell'oggetto a cardinalità multipla in modo che ogni oggetto business viene passato alla mappa secondaria.

Per richiamare una mappa secondaria a cardinalità multipla, eseguire i seguenti passi da Map Designer Express:

1. Chiudere la mappa secondaria ed aprire la mappa principale.
2. Selezionare l'oggetto origine secondario e l'oggetto di destinazione secondario e selezionare Mappa secondaria dalla casella combinata della regola di trasformazione.

   Per la trasformazione da `SAP_CustBankData` a `Customer.CustomerInformation.CustomerBankData` selezionare `SAP_CustBankData` e `CustomerBankData`.

   Risultato: viene visualizzata la finestra di dialogo Mappa secondaria.
3. Selezionare il nome della mappa secondaria e fare clic su OK.

   Esempio: `Sub_SaCwCustBankData`

   In questo esempio non è necessario specificare una condizione, quindi fare clic su OK.
4. Aprire l'Activity Editor dell'oggetto di destinazione secondario.

   Per la trasformazione da `SAP_CustBankData` a `Customer.CustomerInformation.CustomerBankData` in Activity Editor verrà visualizzato del codice simile al seguente:

```
{
BusObjArray srcCollection_For_ObjSAP_Order_SAP_OrderPartners =
   ObjSAP_Order.getBusObjArray("SAP_OrderPartners");

//
// LOOP SOLO SU VETTORI NON VUOTI
// ------------------------------
//
// Esegue il loop solo se il vettore origine non è vuoto.
//
if ((srcCollection_For_ObjSAP_Order_SAP_OrderPartners != null) &&
      (srcCollection_For_ObjSAP_Order_SAP_OrderPartners.size() > 0))
   {
   int currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners;
   int lastInputIndex_For_ObjSAP_Order_SAP_OrderPartners =
   srcCollection_For_ObjSAP_Order_SAP_OrderPartners.getLastIndex();
   for (currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners = 0;
      currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners <=
      lastInputIndex_For_ObjSAP_Order_SAP_OrderPartners;
      currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners++)
      {
      BusObj currentBusObj_For_ObjSAP_Order_SAP_OrderPartners =
(BusObj) (srcCollection_For_ObjSAP_Order_SAP_OrderPartners.elementAt(
      (currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners));

      //
```

```
      // RICHIAMA MAPPA SU OGGETTI VALIDI
      // ---------------------------
      //
      // Richiama la mappa solo per gli oggetti secondari che soddisfano determinati
      // criteri.
      //
      if (currentBusObj_For_ObjSAP_Order_SAP_OrderPartners != null)
         {
         BusObj[] _cw_inObjs =
            { currentBusObj_For_ObjSAP_Order_SAP_OrderPartners };
         BusObj[] _cw_outObjs =
            DtpMapService.runMap(
            "Sub_SaOrderPartners_to_CwCustomerRole", "CwMap",
            _cw_inObjs, cwExecCtx);
   ObjOrder.setWithCreate("AssociatedCustomers",
            _cw_outObjs[0]);
         }
      }
   }
}
```

Notare che `runMap()` è un metodo statico quindi viene richiamato come:

```
DtpMapService.runMap()
```

5. Apportare alcune modifiche al codice:
   - Nell'oggetto business origine, solo l'oggetto principale ha l'istruzione ad esso associato. Nel codice per richiamare le mappe secondarie per mappare gli oggetti secondari, è buona norma impostare l'istruzione dell'oggetto secondario su quella dell'oggetto principale, prima di passare l'oggetto secondario nella mappa secondaria. Utilizzare il metodo `setVerb()` della classe `BusObj`.

     Nota: Se la gestione delle relazioni (riferimenti incrociati) deve essere eseguita nella mappa secondaria, *è richiesta* la trasmissione dell'istruzione. Consultare "Definizione delle regole di trasformazione per una relazione di identità semplice" a pagina 288 per ulteriori informazioni.
   - La riga di codice per richiamare la mappa secondaria deve essere inclusa nel blocco `try/catch` per essere sicuri di rilevare `MapNotFoundException`.

   Questo è il codice modificato (le modifiche sono evidenziate in grassetto):

```
   {
   BusObjArray srcCollection_For_ObjSAP_
Order_SAP_OrderPartners =      ObjSAP_Order.getBusObjArray("SAP_OrderPartners");

   //
   // LOOP SOLO SU VETTORI NON VUOTI
   // ----------------------------
   //
   // Esegue il loop solo se il vettore origine non è vuoto.
   //
   if ((srcCollection_For_ObjSAP_Order_SAP_OrderPartners
      != null) &&
         (srcCollection_For_ObjSAP_Order_SAP_OrderPartners.size()
    > 0))
      {
      int currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners;
      int lastInputIndex_For_ObjSAP_Order_SAP_OrderPartners =
         srcCollection_For_ObjSAP_Order_SAP_OrderPartners.getLastIndex();

      for (currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners
    = 0;
            currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners
    <=
```

```
            lastInputIndex_For_ObjSAP_Order_SAP_OrderPartners;
            currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners++)
          {
            BusObj currentBusObj_For_ObjSAP_Order_SAP_OrderPartners =
(BusObj)
                (srcCollection_For_ObjSAP_Order_SAP_OrderPartners.elementAt(
            currentBusObjIndex_For_ObjSAP_Order_SAP_OrderPartners));

          //
          // RICHIAMA MAPPA SU OGGETTI VALIDI
          // --------------------------
          //
          // Richiama la mappa solo per gli oggetti secondari che soddisfano determinati
          // criteri.
          //
          if (currentBusObj_For_ObjSAP_Order_SAP_OrderPartners != null)
             {
             currentBusObj_For_ObjSAP_Order_SAP_OrderPartners.setVerb(
                ObjSAP_Order.getVerb());            BusObj[] _cw_inObjs

             =  { currentBusObj_For_ObjSAP_Order_SAP_OrderPartners };

             try
                {
                BusObj[] _cw_outObjs = DtpMapService.runMap(
                   "Sub_SaOrderPartners_to_CwCustomerRole", "CwMap",
                   _cw_inObjs, cwExecCtx);
                ObjOrder.setWithCreate("AssociatedCustomers", _cw_outObjs[0]);
                }
             catch (MapNotFoundException me)
                {
                logError(5502, "Sub_SaOrderPartners_to_CwCustomerRole");
                throw new MapFailureException ("Mappa secondaria non trovata");
                }            }
          }
        }
      }
```

6. Una volta aggiunto il richiamo della mappa secondaria, ricompilare la mappa principale.

Ulteriori informazioni sull'utilizzo delle mappe secondarie

Questa sezione fornisce i seguenti suggerimenti sull'utilizzo delle mappe secondarie:

- "Specifica di condizioni durante il richiamo della mappa secondaria"
- "Utilizzo del Generatore di espressioni per il richiamo di una mappa secondaria" a pagina 211
- "Trasmissione di oggetti business di tipo diverso alle mappe secondarie" a pagina 212

Per una introduzione alle mappe secondarie, consultare "Trasformazione con una mappa secondaria" a pagina 47.

Specifica di condizioni durante il richiamo della mappa secondaria

Spesso la mappa richiede della logica di programmazione per stabilire quando richiamare una mappa secondaria. Alcune volte potrebbero esserci delle condizioni che devono essere true, affinché la mappa secondaria possa essere richiamata. Questa logica va nella mappa principale, nell'attributo che contiene l'oggetto business di destinazione della mappa secondaria, e prima della chiamata al metodo

runMap(). Se queste condizioni si inseriscono nell'area Condizioni della finestra di dialogo Mappa secondaria (consultare Figura 22 a pagina 50), Map Designer Express le aggiunge all'istruzione if con la quale racchiude la chiamata runMap().

Linee guida: quando si inseriscono queste condizioni, tenere presente quanto segue:

- E' possibile immettere solo una condizione, anche se la condizione può includere più clausole, combinate con l'operatore AND (&&).
- Non terminare la riga con il punto e virgola perché la condizione immessa diviene una clausola if nel codice generato dell'attributo di destinazione.

Esempio: negli oggetti business mostrati nella Figura 21 a pagina 48 l'oggetto business OrderLine ha un attributo denominato DelSched, che è un oggetto business secondario. In una condizione della mappa secondaria, si può fare riferimento a quell'attributo nel seguente modo:

```
srcBusObj
```

Immettere la seguente condizione della mappa secondaria nell'area Condizioni della finestra di dialogo Mappa secondaria per eseguire una mappa secondaria sull'oggetto business DelSched *solo se* il valore dell'attributo TransportType dello stesso oggetto business è uguale alla stringa AIR.

```
srcBusObj.getString("TransportType").equals("AIR")
```

La seguente condizione esegue la mappa secondaria per gli attributi OrderLine DelSched *solo se* il valore dell'attributo OrderLine LinePrice è maggiore di $10.000,00.

```
ObjOrderLine.getFloat["LinePrice") > 10000.00
```

Esempio: le condizioni sono necessarie per la seguente mappa perché la mappatura deve avvenire *solo* se SAP_CustPartnerFunctions.PartnerId *non è uguale a* SAP_CustomerMaster.CustomerId:

```
SAP_CustomerMaster.SAP_CustSalesAreaData.SAP_
CustPartnerFunctions[n]
```

a

```
Customer.RelatedCustomerRef
```

Le sezioni che seguono descrivono i passi per la creazione del richiamo della mappa:

- "Passi per la creazione della mappa secondaria"
- "Passi per il richiamo della mappa secondaria" a pagina 210

Passi per la creazione della mappa secondaria

Per creare la mappa secondaria Sub_SaCwCustCreditAreaData eseguire i seguenti passi da Map Designer Express:

1. Chiudere la mappa principale ed avviare una nuova mappa secondaria.

   Specificare un oggetto business origine SAP_CustPartnerFunctions ed un oggetto business di destinazione RelatedCustomerRef. Denominare la mappa Sub_SaCwCustPartners.
2. Impostare l'istruzione come descritto in "Impostazione del verbo dell'oggetto business di destinazione" a pagina 39.

3. Mappare i singoli attributi come descritto in “Specifica delle trasformazioni degli attributi standard” a pagina 40 e “Ulteriori metodi di trasformazione attributi” a pagina 183.
4. Compilare la mappa secondaria selezionando Compila dal menu File. Se tutto è corretto, viene visualizzato il seguente messaggio:

   ```
   Convalida mappa riuscita.
   Mappa nativa: creazione codice riuscita.
   ```

Passi per il richiamo della mappa secondaria

Per richiamare la mappa secondaria `Sub_SaCwCustCreditAreaData`, eseguire i seguenti passi da Map Designer Express:

1. Tornare alla mappa principale e nella scheda Diagramma, tenendo premuto il tasto `Ctrl`, trascinare `SAP_CustPartnerFunctions` in `RelatedCustomerRef`. Fare doppio clic su Mappa secondaria nell’elenco della colonna Regola dell’oggetto business di destinazione. Selezionare la mappa `SaCwCustPartners` nella finestra di dialogo mappa secondaria e fare clic su OK.

   In caso di condizione semplice, è possibile digitarla nell’area Condizione della finestra di dialogo Mappa secondaria:

   ```
   srcBusObj.getString("PartnerId").equals("some_id")
   ```

   Map Designer Express genera il codice per il richiamo della mappa secondaria con la condizione aggiunta all’istruzione `if` che precede il richiamo della mappa secondaria.

   Se si lascia vuoto il campo Condizione e si fa clic su OK, Map Designer Express genera il codice per il richiamo della mappa secondaria senza la condizione. Continuare al passo 2 per informazioni su come aggiungere esplicitamente il codice.
2. Modificare il codice generato per aggiungere la condizione della mappa secondaria.

   Per accedere a Activity Editor, fare di nuovo doppio clic sulla colonna Regola e fare clic sul pulsante Visualizza codice della finestra di dialogo Mappa secondaria.

   Per l’attributo `RelatedCustomerRef`, segue il codice che include la condizione:

   ```
   {
         BusObjArray srcBusObjs = null;
         srcBusObjs =
   ObjSAP_CustomerMaster.getBusObjArray
   ("SAP_CustSalesAreaData.SAP_CustPartnerFunctions");
     String parent_custid = ObjSAP_CustomerMaster.getString("CustomerId");
     //
     // LOOP SOLO SU VETTORI NON VUOTI
         // ----------------------------
         //
         // Esegue il loop solo se il vettore origine non è vuoto.
         //
         if ((srcBusObjs != null) && (srcBusObjs.size() > 0))
     {
       int srcIndex;
              int lastSrcIndex = srcBusObjs.getLastIndex();
              for (srcIndex = 0; srcIndex <= lastSrcIndex; srcIndex++)
       {
         BusObj srcBusObjInstance = (BusObj)
   (srcBusObjs.elementAt(srcIndex));
         //
         // RICHIAMA MAPPA SU OGGETTI VALIDI
                    // --------------------------
         //
                    // Richiama la mappa solo per gli oggetti secondari
                    //  che soddisfano determinati criteri.
   ```

```
                    //
          if ((srcBusObjInstance != null))
          {
          // verifica della condizione di esecuzione della mappa secondaria
          if
      (!(srcBusObjInstance.getString("PartnerId")).equals(parent_custid))
          {
             // imposta l'istruzione (verb) dell'oggetto business origine della mappa secondaria
    srcBusObjInstance.setVerb(ObjSAP_CustomerMaster.getVerb());
             try
             {
               BusObj[] _cw_inObjs = { srcBusObjInstance };
                                       BusObj[] _cw_outObjs =
    DtpMapService.runMap("Sub_SaCwCustPartners",
    "CwMap", _cw_inObjs,cwExecCtx);
    ObjCustomer.setWithCreate("RelatedCustomerRef", _cw_outObjs[0]);
             }
             catch(MapNotFoundException me)
                                  {
                                     logError(5502, "Sub_SaCwCustPartners");
             throw new MapFailureException("Mappa secondaria non trovata");
             }
           }
         }
               }
     }
    }
```

3. Una volta aggiunto il richiamo della mappa secondaria, ricompilare la mappa principale.

Utilizzo del Generatore di espressioni per il richiamo di una mappa secondaria

E' possibile utilizzare Map Designer Express per generare automaticamente un richiamo di mappa secondaria mediante la finestra di dialogo Mappa secondaria. Tuttavia è possibile utilizzare anche il Generatore di espressioni per generare il richiamo della mappa secondaria. La codifica del richiamo della mappa consente di scrivere delle operazioni più variate di quante Map Designer Express non ne supporti graficamente. Ad esempio, è possibile utilizzare una mappa secondaria per fornire un valore ad un attributo che non contiene un oggetto business secondario o utilizzare una mappa secondaria che dispone di più input e output.

Passi per l'utilizzo del Generatore di espressioni per richiamare una mappa secondaria

Per utilizzare il generatore di espressioni per generare un richiamo di mappa secondaria, eseguire i seguenti passi:

1. Fare doppio clic sulla regola di trasformazione nella scheda Tabella o Diagramma per visualizzare Activity Editor nella vista Java.
2. fare clic con il pulsante destro del mouse in qualunque punto di Activity Editor e selezionare Generatore di espressioni dal menu contestuale.
3. Nel generatore di espressioni:
   - Nella colonna Categorie principali, espandere la cartella Mappa.
   - In Mappe, espandere il tipo di mappa: Mappa nativa per visualizzare un elenco di tutte le mappe compilate disponibili sul sistema.
   - Selezionare la mappa da richiamare e selezionare il metodo `DtpMapService.runMap()` nella colonna API della mappa.
   - Fare clic sul pulsante Inserisci API per avviare la creazione di codice (vedere la Figura 96).

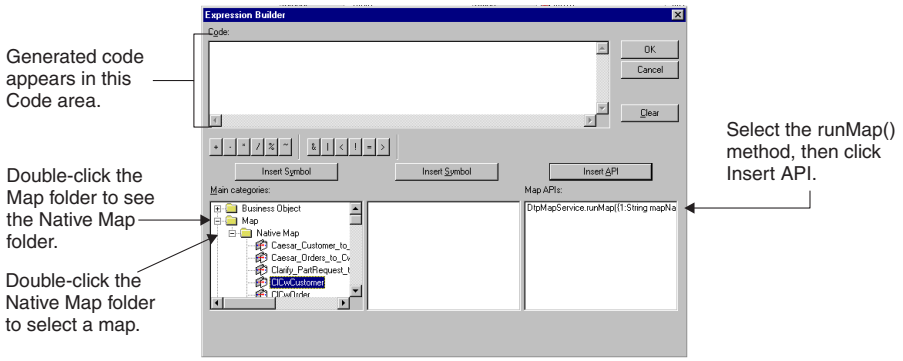


Figura 96. Codifica di un richiamo di mappa secondaria

Risultato: Map Designer Express genera il codice Java necessario a richiamare la mappa specificata come mappa secondaria. Viene visualizzato nell'area del codice in alto nella finestra Generatore di espressioni.

4. Nel codice generato, sostituire i segnaposto degli oggetti di input e output con gli effettivi nomi degli oggetti business origine e di destinazione correnti per la mappa secondaria e scrivere il codice per gestire il valore che restituisce la mappa secondaria.

   Sostituire i segnaposto `InObj`, `OutObj` e `OutAttrName` con i nomi corretti per:

   - La variabile dell'oggetto business origine del livello superiore: `Obj`*`SrcBusObj`*
   - L'attributo origine: il percorso per il nome dell'attributo che contiene l'oggetto business secondario
   - La variabile dell'oggetto business di destinazione: `Obj`*`DestBusObj`*
   - L'attributo di destinazione.

5. Fare clic su OK per chiudere il Generatore di espressioni.

   Risultato: il codice viene inserito in Activity Editor nel punto di inserimento.

Trasmissione di oggetti business di tipo diverso alle mappe secondarie

Per mappare gli oggetti business origine di tipo diverso ad un oggetto business di destinazione è necessario creare una mappa secondaria many-to-one (molti a uno).

Esempio: questo tipo di mappatura è necessario per la seguente mappa:

```
SAP_CustCreditControlAreaData[n]

a
Customer.CustomerInformation.CustomerCreditData[0].
CreditAreaCreditData[n]
```

Alcuni degli attributi devono essere mappati da `SAP_CustomerMaster` e `SAP_CustCreditCentralData`.

La trasformazione di più oggetti business origine di tipo diverso implica i seguenti passi principali:

1. "Passi per la creazione della mappa secondaria many-to-one" a pagina 213—crea una mappa secondaria many-to-one, che trasforma gli attributi necessari contenuti negli oggetti business origine multipli in quelli del singolo oggetto business di destinazione.

2. “Passi per il richiamo della mappa secondaria many-to-one”—Richiama la mappa secondaria dalla mappa principale, nell’attributo che conserva l’oggetto business di destinazione della mappa secondaria.

Passi per la creazione della mappa secondaria many-to-one

Per creare una mappa secondaria che mappa gli oggetti business origine di tipo diverso ad un unico oggetto di destinazione secondario, si crea una mappa con i vari oggetti business di origine ed il singolo oggetto business di destinazione. Questa mappa contiene le trasformazioni degli attributi contenuti nei diversi oggetti origine, nei corrispondenti attributi contenuti nell’oggetto di destinazione.

Per creare una mappa secondaria many-to-one, eseguire i seguenti passi da Map Designer Express:

1. Chiudere la mappa principale ed avviare una nuova mappa. La nuova mappa avrà almeno due oggetti origine e un oggetto di destinazione.

   Trascinare gli oggetti business `SAP_CustCreditControlAreaData`, `SAP_CustCreditCentralData` e `SAP_CustomerMaster` nella metà di sinistra della finestra di mappatura e `CreditAreaCreditData` nella metà di destra. La nuova mappa avrà tre oggetti origine ed un oggetto di destinazione.
2. Salvare la mappa secondaria.

   Raccomandazione: i nomi delle mappe secondarie devono iniziare con il prefisso “`Sub_`”. Ad esempio: `Sub_SaCwCustCreditAreaData`.
3. Impostare l’istruzione dell’oggetto di destinazione come descritto in “Impostazione del verbo dell’oggetto business di destinazione” a pagina 39.

   A meno che nella mappa secondaria non venga eseguita la gestione di relazioni, *non* importa quale oggetto origine viene utilizzato per impostare l’istruzione. Accertarsi che quando si richiama questa mappa secondaria, si imposti l’istruzione per lo stesso oggetto specifico, prima di passarlo alla mappa secondaria. Se uno degli oggetti origine è l’oggetto origine principale e si sceglie di utilizzare la sua istruzione, non è necessario impostare l’istruzione prima di passare quell’oggetto alla mappa secondaria; dispone già di un’istruzione ad esso associata.
4. Mappare i singoli attributi come descritto in “Specifica delle trasformazioni degli attributi standard” a pagina 40 and “Ulteriori metodi di trasformazione attributi” a pagina 183.
5. Compilare la mappa secondaria selezionando Compila dal menu File.

   Risultato: se tutto è corretto, viene visualizzato il seguente messaggio:

   ```
   Convalida mappa riuscita.
   Mappa nativa: creazione codice riuscita.
   ```

Passi per il richiamo della mappa secondaria many-to-one

Per richiamare una mappa secondaria many-to-one, eseguire il metodo `runMap()` dalla mappa principale, nell’attributo che conserva l’oggetto business di destinazione della mappa secondaria. Per richiamare la mappa secondaria `Sub_SaCwCustCreditAreaData`, eseguire i seguenti passi da Map Designer Express:

1. Aprire la mappa principale.

   Per la mappa secondaria `Sub_SaCwCustCreditAreaData`, la mappa principale è la mappa che contiene l’oggetto business `CreditAreaCreditData` come attributo.
2. Fare doppio clic nella colonna della regola di trasformazione dell’oggetto di destinazione per aprire Activity Editor.

   Fare doppio clic nella colonna Regola di trasformazione associata all’attributo `CreditAreaCreditData` per aprire il relativo Activity Editor ed immettere il seguente codice:

```
{
BusObj srcobj1 = ObjSAP_CustomerMaster;
BusObj srcobj2 = (BusObj)
   (ObjSAP_CustomerMaster.getBusObj("SAP_CustCreditCentralData"));
BusObjArray srcobj3 =
   (BusObjArray)(ObjSAP_CustomerMaster.get(
      "SAP_CustCreditControlAreaData"));

//
// RICHIAMO MAPPA SU OGGETTI VALIDI
// --------------------------
//
// Richiama la mappa solo per gli oggetti secondari che soddisfano determinati
// criteri.
//
int i = 0;

// Quando si verificano tutti e 3 gli oggetti origine, != null potrebbe non essere richiesto
if (srcobj1 != null && srcobj2 != null & srcobj3 != null)
   {
   for (i = 0; i < srcobj3.size(); i++)
      {
      BusObj[] _cw_inObjs = new BusObj[3];

      // impostare l'istruzione per uno dei seguenti oggetti
      _cw_inObjs[0] = srcobj1;
      _cw_inObjs[1]= srcobj2;
      _cw_inObjs[2] = srcobj3.elementAt(i);

      try
         {
         BusObj[] _cw_outObjs = DtpMapService.runMap(
            "Sub_SaCwCustCreditAreaData",
            "CwMap",
            _cw_inObjs,
            cwExecCtx);
         ObjCustomer.setWithCreate(
"CustomerInformation.CustomerCreditData[0].CreditAreaCreditData["
            + i + "]", _cw_outObjs[0]);
         }

      catch (MapNotFoundException me)
         {
         logError(5502, "Sub_SaCwCustCreditAreaData");
         throw new MapFailureException ("Mappa secondaria non trovata");
         }
      }
   }
}
```

3. Per tenere traccia dell'esecuzione della mappa, utilizzare il metodo `logInfo()` all'interno del loop `for`. Segue il codice.

   ```
   logInfo("in for loop");
   ```

   o

   ```
   trace("in for loop");
   ```

4. Una volta aggiunto il richiamo della mappa secondaria, ricompilare la mappa principale.

Se tutto funziona in maniera corretta e nessuno degli oggetti origine è null, viene visualizzato il messaggio `in for loop` nel file di log di InterChange Server Express tante volte quante sono le istanze dell'oggetto origine secondario.

Esecuzione delle query nel database

Durante l'esecuzione di una mappa, potrebbe essere necessario ottenere delle informazioni da un database, come il database delle relazioni. Per ottenere o modificare le informazioni di un database, si eseguono le query nelle relative tabelle. Una *query* è una richiesta, di solito sotto forma di istruzione SQL (Structured Query Language), che viene inviata al database per l'esecuzione. La Tabella 68 mostra le attività secondarie per l'esecuzione di una query in un database.

Nota: E' possibile accedere a qualunque database esterno supportato da IBM con JDBC tramite il driver WebLogic.

Tabella 68. Attività secondarie per l'esecuzione di una query

Attività secondarie per l'esecuzione di una query		Procedura associata (vedere . . .)
1.	Ottenimento di una connessione (che è un oggetto `CwDBConnection`) al database.	"Ottenimento di una connessione" a pagina 215
2.	Mediante l'oggetto `CwDBConnection`, invio delle query e gestione delle transazioni nel database.	"Esecuzione della query" a pagina 216 "Gestione della transazione" a pagina 228
3.	Rilascio della connessione.	"Rilascio di una connessione" a pagina 231

Ottenimento di una connessione

Per poter effettuare delle query nel database, è necessario prima ottenere una connessione a quel database con il metodo `getDBConnection()` della classe `BaseDLM`. Per identificare la connessione da ottenere, specificare il nome del pool di connessioni che la contiene. Tutte le connessioni di un particolare pool sono per lo stesso database. Il numero di connessioni contenute nel pool viene stabilito come parte della configurazione del pool di connessioni. E' necessario determinare il nome del pool che contiene le connessioni per il database in cui si desidera eseguire le query.

Importante: Le connessioni vengono aperte quando si carica InterChange Server Express oppure si aprono dinamicamente quando viene configurato un nuovo pool di connessioni. Quindi, il pool che contiene le connessioni al database desiderato deve essere configurato prima dell'esecuzione dell'istanza della mappa che richiede la connessione. I pool di connessioni vengono configurati in IBM System Manager.

Nella Figura 97, la chiamata a `getDBConnection()` ottiene una connessione al database associato alle connessioni contenute nel pool `CustDBConnPool`.

```
CwDBConnection connection =
 getDBConnection("CustDBConnPool");
```

Figura 97. Ottenimento di una connessione da un pool di connessioni

La chiamata `getDBConnection()` restituisce un oggetto `CwDBConnection` nella variabile `connection`, che si può quindi utilizzare per accedere al database associato alla connessione.

Suggerimenti: Il metodo `getDBConnection()` fornisce un'ulteriore forma che consente di specificare il modello di programmazione della

transazione per la connessione. Per ulteriori informazioni, consultare “Gestione della transazione” a pagina 228.

Esecuzione della query

La Tabella 69 mostra i modi in cui possono essere eseguite le query SQL con i metodi della classe CwDBConnection.

Tabella 69. Esecuzione delle query SQL con i metodi CwDBConnection

Tipo di query	Descrizione	Metodo CwDBConnection
Query statica	L'istruzione SQL viene inviata al database come testo.	`executeSQL()`
Query preparata	Dopo la sua esecuzione iniziale, l'istruzione SQL viene salvata nel suo formato compilato eseguibile, in modo che le successive esecuzioni possono utilizzare questa forma precompilata.	`executePreparedSQL()`
Procedura memorizzata	Una procedura definita dall'utente che contiene le istruzioni SQL e la logica delle condizioni	`executeSQL()executePreparedSQL()` `executeStoredProcedure()`

Esecuzione delle query statiche

Il metodo `executeSQL()` invia per l'esecuzione una query statica al database. Una *query statica* è n'istruzione SQL inviata come stringa al database, che analizza la stringa ed esegue l'istruzione SQL che ne risulta. Questa sezione descrive come inviare i seguenti tipi di query SQL al database con `executeSQL()`:

- Query che restituiscono dati dal database (SELECT)
- Query che modificano i dati nel database (INSERT, UPDATE, DELETE)
- Query che eseguono procedure memorizzate definite nel database

Esecuzione di query statiche che restituiscono dati (SELECT): L'istruzione SQL SELECT interroga una o più tabelle per ottenere dei dati. Per inviare un'istruzione SELECT al database per l'esecuzione, specificare una rappresentazione di stringa di SELECT come argomento al metodo `executeSQL()`.

Esempio: la seguente chiamata a `executeSQL()` invia un SELECT di un valore di colonna dalla tabella `customer`:

```
connection.executeSQL(
   "select cust_id from customer where active_status = 1");
```

Nota: Nel codice precedente, la variabile `connection` è un oggetto `CwDBConnection` ottenuto da una chiamata precedente al metodo `getDBConnection()` (vedere la Figura 97).

E' possibile anche inviare un'istruzione SELECT che contiene dei parametri utilizzando il secondo formato del metodo `executeSQL()`.

Esempio: la seguente chiamata a `executeSQL()` esegue la stessa attività dell'esempio precedente tranne che passa lo stato attivo come parametro all'istruzione SELECT:

```
Vector argValues = new Vector();
String active_stat = "1";
argValues.add( active_stat );
connection.executeSQL(
   "select cust_id from customer where
active_status = ?", argValues);
```

L'istruzione SELECT restituisce i dati dalle tabelle del database come righe. Ogni riga è una riga proveniente dai dati che corrispondono alle condizioni contenute nella clausola WHERE di SELECT. Ogni riga contiene i valori delle colonne specificate dall'istruzione SELECT. E' possibile visualizzare i dati restituiti sotto forma di vettore bidimensionale di queste righe e colonne.

Suggerimenti: La sintassi dell'istruzione SELECT deve essere valida per il particolare database che si sta accedendo. Per la sintassi esatta dell'istruzione SELECT consultare la documentazione relativa al database.

Per accedere ai dati restituiti, eseguire i seguenti passi:

1. Ottenere una riga di dati.
2. Ottenere i valori della colonna, uno per uno.

La Tabella 70 mostra i metodi della classe `CwDBConnection` che forniscono l'accesso alle righe dei dati restituiti della query.

Tabella 70. Metodi CwDBConnection per l'accesso alle righe

Attività di accesso alla riga	Metodo CwDBConnection
Verifica dell'esistenza di una riga.	`hasMoreRows()`
Ottenimento di una riga di dati.	`nextRow()`

Controllare il loop mediante le righe restituite con il metodo `hasMoreRows()`. Terminare il loop di righe quando `hasMoreRows()` restituisce `false`. Per ottenere una riga di dati, utilizzare il metodo `nextRow()`. Questo metodo restituisce i valori della colonna selezionata come elementi di un oggetto Java `Vettore`. E' possibile quindi utilizzare la classe `Enumerazione` per accedere ai valori della colonna singolarmente. Entrambe le classi `Vettore` ed `Enumerazione` sono nel pacchetto `java.util`.

La Tabella 71 mostra i metodi Java per l'accesso alle colonne di una riga restituita della query.

Tabella 71. Metodi Java per l'accesso al valore di colonna

Attività di accesso alla colonna	Metodo Java
Determinazione del numero di colonne.	`Vector.size()`
Assegnazione di Vettore a Enumerazione.	`Vector.elements()`
Verifica esistenza di una colonna.	`Enumeration.hasMoreElements()`
Ottenimento di una colonna di dati.	`Enumeration.nextElement()`

Controllare il loop mediante i valori della colonna con il metodo `hasMoreElements()`. Terminare il loop della colonna quando `hasMoreElements()` restituisce `false`. Per ottenere un valore di colonna, utilizzare il metodo `nextElement()`.

Esempio: il seguente esempio di codice semplice richiama un'istanza della classe `CwDBConnection`, che è una connessione ad un database che memorizza le informazioni del cliente. Quindi esegue un'istruzione SELECT che restituisce solo una riga, che contiene una singola colonna, il nome dell'azienda "`CrossWorlds`" per l'id cliente 20987:

```
CwDBConnection connectn = null;
Vector theRow = null;
Enumeration theRowEnum = null;
String theColumn1 = null;

try
   {
   // Ottiene una connessione al database
   connectn = getDBConnection("sampleConnectionPoolName");
   }

catch(CwDBConnectionFactoryException e)
   {
   System.out.println(e.getMessage());
   throw e;
   }

// Testa una serie di risultati a singola colonna, singola riga
try
   {
   // Invia l'istruzione SELECT al database
   connectn.executeSQL(
      "select company_name from customer where cust_id = 20987");

   // Loop attraverso ciascuna riga
   while(connectn.hasMoreRows())
      {
      // Ottiene una riga
      theRow = connectn.nextRow();
      int length = 0;
      if ((length = theRow.size())!= 1)
         {
         return methodName + "Expected result set size = 1," +
            " Actual result state size = " + length;
         }

      // Richiama valori di colonna come oggetto Enumerazione
      theRowEnum = theRow.elements();

      // Verifica l'esistenza dei valori della colonna
      if (theRowEnum.hasMoreElements())
         {
         // Richiama il valore della colonna
         theColumn1 = (String)theRowEnum.nextElement();
         if (theColumn1.equals("CrossWorlds")==false)
            {
            return "Expected result = CrossWorlds,"
               + " Resulting result = " + theColumn1;
            }
         }
      }
   }

// Gestisce le eccezioni prodotte da executeSQL()
catch(CwDBSQLException e)
   {
   System.out.println(e.getMessage());
   }
```

Esempio: il seguente esempio mostra un frammento di codice per un'istruzione SELECT che restituisce più righe, ciascuna riga a due colonne, l'id cliente ed il nome azienda associato:

```
CwDBConnection connectn = null;
Vector theRow = null;
Enumeration theRowEnum = null;
Integer theColumn1 = 0;
```

```
String theColumn2 = null;

try
   {
   // Ottiene una connessione al database
   connectn = getDBConnection("sampleConnectionPoolName");
   }

catch(CwDBConnectionFactoryException e)
   {
   System.out.println(e.getMessage());
   throw e;
   }

// Frammento di codice per una serie di risultati a più righe, più colonne.
// Richiama tutte le righe con le colonne specificate, in cui
// viene soddisfatta la condizione specificata
try
   {
   connectn.executeSQL(
"select cust_id, company_name from customer where active_status = 1");

   // Loop attraverso ciascuna riga
   while(connectn.hasMoreRows())
      {
      // Ottiene una riga
      theRow = connectn.nextRow();

      // Ottiene i valori della colonna come oggetto Enumerazione
      theRowEnum = theRow.elements();
      int length = 0;
      if ((length = theRow.size()) != 2)
      {
         return "Expected result set size = 2," +
            " Actual result state size = " + length;
      }
      // Verifica l'esistenza dei valori della colonna
      if (theRowEnum.hasMoreElements())
         {
         // Richiama i valori della colonna
         theColumn1 =
            ((Integer)theRowEnum.nextElement()).intValue();
         theColumn2 = (String)theRowEnum.nextElement();
         }
      }
   }
catch(CwDBSQLException e)
   {
   System.out.println(e.getMessage());
   }
```

Nota: L'istruzione SELECT *non* modifica il contenuto del database. Quindi, di solito *non* è necessario eseguire la gestione delle trasazioni per le istruzioni SELECT.

Esecuzione di query statiche che modificano i dati: Le istruzioni SQL che modificano i dati di una tabella del database includono:

- INSERT, che aggiunge delle righe nuove ad una tabella del database.
- UPDATE, che modifica le righe esistenti di una tabella del database.
- DELETE, che rimuove le righe da una tabella del database.

Per inviare una di queste istruzioni come query statica al database per l'esecuzione, specificare una rappresentazione di stringa dell'istruzione come argomento per il metodo `executeSQL()`.

Esempio: la seguente chiamata a `executeSQL()` invia un INSERT di una riga nella tabella `abc` del database associato alla connessione corrente:

```
connection.executeSQL("insert into abc values (1, 3, 6)");
```

Nota: Nel codice precedente, la variabile `connection` è un oggetto `CwDBConnection` ottenuto da una precedente chiamata al metodo `getDBConnection()`.

Per un'istruzione UPDATE o INSERT, è possibile determinare il numero di righe nella tabella del database che sono state modificate o aggiunte con il metodo `getUpdateCount()`.

Importante: Poiché le istruzioni INSERT, UPDATE e DELETE modificano il contenuto del database, valutare la necessità della gestione delle transazioni per queste istruzioni. Per ulteriori informazioni, consultare "Gestione della transazione" a pagina 228.

Esecuzione di una procedura memorizzata statica: E' possibile utilizzare il metodo `executeSQL()` per eseguire una chiamata procedura memorizzata finché esistono *entrambe* le seguenti condizioni:

- La procedura memorizzata *non* utilizza i parametri OUT.

  Se la procedura memorizzata utilizza un parametro OUT, si *deve* utilizzare `executeStoredProcedure()` per eseguirla.
- la procedura memorizzata viene richiamata solo una volta.

  Il metodo `executeSQL()`*non* salva l'istruzione preparata per la chiamata procedura memorizzata. Quindi, se si richiama la stessa procedura più più di una volta (ad esempio in un loop), l'utilizzo di `executeSQL()` può essere più lento che richiamare un metodo che non salva l'istruzione preparata: `executePreparedSQL()` o `executeStoredProcedure()`.

Per ulteriori informazioni, consultare "Esecuzione delle procedure memorizzate" a pagina 223.

Esecuzione di query preparate

Il metodo `executePreparedSQL()` invia una query preparata al database per l'esecuzione. Una *query preparata* è un'istruzione SQL già precompilata in un formato eseguibile utilizzato dal database. La prima volta che `executePreparedSQL()` invia una query al database, lo fa in formato stringa. Il database riceve la query, la compila in un formato eseguibile analizzando la stringa, ed esegue l'istruzione SQL risultante (come fa per `executeSQL()`). Tuttavia, il database restituisce questo formato compilato di istruzione SQL a `executePreparedSQL()`, che lo memorizza. Questa istruzione SQL compilata viene denominata *istruzione preparata*.

Nelle esecuzioni successive della stessa query, `executePreparedSQL()` prima verifica se esiste già un'istruzione preparata per quella query. Se esiste, `executePreparedSQL()` la invia al database al posto della stringa di query. Le esecuzioni successive della query sono più efficienti perché il database non deve esaminare la stringa e creare l'istruzione preparata.

E' possibile inviare ad un database i seguenti tipi di query SQL con `executePreparedSQL()`:

- Query che restituiscono dati dal database (SELECT)
- Query che modificano i dati nel database (INSERT, UPDATE, DELETE)
- Query che eseguono procedure memorizzate definite nel database

Esecuzione di query preparate che restituiscono dati (SELECT): Se è necessario eseguire la stessa istruzione SELECT più volte, utilizzare `executePreparedSQL()` per creare una versione precompilata dell'istruzione. Per preparare un'istruzione SELECT tenere presente quanto segue:

- E' possibile utilizzare i parametri in questa istruzione SELECT per passare informazioni specifiche a ciascuna esecuzione dell'istruzione preparata. Per un esempio su come utilizzare i parametri con un'istruzione preparata, vedere la Figura 98.
- Quando si esegue un'istruzione SELECT con `executePreparedSQL()`, si utilizzano sempre gli stessi metodi per accedere ai dati restituiti (Tabella 70 e Tabella 71). Per ulteriori informazioni, consultare "Esecuzione di query statiche che restituiscono dati (SELECT)" a pagina 216.

Esecuzione di query preparate che modificano i dati: Se è necessario eseguire la stessa istruzione INSERT, UPDATE o DELETE più volte, utilizzare `executePreparedSQL()` per creare una versione precompilata dell'istruzione. L'istruzione SQL che viene rieseguita *non* deve essere esattamente la stessa ogni volta, per poter usufruire dell'istruzione preparata. E' possibile utilizzare dei parametri nell'istruzione SQL per fornire dinamicamente le informazioni ad ogni esecuzione dell'istruzione.

Il frammento di codice nella Figura 98 inserisce 50 righe nella tabella `impiegato`. La prima volta che `executePreparedSQL()` viene richiamato, invia la versione stringa dell'istruzione INSERT al database, che la esamina, la esegue e la restituisce nel suo formato eseguibile: un'istruzione preparata. Le successive 49 volte in cui viene eseguita l'istruzione INSERT (presupponendo che tutti gli INSERT abbiano esito positivo), `executePreparedSQL()` riconosce che esiste un'istruzione preparata e la invia al database per l'esecuzione.

```
CwDBConnection connection;
Vector argValues = new Vector();

argValues.setSize(2);

int emp_id = 1;
int emp_id = 2000;

for (int = 1; i < 50; i++)
   {
   argValues.set(0, new Integer(emp_id));
   argValues.set(1, new Integer(emp_num));

   try
      {
      // Invia l'istruzione INSERT al database
      connection.executePreparedSQL(
"insert into employee (employee_id, employee_number) values (?, ?)",
          argValues);

      // Aumenta i valori dell'argomento
      emp_id++;
      emp_num++
      }

   catch(CwDBSQLException e)
      {
      System.out.println(e.getMessage());
      }
   }
```

Figura 98. Trasmissione di valori argomento ad un'istruzione preparata

Suggerimenti: L'esecuzione della versione preparata dell'istruzione INSERT di solito migliora le prestazioni dell'applicazione, anche se non aumenta la memoria dell'applicazione.

Quando si riesegue un'istruzione SQL che modifica il database, è ancora necessario gestire le transazioni secondo il modello di programmazione delle transazioni. Per ulteriori informazioni, consultare "Gestione della transazione" a pagina 228.

Nota: Per semplificare il codice, nella Figura 98 *non* è inclusa la gestione della transazione.

Esecuzione di una procedura preparata memorizzata: E' possibile utilizzare il metodo `executePreparedSQL()` per eseguire una chiamata procedura memorizzata, finché esistono *entrambe* le seguenti condizioni:

- La procedura memorizzata *non* contiene i parametri OUT.

  Se la procedura memorizzata utilizza un parametro OUT, si *deve* utilizzare `executeStoredProcedure()` per eseguirla.
- La procedura memorizzata viene richiamata più di una volta.

  Il metodo `executePreparedSQL()` salva in memoria l'istruzione preparata per la chiamata procedura memorizzata. Quindi, se si richiama la procedura memorizzata solo una volta, l'utilizzo di `executePreparedSQL()` può usare più memoria che richiamare la procedura memorizzata con `executeSQL()`, che non salva l'istruzione preparata.

Per ulteriori informazioni, consultare "Esecuzione delle procedure memorizzate" a pagina 223.

Esecuzione delle procedure memorizzate

Una *procedura memorizzata* è una procedura definita dall'utente che contiene le istruzioni SQL e la logica delle condizioni. Le procedure memorizzate vengono memorizzate in un database insieme ai dati.

Nota: Quando si crea una nuova relazione, Relationship Designer Express crea una procedura memorizzata per conservare ogni tabella di relazioni.

La Tabella 72 mostra i metodi nella classe `CwDBConnection` che richiamano una procedura memorizzata.

Tabella 72. Metodi CwDBConnection per richiamare una procedura memorizzata

Come richiamare la procedura memorizzata	Metodo CwDBConnection	Utilizzo
Inviare al database un'istruzione CALL per eseguire la procedura memorizzata.	`executeSQL()`	Per richiamare una procedura memorizzata che *non* contiene parametri OUT e viene eseguita *solo una volta*
	`executePreparedSQL()`	Per richiamare una procedura memorizzata che *non* contiene parametri OUT e viene eseguita *più di una volta*
Specificare il nome della procedura memorizzata ed un vettore dei suoi parametri per creare una chiamata della procedura, che viene inviata al database per l'esecuzione.	`executeStoredProcedure()`	Per richiamare qualsiasi procedura memorizzata, incluso una con i parametri OUT

Nota: E' possibile utilizzare i metodi JDBC per eseguire direttamente una procedura memorizzata. Tuttavia, l'interfaccia fornito dalla classe `CwDBConnection` è più semplice e riutilizza le risorse del database, che possono aumentare l'efficienza dell'esecuzione. Per eseguire le procedure memorizzate utilizzare i metodi contenuti nella classe `CwDBConnection`.

Una procedura memorizzata può restituire i dati sotto forma di una o più righe. In questo caso, per accedere a queste righe restituite dai risultati della query, si utilizzano gli stessi metodi Java (come `hasMoreRows()` e `nextRow()`) che si utilizzano per i dati restituiti da un'istruzione SELECT. Per ulteriori informazioni, consultare "Esecuzione di query statiche che restituiscono dati (SELECT)" a pagina 216.

Come mostra la Tabella 72, la scelta del metodo da utilizzare per richiamare una procedura memorizzata dipende dai seguenti fattori:

- Se la procedura fornisce dei parametri OUT

  Un parametro OUT è un parametro mediante il quante la procedura memorizzata restituisce un valore al codice della chiamata. Se la procedura memorizzata utilizza un parametro OUT, si *deve* utilizzare `executeStoredProcedure()` per richiamarla.
- Il numero di volte in cui si richiama la procedura memorizzata

  Il metodo `executeStoredProcedure()` salva la versione compilata della procedura memorizzta. Quindi, se si richiama la stessa procedura memorizzata più di una volta, (ad esempio in un loop), l'utilizzo di `executeStoredProcedure()` può essere più rapido di `executeSQL()` poiché il database può riutilizzare la versione precompilata.

le sezioni che seguono descrivono l'utilizzo dei metodi executeSQL() e executeStoredProcedure() per richiamare una procedura memorizzata.

Richiamo di una procedura memorizzata senza parametri OUT: Per richiamare una procedura memorizzata che *non* contieneparametri OUT, è possibile utilizzare uno dei due metodi di CwDBConnection riportati di seguito:

- Il metodo executeSQL() invia una chiamata di procedura memorizzata statica al database.

  Questa chiamata di procedura viene inviata come stringa al database, che la compila in un'istruzione preparata, prima di eseguirla. L'istruzione preparata *non* viene salvata. Quindi, executeSQL() è utile per una procedura memorizzata che deve essere richiamata una sola volta.
- Il metodo executePreparedSQL() invia una chiamata di procedura memorizzata preparata al database.

  Al suo primo richiamo, la chiamata di procedura viene inviata al database, che crea l'istruzione preparata e la salva. Il database quindi rimanda indietro l'istruzione preparata a executePreparedSQL(), che la salva in memoria. Quindi, executePreparedSQL() è utile per una procedura memorizzata che deve essere chiamata more più di una volta (ad esempio in un loop).

Per chiamare una procedura memorizzata con uno di questi metodi specificare, come argomento del metodo, una rappresentazione di stringa dell'istruzione CALL che include la procedura memorizzata e qualsiasi eventuale argomento. Nella Figura 99, la chiamata a executeSQL() invia un'istruzione CALL per eseguire la procedura memorizzata setOrderCurrDate().

```
connection.executeSQL("call setOrderCurrDate(345698)");
```

Figura 99. Richiamo di una procedura memorizzata con executeSQL()

Nella Figura 99, la variabile connection è un oggetto CwDBConnection ottenuto da una precedente chiamata al metodo getDBConnection(). E' possibile utilizzare executeSQL() per eseguire la procedura memorizzata setOrderCurrDate() poiché il suo unico argomento è un parametro IN, cioè il valore viene inviato *solo* nella procedura memorizzata. Questa procedura memorizzata *non* contiene parametri OUT.

E' possibile utilizzare il formato executeSQL() o executePreparedSQL() che accetta la trasmissione di un vettore di parametri nei valori argomento della procedura memorizzata. Tuttavia, *non è possibile* utilizzare questi metodi per chiamare una procedura memorizzata che utilizza un parametro OUT. Per eseguire questo tipo di procedura memorizzata, si *deve* utilizzare executeStoredProcedure(). Per ulteriori informazioni, consultare "Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 224.

Nota: Utilizzare un blocco PL/SQL anonimo se si prevede di richiamare oggetti PL/SQL memorizzati Oracle tramite ODBC con il metodo CwDBConnection .executeSQL(). Quello che segue è un formato accettabile (il nome della procedura memorizzata è myproc):

```
connection.executeSQL("begin myproc(...);
 end;");
```

Richiamo di procedure memorizzate con executeStoredProcedure(): Il metodo executeStoredProcedure() può eseguire qualsiasi procedura memorizzata, incluso quella che utilizza i parametri OUT. Questo metodo salva l'istruzione preparata per la chiamata di procedura memorizzata , come per il metodo executePreparedSQL().

Quindi, executeStoredProcedure() può migliorare le prestazioni della chiamata di una procedura memorizzata che viene seguita più volte.

Passi per il richiamo di una procedura memorizzata con il metodo executeStoredProcedure(): Per richiamare una procedura memorizzata con il metodo executeStoredProcedure(), eseguire i seguenti passi:

1. Specificare come Stringa il nome della procedura memorizzata da eseguire.
2. Creareun vettore parametro Vettore di oggetti CwDBStoredProcedureParam, che fornisce le informazioni sui parametri: il tipo di parametro (in/out) ed il valore di ciascun parametro della procedura memorizzata.

Un *parametro* è un valore che si può inviare all'interno o all'esterno della procedura memorizzata (in/out). Il tipo di parametro in/out determina come la procedura memorizzata utilizza il valore del parametro:

- Un parametro IN è *solo per l'input*: la procedura memorizzata accetta il valore del parametro come input ma *non* utilizza il parametro per restituire un valore al codice della chiamata.
- Un parametro OUT è *solo per l'output*: la procedura memorizzata *non* interpreta il valore del parametro come input ma utilizza il parametro per restituire un valore al codice della chiamata.
- Un parametro INOUT è *sia per l'input che per l'output*: la procedura memorizzata accetta il valore del parametro come input ed utilizza il parametro per restituire un valore al codice della chiamata.

Una L'oggetto CwDBStoredProcedureParam descrive un singolo parametro di una procedura memorizzata. La Tabella 73 mostra le informazioni sul parametro che un oggetto CwDBStoredProcedureParam contiene, oltre ai metodi per richiamarle ed impostarle.

Tabella 73. Informazioni sul parametro in un oggetto CwDBStoredProcedureParam

Informazioni sul parametro	Metodo CwDBStoredProcedureParam
Valore del parametro	getValue()
Tipo di parametro in/out	getParamType()

Passi per la trasmissione dei parametri ad una procedura memorizzata con executeStoredProcedure(): Per passare i parametri ad una procedura memorizzata con executeStoredProcedure(), eseguire i seguenti passi:

1. Creare un oggetto CwDBStoredProcedureParam per conservare le informazioni sul parametro.

   Utilizzare il costruttore CwDBStoredProcedureParam() per creare un nuovo oggetto CwDBStoredProcedureParam. A questo costruttore, passare le seguenti informazioni sul parametro per inizializzare l'oggetto:

   - Il tipo di parametro in/out specifica se il parametro deve essere un parametro IN, INOUT o OUT.
   - Il valore del parametro è un tipo di dato Java che contiene il valore da assegnare al parametro. La classe CwDBStoredProcedureParam fornisce molte versioni del suo costruttore per supportare i diversi tipi di dati che possono essere associati al valore del parametro. Per un parametro OUT, questo valore può essere fittizio ma il tipo di dati deve corrispondere al tipo di dati del parametro OUT nella dichiarazione della procedura memorizzata.
2. Ripetere il passo 1 per ogni parametro della procedura memorizzata.

3. Creare un oggetto Vettore con elementi sufficienti a contenere tutti i parametri della procedura memorizzata.
4. Aggiungere l'oggetto inizializzato CwDBStoredProcedureParam all'oggetto Vettore del parametro.

   Utilizzare il metodo addElement() o add() della classe Vettore per aggiungere l'oggetto CwDBStoredProcedureParam.
5. Dopo aver creato tutti gli oggetti CwDBStoredProcedureParam e averli aggiunti al vettore Vettore del parametro, passare l'array come secondo argomento al metodo executeStoredProcedure().

   Il metodo executeStoredProcedure() invia la procedura memorizzata ed i suoi parametri al database per l'esecuzione.

Esempio: si supponga di avere la procedura memorizzata get_empno() definita in un database nel modo seguente:

```
create or replace procedure get_empno(emp_id IN number,
      emp_number OUT number) as
   begin
      select emp_no into emp_number
      from emp
      where emp_id = 1;
   end;
```

Questa procedura memorizzata get_empno() ha due parametri:

- Il primo, emp_id, è un parametro IN.

  Quindi è necessario inizializzare il suo oggetto associato CwDBStoredProcedureParam con il tipo in/out PARAM_IN, oltre che con il valore appropriato da inviare nella procedura memorizzata. Poiché emp_id è dichiarato come tipo SQL NUMBER (che contiene un valore di numero intero), il valore del parametro è di un Oggetto Java che contiene dei valori di numeri interi: Numero intero.
- il secondo parametro, emp_number, è un parametro OUT.

  Per questo parametro, creare un oggetto CwDBStoredProcedureParam *vuoto* da inviare nella procedura memorizzata. Questo oggetto si inizializza con il tipo in/out type PARAM_OUT. Tuttavia, si fornisce un valore Numero intero fittizio per questo parametro. Una volta terminata l'esecuzione della procedura memorizzata, è possibile ottenere il valore restituito da questo parametro OUT con il metodo getValue().

La Figura 100 esegue la procedura memorizzata get_empno() con il metodo executeStoredProcedure() per ottenere il numero di impiegato per l'id impiegato 65:

```
CwDBConnection connectn = null;

try
   {
   // Richiama la connessione al database
   connectn = getDBConnection("CustomerDBPool");

   // Crea il Vettore del parametro
   Vector paramData = new Vector(2);

   // Crea il parametro IN per l'id impiegato e lo aggiunge al vettore del
   // parametro
   paramData.add(
      new CwDBStoredProcedureParam(PARAM_IN, new Integer(65)));

   // Crea l'argomento fittizio per il parametro OUT e lo aggiunge al
   // vettore del parametro
   paramData.add(
      new CwDBStoredProcedureParam(PARAM_OUT, new Integer(0));

   // Richiama la procedura memorizzata get_empno()
   connectn.executeStoredProcedure("get_empno", paramData);

   // Richiama il risultato dal parametro OUT
   CwDBStoredProcedureParam outParam =
      (CwDBStoredProcedureParam) paramData.get(1);
   int emp_number = ((Integer) outParam.getValue().intValue();
   }
```

Figura 100. Esecuzione della procedura memorizzata get_empno()

Suggerimenti: L'oggetto Java `Vettore` è un'array basato su zero. nel codice precedente, per accedere al valore del parametro OUT dal vettore `Vettore` del parametro, la chiamata `get()` specifica un valore di indice 1 perché il vettore `Vettore` è basato su zero.

Una procedura memorizzata elabora i suoi parametri come tipi di dati SQL. Poiché i tipi di dati SQL e Java *non* sono identici, il metodo `executeStoredProcedure()` deve convertire un valore di parametro fra questi due tipi di dati. Per un parametro IN, `executeStoredProcedure()` converte il valore del parametro da un tipo di dati Java al suo tipo di dati SQL. Per un parametro OUT, `executeStoredProcedure()` converte il valore del parametro dal suo tipo di dati SQL ad un tipo di dati Java.

Il metodo `executeStoredProcedure()` utilizza il tipo di dati JDBC internamente per contenere il valore del parametro inviato alla (e dalla) procedura memorizzata. JDBC definisce una serie di identificativi generici di tipi SQL nella classe `java.sql.Types`. Questi tipi rappresentano i tipi SQL più comunemente usati. JDBC fornisce anche una mappatura standard dai tipi JDBC ai tipi di dati Java.

Esempio: un JDBC INTEGER di solito viene mappato ad un tipo Java `int`. Il metodo `executeStoredProcedure()` utilizza le mappature mostrate nella Tabella 74.

Tabella 74. Mappature fra i tipi di dati Java e JDBC

Tipo di dati Java	Tipo di dati JDBC
`String`	CHAR, VARCHAR o LONGVARCHAR
`Integer, int`	INTEGER
`Long`	BIGINT
`Float, float`	REAL
`Double, double`	DOUBLE

Tabella 74. Mappature fra i tipi di dati Java e JDBC (Continua)

Tipo di dati Java	Tipo di dati JDBC
`java.math.BigDecimal`	NUMERIC
`Boolean, boolean`	BIT
`java.sql.Date`	DATE
`java.sql.Time`	TIME
`java.sql.Timestamp`	TIMESTAMP
`java.sql.Clob`	CLOB
`java.sql.Blob`	BLOB
`byte[]`	BINARY, VARBINARY o LONGVARBINARY
`Array`	ARRAY
`Struct`	STRUCT

Gestione della transazione

Una *transazione* è una serie di operazioni che vengono eseguite come un'unica unità. Tutte le istruzioni SQL che vengono eseguite in una transazione riescono o non riescono come un'unica unità. Questa sezione fornisce le seguenti informazioni sulla gestione delle transazioni:

- "Individuazione del modello di programmazione della transazione"
- "Specifica dell'ambito della transazione" a pagina 229

Individuazione del modello di programmazione della transazione

Il raggruppamento dei passi di esecuzione delle operazioni del database in transazioni viene denominato *bracketing della transazione*. Ad ogni connessione è associato uno dei seguenti modelli di programmazione della transazione:

- Bracketing della transazione implicita—le operazioni del database fanno parte di una *transazione implicita*, che inizia non appena viene acquisita la connessione e termina quando la connessione viene rilasciata; il bracketing della transazione viene gestito implicitamente da InterChange Server Express.
- Bracketing della transazione esplicita—le operazioni del database fanno parte di una *transazione esplicita*, il cui inizio e la cui fine vengono determinati con la programmazione.

Al runtime, un'istanza di mappa stabilisce quale modello di programmazione della transazione utilizzare per ciascuna connessione acquisita. Per impostazione predefinita, una mappa presuppone che *tutte* le connessioni che acquisisce utilizzano il bracketing della transazione implicita come modello di programmazione della transazione. E' possibile sostituire il modello predefinito di programmazione della transazione in uno dei modi elencati nella Tabella 75.

Tabella 75. Sostituzione di un modello di programmazione della transazione di una connessione

Modello di programmazione della transazione da sostituire	Azione da intraprendere
Per specificare un modello di programmazione transazione *per tutte le connessioni* ottenute da una particolare istanza di mappa	Selezionare o deselezionare la casella di controllo Transazione database implicita nella scheda Generale della finestra di dialogo Proprietà mappa. E' anche possibile impostare questa proprietà nella finestra Proprietà mappa di Service Manager.

Tabella 75. Sostituzione di un modello di programmazione della transazione di una connessione (Continua)

Modello di programmazione della transazione da sostituire	Azione da intraprendere
Per specificare un modello di programmazione transazione *per una particolare connessione*	Fornire un valore booleano per indicare il modello di programmazione transazione desiderato (solo per questa connessione) come secondo argomento facoltativo per il metodo `getDBConnection()`. La seguente chiamata `getDBConnection()` specifica il bracketing della transazione esplicita per la connessione ottenuta dal pool di connessione `ConnPool`: `conn = getDBConnection("ConnPool", false);`

E' possibile determinare il modello corrente di programmazione della transazione che le connessioni utilizzeranno con il metodo `BaseDLM.implicitDBTransactionBracketing()`, che restituisce un valore `booleano` che indica se il modello di programmazione della transazione è un bracketing di transazione implicita.

Specifica dell'ambito della transazione

Il modello di programmazione transazione della connessione determina come viene specificato l'ambito della transazione del database. Quindi, questa sezione fornisce le seguenti informazioni:

- "Ambito della transazione con il bracketing della transazione implicita"
- "Ambito della transazione con il bracketing della transazione esplicita"

Ambito della transazione con il bracketing della transazione implicita: InterChange Server Express gestisce le azioni della mappa in una singola transazione implicita. Se la connessione utilizza il bracketing della transazione implicita, InterChange Server Express si occupa anche della gestione della transazione per le operazioni eseguite su un database esterno, uno associato ad una connessione proveniente da un pool di connessioni. Quando una mappa esegue delle operazioni di database, InterChange Server Express gestisce anche quelle come transazione implicita, che è una transazione secondaria della transazione principale (la mappa). La transazione secondaria del database inizia appena la mappa ottiene la connessione. InterChange Server Express termina implicitamente la transazione secondaria quando l'esecuzione della mappa è completata.

La riuscita o meno di questa transazione secondaria del database dipende dall'esito positivo o negativo della mappa:

- Se la mappa ha esito positivo, InterChange Server Express esegue il commit della transazione secondaria del database.
- Se la mappa ha esito negativo, InterChange Server Express esegue il rollback della transazione secondaria del database. Se il rollback non riesce, InterChange Server Express produce l'eccezione `CwDBTransactionException` e registra un errore.

Ambito della transazione con il bracketing della transazione esplicita: Se la connessione utilizza il bracketing della transazione esplicita, InterChange Server Express si aspetta che la definizione della mappa specifichi esplicitamente l'ambito

di ciascuna transazione del database. Il bracketing della transazione esplicita è utile se si deve eseguire del lavoro di database che è indipendente dall'esito della mappa.

Esempio: se si deve eseguire il controllo per indicare che determinate tabelle sono state accedute, quel controllo deve essere eseguito a prescindere dall'esito degli accessi alle tabelle. Se si includono le operazioni di controllo del database in una transazione esplicita, vengono eseguite a prescindere dall'esito della mappa.

La Tabella 76 mostra i metodi contenuti nella classe `CwDBConnection` che fornisce la gestione dei limiti di transazione per le transazioni esplicite.

Tabella 76. Metodi CwDBConnection per la gestione della transazione esplicita

Attività di gestione transazione	Metodo CwDBConnection
Iniziare una nuova transazione.	`beginTransaction()`
Terminare la transazione, eseguendo il commit (salvataggio) di tutte le modifiche apportate al database durante la transazione.	`commit()`
Stabilire se una transazione al momento è attiva.	`inTransaction()`
Terminare la transazione, eseguendo il rollback (regressione) di tutte le modifiche apportate durante la transazione.	`rollBack()`

Passi per la specifica dell'ambito di transazione di una transazione esplicita: Per specificare l'ambito di transazione di una transazione esplicita, eseguire i seguenti passi:

1. Contrassegnare l'inizio della transazione con una chiamata al metodo `beginTransaction()`.
2. Eseguire tutte le istruzioni SQL, che devono riuscire o non riuscire come un'unica unità, *fra* la chiamata a `beginTransaction()` ed il termine della transazione.
3. Terminare la transazione in uno dei due modi riportati di seguito:
   - Richiamare `commit()` per terminare la transazione con esito positivo. Tutte le modifiche effettuate dalle istruzioni SQL vengono *salvate* nel database.
   - Richiamare `rollBack()` per terminare la transazione con esito negativo. Tutte le modifiche effettuate dalle istruzioni SQL vengono *ritirate* dal database.

   E' possibile scegliere quali condizioni causano il fallimento della transazione. Testare la condizione e richiamare `rollBack()` se viene rilevata una condizione di fallimento. Altrimenti, richiamare `commit()` per terminare la transazione con esito positivo.

Importante: Se *non* si utilizza `beginTransaction()` per specificare l'inizio della transazione esplicita, il database esegue ogni istruzione SQL come transazione separata. Se si include `beginTransaction()` ma *non* si specifica il termine della transazione database con `commit()` o `rollback()` prima che venga rilasciata la connessione, InterChange Server Express termmina in modo implicito la transazione in base all'esito positivo della mappa. Se la mappa ha esito positivo, InterChange Server Express esegue il commit della transazione di database. Se la mappa *non* ha esito positivo, InterChange Server Express esegue in modo implicito il rollback della transazione del database. A prescindere dalla riuscita della mappa, InterChange Server Express registra un'avvertenza.

Il seguente frammento di codice aggiorna tre tabelle nel database associato alle connessioni in `CustDBConnPool`. Se *tutti* gli aggiornamenti hanno esito positivo, il frammento di codice esegue il commit delle modifiche con il metodo `commit()`. Se si verificano degli errori di transazione, viene restituita un'eccezione `CwDBTransactionException` ed il frammento di codice richiama il metodo `rollback()`.

```
CwDBConnection connection =
getDBConnection("CustDBConnPool", false);

// Inizia una transazione
connection.beginTransaction();

// Aggiorna alcune tabelle
try
   {
   connection.executeSQL("update table1....");
   connection.executeSQL("update table2....");
   connection.executeSQL("update table3....");

   // Esegue il commit della transazione
   connection.commit();
   }

catch (CwDBSQLException e)
   {
   // Esegue il rollback della transazione se la chiamata executeSQL() produce
   // un'eccezione
   connection.rollback();
   }

// Rilascia la connessione al database
connection.release();
```

Per stabilire se una transazione al momento è attiva, utilizzare il metodo `inTransaction()`.

Attenzione: Utilizzare i metodi `beginTransaction()`, `commit()` e `rollback()` *solo* se la connessione utilizza il bracketing della transazione esplicita. Se la connessione utilizza il bracketing della transazione implicita, l'uso di uno di questi metodi comporta un'eccezione `CwDBTransactionException`.

Rilascio di una connessione

Una volta rilasciata, una connessione viene riportata al relativo pool di connessioni, dove è disponibile per essere utilizzata da altri componenti. Il modo in cui viene rilasciata una connessione al database dipende dal modello di programmazione della transazione. Questa sezione fornisce le seguenti informazioni:

- "Rilascio di una connessione con bracketing della transazione implicita"
- "Rilascio di una connessione con bracketing della transazione esplicita" a pagina 232

Rilascio di una connessione con bracketing della transazione implicita

InterChange Server Express rilascia automaticamente una connessione che utilizza il bracketing della transazione implicita quando ha terminato la transazione di database. InterChange Server Express non termina la transazione di database finché non stabilisce l'esito positivo o negativo della mappa; vale a dire, InterChange Server Express rilascia le connessioni quando l'istanza mappa finisce l'esecuzione. Se la mappa si esegue con esito positivo, InterChange Server Express

esegue automaticamente il commit di tutte le transazioni database ancora attive. Se l'esecuzione della mappa ha esito negativo (ad esempio se viene prodotta un'eccezione che non viene gestita con un'istruzione `catch`), InterChange Server Express esegue automaticamente il rollback di tutte le transazioni ancora attive.

Rilascio di una connessione con bracketing della transazione esplicita

Una connessione che utilizza il bracketing della transazione esplicita termina in uno dei seguenti casi:

- InterChange Server Express rilascia automaticamente una connessione che utilizza il bracketing della transazione esplicita.
- E' possibile rilasciare esplicitamente una connessione con il metodo `release()` della classe `CwDBConnection`.

E' possibile utilizzare il metodo `CwDBConnection.isActive()` per determinare se una connessione è stata rilasciata. Se una connessione è stata rilasciata, `isActive()` restituisce `false`, come mostra il seguente frammento di codice:

```
if (connection.isActive())
   connection.release();
```

Attenzione: *Non* utilizzare il metodo `release()` se al momento è attiva una transazione. Con il bracketing della transazione implicita, InterChange Server Express non termina la transazione di database finché non rileva l'esito della mappa. Pertanto, l'utilizzo di questo metodo per una connessione che utilizza il bracketing della transazione implicita genera un'eccezione `CwDBTransactionException`. Se non si gestisce questa eccezione in modo esplicito, comporta un rollback automatico della transazione attiva. E' possibile utilizzare il metodo `inTransaction()` per stabilire se una transazione è attiva. InterChange Server Express rilascia automaticamente una connessione a prescindere dal modello di programmazione transazione utilizzato. Nella maggior parte dei casi non è necessario rilasciare esplicitamente la connessione.

Ristabilire una connessione

Le connessioni al database possono essere interrotte per diversi motivi, come un problema di rete o la chiusura del database. InterChange Server Express può rilevare le connessioni interrotte in entrambi i database, interno e dell'utente.

Se una connessione viene interrotta, InterChange Server Express tenta automaticamente di ristabilire la connessione in base ad un numero predefinito di tentativi ad intervalli di tempo predefiniti. Consultare il manuale *Guida alla gestione del sistema* per ulteriori informazioni sull'impostazione di numero max di tentativi e intervalli di tempo fra i tentativi.

Nota: Il valore predefinito per il numero max di tentativi è 3 e per intervalli di tempo fra i tentativi è di 60 secondi.

Parte 2. Relazioni

Capitolo 6. Introduzione alle relazioni

Questo capitolo contiene una panoramica delle relazioni di WebSphere Business Integration Server Express, oltre al processo di sviluppo della relazione.

In questo capitolo sono illustrati i seguenti argomenti:

Cos'è una relazione?

Quando gli attributi in un'origine e un oggetto business di destinazione contengono dati equivalenti rappresentati in modo diverso, la fase di trasformazione richiede *relazione*. Una relazione stabilisce un'associazione tra i dati di due o più oggetti business. Ciascun oggetto business viene denominato *partecipante* nella relazione.

Di seguito sono riportati i dati che in genere vengono trasformati utilizzando le relazioni:

- Numeri di ID, come ad esempio un ID cliente o un ID del prodotto
- Altri valori rappresentati come codici, come ad esempio il paese, la valuta o lo stato civile.

Esempio: si supponga che l'applicazione A utilizzi numeri interi sequenziali per gli ID cliente e l'applicazione B utilizzi codici cliente generati. `TashiCo` dispone di un ID cliente `806` nell'applicazione A e `A100` nell'applicazione B. Per trasferire i dati dell'ID cliente tra le applicazioni A e B, è possibile creare una relazione tra l'oggetto business del client dell'applicazione A, l'oggetto business del cliente generico e l'oggetto business del cliente dell'applicazione B in base agli attributi dell'ID cliente.

Questa relazione stabilisce un'associazione tra i clienti dell'applicazione A e B, in base agli attributi chiave degli oggetti Business dei relativi clienti. In Figura 101, ciascuna casella rappresenta un partecipante ad una relazione denominata `CustIden`.

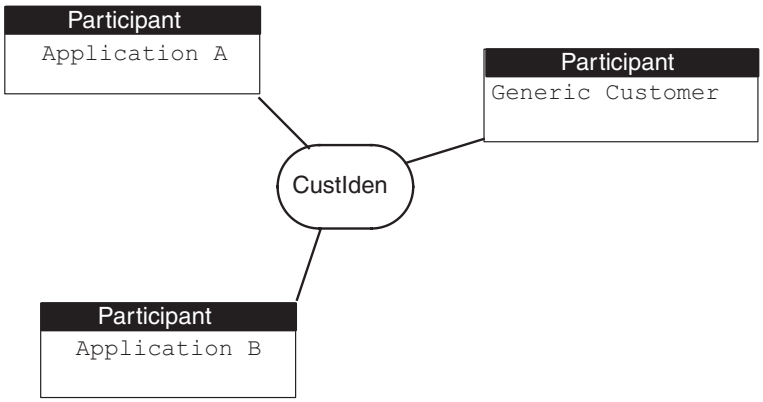


Figura 101. Relazione con tre partecipanti

Le relazioni sono classificate nelle seguenti categorie in base al tipo di dati del partecipante e al numero di istanze di ciascun partecipante che può essere correlato:

- Una *relazione di ricerca* stabilisce un'associazione tra i dati, come ad esempio gli attributi negli oggetti business. I dati possono essere correlati su base *uno a uno*, *uno-più* o *più-più*. In genere, le relazioni di ricerca trasformano gli attributi non chiave i cui valori sono rappresentati con i codici, come ad esempio lo stato civile o il codice della valuta. Utilizzare una relazione di ricerca se questi valori dell'attributo sono statici, ovvero i nuovi valori non sono spesso aggiunti o i valori esistenti non sono spesso rimossi.
- Una *relazione di identità* stabilisce un'associazione tra gli oggetti business o altri dati su base *uno a uno*. Per ciascuna istanza di relazione, può essere presente una sola istanza di ciascun partecipante. In genere, le relazioni di identità trasformano gli attributi chiave degli oggetti business, come ad esempio i numeri di ID e i codici del prodotto. La relazione in Figura 101 è un esempio di una relazione di identità. Utilizzare una relazione di identità se i valori di chiave sono dinamici, ovvero i valori di chiave vengono aggiunti frequentemente o se i valori esistenti vengono frequentemente rimossi.
- Una *relazione di non identità* stabilisce un'associazione tra gli oggetti business o altri dati su base *uno-più* o *più-a-più*. Per ciascuna istanza di relazione, può essere presente una o più istanze di ciascun partecipante. Un esempio di una relazione di non identità è una trasformazione da RMA-a-Ordine, in cui un oggetto business RMA singolo (Return Materials Authorization) può fornire uno o più oggetti business Ordine.

Relazioni di ricerca

Una *relazione di ricerca* correla due parti di dati non chiave. Ad esempio in una mappa da `Clarify_Site` a `Cliente`, è possibile trasformare gli attributi i cui valori sono rappresentati da codici o abbreviazioni, come ad esempio `SiteStatus`, utilizzando una relazione di ricerca. In una relazione di ricerca, è presente un partecipante per ciascun oggetto business specifico dell'applicazione.

La relazione `CustLkUp` in Figura 102 stabilisce una relazione di ricerca tra i codici di stato del cliente dalle applicazioni Clarify e SAP. Ciascuna casella rappresenta un partecipante nella relazione di ricerca `CustLkUp`. Si noti che questa relazione dispone di due partecipanti, uno per ciascun oggetto business specifico dell'applicazione.

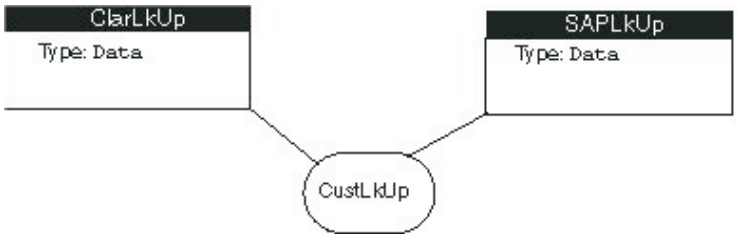


Figura 102. Definizione della relazione di ricerca CustLkUp

Nota: Poiché una relazione di ricerca non indica gli attributi in correlazione, i relativi partecipanti utilizzano un tipo speciale denominato `Dati`. Per ulteriori informazioni, consultare "Tipo di partecipante" a pagina 246.

Esempio: si supponga che l'applicazione Clarify rappresenti un cliente non attivo con uno stato di posizione `Inattivo` mente in SAP il valore corrispondente è `05`. Sebbene questi codici di stato del cliente siano diversi, rappresentano lo stesso stato, come viene illustrato in Figura 103.

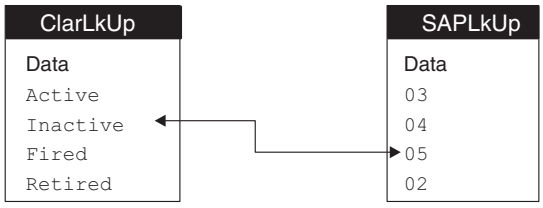


Figura 103. Dati di relazione per la relazione di ricerca CustLkUp

Tabella 77 illustra le attività secondarie per la creazione di una relazione di ricerca.

Tabella 77. Attività secondarie per la creazione di una relazione di ricerca

Attività secondaria		Per ulteriori informazioni
1.	Definizione di una relazione di ricerca in Relationship Designer Express.	"Definizione delle relazioni di ricerca" a pagina 259
2.	Personalizzazione del codice di mappatura per conservare la relazione di ricerca.	"Utilizzo delle relazioni di ricerca" a pagina 273
3.	Test della relazione di ricerca per verificare che sia implementata correttamente.	"Test di una relazione di ricerca" a pagina 104

Relazioni di identità

Una *relazione di identità* stabilisce un'associazione tra gli oggetti business o altri dati su base *uno-a-uno*. Per conservare una relazione uno-a-uno, ciascun oggetto business deve disporre di una chiave, ovvero l'oggetto contiene almeno un attributo (un *attributo chiave*) il cui valore identifica in modo univoco l'oggetto. Se entrambi gli oggetti business contengono una chiave, possono partecipare ad una relazione di identità.

Il sistema WebSphere Business Integration Server Express supporta i seguenti tipi di relazioni di identità:

- "Relazioni di identità semplici" a pagina 238
- "Relazioni di identità composite" a pagina 239

Entrambi i tipi di relazioni di identità coinvolgono gli attributi dell'oggetto business correlato. Pertanto, ciascun partecipante di una relazione di identità dispone di un oggetto business come relativo tipo di partecipante. Per ulteriori informazioni sui tipi di partecipante, consultare "Tipo di partecipante" a pagina 246.

Relazioni di identità semplici

Una *relazione di identità semplice* correla due oggetti business mediante un unico attributo chiave, ovvero ciascun oggetto business contiene un singolo valore che identifica in modo univoco l'oggetto.

Esempio: si supponga che la relazione `CustIden` (fare riferimento a Figura 101) viene raffinata ulteriormente per stabilire un'associazione tra i clienti delle applicazioni Clarify e SAP, in base agli attributi chiave degli oggetti business dei relativi clienti. In Figura 104, ciascuna casella rappresenta un partecipante della relazione di identità del cliente. Si noti che questa relazione dispone di un partecipante per ciascun oggetto business specifico dell'applicazione e oggetto business generico.

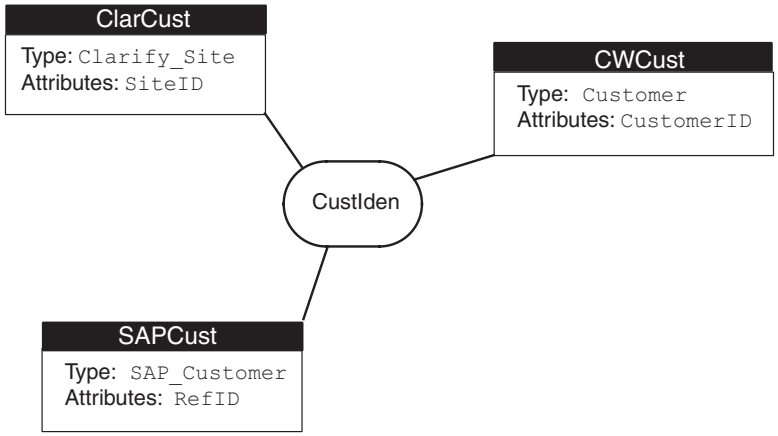


Figura 104. Definizione della relazione di identità semplice CustIden

L'azienda `TashiCo` è identificata con il valore di chiave `A100` nell'applicazione Clarify mentre questa stessa azienda è identificata con un valore di chiave `806` nell'applicazione SAP. Sebbene questi ID dell'applicazione siano diversi, rappresentano lo stesso cliente, come illustrato in Figura 105.

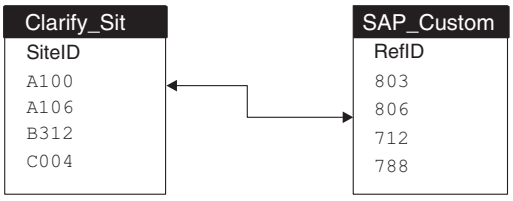


Figura 105. Dati di relazione per la relazione di identità semplice custIden

Pertanto, le seguenti mappe utilizzano una relazione di identità semplice per conservare le trasformazioni tra gli attributi:

- Le mappe in entrata (tra gli oggetti business specifici dell'applicazione Clarify e l'oggetto business `Cliente` generico) utilizzano una relazione di identità semplice

per conservare la trasformazione tra l'attributo SiteID dell'oggetto business Clarify_Site e l'attributo CustomerID generico dell'oggetto business Cliente generico.

- Anche le mappe in uscita (tra l'oggetto business Cliente generico e l'oggetto business specifico dell'applicazione SAP) utilizzano una relazione di identità semplice per conservare la trasformazione tra l'attributo RefID dell'oggetto business SAP_Customer e l'attributo CustomerID generico dell'oggetto Customer generico.

Tabella 78 illustra le attività secondarie per la creazione di una relazione di identità semplice.

Tabella 78. Attività secondarie per la creazione di una relazione di identità semplice

Attività secondaria		Per ulteriori informazioni
1.	Definizione di una relazione di identità semplice in Relationship Designer Express.	"Definizione delle relazioni di identità" a pagina 257
2.	Personalizzazione del codice di mappatura per conservare la relazione di identità semplice.	"Utilizzo di relazioni di identità semplici" a pagina 277
3.	Test della relazione di identità semplice per verificare che sia implementata correttamente.	"Test di una relazione d'identità" a pagina 101

Relazioni di identità composite

Una *relazione di identità composita* correla due oggetti business mediante una chiave composita. Come indicato dal termine "composito", una chiave composita è costituita da vari attributi. I valori di *tutti* gli attributi sono necessari per identificare in modo univoco un oggetto. Una chiave composita è costituita da una chiave univoca di un oggetto business parent e una chiave non univoca di un oggetto business child.

Esempio: si supponga che un ordine particolare di TashiCo nell'applicazione Clarify sia identificato con il valore di chiave 8765. Questo stesso ordine nell'applicazione SAP è identificato con il valore di chiave 0003411. Poiché questi due numeri di origine identificano in modo univoco lo stesso ordine, i relativi attributi chiave sono correlati mediante una relazione di identità semplice. Tuttavia, un ordine contiene anche delle righe. Se tutte le applicazioni partecipanti identificano tali righe dell'ordine con un valore univoco, una relazione di identità semplice può conservare le relative trasformazioni.

Tuttavia, spesso si verifica il caso in cui questa applicazione utilizza solo il numero di riga per identificare una voce della riga dell'ordine. Ovvero, ciascun ordine contiene una voce di riga identificata con 1, con successive voci con i numeri 2, 3 e così via. Tali numeri di righe *non* identificano in modo univoco le voci della riga dell'ordine. Per identificare in modo univoco tali voci, l'applicazione utilizza una chiave composta che è costituita dal numero di origine (dall'oggetto business dell'ordine parent) e il numero di riga (dall'oggetto business della riga dell'ordine).

In Figura 106, la relazione OrdrLine stabilisce una relazione tra le righe dell'ordine delle applicazioni Clarify e SAP, in base ai relativi attributi chiave compositi: l'attributo chiave univoco del relativo oggetto business dell'ordine parent combinato con il numero di riga dell'ordine nell'oggetto business della riga dell'ordine child. Ciascuna casella rappresenta un partecipante nella relazione di identità composita OrdrLine. Si noti che ciascun partecipante dispone di due attributi.

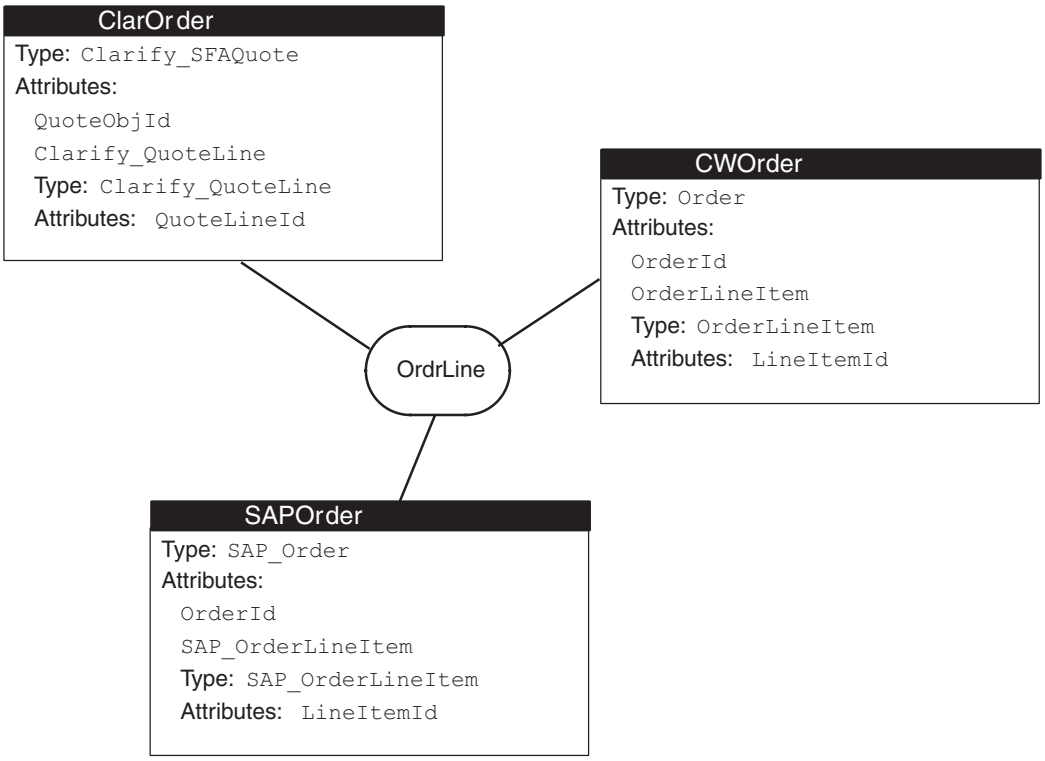


Figura 106. Definizione della relazione di identità composita OrdrLine

Esempio: si supponga che l'applicazione Clarify (rappresentata dal partecipante `ClarOrder` in Figura 106) utilizzi numeri interi sequenziali per identificare le voci della riga dell'ordine, mentre l'applicazione SAP utilizza il numero di riga per identificare tali voci. L'applicazione Clarify identifica in modo univoco ciascuna voce di riga dell'ordine. Pertanto le mappe tra l'oggetto business specifico dell'applicazione Clarify e l'oggetto business `Ordine` generico (rappresentato dal partecipante `CWOrder`) possono utilizzare una relazione di identità semplice per conservare la trasformazione delle voci della riga dell'ordine.

Tuttavia, l'applicazione SAP (rappresentata dal partecipante `SAPOrder`) identifica le voci della riga dell'ordine con il relativo numero di riga. Le relative voci non sono identificate in modo univoco: ciascun ordine contiene un numero di riga identificato con 1, con eventuali voci successive numerate 2, 3 e così via. Per identificare in modo univoco la terza voce della riga dell'ordine 0003411, è necessario utilizzare una chiave composita, che comprende sia il numero di ordine (0003411) sia il numero di voce (3). Pertanto, le mappe tra l'oggetto business specifico dell'applicazione SAP e l'oggetto business `Ordine` generico devono utilizzare una relazione di identità composita per conservare la trasformazione delle voci della riga dell'ordine.

La voce della terza riga dell'ordine `TashiCo` (`8765`) è identificata nell'applicazione Clarify con il valore di chiave semplice `1171`. Tuttavia, questa stessa voce della riga è identificata nell'applicazione SAP con il valore della chiave composita `0003411` (numero ordine) e `3` (numero riga). Sebbene tali righe dell'ordine siano identificate in modo diverso, rappresentano la stessa voce della riga dell'ordine, come illustrato in Figura 107.

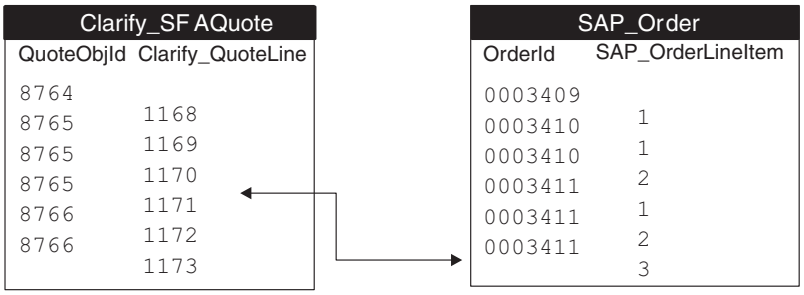


Figura 107. Dati di relazione per la relazione di identità composita OrdrLine

Tabella 79 illustra le attività secondarie per la creazione di una relazione di identità composita.

Tabella 79. Attività secondarie per la creazione di una relazione di identità composita

Attività secondaria		Per ulteriori informazioni
1.	Definizione di una relazione di identità composita in Relationship Designer Express.	"Definizione delle relazioni di identità" a pagina 257
2.	Personalizzazione del codice di mappatura per conservare la relazione di identità composita.	"Utilizzo delle relazioni di identità composite" a pagina 290
3.	Test della relazione di identità composita per verificare che sia implementata correttamente.	"Test di una relazione d'identità" a pagina 101

Relazioni: una visualizzazione più dettagliata

Per comprendere i tipi di relazioni supportate dal sistema WebSphere Business Integration Server Express, è necessario conoscere il modo in cui la IBM implementa i concetti di seguito riportati:

- "Relazioni"
- "Partecipanti" a pagina 245

Relazioni

Come illustrato in Tabella 80, una relazione è un'entità costituita da due parti, che comprende un'entità di archivio e un oggetto di runtime.

Tabella 80. Parti di una relazione

Entità di archivio	Oggetto di runtime
Definizione della relazione	Istanza della relazione

Definizione della relazione

Una relazione viene definita nel sistema WebSphere Business Integration Server Express con una *definizione della relazione*. Le definizioni della relazione identificano ciascun partecipante e specifica il modo in cui sono correlati tali partecipanti. In Figura 101, `CustIden` è la definizione della relazione e comprende le informazioni sui tre partecipanti, ovvero Applicazione A, Applicazione B e Cliente generico.

Il sistema memorizza le definizioni della relazione nell'archivio. Lo strumento Relationship Designer Express fornisce i dialoghi per consentire la creazione delle definizioni delle relazioni. Utilizzando questo strumento, la definizione della relazione completata viene memorizzata nel repository.

Suggerimenti: Per ulteriori informazioni sull'utilizzo di Relationship Designer Express per creare definizioni delle relazioni, consultare "Personalizzazione della finestra principale" a pagina 252.

La definizione della relazione fornisce le seguenti informazioni sulla relazione:

- Il nome della relazione
- Il nome del database della relazione

Nome della definizione della relazione: Una definizione della relazione è semplicemente un modello o una descrizione della relazione, quindi *non* è un oggetto business corrente. Pertanto, il nome della definizione della relazione *non* deve essere uguale al nome dell'oggetto business associato.

Database della relazione: Il *database della relazione* conserva le tabelle di relazione di una relazione. La relazione utilizza tali tabelle di relazione per tenere traccia dei valori correlati specifici dell'applicazione. Per ulteriori informazioni, consultare "Tabelle di relazioni" a pagina 243.

Per accedere al database della relazione al runtime, il sistema deve disporre delle seguenti informazioni:

- Il tipo di DBMS (Database management system) che gestisce il database della relazione
- Il nome e la password dell'account utente che accede al database della relazione
- La posizione del database della relazione

Per impostazione predefinita, il database della relazione è il repository del sistema WebSphere Business Integration Server Express, ovvero Relationship Designer Express crea tutte le tabelle delle relazioni nel repository. Relationship Designer Express consente di specificare la posizione delle tabelle della relazione in uno dei modi di seguito riportati:

- Modificare la posizione predefinita dei database della relazione di *ogni* relazione.

  Per ulteriori informazioni, consultare "Impostazioni predefinite globali" a pagina 265.
- Personalizzare la posizione di ciascuna tabella di relazioni come parte del processo di creazione della definizione di una relazione.

  Per ulteriori informazioni, consultare "Impostazioni avanzate per le definizioni di relazioni" a pagina 262.

Istanza della relazione

La definizione della relazione è un modello per l'esecuzione dell'istanza del runtime della relazione, denominata *istanza della relazione*. Durante l'esecuzione della mappa, il sistema crea delle istanze della relazione in base alla definizione della relazione e utilizzando i valori degli oggetti business correnti che stanno per essere trasformati.

Esempio: i dati della relazione per la relazione di ricerca `CustLkUp` (consultare Figura 103) illustra che lo stato di un cliente `Inattivo` in un'applicazione Clarify è uguale allo stato del cliente `05` in un'applicazione SAP. Sebbene, tali codici di stato siano diversi, rappresentano lo stesso stato del cliente e pertanto si trovano nella stessa istanza della relazione, come illustrato in Figura 108.

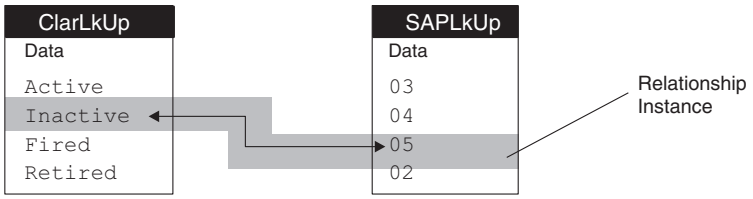


Figura 108. Un'istanza della relazione per la relazione CustLkUp

Una istanza della relazione è rappresentata nella mappatura dell'API da un'istanza della classe `Relationship` o `IdentityRelationship`.

Per trovare un'istanza della relazione, il sistema richiede le informazioni di seguito riportate:

- Una *tabella di relazioni* per identificare la tabella contenente le istanze della relazione per un determinato partecipante
- Un *ID dell'istanza della relazione* per identificare l'istanza della relazione corrente all'interno della tabella di relazione

Tabelle di relazioni: Una *tabella di relazioni* è una tabella di database che contiene i dati di runtime della relazione per un partecipante della relazione. InterChange Server Express memorizza le istanze della relazione nelle tabelle di relazioni, con una tabella (talvolta denominata *tabella del partecipante*) che memorizza informazioni per un partecipante della relazione. Ad esempio, per la relazione di ricerca `CustLkUp` in Figura 102, InterChange Server Express richiede due tabelle di partecipanti, come illustrato in Figura 108.

Quando si crea un definizione di relazione, Relationship Designer Express crea automaticamente gli schemi della tabella richiesti dalla relazione, ovvero crea le tabelle di relazioni con le colonne necessarie per ciascun partecipante. Al runtime, tali tabelle conservano i dati per le istanze della relazione.

Nota: Per una relazione di identità, InterChange Server Express popola automaticamente le tabelle di relazioni. Per una relazione di ricerca, è necessario popolare le tabelle di relazioni con i dati appropriati. Per ulteriori informazioni, consultare "Popolazione delle tabelle di ricerca con dati" a pagina 274.

Per accedere a una tabella di relazioni al runtime, il sistema deve disporre delle seguenti informazioni:

- Il nome della tabella di relazioni

  Poiché la tabella di relazioni è associata al partecipante, il nome di tale tabella è definito come parte della definizione del partecipante. Per impostazione predefinita, qualunque tabella di relazione dispone di un nome nel seguente formato:

  `RelationshipDefName_ParticipantDefName`

  Relationship Designer Express consente di personalizzare il nome di una tabella di relazioni come parte del processo di creazione di una definizione del partecipante.

  Per ulteriori informazioni, consultare "Impostazioni avanzate per le definizioni del partecipante" a pagina 263.
- Il nome del database che contiene la tabella di relazioni

Il nome del database di relazioni è impostato come parte della definizione della relazione. Per impostazione predefinita, il database della relazione è il repository del sistema. Per ulteriori informazioni, consultare “Impostazioni avanzate per le definizioni di relazioni” a pagina 262.

Nelle fasi di trasformazione della mappa, le tabelle di relazioni sono gestite utilizzando i metodi contenuti `Relationship`, nelle classi `IdentityRelationship` e `Participant`. Alcune mappature dei metodi API gestiscono automaticamente le tabelle di relazioni. Inoltre, è possibile accedere esplicitamente a tali tabelle di relazioni per ottenere questi dati della relazione.

ID istanza della relazione: Il sistema WebSphere Business Integration Server Express identifica in modo univoco ciascuna istanza di relazione assegnandole un valore intero univoco, denominato *ID istanza relazione*. Questo ID dell'istanza consente al sistema di correlare i valori del partecipante. In generale, dato un partecipante di una relazione, è possibile richiamare i dati relativi ad altri partecipanti della relazione specificando l'ID di istanza della relazione.

Esempio: per la relazione tra i codici di stato del cliente di un'applicazione Clarify e di un'applicazione SAP, il sistema WebSphere Business Integration Server Express assegna un ID di istanza della relazione a ciascuna istanza di relazione della relazione di ricerca. Figura 109 illustra il modo in cui l'ID istanza `47` associa i due partecipanti specifici dell'applicazione `ClarLkUp` e `SAPLkUp`. L'ID 47 associa lo stato del cliente Clarify `Inattivo` con il valore di stato del cliente SAP `05`. Si noti che la relazione è, in genere, uguale a quella in Figura 108, con l'aggiunta dell'ID di istanza della relazione.

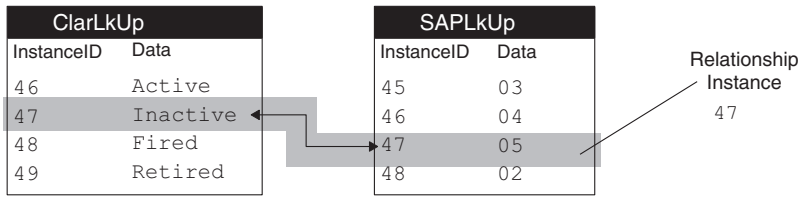


Figura 109. Una relazione di ricerca con ID istanza della relazione

Inoltre, il sistema WebSphere Business Integration Server Express utilizza un ID istanza della relazione per la relazione tra i partecipanti in una relazione di identità. Nella relazione `CustIden` (consultare Figura 104), questo ID istanza associa l'ID cliente memorizzato nell'attributo `SiteID` dell'oggetto business `Clarify_Site`, l'attributo `CustomerID` dell'oggetto business `Customer` generico e l'attributo `RefID` generico dell'oggetto business `SAP_Customer`. Figura 110 illustra il modo in cui sono associati i dati di istanza della relazione per ciascun partecipante della relazione `CustIden` utilizzando l'ID istanza della relazione.

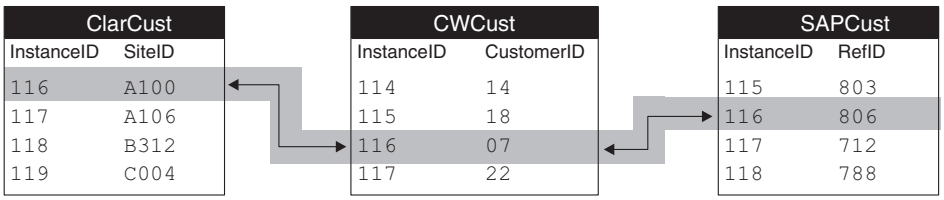


Figura 110. Una relazione di identità del client con ID istanza della relazione

In Figura 110, la tabella di relazioni per il partecipante `CWCust` è compresa per chiarezza, sebbene la tabella non sia necessaria. Infatti, le tabelle di relazione del

partecipante che rappresentano l'oggetto business generico nella relazione sono necessarie *solo* se si desidera generare un ID generico per l'attributo associato all'oggetto business generico. La relazione in Figura 110 genera un ID generico (07) per l'attributo CustomerID nell'oggetto business Customer generico.

E' possibile semplificare la definizione di relazione e ottimizzare le prestazioni eliminando le tabelle di relazione per il partecipante che rappresenta l'oggetto business generico. E' possibile effettuare questa operazione selezionando l'opzione gestito per il partecipante quando si crea la definizione della relazione. Per ulteriori informazioni su questa impostazione, consultare "Impostazioni avanzate per le definizioni del partecipante" a pagina 263.

Figura 111 illustra il modo in cui i dati di istanza della relazione sono associati nella relazione CustIden quando è specificata l'impostazione gestito per il partecipante CWCust.

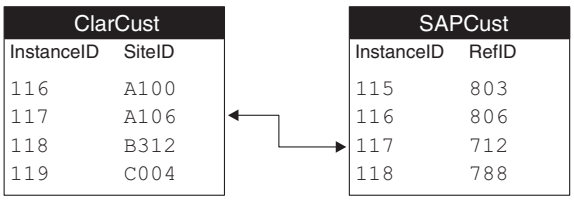


Figura 111. Un'istanza della relazione di identità senza tabella generica

Il sistema WebSphere Business Integration Server Express memorizza l'ID di istanza della relazione in tabella di relazione per ciascun partecipante. Come viene illustrato in Figura 109 e in Figura 111, ciascuna tabella di relazione in una relazione dispone di una colonna che contiene l'ID di istanza della relazione. InterChange Server Express crea automaticamente la colonna ID istanza della relazione quando crea lo schema della tabella.

Partecipanti

Una relazione contiene *partecipanti*, che descrive le entità partecipanti alla relazione. Come illustrato in Tabella 81, un partecipante è un'entità di due parti, costituita da una definizione di repository e un oggetto di runtime.

Tabella 81. Parti di un partecipante

Entità di archivio	Oggetto di runtime
Definizione del partecipante	Istanza del partecipante

Definizioni del partecipante

La definizione di relazione contiene un elenco di *definizioni del partecipante*. Ad esempio, la definizione di relazione CustIden in Figura 104 associa gli oggetti business cliente in Clarify e SAP e contiene queste definizioni del partecipante: SAPCust, CWCust e ClarCust.

Il sistema WebSphere Business Integration Server Express memorizza le definizioni del partecipante nel repository. Lo strumento Relationship Designer Express fornisce i dialoghi per consentire la creazione delle definizioni del partecipante. Utilizzando questo strumento, è inoltre possibile memorizzare la definizione del partecipante completata nel repository.

La definizione del partecipante fornisce le seguenti informazioni sul partecipante:

- Il nome del partecipante
- Il tipo di partecipante
- Il nome della tabella del partecipante e le procedure memorizzate

Nome della definizione del partecipante: Una definizione del partecipante è semplicemente un modello o una descrizione del partecipante; *non* è un oggetto business corrente. Pertanto, il nome della definizione del partecipante *non* deve essere il nome dell'oggetto business associato.

Tipo di partecipante: Come gli attributi in una definizione di oggetto business, i partecipanti nella definizione di relazioni dispongono di un tipo associato. Il tipo di partecipante specifica il tipo di dati associati alle istanze del partecipante. Il tipo di partecipante può essere uno dei seguenti:

- Il nome di una definizione dell'oggetto business

  Le relazioni con i partecipanti di questo tipo stabiliscono un'associazione tra gli oggetti business completi. In questo caso, specificare gli attributi dell'oggetto business che correlano il partecipante ad altri partecipanti della relazione. Gli attributi selezionati, in genere gli attributi chiave dell'oggetto business, diventano *Identificativi di istanza del partecipante*.
- Il termine `Dati`

  Nella definizione di partecipante, i `Dati` rappresentano un tipo di dati dell'attributo supportato, come ad esempio `Stringa`, `long`, `int`, `double`, `float` o `boolean`. Specificare i `Dati` come tipo dei partecipanti delle relazioni che stabiliscono le associazioni tra attributi specifici negli oggetti business. I partecipanti delle relazioni di ricerca che dispongono di un tipo di `Dati`.

Per informazioni sul modo in cui definire il tipo di partecipante, consultare "Creazione della definizione di una relazione" a pagina 255.

Tabella del partecipante e procedure memorizzate: Per ciascun partecipante, InterChange Server Express crea le seguenti entità di database:

- Una tabella del partecipante che contiene gli ID dell'istanza di relazione e il valore specifico dell'applicazione del partecipante associato
- Le procedure memorizzate per eseguire le operazioni di Richiamo (Seleziona), Elimina e Aggiorna nella tabella del partecipante

Per impostazione predefinita, Relationship Designer Express assegna i nomi dei seguenti dalla tabella del partecipante e dalla procedura memorizzata: *`RelName`* *`_ParticipantName_X`*, dove *`RelName`* è il nome della definizione della relazione, *`ParticipantName`* è il nome della relazione del partecipante e *`X`* è `T` per la tabella del partecipante o `SP` per la procedura memorizzata. Per impostazione predefinita, Relationship Designer Express crea le tabelle di relazione nel repository del sistema WebSphere Business Integration Server Express.

Relationship Designer Express consente di personalizzare i nomi della tabella del partecipante e delle procedure memorizzate. Per ulteriori informazioni sulla denominazione della tabella del partecipante e delle procedure memorizzate, consultare "Impostazioni avanzate per le definizioni del partecipante" a pagina 263.

Istanze del partecipante

La definizione del partecipante è un modello per l'esecuzione dell'istanza del runtime del partecipante, denominata *Istanza del partecipante*. Durante l'esecuzione della mappa, il sistema WebSphere Business Integration Server Express crea delle

istanze del partecipante basate sulla definizione del partecipante e i valori dell'attributo dagli oggetti business correnti che si stanno per trasformare.

Il sistema WebSphere Business Integration Server Express memorizza le istanze del partecipante come una colonna nella tabella della relazione del partecipante.

Esempio: per la relazione `CustIden` in Figura 104, il partecipante `ClarCust` dispone di una colonna denominata `SiteID` nella relativa tabella del partecipante per conservare i valori delle relative istanze del partecipante. Il partecipante `SAPCust` dispone di una colonna `RefID` nella relativa tabella del partecipante per conservare i valori delle relative istanze del partecipante.

Ciascuna istanza del partecipante contiene le informazioni di seguito riportate:

- Il nome della definizione della relazione
- L'ID istanza della relazione
- Nome della definizione del partecipante
- I dati per associare con il partecipante

Un'istanza del partecipante è rappresentata nella mappatura dell'API da un'istanza della classe del `partecipante`.

Panoramica del processo di sviluppo della relazione

Una relazione nel sistema WebSphere Business Integration Server Express è un'entità costituita da due parti:

- Una relazione di una definizione, memorizzata nell'archivio, per definire i partecipanti
- Il codice all'interno di una mappa per implementare la relazione accedendo alle tabella di relazione

Per definire una relazione nel sistema WebSphere Business Integration Server Express, è necessario eseguire i passi di base indicati:

1. Determinare il tipo di relazione necessaria.
2. All'interno di Relationship Designer Express definire una definizione di relazione e i partecipanti compositi.
3. In Map Designer Express personalizzare la regola di trasformazione, se necessario, per conservare la relazione.
4. Ricompilare le mappe interessate.
5. Distribuire le relazioni e le mappe in InterChange Server Express con l'opzione Crea schema.
6. Verificare che il/i database della relazione esista e che sia definito correttamente all'interno della relazione.
7. Popolare le tabelle di relazione per eventuali relazioni di ricerca. Facoltativamente, popolare altre tabelle di relazione con i dati di test per tale fase.
8. Per ciascuna mappa, avviare tutte le relazioni nella mappa.
9. Eseguire il test della relazione con il Connettore test. Assicurarsi di impostare il contesto di richiamo appropriato come parte dei ciascuno dei test.

Figura 112fornisce una panoramica di visualizzazione del processo di sviluppo della relazione e fornisce un rapido riferimento ai capitoli in cui è possibile trovare informazioni sugli argomenti specifici. Si noti che se un team di personale è

disponibile per lo sviluppo della mappa, le principali attività di sviluppo di una mappa possono essere effettuate in parallelo da membri diversi del team di sviluppo.

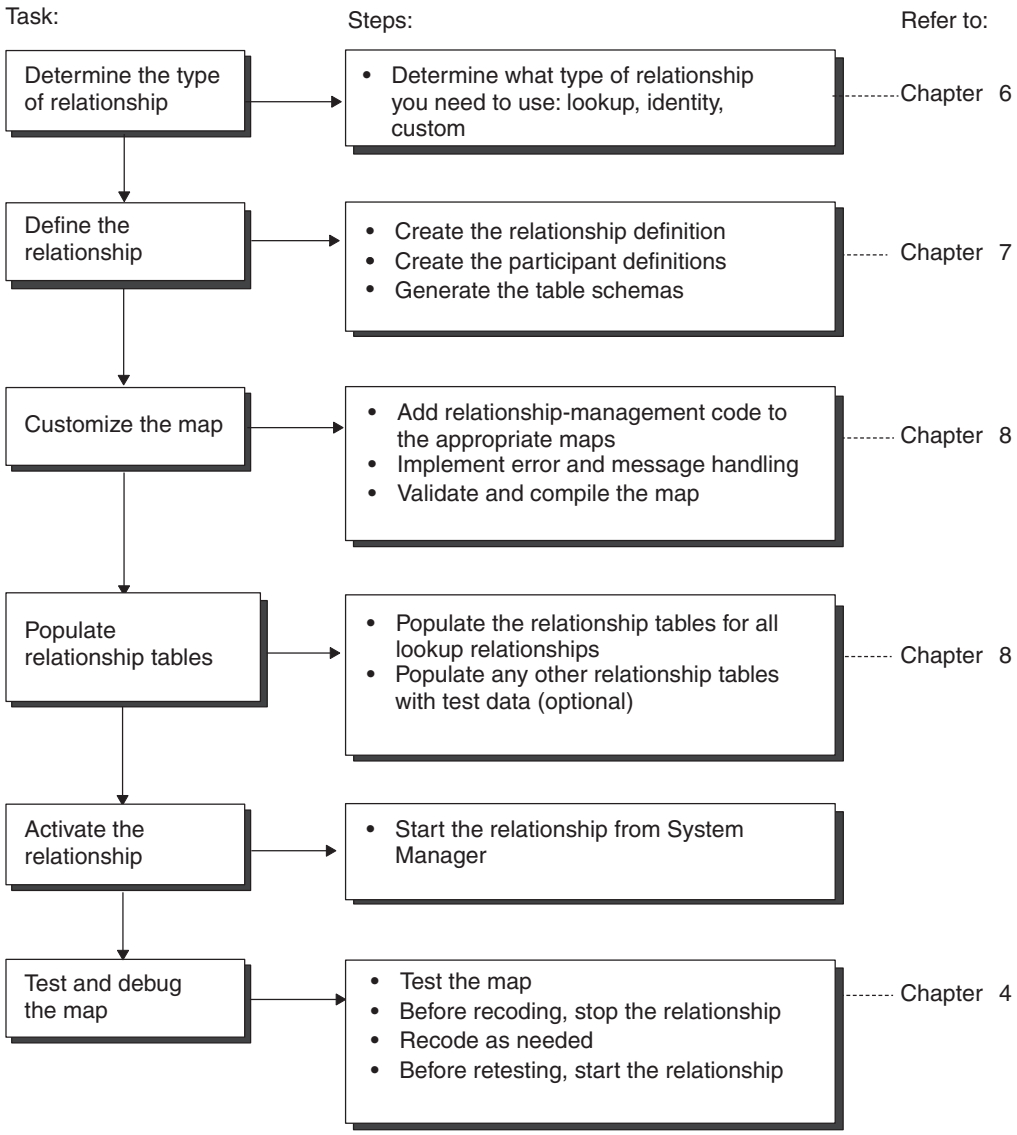


Figura 112. Panoramica dell'attività di sviluppo della relazione

Capitolo 7. Creazione delle definizioni di relazione

Questo capitolo descrive il modo in cui creare e modificare le definizioni di relazione utilizzando Relationship Designer Express. Per informazioni secondarie sul modo in cui il sistema WebSphere Business Integration Server Express utilizza le relazione nella mappatura, consultare Capitolo 6, "Introduzione alle relazioni", a pagina 235. Per informazioni sulla personalizzazione delle relazioni nelle mappe, consultare Capitolo 5, "Personalizzazione di una mappa", a pagina 109.

In questo capitolo sono illustrati i seguenti argomenti:

Panoramica di Relationship Designer Express

Relationship Designer Express è uno strumento di sviluppo grafico per la creazione e la modifica delle definizioni di relazione. Una *definizione di relazione* stabilisce un'associazione tra due o più partecipanti. E' possibile creare una definizione di relazione specificando i partecipanti della relazione e la definizione dell'origine dati, oltre ad altre proprietà associate a ciascun partecipante.

Questa sezione illustra gli argomenti di seguito riportati per la presentazione di Relationship Designer Express:

Avvio di Relationship Designer Express

Per avviare Relationship Designer Express, procedere nel modo seguente:

- Da System Manager, eseguire una delle azioni indicate:
  - Dal menu Strumenti, selezionare Relationship Designer Express.
  - Fare clic su una cartella Relazione in un progetto per abilitare l'icona Relationship Designer Express nella barra degli strumenti di System Manager. Quindi, fare clic sull'icona Relationship Designer Express.
  - Fare clic con il tastino destro del mouse sulla cartella Relazione in un progetto, quindi selezionare Designer Express dal menu Contesto.

– Fare clic con il tastino destro del mouse su una relazione nella cartella Dinamica o Statica, quindi selezionare Modifica definizioni dal menu Contesto.

Risultato: Relationship Designer Express avvia ed evidenzia la relazione selezionata.

- Da uno strumento di sviluppo, come ad esempio Business Object Designer Express, Map Designer Express, eseguire una delle azioni indicate:
  – Dal menu Strumenti, selezionare Relationship Designer Express.
  – Nella barra degli strumenti Programmi, fare clic sul pulsante Relationship Designer Express.
- Utilizzare un collegamento di sistema:

```
Avvio > Programmi > IBM WebSphere Business Integration Server Express
  > Toolset Express > Sviluppo > Relationship
 Designer Express
```

Importante: Affinché Relationship Designer Express sia in grado di accedere alle relazioni memorizzate in System Manager, Relationship Designer Express deve essere connesso ad un'istanza di System Manager. I passi precedenti suppongono che System Manager sia già stato avviato. Se System Manager è già in esecuzione Relationship Designer Express si collega direttamente ad esso.

Operazioni con i progetti

System Manager è l'unico strumento che interagisce con il server. Importa ed esporta le entità (relazioni, mappe) tra i progetti InterChange Server Express e System Manager. I vari strumenti, quali Relationship Designer Express, si collegano a System Manager e visualizzano, editano e modificano tali entità in base al progetto.

Un *progetto* è semplicemente un raggruppamento logico di entità per scopi di gestione e distribuzione. Una volta distribuite le entità a InterChange Server Express, il progetto dal quale sono state originate non ha più alcun significato.

System Manager consente di creare più progetti. Prima di effettuare attività con una relazione, è necessario selezionare il progetto in cui si trova la relazione.

Passi per la selezione di un progetto

Per selezionare un progetto con cui effettuare delle attività, eseguire i passi indicati qui:

1. Dal menu File, selezionare Passa al progetto.
2. Nel menu secondario Passa al progetto, selezionare il nome del progetto.

Risultato: ora è possibile effettuare delle attività con le relazioni contenute nel progetto. Prima di passare ad un altro progetto, è necessario salvare le relazioni modificate nel progetto corrente.

Figura 113 illustra l'opzione Passa al progetto per ricercare un progetto.

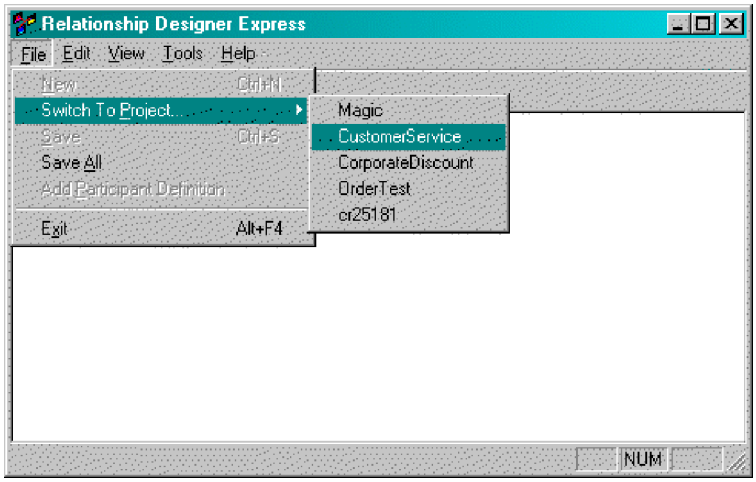


Figura 113. Ricerca di un progetto

Quando Relationship Designer Express stabilisce una connessione a System Manager, ottiene un elenco di oggetti business definiti nel progetto corrente. Questo elenco aiuta a definire i partecipanti.

Se si aggiunge o si elimina un oggetto business utilizzando Business Object Designer Express, System Manager invia una notifica a Relationship Designer Express, che aggiorna in modo dinamico l'elenco delle definizioni dell'oggetto business.

Layout di Relationship Designer Express

Nella finestra di Relationship Designer Express, al lato sinistro viene visualizzato un elenco delle definizioni di relazione nel progetto corrente. In questo elenco di definizione delle relazioni, viene visualizzato il contenuto di ciascuna definizione di relazione in formato gerarchico simile a Esplora risorse. E' possibile espandere il nome della relazione facendo clic sul simbolo più (+) acanto al relativo nome per visualizzare un elenco delle definizioni del partecipante, dei tipi di partecipante e degli attributi associati. Figura 114 illustra un elenco di definizione della relazione.

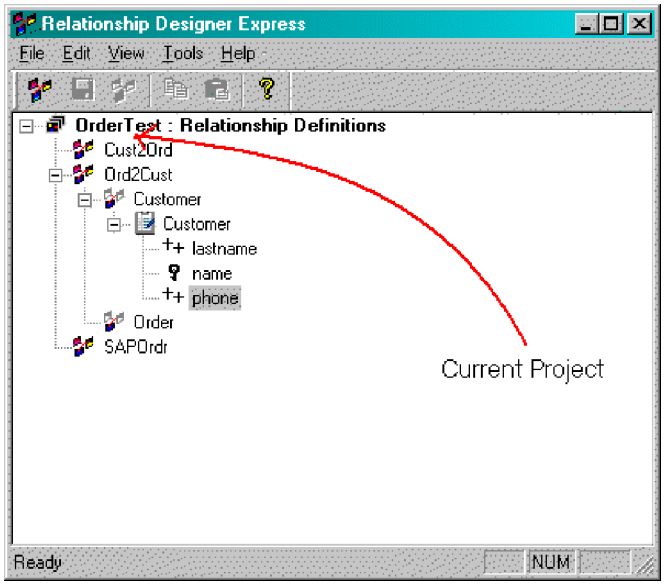


Figura 114. Elenco di definizione della relazione

La finestra Tipi di partecipante visualizza un elenco di tipi di dati disponibili nel progetto corrente che è possibile associare al partecipante.

Figura 115 illustra la finestra principale di Relationship Designer Express, con l'elenco Definizione di relazioni e la finestra Tipi di partecipante.

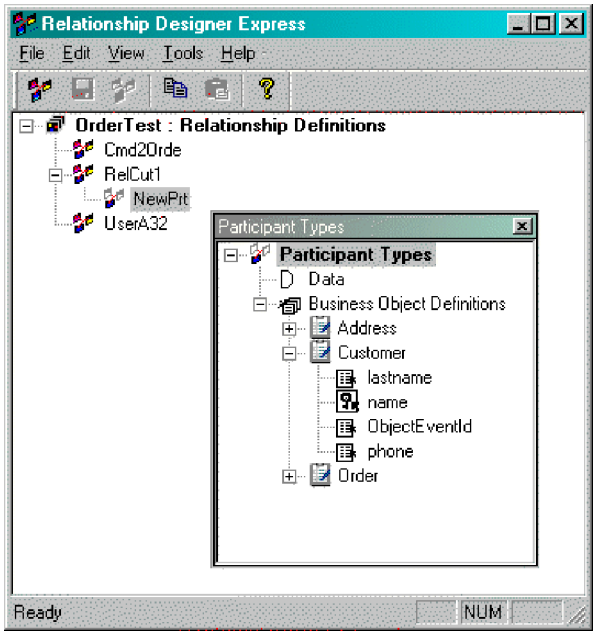


Figura 115. Finestra principale di Relationship Designer Express

Personalizzazione della finestra principale

Relationship Designer Express consente di personalizzare la propria finestra principale come segue:

- "Selezione delle finestre da visualizzare" a pagina 252
- "Spostamento di una finestra divisibile" a pagina 253

Selezione delle finestre da visualizzare

Quando si apre per la prima volta Relationship Designer Express, nella finestra principale viene visualizzato solo l'elenco di definizione della relazione. La finestra Tipi di partecipante non viene visualizzata. E' possibile personalizzare l'aspetto della finestra principale con le opzioni dal menu a discesa Visualizza. Tabella 82 descrive le opzioni del menu Visualizza e il modo in cui condizionano l'aspetto della finestra principale di Relationship Designer Express.

Tabella 82. Opzioni del menu vista per la personalizzazione della finestra principale

Opzioni del menu Visualizza	Elemento visualizzato
Tipi di partecipanti	Viene visualizzata la finestra Tipi di partecipante
Barra degli strumenti	La Barra degli strumenti standard, che dispone delle funzioni principali di Relationship Designer Express
Barra di stato	Un riquadro con un'unica riga in cui vengono visualizzate le informazioni sullo stato di Relationship Designer Express

Suggerimento: quando viene visualizzata un'opzione di menu con un contrassegno a sinistra, viene visualizzato anche l'elemento associato. Per disattivare la visualizzazione dell'elemento, selezionare l'opzione di menu associata. Il contrassegno non viene più visualizzato indicando che l'elemento non viene al momento visualizzato. Invece, è possibile attivare la visualizzazione di un elemento non visualizzato, selezionando l'opzione di menu associata. In questo caso, il contrassegno viene visualizzato accanto all'elemento di visualizzazione.

Spostamento di una finestra divisibile

Relationship Designer Express supporta le seguenti funzioni della finestra principale, come finestre divisibili:

- Barra degli strumenti Standard
- Finestra Tipi di partecipante

Suggerimento: per impostazione predefinita, una finestra divisibile è, in genere, posizionata lungo il margine della finestra principale e si sposta come parte di quest'ultima. Quando si sposta una finestra divisibile, la si stacca dalla finestra principale, consentendo di funzionare come finestra indipendente. Per spostare una finestra divisibile, tenere premuto il pulsante sinistro del mouse, spostare il bordo della finestra e trascinarlo fuori della finestra principale o sul desktop.

Utilizzo delle funzioni di Relationship Designer Express

E' possibile accedere alle funzioni di Relationship Designer Express utilizzando uno degli elementi di seguito riportato:

- Menu a discesa
- Menu Contesto
- Pulsanti della Barra degli strumenti
- Tasti di accesso rapido

Menu a discesa di Relationship Designer Express

Relationship Designer Express dispone dei seguenti menu a discesa:

- Menu File
- Menu Modifica
- menu Visualizza
- menu Strumenti
- Menu Guida

Le sezioni di seguito riportate descrivono le opzioni di ciascuno di questi menu. I tasti di accesso rapido sono disponibili per alcune di queste opzioni, come indicato.

Funzioni del menu File: Il menu a discesa File di Relationship Designer Express dispone delle opzioni illustrate in Tabella 83. Salvo per l'opzione Passa la progetto, tutte le opzioni del menu File condizionano gli oggetti del progetto corrente.

Tabella 83. Opzioni del menu File in Relationship Designer Express

Opzione del menu File	Descrizione	Per ulteriori informazioni
Nuovo (Ctrl+N)	Crea una nuova definizione di relazione	"Creazione della definizione di una relazione" a pagina 255
Passa al progetto (Ctrl+S)	Elenca gli altri progetti	"Operazioni con i progetti" a pagina 250
Salva	Salva la definizione della relazione corrente in un file	"Creazione della definizione di una relazione" a pagina 255

Tabella 83. Opzioni del menu File in Relationship Designer Express (Continua)

Opzione del menu File	Descrizione	Per ulteriori informazioni
Salva tutto	Salva tutte le definizioni di relazione aperte	N/D
Aggiungi definizione partecipante	Aggiunge una nuova definizione del partecipante alla definizione di relazione corrente	"Creazione della definizione di una relazione" a pagina 255

Funzioni del menu Modifica: Il menu a discesa Modifica di Relationship Designer Express dispone delle seguenti opzioni:

- Rinomina — ridenomina la definizione della relazione
- Copia (Ctrl+C) — Copia la definizione di relazione corrente.
- Incolla (Ctrl+V) — Incolla la definizione di relazione copiata.
- Taglia (Ctrl+X) — Taglia la definizione di relazione corrente.
- Impostazioni avanzate — Visualizza la finestra Impostazioni avanzate.

Funzioni del menu Visualizza: Il menu a discesa Visualizza di Relationship Designer Express dispone delle seguenti opzioni:

- Tipi di partecipante — Visualizza la finestra Tipi di partecipante.
- Espandi struttura — Visualizza i membri del livello corrente dell'elenco delle definizioni di relazione (come facendo clic sul simbolo più accanto al nome del livello).
- Riduci struttura — Riduce il livello corrente dell'elenco delle definizioni di relazione, in modo che i relativi membri non siano visualizzati (come farcendo clic sul simbolo meno accanto al nome del livello).
- Barra degli strumenti — Quando abilitata, viene visualizzata la Barra degli strumenti standard.
- Barra di stato — Quando abilitata, visualizza un messaggio di stato di una riga in basso nella finestra principale.

Per informazioni sulle opzioni del menu Visualizza che controlla la visualizzazione, consultare "Selezione delle finestre da visualizzare" a pagina 252.

Funzioni del menu Strumenti: Il menu a discesa Strumenti di Relationship Designer Express dispone delle opzioni per avviare ciascuno strumento di InterChange Server Express:

- Relationship Manager
- Process Designer Express

  Limitazione: questo strumento è disponibile solo in WebSphere Business Integration Server Express Plus.
- Map Designer Express
- Business Object Designer Express

Funzioni del menu Guida: Relationship Designer Express dispone di un menu Guida standard con le seguenti opzioni:

- Argomenti della guida (F1)
- Documentazione
- Informazioni su Relationship Designer Express

Menu Contesto: Il menu Contesto è un menu di accesso rapido disponibile facendo clic con il tastino destro del mouse in varie posizioni nell'applicazione. Viene aperto un menu contenente comandi utili, che cambiano in base alla posizione in cui si fa clic.

Barra degli strumenti standard di Relationship Designer Express

Relationship Designer Express dispone di una barra degli strumenti standard per le attività comuni da eseguire. Questa barra degli strumenti è divisibile, ovvero è possibile dividerla dalla tavolozza della finestra principale e spostarla fuori della finestra principale o sul desktop.

Suggerimento: per identificare lo scopo di ciascun pulsante della barra degli strumenti, passare con il cursore su ciascun pulsante.

Figura 116 illustra la barra degli strumenti standard di Relationship Designer Express.

Figura 116. Barra degli strumenti standard di Relationship Designer Express

L'elenco di seguito riportato fornisce la funzione di ciascun pulsante della barra degli strumenti standard, da sinistra a destra:

- Nuova relazione
- Salva relazione
- Nuovo partecipante
- Copia
- Incolla
- Guida

Creazione della definizione di una relazione

Effettuare i seguenti passi per creare una definizione di relazione:

1. Creare un nome della relazione eseguendo uno dei passi indicati:
   - Dal menu File, selezionare Nuova definizione della relazione
   - Utilizzare la combinazione di tasti di accesso rapido `Ctrl+N`.
   - Nella Barra degli strumenti standard, fare clic sul pulsante Nuova relazione.
2. Denominare l'icona di definizione della relazione.

   Regola: i nomi di definizione della relazione possono essere di una lunghezza fino a 8 caratteri, possono contenere solo lettere e numeri e devono iniziare con una lettera.
3. Creare una definizione del partecipante per ciascun oggetto business da correlare.

   Per effettuare questa operazione, selezionare un nome di definizione della relazione ed eseguire una delle azioni indicate:
   - Dal Menu File, selezionare Aggiungi definizione partecipante.
   - Nella Barra degli strumenti standard, fare clic sul pulsante Nuovo partecipante.
4. Per ciascuna definizione del partecipante, denominare l'icona per la definizione del partecipante.

Regola: i nomi di definizione del partecipante possono essere di una lunghezza fino a 8 caratteri, possono contenere solo lettere e numeri e devono iniziare con una lettera.

5. Associare un tipo di dati a ciascun partecipante trascinando il tipo dallafinestra Tipi di partecipante nella definizione del partecipante.

   Suggerimento: per visualizzare la finestra Tipi di partecipante, selezionare i Tipi di partecipante dal Menu Visualizza.

   - Per associare un tipo di dati di un oggetto business, trascinare la definizione dell'oggetto business dalla finestra Tipi di partecipante.

     I partecipanti nella relazione di identità utilizzano le definizioni dell'oggetto business come tipo di partecipante. Per ulteriori informazioni, consultare "Definizione delle relazioni di identità" a pagina 257.
   - Per associare un tipo di dati Java, trascinare il tipo di partecipante `Dati` dalla finestra Tipi di partecipante.

     Nella definizione di relazione, il tipo di partecipante `Dati` rappresenta *tutti* i tipi di dati diversi dai tipi dell'oggetto business. I partecipanti della relazione di ricerca utilizzano i `Dati` come tipo di partecipante. Per ulteriori informazioni, consultare "Definizione delle relazioni di ricerca" a pagina 259.
6. Per i tipi di partecipante che sono definizioni di oggetti business, aggiungere o modificare gli attributi da associare al partecipante.

   Gli attributi selezionati diventano la base su cui gli oggetti business sono correlati.
7. Salvare la definizione della relazione procedendo in uno dei modi riportati:
   - Dal menu File, selezionare Salva definizione della relazione.
   - Utilizzare la combinazione di tasti di accesso rapido `Ctrl+S`.
   - Nella Barra degli strumenti standard, fare clic sul pulsante Salva relazione.
8. Prima di eseguire una mappa che utilizza la definizione della relazione, effettuare i seguenti passi:
   a. Attivare la relazione. Una volta distribuita la relazione a InterChange Server Express, questa nuova relazione *non* viene attivata. Tuttavia, affinché la Mappatura dei metodi API sia in grado di accedere alle tabelle di relazione, è necessario che sia attiva una tabella di relazioni. Per attivare la relazione, fare clic sul nome della relazione in System Manager, quindi selezionare l'opzione Avvia dal menu Componente.
   b. Compilare e distribuire la mappa che utilizza la relazione.

      Risultato: se la mappa viene distribuita e compilata correttamente in InterChange Server Express, InterChange Server Express crea il codice di mappatura eseguibile e attiva la mappa. Per ulteriori informazioni, consultare "Compilazione di una mappa" a pagina 89.

Limitazioni:

1. L'IBM supporta la creazione delle tabelle di relazione solo nei database e nelle piattaforme supportate per il repository di InterChange Server Express.
2. Se si crea o si effettua una modifica a una definizione di relazione, è necessario arrestare prima la relazione mediante il menu Relazione di System Manager, effettuare le modifiche alla relazione, quindi riavviare la relazione.

Definizione delle relazioni di identità

Una *relazione di identità* stabilisce un'associazione tra due o più oggetti business su base uno-a-uno. Ovvero, per una determinata istanza di relazione, può essere presente solo un'istanza per ciascun partecipante. In genere, viene creata una relazione di identità per trasformare gli attributi chiave in un oggetto business, come ad esempio un ID cliente o del prodotto. Per ulteriori informazioni secondarie, consultare "Relazioni di identità" a pagina 237.

InterChange Server Express supporta i tipi di relazioni di identità illustrati in Tabella 84.

Tabella 84. Tipi di Relazioni di identità

Tipo di relazione di identità	Descrizione	Per ulteriori informazioni
Relazione di identità semplice	Correla due oggetti business mediante un attributo chiave singolo	"Utilizzo di relazioni di identità semplici" a pagina 277
Relazione di identità composita	Correla due oggetti business mediante una chiave composita (costituita da più di un attributo)	"Utilizzo delle relazioni di identità composite" a pagina 290

Passi per la definizione delle relazioni di identità

Per definire una relazione di identità utilizzando Relationship Designer Express, procedere nel modo seguente:

1. Creare una definizione di relazione e le definizioni del partecipante seguendo i passi da 1 a 4 indicati in "Creazione della definizione di una relazione" a pagina 255.

   Linee guida: Creare una definizione di partecipante per ciascun oggetto business da correlare. Le relazioni di identità richiedono i partecipanti per l'oggetto business generico, oltre agli oggetti business specifici dell'applicazione.

2. Associare un oggetto business a ciascuna definizione del partecipante, trascinando la definizione dell'oggetto business dalla finestra Tipi di partecipante nella definizione del partecipante. Rilasciare il pulsante di trascinamento quando viene visualizzato il simbolo (+) nella finestra principale di Relationship Designer Express. Per informazioni sull'apertura della finestra Tipi di partecipante, fare riferimento ai passi 5 in "Creazione della definizione di una relazione" a pagina 255.

   Per le relazioni di identità, il tipo di partecipante è un oggetto business. Ciascuna relazione di identità dispone di un partecipante con un tipo di partecipante dell'oggetto business generico più un partecipante per ciascun oggetto specifico dell'applicazione.

3. Per ciascun oggetto business che si associa ad una definizione del partecipante, aggiungere gli attributi che correlano l'oggetto business agli altri partecipanti.

   Per effettuare tale operazione, espandere l'oggetto business associato nella finestra Tipi di partecipante, selezionare un attributo e trascinarlo sull'oggetto business nella finestra principale di Relationship Designer Express. Gli attributi selezionati costituiscono la base della relazione tra gli oggetti business.

   Per le relazioni di identità, gli attributi sono in genere gli attributi chiave di ciascuna definizione dell'oggetto business. Il tipo di chiave determina il tipo di relazione di identità:

   - Per una singola chiave, utilizza una relazione di identità semplice. Ciascun partecipante può essere costituito da un solo attributo: la chiave univoca

dell'oggetto business. Per ulteriori informazioni, consultare "Passi per la creazione della definizione di relazioni child" a pagina 289.

- Per una chiave composita, utilizzare una relazione di identità composita. Specificare un chiave composita aggiungendo ciascun attributo chiave nell'ordine di visualizzazione che appare nella chiave composita. Ciascun partecipante può contenere vari attributi: in genere, la chiave univoca dall'oggetto business parent e l'ultimo attributo dall'oggetto business child (all'interno dell'oggetto business parent). Quando distribuito al server, la relazione viene salvata in una tabella, il cui nome è la concatenazione degli attributi nell'ordine di visualizzazione nella definizione del partecipante. Per ulteriori informazioni, comprese le limitazioni di dimensione dell'indice di alcuni database, consultare "Creazione delle definizioni della relazione di identità composita" a pagina 290.

4. Evidenziare il nome della definizione di relazione, quindi selezionare Impostazioni avanzate dal menu Modifica.

   Inizialmente, la finestra Impostazioni avanzate visualizza le impostazioni di definizione della relazione, come illustrato in Figura 118 a pagina 263.

   a. Modificare le impostazioni di definizione della relazione nel modo seguente:
      - In `Tipo di relazione`, selezionare la casella `Identità`.

        Risultato: questa impostazione istruisce InterChange Server Express per l'elaborazione della relazione come relazione di identità impostando un vincolo di univocità nell'ID istanza di relazione e negli attributi chiave per ciascun partecipante. Questa operazione garantisce una corrispondenza uno-a-uno tra tutti i partecipanti in ciascuna istanza di relazione.
      - Se si desidera che le tabelle di relazione si trovino in un database di verso da quello predefinito (il repository del sistema WebSphere Business Integration Server Express, per impostazione predefinita), immettere le informazioni sul database appropriate nell'area `Impostazioni DBMS` della finestra. Per ulteriori informazioni, consultare "Impostazioni avanzate per le definizioni di relazioni" a pagina 262.

   b. Modificare le impostazioni avanzate per la definizione del partecipante.
      - Nel browser degli oggetti della finestra Impostazioni avanzate, espandere la definizione della relazione ed evidenziare la definizione del partecipante che rappresenta l'oggetto business generico per visualizzare le impostazioni di definizione del partecipante (consultare Figura 119 a pagina 264). Selezionare la casella contrassegnata `IBM WBI-managed`.

        Risultato: questa operazione istruisce Relationship Designer Express a *non* creare tabelle di relazioni per l'oggetto business generico. Quando si conserva la relazione con il metodo `maintainSimpleIdentityRelationship()`, il sistema WebSphere Business Integration Server Express utilizza gli ID di istanza della relazione memorizzati nelle tabelle di relazione specifiche dell'applicazione per trasformare gli attributi di relazione.
      - Se si desidera personalizzare il nome di questa tabella di relazione del partecipante o procedura memorizzata, immettere il nome del campo appropriato che si trova nella finestra. Per ulteriori informazioni, consultare "Impostazioni avanzate per le definizioni del partecipante" a pagina 263.

   c. Fare clic su OK per chiudere la finestra Impostazioni avanzate.

5. Salvare la definizione della relazione come descritto nei passi 7 e 8 in "Creazione della definizione di una relazione" a pagina 255.

Correlazione di oggetti business child

Quando si creano delle relazioni di identità, gli oggetti business che si correlano dispongono spesso di oggetti business child. Ad esempio, alcuni oggetti business del client dispongono di oggetti business child per la memorizzazione delle informazioni sull'indirizzo. Un oggetto business child può partecipare ai tipi di relazioni illustrati in Tabella 85.

Tabella 85. Relazioni per gli oggetti business child

Condizione di un oggetto business child	Tipo di relazione	Per ulteriori informazioni
La chiave dell'oggetto business child identifica *in modo univoco* il child appartenente al contesto del relativo parent	Relazione di identità semplice	"Codifica di una relazione di identità semplice a livello di child" a pagina 288
La chiave per l'oggetto business *non* lo identifica in modo univoco nel contesto del relativo parent	Relazione di identità composita	
Per conservare gli oggetti business child durante un'operazione di Aggiornamento come parte di una relazione di identità	Relazione parent/child	"Gestione delle istanze child" a pagina 298

Se il child è un oggetto business con cardinalità multipla, è possibile modificare l'indice per rendere il riferimento del partecipante un child specifico. Per effettuare questa operazione, selezionare l''attributo chiave child, fare clic con il tastino destro del mouse, quindi selezionare Modifica indice dal menu Contesto. Se i child di origine e di destinazione in una mappa corrispondono uno a uno, l'indice non è significativo e non è necessario modificarlo. Tuttavia, se la mappa trasforma il child in qualunque altro modo, è possibile immettere un numero di indice specifico. Ad esempio, se gli oggetti business child rappresentano gli indirizzi e il terzo indirizzo di origine corrisponde al primo indirizzo di destinazione, è possibile modificare gli indici rispettivamente su `2` e `0`.

Definizione delle relazioni di ricerca

Una *relazione di ricerca* associa i dati che sono equivalenti tra gli oggetti business, ma possono essere rappresentati in modi diversi. In questo caso, dato un valore in un oggetto business, la relazione può ricercarne l'equivalente nelle tabelle di relazioni per un altro oggetto business. Il più comune esempio di attributi che potrebbero richiedere le ricerche sono i codici (`EmployeeType`, `PayLevel`, `OrderStatus`) e le abbreviazioni (`Stato`, `Paese`, `Valuta`). Per ulteriori informazioni secondarie, consultare "Relazioni di ricerca" a pagina 236.

Quando si crea una definizione di relazione per una ricerca, si aggiunge una definizione del partecipante per ciascun oggetto business contenente gli attributi da correlare. Tuttavia, non associare le definizioni dell'oggetto business corrente o i nomi degli attributi alle definizioni del partecipante. Invece, specificare `Dati` come tipo di partecipante per ciascuna definizione del partecipante.

Passi per la definizione delle relazioni di ricerca

Per definire una relazione di ricerca utilizzando Relationship Designer Express, procedere nel modo seguente:

1. Creare una definizione di relazione e le definizioni del partecipante seguendo i passi da 1 a 4 indicati in “Creazione della definizione di una relazione” a pagina 255.

   Suggerimento: Creare una definizione di partecipante per ciascun oggetto business da correlare.

2. Per ciascuna definizione del partecipante, specificare `Dati` come tipo di partecipante trascinando il tipo di partecipante `Dati` dalla finestra Tipi di partecipante nella definizione del partecipante.

   Nella definizione di relazione, il tipo di partecipante `Dati` rappresenta *tutti* i tipi di dati diversi dai tipi dell'oggetto business. Quando viene creata la mappa e si effettuano attività con le istanze della relazione utilizzando i metodi nelle classi `Relationship`, `IdentityRelationship` e `Participant`, è possibile utilizzare i dati di qualunque tipo di dati Java supportati, quali `Stringa`, `int`, `long`, `float`, `double` o `boolean`.

3. Prendere nota del nome della tabella per la memorizzazione dei valori di ricerca per ciascuna definizione del partecipante. E' necessario conoscere il nome della tabella per popolare le tabelle con i valori di ricerca per ciascuna definizione del partecipante. Oppure, se si dispone già di tabelle contenenti i valori di ricerca, è possibile sostituire il nome di tabella generato con il nome di tabella proprio.

   Per richiamare i nomi di tabella per ciascuna definizione di partecipante nella definizione della relazione o per specificare i nomi di tabella propri:

   a. Selezionare la definizione del partecipante e le Impostazioni avanzate dal menu Modifica.

      Risultato: viene visualizzata la finestra di dialogo Impostazioni avanzate con le impostazioni di memorizzazione per il partecipante. Per ulteriori informazioni su queste impostazioni, consultare “Specifica delle impostazioni di relazione avanzate” a pagina 262.

   b. Scrivere le impostazioni di memorizzazione per il partecipante o sovrascrivere le impostazioni con le proprie informazioni sulla tabella.

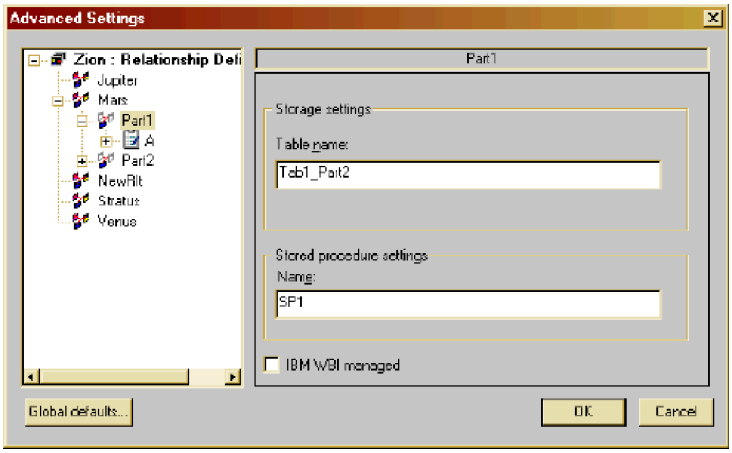


Figura 117. Dialogo Impostazioni avanzate

   c. Ripetere i passi 3a e 3b per ciascuna definizione del partecipante.

   d. Fare clic su OK per chiudere la finestra di dialogo Impostazioni avanzate.

4. Salvare la definizione della relazione come descritto nei passi 7 e 8 in “Creazione della definizione di una relazione” a pagina 255.

Suggerimento: per creare le tabelle di relazione, selezionare la casella Crea schema nella finestra di dialogo Distribuisci progetto in System Manager. Per ulteriori informazioni sulla creazione dello schema di runtime, consultare "Creazione dello schema della tabella di relazione" a pagina 261.

5. Utilizzando le informazioni raccolte al passo 3, popolare le tabelle di relazione con i valori di ricerca per ciascun partecipante oppure aggiungere le proprie tabelle dei valori di ricerca al database. Per ulteriori informazioni, consultare "Popolazione delle tabelle di ricerca con dati" a pagina 274.

Creazione dello schema della tabella di relazione

Per ciascuna definizione di relazione creata, InterChange Server Express utilizza i seguenti oggetti di database per conservare i dati di runtime per le istanze di relazione:

- Le tabelle nel database di relazione conservano i dati delle istanze di relazione.
- Le procedure memorizzate nel database di relazione conservano le tabelle di relazione.

Copia delle definizioni della relazione e del partecipante

Per creare una nuova definizione di relazione simile ad una esistente, copiare la definizione esistente e modificarla in base alle necessità. Inoltre, è possibile copiare una definizione del partecipante da una definizione di relazione e incollarla nella stessa definizione di relazione o in un'altra.

Passi per copiare le definizioni di relazioni nel progetto corrente

Per copiare una definizione di relazione, effettuare i seguenti passi:

1. Selezionare la definizione della relazione da copiare (ad esempio, CustToClient) e selezionare Salva definizione dal menu File.
2. Selezionare la definizione della relazione da copiare, quindi selezionare Copia dal menu Modifica.
3. Selezionare il nome del Progetto (nodo struttura root), quindi selezionare Incolla dal menu Modifica.

   Risultato: Relationship Designer Express crea una nuova definizione di relazione con il nome `Copy of CustToClient`. Il nome della definizione viene visualizzato in modalità di modifica.
4. Immettere un nuovo nome per la definizione di relazione, quindi premere Invio.
5. Per salvare la nuova definizione nel repository, selezionare Salva definizione della relazione dal menu File (o utilizzare la combinazione di tasti di accesso rapido `Ctrl+S`).

Suggerimenti: Per copiare una definzione di relazione da un InterChange Server Express ad un altro, utilizzare il comando `repos_copy`. Il comando `repos_copy` copia gli oggetti all'interno e all'esterno del repository di InterChange Server Express.

Passi per la copia delle definizioni del partecipante nel progetto corrente

Per copiare una definizione del partecipante, effettuare i seguenti passi:

1. Selezionare la definizione di relazione cui appartiene la relazione che si desidera copiare e selezionare Salva relazione di definizione dal menu File.
2. Selezionare la definizione del partecipante che si desidera copiare, quindi selezionare Copia dal menu Modifica.
3. Selezionare la definizione di relazione in cui si desidera copiare la definizione del partecipante, quindi selezionare Incolla dal menu Modifica.

   Risultato: Relationship Designer Express crea una nuova definzione del partecipante con il nome `Copia`. Il nome della definizione viene visualizzato in modalità di modifica.
4. Immettere un nuovo nome per la definizione del partecipante, quindi premere Invio.

Ridenominazione delle definizioni di relazione o del partecipante

E' possibile ridenominare una definizione di relazione o del partecipante prima di salvarla nel repository. Per modificare il nome di una definizione dopo averlo salvato, è necessario copiare la definizione in un nuovo nome ed eliminare il nome precedente. Per una guida alla copia delle definizioni, consultare "Copia delle definizioni della relazione e del partecipante" a pagina 261.

Specifica delle impostazioni di relazione avanzate

Per ciascuna definizione di relazione creata, Relationship Designer Express conserva le impostazioni avanzate che condizionano la memorizzazione e l'elaborazione dei dati di istanza della relazione.

Nota: Se si modifica una qualunque impostazione relativa al database, come ad esempio un nome account di login, una password o un nome di tabella dopo la creazione degli schemi della tabella di relazioni, è necessario creare di nuovo gli schemi della tabella di relazione utilizzando System Manager affinché siano applicate le modifiche.

Per visualizzare o modificare le impostazioni, selezionare Impostazioni avanzate dal menu Modifica. Nella finestra di dialogo Impostazioni avanzate, le impostazioni visualizzate al lato destro sono diverse in base alle voci selezionate a sinistra:

- Definizione della relazione
- Definizione del partecipante
- Attributo

Impostazioni avanzate per le definizioni di relazioni

Per visualizzare o modificare le impostazioni di una definizione di relazione, selezionare il nome della relazione. La figura di seguito riportata illustra un esempio delle impostazioni avanzate a questo livello:

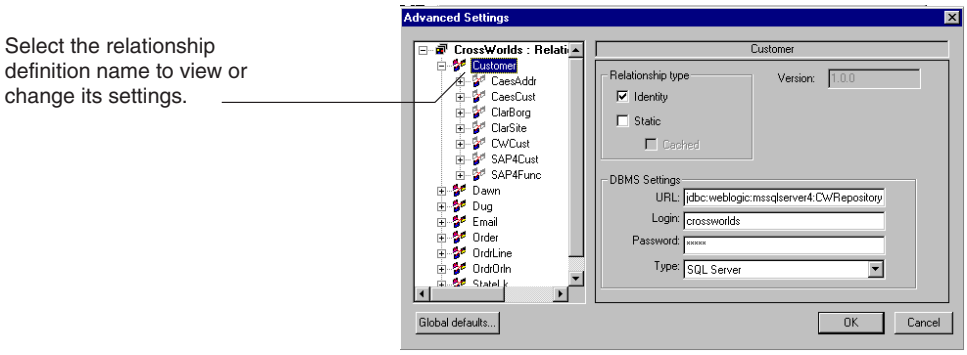


Figura 118. Impostazioni avanzate per la definizione di relazione

Tabella 86 riepiloga le impostazioni disponibili per le definizioni di relazione. I valori predefiniti per le impostazioni DBMS derivano dalla finestra di dialogo Impostazioni predefinite globali descritte in "Impostazioni predefinite globali" a pagina 265.

Tabella 86. Riepilogo delle impostazioni avanzate per le definizioni di relazione

Impostazione	Descrizione
Tipo relazione	
Dinamica (identità)	Quando viene abilitata questa opzione, la relazione è dinamica (identità). Per ulteriori informazioni, consultare "Definizione delle relazioni di identità" a pagina 257.
Statica (ricerca)	Quando viene abilitata questa opzione, la relazione è statica (ricerca). Per ulteriori informazioni, consultare "Definizione delle relazioni di ricerca" a pagina 259.
Memorizzata nella cache	Quando è abilitato il campo Statico, anche questo campo è abilitato. Selezionare questo campo per disporre di tabelle di relazione memorizzate nella cache. Per ulteriori informazioni, consultare "Ottimizzazione di una relazione" a pagina 267.
Versione	Questo campo è in sola lettura. Le versioni per le definizioni della relazione non sono supportate in questo rilascio.
Impostazioni DBMS	
URL	Il percorso JDBC in cui si trovano le tabelle di relazione per questa definizione di relazione. La posizione predefinita per tutte le tabelle di relazione è specificata nelle Impostazioni predefinite globali (consultare 265).
Login	Il nome utente per il login al database di relazione.
Password	La password per il login al database di relazione.
Tipo	La tipo di database di relazione, come ad esempio Server SQL e DB2.

Nota: Se si specifica un database per le tabelle di relazione diverso dal database del repository di InterChange Server Express, potrebbe essere necessario incrementare l'impostazione al numero massimo di pool di connessioni che il server può creare. Il parametro di configurazione del server che specifica il numero dei pool di connessione è `MAX_CONNECTION_POOLS`. Il valore predefinito è 10.

Impostazioni avanzate per le definizioni del partecipante

Per visualizzare o modificare le impostazioni per le definizioni del partecipante, selezionare il nome di definzione del partecipante. La seguente figura illustra un esempio delle impostazioni avanzate a questo livello:

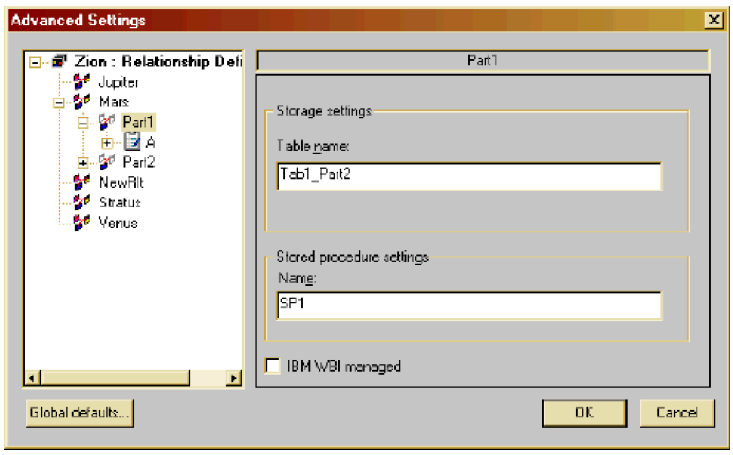


Figura 119. Impostazioni avanzate per una definizione del partecipante

Tabella 87 riepiloga le impostazioni disponibili per le definizioni del partecipante.

Tabella 87. Riepilogo delle impostazioni avanzate per le definizioni del partecipante

Impostazione	Descrizione
Nome della tabella	Nome della tabella di relazione nel database di relazione contenente i dati di relazione per l'istanza del partecipante. **Regola:** i database di relazione DB2 possono utilizzare solo un massimo di 17 caratteri nei nomi della tabella di relazione. Sebbene i nomi di tabella *non* dispongano di un limite in DB2, i nomi di indice dispongono di tale limite. Poiché Relationship Designer Express genera i nomi di indice per le tabelle di relazione in base ai relativi nomi di tabella, i nomi della tabella di relazione per un database DB2 devono essere costituiti da 17 caratteri o meno.
Nome della procedura memorizzata	Nome della procedura memorizzata che conserva la tabella di relazione.
InterChange Server Express gestito	Se selezionato, impedisce la creazione delle tabelle di relazione da parte di questo partecipante. Selezionare questa impostazione solo quando: • L'oggetto business associato alla definizione del partecipante è un oggetto business generico. • E' presento un solo attributo associato al partecipante e si tratta di un attributo chiave.

Impostazioni avanzate per gli attributi

Per visualizzare o modificare le impostazioni avanzate di un attributo, selezionare l'attributo. La figura di seguito riportata illustra un esempio delle impostazioni avanzate:

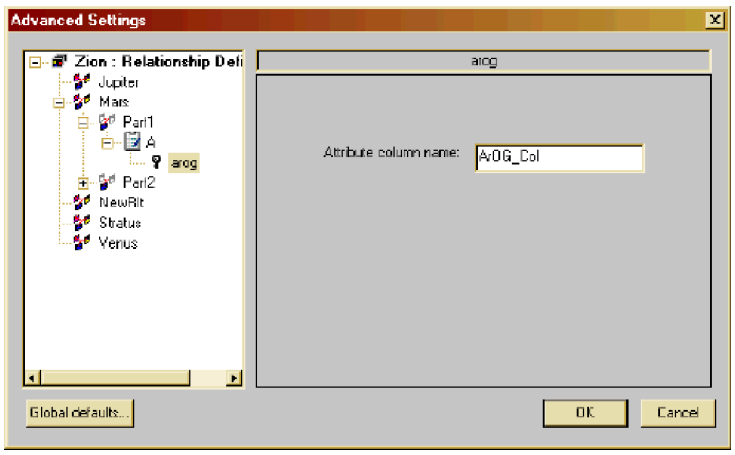


Figura 120. Impostazioni avanzate per gli attributi

Per gli attributi, la sola impostazione disponibile è il nome della colonna. Il nome della colonna è il nome della colonna nella tabella di relazione contenente i valori per l'attributo selezionato. In genere, si tratta dello stesso nome dell'attributo. Potrebbe essere necessario modificare il nome della colonna se si utilizzando le tabelle create invece delle tabelle predefinite create da Relationship Designer Express.

Impostazioni predefinite globali

Quando si salva una nuova definizione di relazione e si creano gli schemi di tabella della relazione, Relationship Designer Express deve conoscere la posizione del database delle tabelle di relazione, il tipo di database, e il modo in cui accedere il database con un nome utente e una password. Relationship Designer Express conserva i valori predefiniti per queste impostazioni, che utilizza per tutte le nuove definizioni della relazione creata. Una volta creata una definizione di relazione, queste impostazioni vengono memorizzate con la definzione di relazione ed è possibile modificare le impostazioni per ciascuna definizione di relazione singolarmente.

Per impostazione predefinita, il nome del database e le informazioni di accesso sono le stesse utilizzate dal repository di InterChange Server Express. Se si desidera memorizzare le tabelle di relazione in un'altra posizione, è possibile modificare le impostazioni globali.

Passi per la visualizzazione o la modifica delle impostazioni predefinite globali

Per visualizzare o modificare le impostazioni predefinite globali, effettuare i seguenti passi:

1. In Relationship Designer Express, selezionare Impostazioni avanzate dal menu Modifica.

   Risultato: viene visualizzata la finestra di dialogo Impostazioni avanzate.

2. Fare clic sul pulsante Impostazioni predefinite globali.

   Risultato: viene visualizzata la finestra di dialogo Impostazioni predefinite globali.

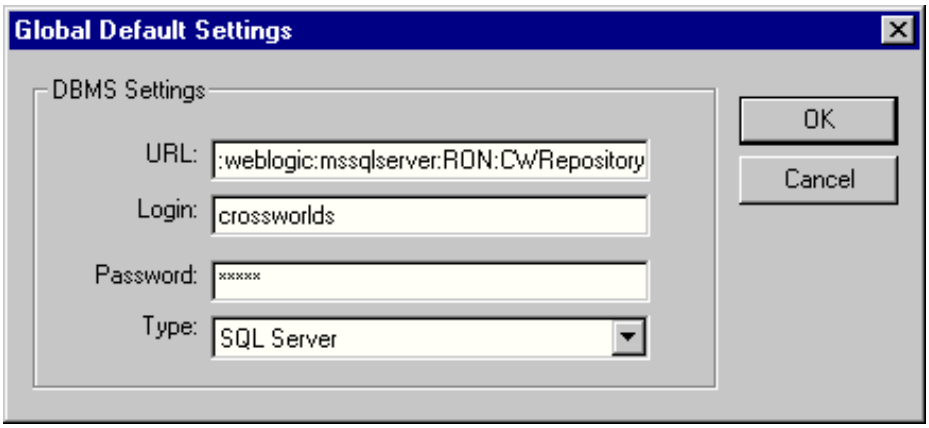


Figura 121. Dialogo impostazioni predefinite globali

Tabella 88 descrive le impostazioni predefinite globali per le relazioni.

Tabella 88. Impostazioni predefinite globali della relazione

Impostazione	Descrizione
URL	Il percorso JDBC in cui si trova il database della relazione. Il valore predefinito è il database del repository InterChange Server Express.
Login	Il nome utente per il login nel database di relazione.
Password	La password per il login al database di relazione.
Tipo	Il tipo di database della relazione, come ad esempio il server SQL o DB2.

Nota: Se si specifica un database per le tabelle di relazione diverso dal database del repository di InterChange Server Express, potrebbe essere necessario incrementare l'impostazione del numero massimo di pool di connessioni che il server può creare. Il parametro di configurazione del server che specifica il numero dei pool di connessione è `MAX_CONNECTION_POOLS`. Il valore predefinito è 10.

3. Una volta terminata la visualizzazione o la modifica, fare clic su OK per salvare o su Annulla per uscire senza salvare.

Nota: Le modifiche effettuate alle impostazioni predefinite globali si applicano solo alle nuove definizioni della relazione. Non condizionano le relazioni esistenti. Se si desidera modificare le impostazioni di una relazione esistente, consultare "Specifica delle impostazioni di relazione avanzate" a pagina 262.

Eliminazione di una definzione di relazione

E' possibile eliminare una definizione di relazione elencata nella finestra principale di Relationship Designer Express in uno dei seguenti modi:

- Evidenziare la definizione, quindi selezionare Elimina dal menu Modifica.
- Fare clic con il tastino destro del mouse sulla definizione, quindi selezionare Elimina.

Ottimizzazione di una relazione

Per impostazione predefinita, ciascuna relazione delle tabelle di relazione viene memorizzata nel database di relazione. Ogni volta che una relazione richiama o modifica i dati di runtime, utilizza le istruzioni SQL per accedere a questo database. Se le tabelle di relazione vengono accedute frequentemente, tali accessi possono avere un impatto significativo sulle prestazioni in termini di utilizzo di CPU e risorse di InterChange Server Express. Come parte del progetto di una relazione, è possibile determinare se memorizzare nella cache queste tabelle di relazione in memoria.

Per prendere tale decisione, è necessario determinare la frequenza con cui cambiano i dati di runtime della relazione. Il sistema WebSphere Business Integration Server Express consente di classificare la relazione in due modi:

- Relazione dinamica — una relazione in cui dati di runtime cambiano frequentemente, ovvero le relative tabelle di relazione hanno frequenti operazioni Inserisci, Aggiorna o Elimina. Tutte le relazioni sono dinamiche per impostazione predefinita.
- Relazione statica — una relazione i cui dati di runtime cambiano veramente poco, ovvero le relative tabelle di relazione hanno pochissime operazioni Inserisci, Aggiorna o Elimina. Ad esempio, poiché le tabelle di ricerca memorizzano le informazioni, come ad esempio valori di stato e codici, i relativi dati molto spesso non statici. Tali tabelle sono buoni candidati per essere memorizzati nella cache.

Nota: System Manager classifica le relazioni in queste due categorie. Quando si espande la cartella Relazioni, System Manager visualizza due cartelle secondarie: Dinamica e Statica.

Definire se una relazione è dinamica o statica dalla finestra di dialogo Impostazioni avanzate per la definizione della relazione. Le sezioni di seguito riportate riepilogano il modo in cui definire una relazione dinamica o statica da questa finestra di dialogo. Per informazioni sulla visualizzazione della finestra di dialogo Impostazioni avanzate, consultare “Specifica delle impostazioni di relazione avanzate” a pagina 262.

Definizione di una relazione dinamica

Per una relazione dinamica, InterChange Server Express accede i dati di run-time dalle relative tabelle di relazione nel database di relazione. Per impostazione predefinita, InterChange Server Express suppone che una relazione sia dinamica. Pertanto, non è necessario eseguire dei passi particolari per definire una relazione dinamica:

- Per una relazione di identità, fare clic sul campo Dinamica (identità) nella finestra di dialogo Impostazioni avanzate, come descritto in “Definizione delle relazioni di identità” a pagina 257.
- Per una relazione di ricerca, verificare che il campo Dinamico (identità) *non* sia selezionato, come descritto in “Definizione delle relazioni di ricerca” a pagina 259.

Nota: Per una relazione dinamica, *non* fare clic nel campo Statico (ricerca) o Memorizzato nella cache nella finestra di dialogo Impostazioni avanzate.

System Manager elenca tutte le relazioni dinamiche nella cartella denominata Dinamico nella cartella Relazioni.

Definizione di una relazione statica

Per una relazione statica, InterChange Server Express può accedere ai dati di runtime dalle tabelle di relazione memorizzate nella cache. Con la memorizzazione nella cache abilitata per una relazione statica, InterChange Server Express memorizza una copia delle tabelle di relazione in memoria. Quando si decide di memorizzare nella cache le tabelle di relazione, provare a equilibrare le seguenti condizioni:

- In genere, le prestazioni migliorano se si consente a InterChange Server Express di memorizzare nella cache le tabelle di relazioni.

  In questo caso, al server non occorre utilizzare le istruzioni SQL per accedere al database di relazioni per i dati di runtime. Invece, è possibile accedere alla memoria di questi dati, che è un'operazione più veloce. Se i dati di runtime per una relazione statica non sono al momento in memoria InterChange Server Express legge le tabelle di relazioni appropriate dal database in memoria quando i dati vengono acceduti per la prima volta. Per successivi accessi, InterChange Server Express utilizza la versione delle tabelle memorizzata nella cache.

  Tuttavia, una volta letta la tabella in memoria, InterChange Server Express deve conservare una coerenza tra le tabelle di relazione nel database e le tabelle memorizzate nella cache. Per le operazioni Aggiorna, Inserisci ed Elimina, InterChange Server Express deve modificare *entrambe* le tabelle di database *e* le tabelle memorizzate nella cache. Questo doppio aggiornamento può rallentare le prestazioni. Quando si determina se memorizzare nella cache delle tabelle di relazioni, considerare la durata prevista e l'intervallo di aggiornamento dei dati.
- L'utilizzo della memoria aumenta quando le tabelle di relazioni vengono memorizzate nella cache. La quantità di memoria utilizzata è approssimativamente equivalente alla dimensione di tutte le tabelle in memoria.

  Suggerimento: non memorizzare nella cache una tabella di relazioni contenente più di 1000 righe.

  Importante: InterChange Server Express *non* verifica l'eccessivo utilizzo di memoria. E' necessario assicurarsi che l'utilizzo di memoria resti entro i limiti imposti dal sistema.

Per definire una relazione statica, visualizzare la finestra di dialogo Impostazioni avanzate (consultare Figura 118) per la definizione delle relazioni e l'impostazione del campo Statico da questa finestra di dialogo come segue:

- Per una relazione di identità, abilitare entrambi i campi Dinamico (identità) e Statico (ricerca). Per ulteriori informazioni sull'utilizzo del campo Dinamico (identità), consultare "Definizione delle relazioni di identità" a pagina 257.
- Per una relazione di ricerca, abilitare il campo Statico (ricerca), *non* il campo Dinamico (identità).

Quando il campo Statico (ricerca) è abilitato, la finestra di dialogo Impostazioni avanzate abilita anche il campo Memorizzato nella cache. Il campo Memorizzato nella cache consente di controllare quando InterChange Server Express memorizza nella cache la tabella di relazioni:

- Quando l'opzione Memorizzato nella cache è abilitata InterChange Server Express può memorizzare nella cache le tabelle di relazioni per una relazione statica. Memorizza nella cache *tutte* le tabelle di relazioni coinvolte nella relazione.
- Quando l'opzione Memorizzato nella cache è disabilitata, InterChange Server Express non memorizza nella cache le tabelle di relazioni. Invece, utilizza le tabelle nel database di relazione per accessi futuri.

E' possibile controllare la memorizzazione nella cache per una relazione definita come statica.

Note:

1. Un volta modificata uno stato di relazione Memorizzato nella cache o Statico dalla finestra di dialogo Impostazioni avanzate, assicurarsi di salvare la definizione della relazione affinché la modifica venga memorizzata nel progetto.
2. E' possibile modificare le proprietà di relazione memorizzate nella cache e ricaricarle dalla vista di gestione del componente del server. Per effettuare questa operazione, fare clic con il tastino destro del mouse sulla relazione statica e selezionare le proprietà dal menu Contesto.
   - Memorizzato nella cache — controlla la memorizzazione nella cache delle tabelle di relazione.
   - Ricarica — istruisce InterChange Server Express a rileggere le tabelle di relazioni in memoria.

Capitolo 8. Implementazione delle relazioni

Gli attributi della relazione sono quelli che vengono trasformati utilizzando le relazioni. *Non* è possibile trasformare gli attributi della relazione trascinando l'attributo di origine in quello di destinazione. Invece creare una trasformazione personalizzata e personalizzare la regola di trasformazione per l'attributo della relazione di destinazione utilizzando i blocchi della funzione in Activity Editor o scrivere il codice per l'attributo della relazione di destinazione utilizzando i metodi delle classi `Relationship`, `IdentityRelationship` e `Participant`.

Questo capitolo descrive il modo in cui sviluppare il codice all'interno di una mappa per implementare i diversi tipi di relazioni. Sono descritte le seguenti attività.

Nota: In questo capitolo si suppone che siano già state create le definizioni di relazione per le relazioni. Per informazioni, consultareCapitolo 7, "Creazione delle definizioni di relazione", a pagina 249.

Implementazione di una relazione

Una volta creata una definizione della relazione all'interno di Relationship Designer Express, si è pronti per implementare la relazione all'interno della mappa. Per istruzioni sulla creazione delle definizioni, consultare Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249.

Per implementare una relazione, è possibile utilizzare i blocchi di funzione della relazione nell'oggetto di destinazione della mappa oppure aggiungere la mappatura dei metodi API al codice degli attributi dell'oggetto di destinazione della mappa.

Tabella 89 illustra i blocchi di funzione da utilizzare.

Tabella 89. Blocchi di funzione per la relazione

Tipo di relazione	Blocco di funzione	Per ulteriori informazioni
Ricerca	Generale/API/Relazione/Richiama istanze Generale/API/Relazione/Ottieni partecipanti	"Utilizzo delle relazioni di ricerca" a pagina 273

Tabella 89. Blocchi di funzione per la relazione (Continua)

Tipo di relazione	Blocco di funzione	Per ulteriori informazioni
Identità semplice	Generale/API/Relazione di identità/Conserva relazione di identità semplice Generale/API/Relazione di identità/Conserva child istruzione	"Utilizzo di relazioni di identità semplici" a pagina 277
Identità composita	Generale/API/Relazione di identità/Conserva Relazione composita Generale/API/Relazione di identità/Conserva child Istruzione Generale/API/Relazione di identità/Aggiorna child personali (facoltativo)	"Utilizzo delle relazioni di identità composite" a pagina 290
Personalizza	Generale/API/Relazione/Crea relazione Generale/API/Relazione di identità/Aggiungi child personali Generale/API/Relazione/Aggiungi partecipante	

Tabella 89 illustra la mappatura dei metodi API che conservano tipi diversi di relazioni.

Tabella 90. Mappatura dei metodi API per le relazioni

Tipo di relazione	Mappatura del metodo API	Per ulteriori informazioni
Ricerca	`retrieveInstances()retrieveParticipants()`	"Utilizzo delle relazioni di ricerca" a pagina 273
Identità semplice	`maintainSimpleIdentityRelationship()` `maintainChildVerb()`	"Utilizzo di relazioni di identità semplici" a pagina 277
Identità composita	`maintainCompositeRelationship()maintainChildVerb()` `updateMyChildren()`(facoltativo)	"Utilizzo delle relazioni di identità composite" a pagina 290
Personalizza	`create()addMyChildren()addParticipant()`	"Conservazione delle relazioni personalizzate" a pagina 312

Quando si trasformano gli attributi di relazione, una mappa deve conoscere il contesto di richiamo della mappa stessa. Per determinare il contesto di richiamo, la mappa deve disporre delle seguenti informazioni dal contesto di esecuzione della mappa:

- Il contesto di richiamo della mappa, che è parte del contesto di esecuzione della mappa

  Per ulteriori informazioni, consultare "Contesti di chiamata" a pagina 200.
- IL verbo, che è parte dell'oggetto business

Questi due fattori indicano alla mappa le azioni da eseguire sulle tabelle di relazioni.

Per le relazioni contenute in Tabella 89, la mappatura dei metodi API esegue le operazioni appropriate sulle tabelle di relazioni. Pertanto, tali metodi richiedono che il contesto di richiamo e il verbo dell'oggetto business siano inoltrati come argomenti.

Utilizzo delle relazioni di ricerca

Una *relazione di ricerca* associa i dati che sono equivalenti tra gli oggetti business, ma possono essere rappresentati in modi diversi. Di seguito sono riportate le sezioni che descrivono i passi per l'utilizzo delle relazioni di ricerca:

- "Creazione delle definizioni della relazione di ricerca"
- "Popolazione delle tabelle di ricerca con dati" a pagina 274
- "Personalizzazione delle trasformazioni della mappa per una relazione di ricerca" a pagina 275

Nota: Per informazioni aggiuntive, consultare "Relazioni di ricerca" a pagina 236.

Creazione delle definizioni della relazione di ricerca

Le definizioni della relazione di ricerca sono diverse dalle definizioni della relazione di identità, poiché i tipi di partecipante *non* sono oggetti business di tipo `dati` (la prima selezione nell'elenco dei tipi di partecipante). Per ulteriori informazioni sulla creazione di una definizione della relazione per una relazione di ricerca, consultare "Definizione delle relazioni di ricerca" a pagina 259.

Esempio: si supponga di creare una relazione di ricerca denominata `StatAdtp` per i valori `AddressType`. In Figura 122, ciascuna casella rappresenta un partecipante nella relazione di ricerca `StatAdtp`. Si noti che ciascun partecipante di questa relazione sia del tipo `Data`.

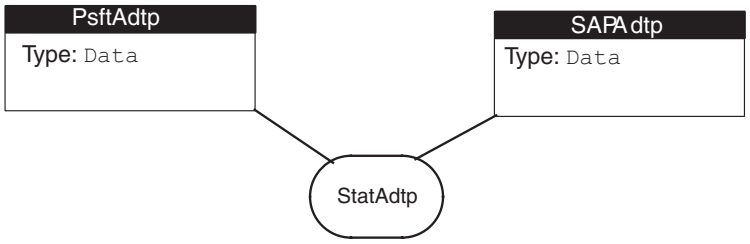


Figura 122. Definizione della relazione di ricerca StatAdtp

Poiché una relazione di ricerca non indica gli attributi correlati, è possibile utilizzare una definizione della relazione di ricerca per la trasformazione di vari attributi. Infatti, è possibile utilizzare una definizione di relazione di ricerca per ciascun attributo che richiede una ricerca indipendentemente dall'oggetto business che si sta trasformando. Tuttavia, poiché viene creata solo una serie di tabelle per ciascuna definizione della relazione, utilizzando una definizione della relazione per tutte le relazioni di ricerca, le tabelle diventano grandi e difficili da conservare.

Una strategia migliore potrebbe essere quella di creare una definizione della relazione di ricerca per unità comune di dati, come ad esempio il codice postale o lo stato. In questo modo, ciascuna tabella di relazione contiene le informazioni correlate per significato. Le relazioni definite in questo modo sono più modulari, poiché è possibile aggiungere nuovi partecipanti, dal momento che sono supportate nuove collaborazioni o applicazioni e si riutilizza la stessa definizione di relazione. Ad esempio, si supponga di creare una definizione della relazione di ricerca per il codice postale per trasformare gli oggetti business `Clarify_Site` in `SAP_Customer`. In seguito, se si aggiungono nuove collaborazioni o una nuova applicazione, è possibile riutilizzare la stessa definizione di relazione per ciascuna trasformazione che comprende un codice di avviamento postale.

Popolazione delle tabelle di ricerca con dati

Quando si distribuisce la definizione della relazione con l'opzione Crea schema abilitata, InterChange Server Express genera una tabella di relazioni (denominata anche *tabella di ricerca*) per ciascun partecipante. Ciascuna tabella di ricerca dispone di un nome nel seguente formato:

RelationshipDefName_ParticipantDefName

Quando si distribuisce la definizione della relazione StatAdtp (consultare Figura 122) con l'opzione Crea schema abilitata, InterChange Server Express genera le seguenti tabelle di ricerca:

- StatAdtp_PsftAdtp_T
- StatAdtp_SAPAdtp_T

Una tabella di ricerca contiene una colonna per l'ID istanza della relazione INSTANCEID) ed i dati dell'istanza del partecipante associati dati). Figura 123 illustra le tabelle di ricerca per i partecipanti PsftAdtp e SAPAdtp nella relazione di ricerca StatAdtp. Queste due tabelle utilizzano l'ID istanza della relazione per correlare i partecipanti. Ad esempio, l'ID d'istanza 116 correla il valore PsftAdtp Fired e il valore SAPAdtp 04.

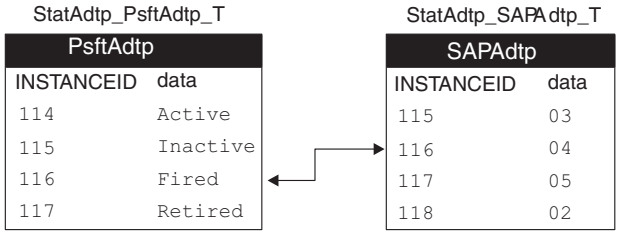


Figura 123. Tabelle di relazione per la relazione di ricerca CustLkUp

Diversamente dalle tabelle di relazioni che conservano i dati per le relazioni di identità, le tabelle di ricerca *non* vengono popolate automaticamente. E' necessario popolare queste tabelle inserendo dati nelle relative colonne. E' possibile popolare una tabella di ricerca in uno dei seguenti modi:

- Creare uno script contenente le istruzioni SQL INSERT per compilare la tabella di ricerca con i dati desiderati.
- Utilizzare Relationship Manager per aggiungere righe alla tabella di ricerca.

Inserimento delle istanze del partecipante con SQL

E' possibile inserire i dati del partecipante in una tabella di ricerca con l'istruzione SQLINSERT. Questo metodo è utile quando è necessario aggiungere più righe di dati alla tabella di ricerca. E' possibile creare la sintassi per un'istruzione INSERT quindi utilizzare l'editor per copiare e incollare questa riga tante volte quante sono le righe da inserire. In ciascuna riga, è necessario modificare solo i dati da inserire (in genere in una clause VALUES dell'istruzione INSERT.

Per utilizzare l'istruzione INSERT, è necessario conoscere il nome della tabella di relazioni di ricerca e delle relative colonne. Tabella 91 illustra i nomi delle colonne di una tabella di ricerca.

Tabella 91. Colonne di una tabella di ricerca

Colonne di una tabella di ricerca	Descrizione
INSTANCEID	L'ID di istanza della relazione.

Tabella 91. Colonne di una tabella di ricerca (Continua)

Colonne di una tabella di ricerca	Descrizione
dati	I dati del partecipante
STATUS	Impostare su zero (0) quando il partecipante è attivo
LOGICAL_STATE	Indica se l'istanza del partecipante è stata eliminata in modo logico (zero indica "no")
TSTAMP	Data dell'ultima modifica per l'istanza del partecipante.

Attenzione: Quando si utilizzano le istruzioni SQL per inserire i dati del partecipante in una tabella di ricerca, verificare di fornire un valore per le colonne STATUS, LOGICAL_STATE e TSTAMP. Tutti i valori sono richiesti affinché per gli strumenti di IBM WebSphere Business Integration Server Express funzionino correttamente. In particolare, l'omissione del valore TSTAMP non consente a Relationship Manager di richiamare i dati del partecipante, se non esiste alcun valore data/ora, Relationship Manager restituisce un'eccezione.

Esempio: si supponga che si desidera aggiungere i dati del partecipante nella tabella di relazioni contenente le informazioni per il tipo di indirizzo, illustrato in Tabella 92.

Tabella 92. Valori di esempio per il tipo di indirizzo del partecipante PsftAdtp

INSTANCEID	STATUS	LOGICAL_STATE	TSTAMP	dati
1	0	0	data corrente	Inizio
2	0	0	data corrente	Mailing

Le seguenti istruzioni INSERT creano i dati del partecipante Tabella 92 nella tabella di ricerca PstfAdtp:

```
INSERT INTO StatAdtp_PsftAdtp_T

   (INSTANCEID, STATUS, LOGICAL_STATE, TSTAMP, dati)
   VALUES (1, 0, 0, getDate(), 'Home')

INSERT INTO StatAdtp_PsftAdtp_T
   (INSTANCEID, STATUS, LOGICAL_STATE, TSTAMP, dati)
   VALUES (2, 0, 0, getDate(), 'Mailing')
```

Nota: La sintassi INSERT precedente è compatibile con MicroSoft SQL Server 7.0. Se si utilizza un altro server di database per la tabella di relazioni, assicurarsi di utilizzare la sintassi INSERT compatibile con quel server.

Inserimento delle istanze del partecipante con Relationship Manager

Relationship Manager è uno strumento IBM WebSphere Business Integration Server Express che visualizza graficamente i dati di runtime in una tabella di relazioni. Relationship Manager è utile quando è necessario aggiungere solo alcune righe alla tabella di ricerca.

Personalizzazione delle trasformazioni della mappa per una relazione di ricerca

Una volta creata la definizione della relazione e le definizioni del partecipante per la relazione di ricerca, è possibile personalizzare la regola di trasformazione della

mappa per eseguire le ricerche. Per informazioni sulla personalizzazione delle relazioni di ricerca in Activity Editor, consultare “Esempio 3: utilizzo della ricerca statica per la conversione” a pagina 164.

Tabella 93 illustra la mappatura dei metodi API necessaria per implementare una relazione di ricerca. Inoltre, questa tabella elenca la mappa in cui è necessario il richiamo API.

Tabella 93. Mappatura dei metodi API per le relazioni di ricerca

Passo nella relazione di ricerca	Mappa	Mappatura del metodo API
Ottenere l'ID di istanza della relazione per i dati del partecipante dall'oggetto business di origine. L'ID istanza viene salvato nell'oggetto business generico.	Mappa in entrata	`retrieveInstances()`
Ottenere le istanze del partecipante per l'ID di istanza della relazione dall'oggetto business generico. I dati del partecipante vengono salvati nell'oggetto business specifico dell'applicazione.	Mappa in uscita	`retrieveParticipants()`

Suggerimenti: I metodi `retrieveInstances()` e `retrieveParticipants()` *non* popolano le tabelle di ricerca. Presumono che i dati del partecipante esistono già nella tabella. Verificare di popolare le tabelle di ricerca prima di eseguire una mappa contenente una relazione di ricerca. Per ulteriori informazioni, consultare “Popolazione delle tabelle di ricerca con dati” a pagina 274.

Codifica della mappa in entrata

Il metodo `retrieveInstances()` viene richiamato nella mappa in entrata per ripristinare gli ID di istanza della relazione per i dati del partecipante nell'oggetto business di origine. Di seguito viene riportata una parte del codice che esegue una ricerca nella mappa in entrata:

```
String addrtype = ObjSrcObj.getString("SrcAttr");
int[] generic_ids = Relationship.retrieveInstances(
   "RelationshipDefName", "ParticipantDefName", dataFromSrcObj);
if (generic_ids != null && generic_ids.length > 0)
   {
   ObjDestObj.setWithCreate("DestAttr", generic_ids[0]);
   }

else
   {
   throw new MapFailureException(
   logError("No generic instance ID for lookup found");
      "No generic instance ID for lookup found");
   }
```

Suggerimenti: durante la codifica del metodo `retrieveInstances()` tenere presente le seguenti considerazioni:

- Il metodo `retrieveInstances()` *non* restituisce un'eccezione se non trova ID di istanza corrispondenti per determinati dati del partecipante. Per inoltrare un'eccezione se non viene trovato alcun ID di istanze corrispondenti, il frammento del codice precedente verifica gli ID di istanze restituite (`generic_ids`) per un valore null *prima* che venga impostato l'oggetto business di destinazione con `setWithCreate()`.
- Il metodo `retrieveInstances()` restituisce un *vettore* degli ID di istanza della relazione. In genere, una relazione di ricerca è strutturata in modo che ciascuna parte di dati del partecipante è associata a un solo ID di istanza. Tuttavia,

retrieveInstances() non suppone tale corrispondenza uno-a-uno. Restituisce un vettore in modo che possa restituire più ID di istanza di relazioni.

Codifica della mappa in uscita

Il metodo retrieveParticipants() viene richiamato nella mappa in uscita per ripristinare gli ID di istanza della relazione per i dati del partecipante nell'oggetto business di origine. Di seguito viene riportata una parte del codice che esegue una ricerca nella mappa in uscita:

```
int addrtype = ObjSrcObj.getInt("SrcAttr");

if (addrtype != null)
   {
   Participant[] psft_part = Relationship.retrieveParticipants(
      "RelationshipDefName", "ParticipantDefName", addrtype);
   if (psft_part != null && psft_part.length > 0)
      ObjDestObj.setWithCreate("DestAttr", psft_part[0].getString());
   }
```

Suggerimenti: durante la codifica del metodo retrieveParticipants() tenere presente le seguenti considerazioni:

- Se la mappa in uscita non può supporre che l'oggetto business generico contiene un ID di istanza della relazione, è possibile verificare un ID di istanza con valore null *prima* di richiamare retrieveParticipants(). Il metodo retrieveParticipants() restituisce un'eccezione RelationshipRuntimeException se riceve un ID istanza di valore null.
- Il metodo retrieveParticipants() restituisce un *vettore* delle istanze del partecipante. In genere, una relazione di ricerca è strutturata in modo che ciascun ID di istanza della relazione sia associato ad una parte di dati del partecipante. Tuttavia, retrieveParticipants() non suppone tale corrispondenza uno-a-uno. Restituisce un vettore in modo che possa restituire più istanze del partecipante.

Utilizzo di relazioni di identità semplici

Una relazione di identità stabilisce un'associazione tra gli oggetti business o altri dati su base *uno-a-uno*. Una relazione di identità semplice correla due oggetti business mediante un attributo chiave semplice. Le sezioni di seguito riportate descrivono i passi per le attività con le relazioni di identità semplice:

- "Creazione delle definizioni della relazione di identità semplice"
- "Accesso alle tabelle di relazione di identità" a pagina 278
- "Definizione delle regole di trasformazione per una relazione di identità semplice" a pagina 288

Creazione delle definizioni della relazione di identità semplice

Le definizioni della relazione di identità sono diverse dalle definizioni della relazione di ricerca, poiché i tipi di partecipante sono oggetti business, *non* tipi di Dati (la prima selezione nell'elenco dei tipi di partecipante). Per una relazione di identità semplice, la relazione è costituita dall'oggetto business generico e almeno un oggetto business specifico dell'applicazione. Il tipo di partecipante per una relazione di identità semplice è un oggetto business per *tutti* i partecipanti. L'attributo del partecipante per ciascun partecipante è un attributo con chiave singola dell'oggetto business. (Per ulteriori informazioni sulla creazione di una definizione di relazione per una relazione di identità semplice, consultare "Definizione delle relazioni di identità" a pagina 257.)

Accesso alle tabelle di relazione di identità

Per un riferimento ad una relazione di identità semplice, definire una regola di trasformazione del riferimento incrociato tra l'oggetto business specifico dell'applicazione e oggetto business generico. Per ulteriori informazioni, consultare "Relazioni di identità con riferimento incrociato" a pagina 51.

Esempio: la relazione `CustIden` (consultare Figura 104) trasforma un attributo chiave `SiteID` nel cliente Clarify in un attributo chiave `RefID` nel cliente SAP. Comprende mappe tra i seguenti oggetti:

- Mappa in entrata: da `Clarify_Site` a `Customer`

  Ottenere dalla tabella di relazioni `ClarCust` l'ID di istanza della relazione associato al valore di chiave `SiteID`.
- Mappa in uscita: da `Customer` a `SAP_Customer`

  Ottenere dalla tabella di relazione `SAPCust` il valore di chiave `RefID` associato all'ID di istanza della relazione.

Figura 124 illustra il modo in cui utilizzare le tabelle di relazione `CustIden` per trasformare un valore `SiteID` di `A100` in un valore `RefID` di `806`.

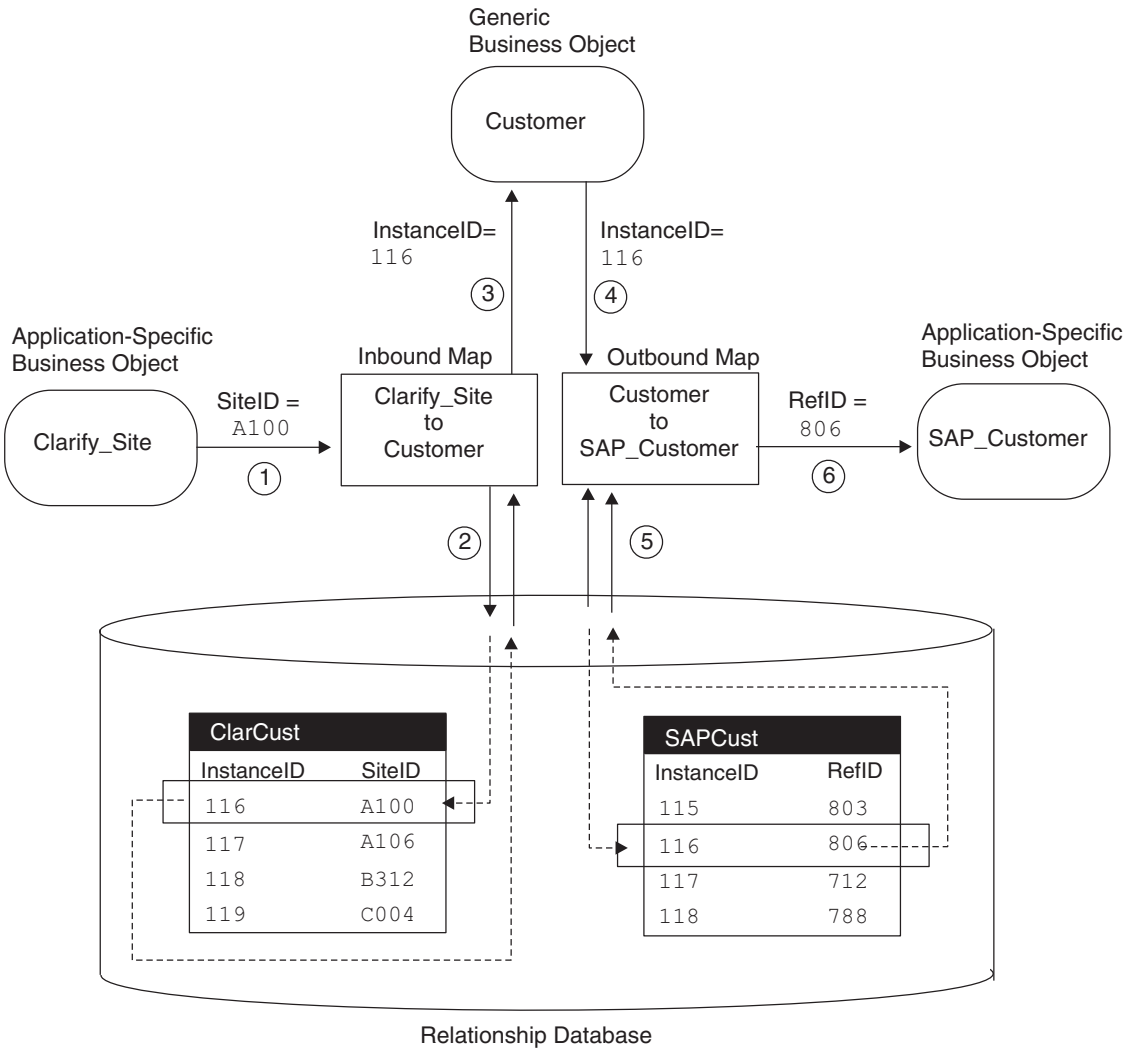


Figura 124. Utilizzo di tabelle di relazione per trasformare un SiteID in un RefID

Il metodo maintainSimpleIdentityRelationship() deve gestire le tabelle di relazione per assicurare che le chiavi specifiche dell'applicazione correlate restino associate ad un singolo ID di istanza della relazione. Ad un livello alto, la regola di trasformazione dei riferimento incrociato genera il codice per le seguenti operazioni:

1. Esegue le convalide sugli argomenti inoltrati. Se un argomento non è valido, il metodo restituisce l'eccezione `RelationshipRuntimeException`. Per un elenco delle convalide eseguite dal codice Java generato dalla trasformazione del riferimento incrociato, consultare l'API `maintainSimpleIdentityRelationship()` in Capitolo 21, "Classe IdentityRelationship", a pagina 455.
2. Intraprendere le azioni appropriate per conservare le tabelle di relazioni in base al contesto di richiamo, che comprende i seguenti fattori:
   - Il verbo di un oggetto business

     La trasformazione del riferimento incrociato ottiene questo verbo dall'oggetto business di origine. Per le mappe in entrata, l'origine è l'oggetto specifico dell'applicazione, per le mappe in uscita, l'origine è l'oggetto business generico.
   - Il valore del contesto di richiamo

La regola di trasformazione del riferimento incrociato ottiene automaticamente il contesto di richiamo dal contesto di esecuzione della mappa.

Questa trasformazione è correlata ai contesti di richiamo illustrati in Tabella 94.

Tabella 94. Contesti di richiamo con maintainSimpleIdentityRelationship()

Contesto di richiamo	Descrizione
EVENT_DELIVERY	Un connettore ha inviato un evento dall'applicazione a InterChange Server Express (flusso trigger di eventi).
ACCESS_REQUEST	Un client di accesso ha inviato una richiesta di accesso da un'applicazione esterna a InterChange Server Express
SERVICE_CALL_REQUEST	Una collaborazione è spedita all'oggetto business elaborato dall'applicazione attraverso una richiesta di chiamata di servizio.
SERVICE_CALL_RESPONSE	Un oggetto business è stato ricevuto dall'applicazione come risultato di una risposta corretta ad una richiesta di chiamata di servizio.
SERVICE_CALL_FAILURE	Una richiesta di richiamo del servizio di collaborazione non è riuscita. Ad esempio, è possibile che sia necessario intraprendere un'azione.
ACCESS_RESPONSE	L'oggetto business sorgente viene rispedito al client di accesso sorgente in risposta ad una richiesta di un rilascio sottoscritto.

Le sezioni di seguito riportate descrivono il comportamento della trasformazione del riferimento incrociato con ciascuno dei contesti di richiamo in Tabella 94.

Nota: Per ulteriori informazioni sul richiamo delle mappe all'interno delle collaborazioni, consultare la sezione "Richiamo di una mappa nativa" nel manuale *Collaboration Development Guide*

Contesti di richiamo EVENT_DELIVERY e ACCESS_REQUEST

Quando un contesto di richiamo è EVENT_DELIVERY o ACCESS_REQUEST, la mappa che sta per essere chiamata è una mappa in entrata, ovvero trasforma un oggetto business specifico dell'applicazione in un oggetto business generico. Un connettore invia il contesto di richiamo EVENT_DELIVERY, client di accesso invia un contesto di richiamo ACCESS_REQUEST. In entrambi i casi, la mappa in entrata riceve un oggetto business specifico dell'applicazione come input e restituisce un oggetto business generico come output. Pertanto, l'attività per la trasformazione del riferimento incrociato è di ottenere dalla tabella di relazione un ID di istanza di relazione per un determinato valore della chiave specifico dell'applicazione.

Per i contesti di richiamo EVENT_DELIVERY e ACCESS_REQUEST, il codice Java generato dalla trasformazione del riferimento incrociato intraprende le azioni di seguito riportate:

1. Ricerca l'istanza di relazione nella tabelle di relazione corrispondente al valore di chiave dell'oggetto business specifico dell'applicazione. Tabella 95 illustra le azioni che il codice Java generato dalla trasformazione del riferimento incrociato intraprende nella tabella di relazione in base al verbo dell'oggetto business specifico dell'applicazione.
2. Ottiene l'ID di istanza dall'istanza di relazione ripristinata.

3. Copia un ID di istanza in un oggetto business generico.

Tabella 95. Azioni per i contesti di richiamo EVENT_DELIVERY e ACCESS_REQUEST

Verbo dell'oggetto business specifico dell'applicazione	**Azione eseguita da maintainSimpleIdentityRelationship()**
Creare	Inserire una nuova voce nella tabelle di relazione per il valore di chiave dell'oggetto business specifico dell'applicazione. Se una voce per questo valore di chiave già esiste, ripristinare quello esistente, *non* aggiungerne un altro alla tabella.
Aggiorna	Richiamare la voce della relazione dalla tabella di relazioni per un determinato valore di chiave dell'oggetto business specifico dell'applicazione. Se una voce per questo valore di chiave *non* esiste, aggiungere una alla tabella.
Elimina	1. Richiamare la voce della relazione dalla tabella di relazioni per un determinato valore di chiave dell'oggetto business specifico dell'applicazione. 2. Contrassegnare la voce della relazione come "disattiva".
Recupera	Richiamare la voce della relazione dalla tabella di relazioni per un determinato valore di chiave dell'oggetto business specifico dell'applicazione. Se una voce per questo valore di chiave *non* esiste, inoltrare un'eccezione `RelationshipRuntimeException`.

Per una relazione di identità che supporta la trasformazione di un oggetto business specifico dell'applicazione `AppA` ad un oggetto business specifico dell'applicazione `AppB`, Figura 125 illustra il modo in cui il codice Java generato dalla trasformazione del riferimento incrociato accede ad una tabella di relazioni con il partecipante `AppA` quando un contesto di richiamo è `EVENT_DELIVERY` (o `ACCESS_REQUEST`) e il verbo dell'oggetto business specifico dell'applicazione `AppA` è Crea o Aggiorna.

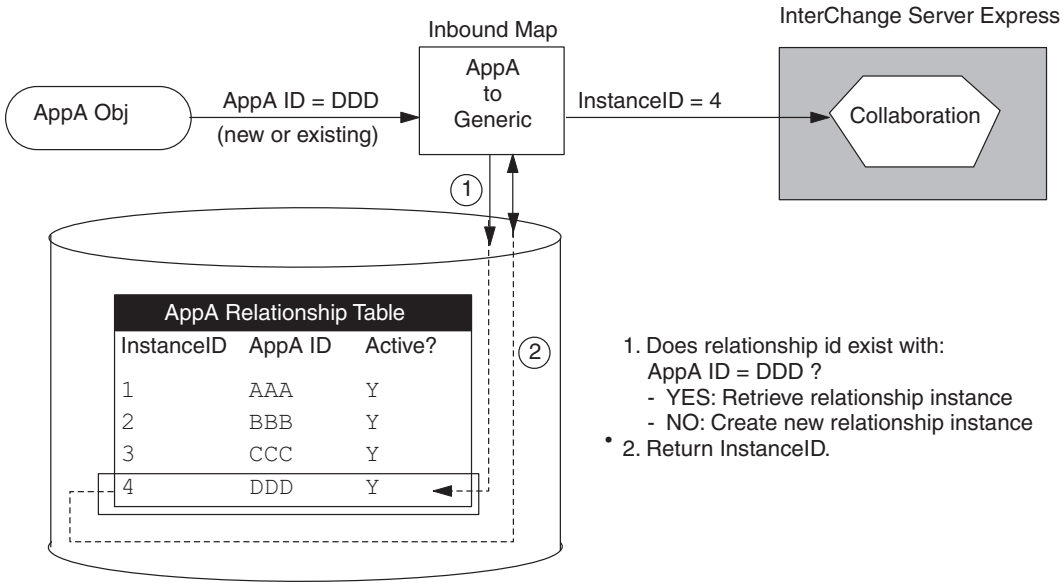


Figura 125. EVENT_DELIVERY e ACCESS_REQUEST con un verbo Aggiorna o Crea

Per un contesto di richiamo `EVENT_DELIVERY` (o `ACCESS_REQUEST`) e un verbo specifico dell'applicazione Crea o Aggiorna, Figura 126 illustra la scrittura che il codice Java generato dalla trasformazione del riferimento incrociato effettua nella

tabella di relazioni quando non esiste alcuna voce corrispondente al valore della chiave specifica dell'applicazione AppA.

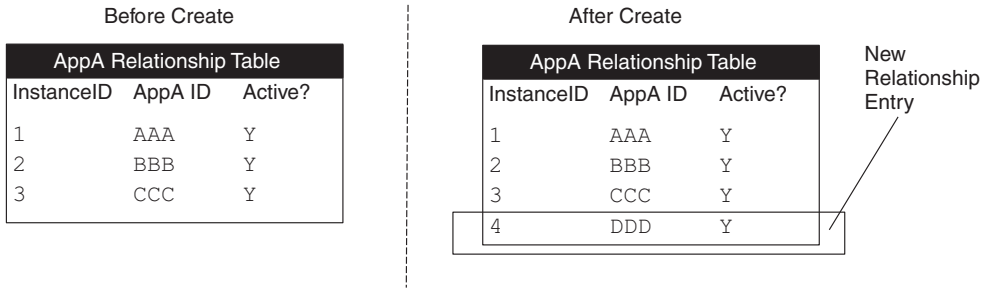


Figura 126. La scrittura nella tabella di relazione per una nuova voce di relazione

Per un contesto di richiamo EVENT_DELIVERY (o ACCESS_REQUEST) e un verbo specifico dell'applicazione Elimina, Figura 127 illustra la scrittura eseguita dal codice Java generato dalla trasformazione del riferimento incrociato nella tabella di relazione AppA.

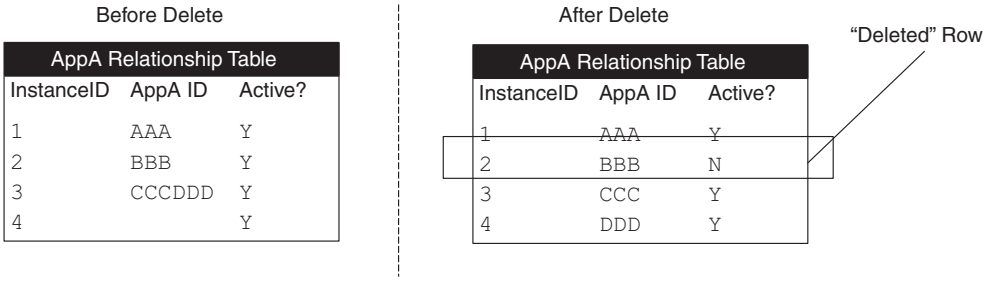


Figura 127. La scrittura nella tabella di relazione per un verbo Elimina

Contesto di richiamo SERVICE_CALL_REQUEST

Quando il contesto di richiamo è SERVICE_CALL_REQUEST, la mappa richiamata è una mappa in uscita, ovvero trasforma un oggetto business generico in un oggetto business specifico dell'applicazione. La mappa in uscita riceve un oggetto business generico come input e restituisce un oggetto business specifico dell'applicazione come output. Pertanto, l'attività per la trasformazione del riferimento incrociato è di ottenere dalla tabella di relazione un valore di chiave dell'oggetto business specifico dell'applicazione per un determinato ID di istanza *solo* se il verbo è Aggiorna, Elimina o Recupera. La trasformazione del riferimento incrociato *non* ottiene il valore di chiave specifico dell'applicazione per il verbo Crea.

Tabella 96 illustra l'azione che la trasformazione del riferimento incrociato intraprende sulla tabella di relazioni in base al verbo di un oggetto business generico.

Tabella 96. Azioni per il contesto di richiamo SERVICE_CALL_REQUEST

Verbo di un oggetto business generico	**Azione eseguita dalla trasformazione del riferimento incrociato**
Crea	Non effettuare alcuna azione. Quando il contesto di richiamo è SERVICE_CALL_RESPONSE, il metodo al momento scrive una nuova voce nella tabella di relazioni. Per ulteriori informazioni, consultare "Contesto di richiamo SERVICE_CALL_RESPONSE" a pagina 284.

Tabella 96. Azioni per il contesto di richiamo SERVICE_CALL_REQUEST (Continua)

Verbo di un oggetto business generico	**Azione eseguita dalla trasformazione del riferimento incrociato**
Aggiorna Elimina Recupera	1. Ottenere il valore di chiave dell'oggetto business generico (l'ID di istanza della relazione) dall'oggetto business della richiesta di origine dal contesto di esecuzione della mappa. 2. Recuperare la voce dalla tabella di relazioni per un determinato valore di chiave dell'oggetto business generico. Se una voce per questo valore di chiave *non* esiste, inoltrare un'eccezione `RelationshipRuntimeException`. Se non viene trovato alcun partecipante quando il verbo è Recupera, viene inoltrata un'eccezione `CxMissingIDException`. 3. Ottenere il valore di chiave specifico dell'applicazione dalla voce della relazione recuperata. 4. Copiare il valore di chiave specifico dell'applicazione nell'oggetto business specifico dell'applicazione.

Come illustrato in Tabella 96, quando il verbo è Crea, il codice Java generato dalla trasformazione del riferimento incrociato *non* scrive una nuova voce nella tabella di relazioni. Non viene eseguita questa operazione di scrittura, poiché non si dispone ancora del valore della chiave specifica dell'applicazione corrispondente all'ID di istanza della relazione. Quando il connettore elabora l'oggetto business specifico dell'applicazione, viene notificata all'applicazione la necessità di inserire una nuova riga (o righe). Se questo inserimento viene eseguito correttamente, l'applicazione invia una notifica al connettore, che crea l'oggetto business specifico dell'applicazione appropriato con un verbo Crea e il valore della chiave dell'applicazione.

Per i restanti verbi (Aggiorna, Elimina e Recupera), il codice Java generato dalla trasformazione del riferimento incrociato esegue un'operazione in lettura nella tabella di relazioni. Per una relazione di identità che supporta la trasformazione da un oggetto business specifico dell'applicazione `AppA` ad un oggetto business specifico dell'applicazione AppB, Figura 128 illustra il modo in cui la trasformazione del riferimento incrociato accede alla tabella di relazione associata al partecipante `AppB` quando un contesto di richiamo è `SERVICE_CALL_REQUESTe` il verbo dell'oggetto business generico è Aggiorna, Elimina o Recupera.

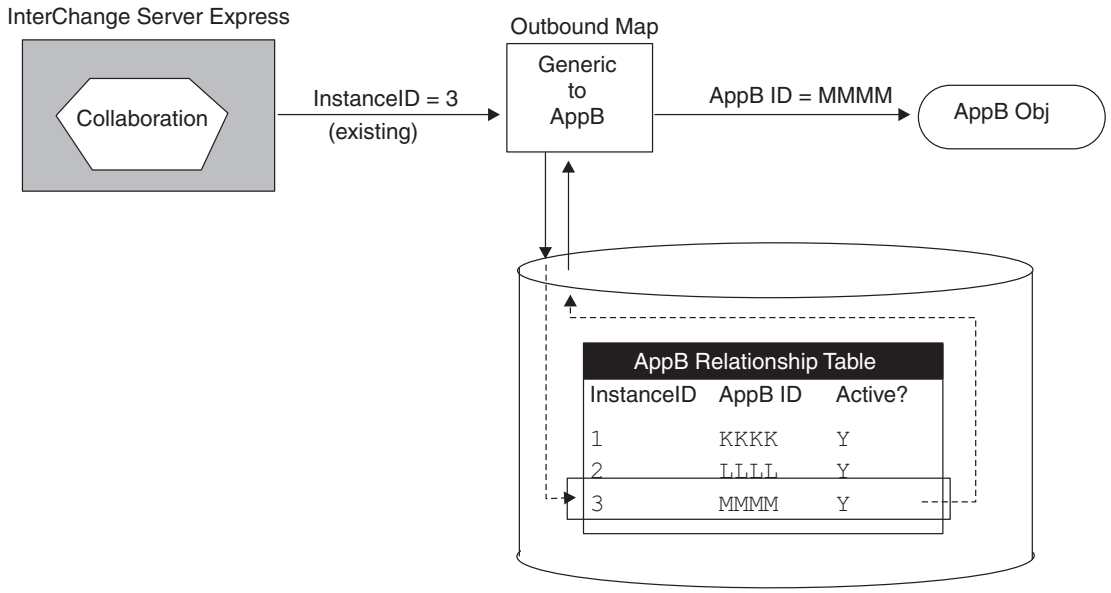


Figura 128. SERVICE_CALL_REQUEST con un verbo aggiorna, elimina o recupera

Contesto di richiamo SERVICE_CALL_RESPONSE

Quando il contesto di richiamo è `SERVICE_CALL_RESPONSE`, la mappa che viene richiamata è una mappa in entrata, ovvero trasforma un oggetto business specifico dell'applicazione in un oggetto business generico. La mappa in entrata riceve un oggetto business specifico dell'applicazione come input e restituisce un oggetto business generico come output. Il contesto di richiamo `SERVICE_CALL_RESPONSE` è importante per Crea verbo, per indicare che l'applicazione di destinazione era in grado di creare un valore univoco per la nuova entità e il connettore ha restituito un oggetto specifico dell'applicazione.

L'attività per la regola di trasformazione del riferimento incrociato è di conservare un valore di chiave dell'oggetto business specifico dell'applicazione nella tabella di relazione per un ID istanza di relazione esistente. Per il contesto di richiamo `SERVICE_CALL_RESPONSE`, il codice Java generato dalla trasformazione del riferimento incrociato intraprende le seguenti azioni:

1. Determina se l'oggetto business generico è null:
   - Per i verbi Aggiorna, Elimina e Recupera, la trasformazione inoltra `RelationshipRuntimeException` se l'oggetto business generico è null.
   - Per un verbo Crea, un oggetto business generico di valore null è valido.
2. Ricerca la voce nella tabella di relazioni che corrisponde al valore della chiave di un determinato oggetto business specifico dell'applicazione. Tabella 97 illustra l'azione che il codice Java generato dalla trasformazione del riferimento incrociato intraprende sulla tabella di relazioni in base al verbo dell'oggetto business specifico dell'applicazione.

Tabella 97. Azioni per il contesto di richiamo SERVICE_CALL_RESPONSE

Verbo dell'oggetto business specifico dell'applicazione	Azione eseguita da maintainSimpleIdentityRelationship()
Crea	Per una determinata chiave specifica dell'applicazione, inserire nella tabella di relazioni la nuova voce della relazione contenente il valore di chiave dell'oggetto business specifico dell'applicazione e il relativo ID istanza della relazione. Il metodo ottiene l'ID istanza della relazione dall'oggetto business della richiesta di origine nel contesto di esecuzione della mappa (`cwExecCtx`). Se una voce per questo valore di chiave già esiste, ripristinare quello esistente, *non* aggiungerne un altro alla tabella.
Elimina	1. Richiamare la voce della relazione dalla tabella di relazioni per un determinato valore di chiave dell'oggetto business specifico dell'applicazione. 2. Contrassegnare la voce della relazione come "disattiva."
Aggiorna	Richiamare la voce della relazione dalla tabella di relazioni per un determinato valore di chiave dell'oggetto business specifico dell'applicazione.
Recupera	Richiamare la voce della relazione dalla tabella di relazioni per un determinato valore di chiave dell'oggetto business specifico dell'applicazione.

Per una relazione di identità che supporta la trasformazione di un oggetto business specifico dell'applicazione `AppA` ad un oggetto business specifico dell'applicazione AppB, Figura 129 illustra il modo in cui il codice Java generato dalla trasformazione del riferimento incrociato accede ad una tabella di relazioni associata al partecipante `AppB` quando un contesto di richiamo è `SERVICE_CALL_RESPONSE` e il verbo dell'oggetto business specifico dell'applicazione `AppB` è Crea.

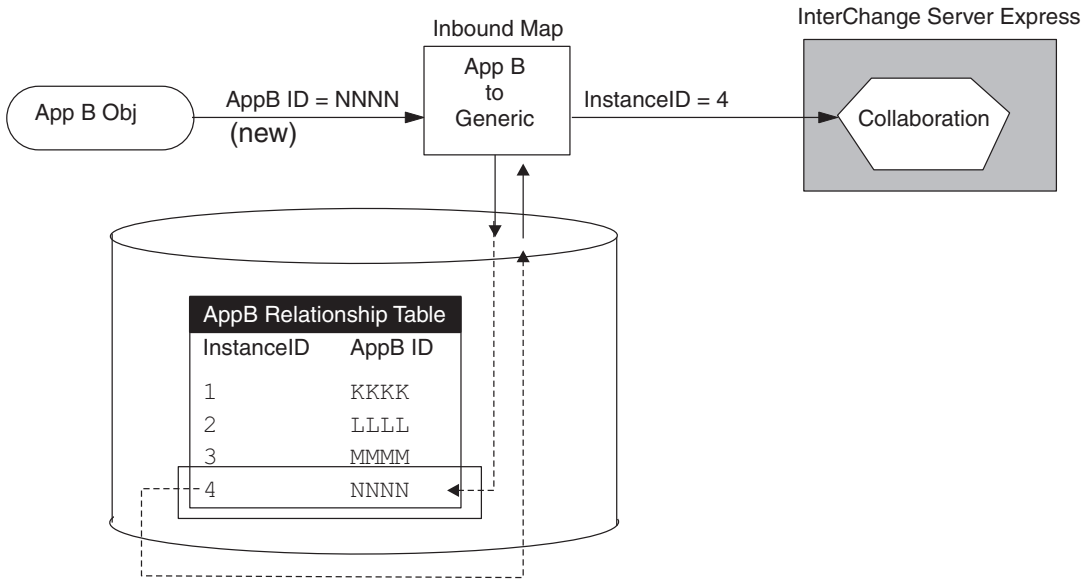


Figura 129. SERVICE_CALL_RESPONSE con verbo crea

Quando il contesto di richiamo è SERVICE_CALL_RESPONSE e il verbo è Crea, la mappa in entrata è stata richiamata dal controller del connettore in risposta alle seguenti azioni:

- Al connettore è stato notificato che l'applicazione ha inserito una nuova riga.

  Il connettore ha inviato questa richiesta di inserimento all'applicazione quando è stato ricevuto l'oggetto business specifico dell'applicazione con il verbo Crea dalla mappa in uscita. Questa mappa in uscita disponeva di un contesto di richiamo SERVICE_CALL_REQUEST. Quando il contesto di richiamo era SERVICE_CALL_REQUEST, la trasformazione del riferimento incrociato non può scrivere una nuova istanza di relazione nella tabella di relazioni, poiché non disponeva ancora del valore di chiave specifico dell'applicazione che corrispondeva all'ID di istanza.
- Il connettore ha generato un nuovo oggetto business specifico dell'applicazione in base ai valori nella nuova riga specifica dell'applicazione e con un verbo Crea.

  Il connettore invia questo oggetto business specifico dell'applicazione a InterChange Server Express dove viene ricevuto dal controller del connettore.
- Il controller del connettore ha richiamato la mappa in entrata per convertire l'oggetto business specifico dell'applicazione in un oggetto business generico.

  La mappa in entrata contiene una trasformazione del riferimento incrociato per creare una voce nella tabella di relazioni per la nuova chiave specifica dell'applicazione.

Per un contesto di richiamo SERVICE_CALL_RESPONSE e un verbo Crea specifico dell'applicazione, Figura 130 illustra la scrittura che il codice Java generato dalla trasformazione del riferimento incrociato effettua nella tabella di relazioni.

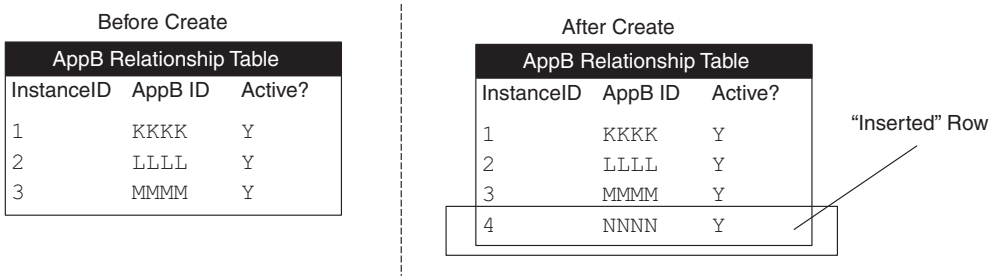


Figura 130. La scrittura nella tabella di relazioni per un verbo Crea

La trasformazione del riferimento incrociato deve associare la nuova chiave specifica dell'applicazione AppB al valore equivalente nell'applicazione AppA. Per il contesto di richiamo EVENT_DELIVERY o ACCESS_REQUEST, la trasformazione del riferimento incrociato potrebbe solo generare un nuovo ID di istanza di relazioni. Tuttavia, per SERVICE_CALL_RESPONSE, la trasformazione del riferimento incrociato non può generare un nuovo ID istanza. Invece, deve assegnare lo stesso ID istanza della relazione al valore della chiave AppB come già assegnato al valore della chiave AppA. Il metodo ottiene l'ID istanza associato al valore della chiave AppA dall'oggetto business della richiesta di origine, che è parte del contesto di esecuzione della mappa.

In Figura 130, il codice Java generato dalla trasformazione del riferimento incrociato effettua i seguenti passi per il contesto di richiamo SERVICE_CALL_RESPONSE e Crea verbo:

- Ottiene l'ID istanza di 4 dall'oggetto business della richiesta di origine nel contesto di esecuzione della mappa.

- Crea una nuova voce nella tabella di relazioni AppB per questo ID di istanza (4) e la nuova chiave specifica dell'applicazione (NNNN).

Quando le esecuzioni della mappa con entrambi i contesti di richiamo EVENT_DELIVERY (o ACCESS_REQUEST) e SERVICE_CALL_RESPONSE (e il verbo Crea) sono completati, le tabelle di relazione per AppA e AppB utilizzano ID di istanza di relazioni comuni da associare alle relative chiavi, come illustrato in Figura 131.

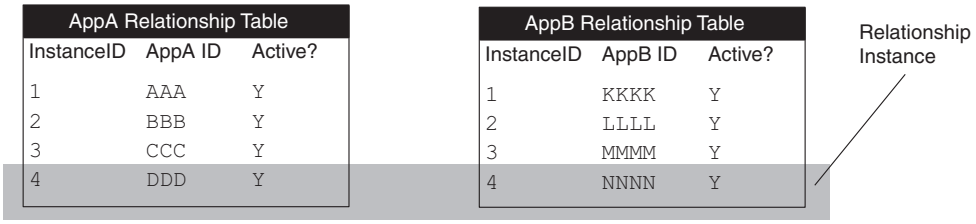


Figura 131. Creazione dell'istanza di relazioni

Per i verbi Aggiorna ed Elimina (e Recupera, se l'ID di istanza già esiste nella tabella di relazioni), la trasformazione del riferimento incrociato recupera l'ID di istanza della relazione dalla tabella di relazioni. Per un contesto di richiamo SERVICE_CALL_RESPONSE e un verbo Elimina specifico dell'applicazione, la trasformazione del riferimento incrociato deve effettuare ulteriori passi per disattivare l'istanza di relazione, come illustrato in Figura 132.

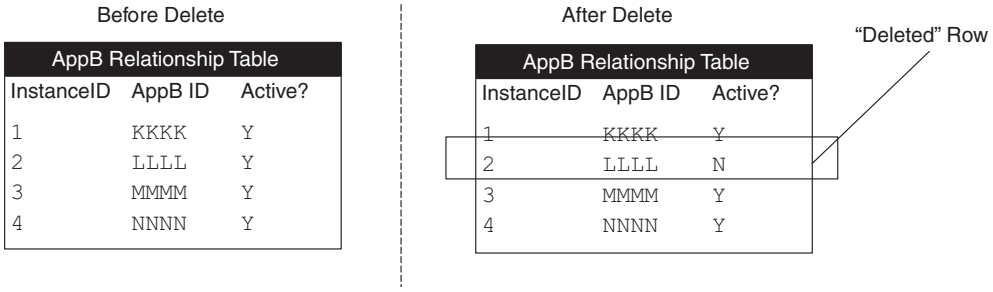


Figura 132. La scrittura nella tabella di relazioni per SERVICE_CALL_RESPONSE e un verbo Elimina

Contesto di richiamo SERVICE_CALL_FAILURE

Quando il contesto di richiamo è SERVICE_CALL_FAILURE, la mappa che sta per essere richiamata è una mappa in entrata, ovvero trasforma un oggetto business specifico dell'applicazione in un oggetto business generico. Per SERVICE_CALL_FAILURE, la mappa in entrata riceve un oggetto business specifico dell'applicazione come input e restituisce un oggetto business generico come output. Il contesto di richiamo SERVICE_CALL_FAILURE è importante per il verbo Crea, indica che l'applicazione di destinazione non è stata in grado di creare un valore univoco per la nuova entità, pertanto il connettore non era in grado di restituire un oggetto business specifico dell'applicazione. L'attività per la trasformazione del riferimento incrociato è uguale per tutti i verbi, come illustrato in Tabella 98.

Tabella 98. Azioni per il contesto di richiamo SERVICE_CALL_FAILURE

Verbo dell'oggetto business specifico dell'applicazione	Azione eseguita da maintainSimpleIdentityRelationship()
Crea Elimina Aggiorna Recupera	1. Ottenere il valore di chiave (ID istanza di relazione) dall'oggetto business generico. Tale oggetto business generico si trova nel contesto di esecuzione della mappa. 2. Copiare l'ID istanza recuperata nell'oggetto business generico.

Contesto di richiamo ACCESS_RESPONSE

Quando il contesto di richiamo è `ACCESS_RESPONSE`, la mappa che sta per essere richiamata è una mappa in uscita come risultato del flusso di trigger di un richiamo. Trasforma un oggetto business generico in un oggetto business specifico dell'applicazione. La mappa in uscita riceve un oggetto business generico come input e restituisce un oggetto business specifico dell'applicazione come output. Pertanto, l'attività per la trasformazione del riferimento incrociato è uguale per tutti i verbi, come illustrato in Tabella 99.

Tabella 99. Azioni per il contesto di richiamo ACCESS_RESPONSE

Verbo di un oggetto business generico	Azione eseguita da maintainSimpleIdentityRelationship()
Crea Elimina Aggiorna Recupera	1. Ottenere il valore di chiave (ID istanza di relazione) dall'oggetto business generico. Tale oggetto business generico si trova nel contesto di esecuzione della mappa. 2. Convertire l'ID di istanza della relazione in un valore intero. Se la conversione non riesce, viene restituita un'eccezione. 3. Copiare i valori di chiave dall'oggetto business della richiesta di origine nell'oggetto business specifico dell'applicazione.

Poiché l'oggetto business della richiesta di origine per `ACCESS_RESPONSE` è l'oggetto business specifico dell'applicazione, la trasformazione del riferimento incrociato ottiene automaticamente questo valore di chiave dall'oggetto business della richiesta di origine nel contesto di esecuzione della mappa (`cwExecCtx`).

La trasformazione del riferimento incrociato può eseguire queste attività in Tabella 99 fino a quando ha accesso all'oggetto business della richiesta di origine. Tuttavia, in alcuni casi, potrebbe non essere possibile accedere a questo oggetto business. Ad esempio, se la trasformazione del riferimento incrociato sta elaborando un oggetto child che non esisteva nella richiesta principale, il metodo prova a richiamare quell'ID istanza di relazione dell'oggetto child. Se il metodo non riesce a trovare l'istanza di relazione, popola solo le chiavi di questo oggetto child con il valore `CxIgnore`.

Definizione delle regole di trasformazione per una relazione di identità semplice

Per informazioni sulla definizione di una relazione del riferimento incrociato, consultare "Relazioni di identità con riferimento incrociato" a pagina 51.

Codifica di una relazione di identità semplice a livello di child

Se gli oggetti business child dispongono di un attributo chiave univoco, è possibile correlare questi oggetti business child in una relazione di identità semplice.

Le sezioni di seguito riportate descrivono i passi per la codifica di questa relazione di identità semplice:

- "Passi per la creazione della definizione di relazioni child" a pagina 289
- "Passi per la personalizzazione della mappa parent" a pagina 289
- "Passi per la personalizzazione della mappa secondaria"

Passi per la creazione della definizione di relazioni child: Per creare una definizione di relazione per una relazione di identità semplice tra gli oggetti business child, eseguire i passi riportati:

1. Creare una definizione del partecipante il cui tipo di partecipante è l'oggetto business child.
2. Impostare l'attributo del partecipante sull'attributo chiave dell'oggetto business child.

   Suggerimento: espandere l'oggetto business child e selezionare l'attributo chiave.
3. Ripetere i passi 1 e 2 per ciascuno dei partecipanti. Come con tutte le relazioni di identità semplici, questa relazione contiene un partecipante per l'oggetto business generico e almeno un partecipante per un oggetto business specifico dell'applicazione. Ciascun partecipante contiene un singolo attributo: la chiave dell'oggetto business.

Passi per la personalizzazione della mappa parent: Nella mappa per l'oggetto business parent (la mappa principale), aggiungere il codice di mappatura all'attributo che contiene l'oggetto business child. In Activity Editor per questo attributo, eseguire i passi riportati per codificare una relazione di identità semplice:

1. Se si crea una mappa secondaria per l'oggetto child, richiamare questa mappa secondaria dall'attributo child della mappa principale. In genere la mappatura delle trasformazioni per un oggetto child sono effettuate nell'ambito di una maschera secondaria, in particolare se l'oggetto child dispone di una cardinalità multipla.
2. Utilizzare il blocco di funzione Generale/API/Relazione di identità/Conserva verbo child per impostare i verbi degli oggetti child di origine.

   L'ultimo parametro del blocco di funzione Generale/API/Relazione di identità/Conserva verbo child è un flag booleano che indica se gli oggetti child partecipano ad una relazione composita. Assicurarsi di inoltrare un valore `false` come ultimo argomento del blocco di funzione Generale/API/Relazione di identità/Conserva verbo child, poiché questo oggetto child partecipa a una relazione di identità semplice e non composita. Se l'oggetto child dispone di una mappa secondaria, richiamare il blocco di funzione Generale/API/Relazione di identità/Conserva verbo child *prima* del richiamo della mappa secondaria. Per ulteriori informazioni, consultare "Impostazione del verbo del child di origine" a pagina 303.

Nota: Se l'attributo chiave di un oggetto business parent partecipa anche ad una relazione di identità semplice, definire una trasformazione del riferimento incrociato nella mappa principale, come descritto in "Relazioni di identità con riferimento incrociato" a pagina 51.

Passi per la personalizzazione della mappa secondaria: Nella mappa secondaria, eseguire i passi riportati:

1. Definire una trasformazione Sposta o Imposta valore per l'oggetto business child.

2. Definire una trasformazione del riferimento incrociato per l'oggetto business child e specificare il nome della relazione e il partecipante. Per ulteriori informazioni, consultare "Relazioni di identità con riferimento incrociato" a pagina 51.

Utilizzo delle relazioni di identità composite

Una relazione di identità stabilisce un'associazione tra gli oggetti business o altri dati su base *uno-a-uno*. Una relazione di identità composita correla due oggetti business mediante un attributo della chiave composita.

Le sezioni di seguito riportate descrivono i passi per le attività con le relazioni di identità composite:

- "Creazione delle definizioni della relazione di identità composita" a pagina 290
- "Determinazione dell'azione di relazione" a pagina 291
- "Personalizzazione delle regole della mappa per una relazione di identità composita" a pagina 293

Creazione delle definizioni della relazione di identità composita

Le definizioni della relazione di identità sono diverse dalle definizioni della relazione di ricerca, poiché i tipi di partecipante sono oggetti business, *non* tipi di `Dati` (la prima selezione nell'elenco dei tipi di partecipante). Come con una relazione di identità semplice:

- La relazione di identità composita è costituita dall'oggetto business generico e almeno un oggetto business specifico dell'applicazione.
- Il tipo di partecipante è un oggetto business per *tutti* i partecipanti.

Tuttavia, per una relazione di identità composita, l'attributo del partecipante per ciascun partecipante è una chiave composita. Questa chiave composita, in genere, è costituita da una chiave univoca di un oggetto business parent e una chiave non univoca di un oggetto business child.

Passi per la creazione delle definizioni di una relazione di identità composita

Per creare una definizione di relazione per una relazione di identità composita, eseguire i passi riportati:

1. Creare una definizione del partecipante il cui tipo di partecipante è l'oggetto business parent.
2. Impostare il primo attributo del partecipante sull'attributo chiave dell'oggetto business parent.

   Suggerimento: espandere l'oggetto business parent e selezionare l'attributo chiave.
3. Impostare il secondo attributo del partecipante sulla chiave dell'attributo child.

   Suggerimento: espandere l'oggetto business parent, quindi espandere l'attributo child all'interno del parent. Selezionare l'attributo chiave dall'oggetto child.
4. Ripetere i passi 1-3 per ciascuno dei partecipanti. Come con tutte le relazioni di identità composite, questa relazione contiene un partecipante per l'oggetto business generico e almeno un partecipante per un oggetto business specifico dell'applicazione. Ciascun partecipante è costituito da due attributi: la chiave

dell'oggetto business parente e la chiave dell'oggetto business child (dell'attributo all'interno dell'oggetto business parent).

Limitazione: Per gestire le relazioni composite, il server crea delle tabelle interne. Una tabella viene creata per ciascun ruolo nella relazione. Un *indice* univoco viene creato in tali tabelle tra tutti gli *attributi chiave* della relazione. (In altre parole, le colonne che corrispondono agli attributi chiave della relazione sono partecipanti dell'indice). Le dimensioni di colonna delle tabelle interne hanno una relazione diretta con gli attributi della relazione e sono determinati dal valore dell'attributo MaxLength per la relazione.

In genere, i database dispongono di limitazioni sulla dimensione degli indici che possono essere creati. Ad esempio, DB2 dispone di una limitazione di indice di 1024 byte con la dimensione di pagina predefinita. Pertanto, in base all'attributo MaxLength di una relazione e al numero degli attributi di una relazione, è possibile eseguire una limitazione in una dimensione di indice durante la creazione delle relazioni composite.

Importante:

- E' necessario assicurarsi che i valori MaxLength siano impostati nel file del repository per tutti gli *attributi chiave* delle limitazioni di dimensione indice del DBMS sottostante.

  Se l'attributo MaxLength per il tipo Stringa non è specificato, il valore predefinito è nvarchar(255) in SQLServer. Pertanto, se una relazione ha *N* chiavi, tutti i tipi Stringa e l'attributo MaxLength predefinito di 255 byte, la dimensione di indice sarebbe ((N*255)*2) + 16 byte. Si noti che potrebbe essere facilmente superato il limite SQLServer 7 di 900 byte quando N assume il valore di >=2 per il valore MaxLength predefinito di 255 byte per il tipo Stringa.
- Inoltre, tenere presente che anche quando alcuni DBMS supportano indici di ampie dimensioni, le prestazioni non sono più ottimali, quindi è sempre bene tenere le dimensioni di indice al minimo.

Per ulteriori informazioni sulla creazione di una definizione di relazione per una relazione di identità composita, consultare "Definizione delle relazioni di identità" a pagina 257.

Determinazione dell'azione di relazione

Tabella 100 illustra i blocchi di funzione dell'attività forniti dalla mappatura dell'API per conservare una relazione di identità composita dall'attributo child dell'oggetto di origine parent. Le azioni che possono essere intraprese da questi metodi dipendono dal verbo dell'oggetto di origine e dal contesto di richiamo.

Tabella 100. Conservazione di una relazione di identità composita dall'attributo child

Blocco di funzione	Descrizione
Generale/API/Relazione di identità/ Conservazione del verbo child	Impostare correttamente il verbo child di origine
Generale/API/Relazione di identità/ Conservare la relazione composita	Eseguire le azioni appropriate nelle tabelle di relazioni

Azioni di Generale/API/Relazione di identità/Conserva relazione composita

Il blocco di funzione Conserva relazione composita genera un codice Java che richiama la mappatura degli API `maintainCompositeRelationship()`, che gestiscono le tabelle di relazione per una relazione di identità composita. Questo metodo

assicura che le istanze della relazione contengono i valori chiave specifici dell'applicazione associati per ciascun ID di istanza della relazione. Questo metodo gestisce automaticamente tutte le aggiunte e le eliminazioni di base dei partecipanti e le istanze di relazione per una relazione di identità composita.

Le azioni che `maintainCompositeRelationship()` effettua sono basate sul valore del verbo dell'oggetto business e del contesto di richiamo. Il metodo viene iterato mediante gli oggetti child di un determinato partecipante, richiamando `maintainSimpleIdentityRelationship()` su ciascuno per impostare correttamente il valore di chiave del child. Come con `maintainSimpleIdentityRelationship()`, l'azione che `maintainCompositeRelationship()` intraprende si basa sulle seguenti informazioni:

- Il contesto di richiamo: `EVENT_DELIVERY`, `ACCESS_REQUEST`, `SERVICE_CALL_REQUEST`, `SERVICE_CALL_RESPONSE`, `SERVICE_CALL_FAILURE` e `ACCESS_RESPONSE`
- Il verbo dell'oggetto business di origine: Crea, Aggiorna, Elimina o Recupera

Per informazioni sulle azioni effettuate da `maintainSimpleIdentityRelationship()`, consultare `"Accesso alle tabelle di relazione di identità" a pagina 278`.

Il metodo `maintainCompositeRelationship()` è in relazione *solo* con le chiavi composite che si estendono solo ai due livelli nidificati. In altre parole, il metodo non può gestire il caso in cui la chiave composita dell'oggetto child dipende dal valore dei relativi oggetti parent del parent.

Esempio: se A è un oggetto business di livello superiore, B è il child di A e C è il child di B, i due metodi *non* supportano le definizioni del partecipante per l'oggetto child C, che sono le seguenti:

- Il tipo di partecipante A e gli attributi sono:

```
attributo chiave di A: ID
attributo chiave di B: B[0].ID
attributo chiave di C: B[0].C[0].ID
```

- Il tipo di partecipante A e gli attributi sono:

```
attributo chiave di A: ID
attributo chiave di C: B[0].C[0].ID
```

Per accedere ad un oggetto child del child, tali metodi supportano solo le definizioni del partecipante di seguito riportate:

- Il tipo di partecipante B e gli attributi sono:

```
attributo chiave di B: ID
attributo chiave di C: C[0].ID
```

- Il tipo di partecipante B e gli attributi sono:

```
attributo chiave di B: ID
primo attributo chiave di C: C[0].ID1
secondo attributo chiave di C: C[0].ID2
```

Azioni di Generale/API/Relazione di identità/Conserva verbo child

Il blocco di funzione Conserva verbo child genera il codice Java che richiama la mappatura dell'API `maintainChildVerb()`, che conserva il verbo degli oggetti child nell'oggetto business di destinazione. Può gestire oggetti child i cui attributi di chiave sono parte di una relazione di identità composita. Quando si richiama `maintainChildVerb()` come parte di una relazione composita, assicurarsi che l'ultimo parametro disponga del valore `true`. Questo metodo assicura che le impostazioni del verbo siano appropriate dato il verbo nell'oggetto di origine

parent e il contesto di richiamo. Per ulteriori informazioni sulle azioni di `maintainChildVerb()`, consultare “Impostazione del verbo del child di origine” a pagina 303.

Personalizzazione delle regole della mappa per una relazione di identità composita

Una volta creata la definizione della relazione e le definizioni del partecipante per la relazione di identità composita, è possibile personalizzare la mappa per conservare la relazione di identità composita. Una relazione di identità composita gestisce una chiave composita. Pertanto, la gestione di questo tipo di relazione coinvolge la gestione di entrambe le parti della chiave composita. Per codificare una relazione di identità composita, è necessario personalizzare le regole di trasformazione della mappatura per entrambi gli oggetti business parent e child, come illustrato in Tabella 101.

Tabella 101. Blocchi delle funzioni di attività per una relazione di identità composita

Mappa coinvolta	Oggetto business coinvolto	Attributo	Blocchi di funzione dell'attività
Principale	Oggetto business parent	Oggetto business di livello superiore	Utilizzare una regola di trasformazione del riferimento incrociato
		Attributo child (oggetto business child)	Generale/API/Relazione di identità/Conserva composita Relazione Generale/API/Relazione di identità/Conserva verbo child Generale/API/Relazione di identità/Aggiorna i child (facoltativo)
Mappa secondaria	Business child oggetto	Attributo chiave (chiave non univoca)	Definire una trasformazione Sposta o Imposta valore per il verbo.

Se gli oggetti business child dispongono di un attributo chiave non univoco, è possibile correlare questi oggetti business child in una relazione di identità composita.

Le sezioni di seguito riportate descrivono i passi per la personalizzazione di questa relazione di identità composita:

- “Passi per la personalizzazione della mappa principale”
- “Personalizzazione della mappa secondaria” a pagina 298
- “Gestione delle istanze child” a pagina 298

Passi per la personalizzazione della mappa principale

Nella mappa per l'oggetto business parent (la mappa principale), aggiungere il codice di mappatura agli attributi parent:

1. Mappare il verbo dell'oggetto business di livello superiore definendo una regola di trasformazione Sposta o Imposta valore.
2. Definire una trasformazione del riferimento incrociato tra gli oggetti business di livello superiore.
3. Definire una trasformazione personalizzata per l'attributo child ed utilizzare il blocco di funzione Generale/API/Relazione di identità/Conserva relazione composita in Activity Editor.

Passi per la codifica dell'attributo child

L'attributo child dell'oggetto parent contiene l'oggetto business child. Questo oggetto child è in genere un oggetto business con cardinalità multipla. Contiene un

attributo chiave il cui valore identifica il child. Tuttavia, non è richiesto che tale valore di chiave sia univoco. Pertanto, non identifica in modo univoco un oggetto child tra quelli per lo stesso parent né è sufficiente per identificare l'oggetto child tra gli oggetti child per tutte le istanze dell'oggetto parent.

Per identificare tale oggetto child in modo univoco, la relazione utilizza una chiave composita. Nella chiave composita, la chiave parent identifica in modo univoco l'oggetto parent. La combinazione di chiave parent e chiave child identifica in modo univoco l'oggetto child. Nella mappa per l'oggetto business parent (la mappa principale), aggiungere il codice di mappatura all'attributo che contiene l'oggetto business child. In Activity Editor per questo attributo, eseguire i passi riportati per codificare una relazione di identità composita:

1. Definire una trasformazione di Mappa secondaria per l'attributo dell'oggetto business child della mappa principale. In genere la mappatura delle trasformazioni per un oggetto child sono effettuate nell'ambito di una maschera secondaria, in particolare se l'oggetto child dispone di una cardinalità multipla.
2. Nella mappa principale, definire una regola di trasformazione personalizzata per il verbo child ed utilizzare il blocco di funzione Generale/API/Relazione di identità/Conserva verbo child per conservare il verbo dell'oggetto business child.

   L'ultimo parametro di input del blocco di funzione Generale/API/Relazione di identità/Conserva verbo child è un flag booleano che indica se gli oggetti child partecipano ad una relazione composita. Assicurarsi di inoltrare un valore `true` come ultimo argomento di `maintainChildVerb()`, poiché questo oggetto child partecipa ad una relazione di identità composita, non semplice. Assicurarsi di richiamare `maintainChildVerb()` *prima* del codice che richiama la mappa secondaria. Per ulteriori informazioni, consultare "Impostazione del verbo del child di origine" a pagina 303.
3. Per conservare questa chiave composita per l'oggetto di origine parent, personalizzare la regola di mappatura per utilizzare il blocco di funzione Generale/API/Relazione di identità/Conserva relazione composita.
4. Per conservare le tabelle di relazione nel caso in cui un oggetto parent dispone di un verbo Aggiorna causato dagli oggetti child eliminati, personalizzare la regola di mappatura per utilizzare il blocco di funzione Generale/API/Relazione di identità/Aggiorna i child.

   Suggerimento: assicurarsi che la regola di trasformazione contenente il blocco di funzione Aggiorna i child disponga di un ordine di esecuzione dopo la regola di trasformazione che contiene il blocco di funzione Conserva relazione composita.

Esempio di personalizzazione della mappa per una relazione di identità composita

L'esempio di seguito riportato descrive il modo in cui personalizzare la mappa per una Relazione di identità composita.

1. Nella mappa principale, definire una regola di trasformazione personalizzata tra i verbi dell'oggetto business child. Utilizzare il blocco di funzione Generale/API/Relazione di identità/Conserva verbo child nell'attività personalizzata per conservare il verbo per gli oggetti business child.

   Lo scopi di questa attività personalizzata è di utilizzare l'API maintainChildVerb() per impostare il verbo dell'oggetto business child in base al contesto di esecuzione della mappa e al verbo dell'oggetto business parent. Figura 133 a pagina 295 illustra questa attività personalizzata.

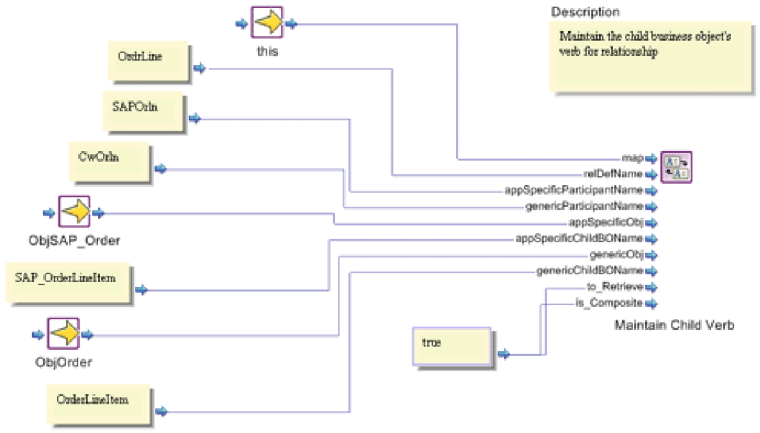


Figura 133. Utilizzo del blocco di funzione Conserva verbo child

2. Se necessario, definire una regola di trasformazione della mappa secondaria tra l'oggetto business child per eseguire qualunque mappatura sia necessaria a livello di child.
3. Definire una regola di trasformazioni personalizzata tra gli oggetti business di livello superiore. Utilizzare il blocco di funzione Generale/API/Relazione di identità/Conserva relazione composita nell'attività personalizzata per conservare la relazione di identità composita per questa mappa.

   L'obiettivo di questa attività personalizzata è di utilizzare l'API maintainCompositeRelationship() per conservare una relazione di identità composita all'interno della mappa. Figura 134 illustra questa attività personalizzata.

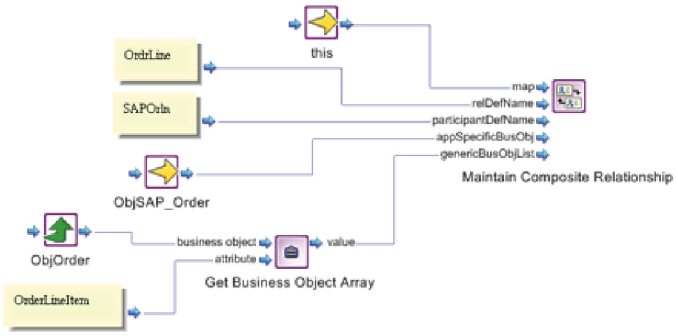


Figura 134. Utilizzo del blocco di funzione Conserva relazione composita

4. Definire una regola di trasformazione personalizzata effettuando una mappatura dall'oggetto business di livello superiore di origine all'attributo dell'oggetto business child di destinazione. Utilizzare il blocco di funzione Generale/API/Relazione di identità/Aggiorna i child nell'attività personalizzata per conservare le istanze del child nella relazione.

   L'obiettivo di questa attività personalizzata è di utilizzare l'API updateMyChildren() per aggiungere o eliminare le istanze del child nella relazione parent/child specificata della relazione di identità. Figura 135 a pagina 296 illustra questa attività personalizzata.

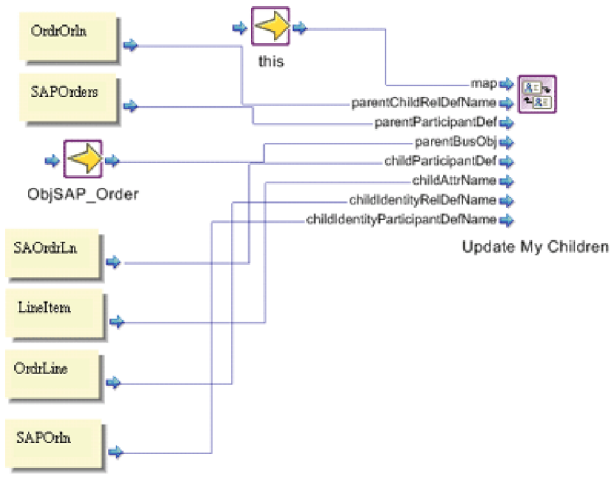


Figura 135. Utilizzo del blocco di funzione Aggiorna i child

Esempio di codifica dell'attributo child

Di seguito è illustrato un esempio di visualizzazione del codice dell'attributo child della mappa parent. Questo frammento di codice esiste nell'attributo Ordina voce riga di un oggetto business Ordine SAP. Utilizza `maintainChildVerb()` per impostare i verbi dell'oggetto child, quindi richiama una mappa secondaria `Sub_SaOrderLieItem_to_CwOrderLineItem`) in un loop `for` per gestire la mappatura dell'oggetto child Ordina voci della riga:

```
{
BusObjArray srcCollection_For_ObjSAP_Order_SAP_OrderLineItem =
   ObjSAP_Order.getBusObjArray("SAP_OrderLineItem");

//
// LOOP ONLY ON NON-EMPTY ARRAYS
// -----------------------------
//
// Perform the loop only if the source array is non-empty.
//
if ((srcCollection_For_ObjSAP_Order_SAP_OrderLineItem != null) &&
      (srcCollection_For_ObjSAP_Order_SAP_OrderLineItem.size() > 0))
   {
   int currentBusObjIndex_For_ObjSAP_Order_SAP_OrderLineItem;
   int lastInputIndex_For_ObjSAP_Order_SAP_OrderLineItem =
   srcCollection_For_ObjSAP_Order_SAP_OrderLineItem.getLastIndex();

   // ----
   IdentityRelationship.maintainChildVerb(
      "OrdrLine",
      "SAPOrln",
      "CWOrln",
      ObjSAP_Order,
      "SAP_OrderLineItem",
      ObjOrder,
      "OrderLineItem",
      cwExecCtx,
      true,
      true);

   // ----
   for (currentBusObjIndex_For_ObjSAP_Order_SAP_
         OrderLineItem = 0;
         currentBusObjIndex_For_ObjSAP_Order_SAP_OrderLineItem <=
            lastInputIndex_For_ObjSAP_Order_SAP_OrderLineItem;
```

```
        currentBusObjIndex_For_ObjSAP_Order_SAP_OrderLineItem++)
     {
     BusObj currentBusObj_For_ObjSAP_Order_SAP_OrderLineItem =
(BusObj) (srcCollection_For_ObjSAP_Order_SAP_OrderLineItem.elementAt(
        currentBusObjIndex_For_ObjSAP_Order_SAP_OrderLineItem));

     //
     // INVOKE MAP ON VALID OBJECTS
     // ---------------------------
     //
     // Invoke the map only on those children objects that meet
     // certain criteria.
     //
     if (currentBusObj_For_ObjSAP_Order_SAP_OrderLineItem != null)
        {
        BusObj[] _cw_inObjs = new BusObj[2];
        _cw_inObjs[0] =
              currentBusObj_For_ObjSAP_Order_SAP_OrderLineItem;
        _cw_inObjs[1] = ObjSAP_Order;
        logInfo ("*** Inside SAPCW header, verb is: " +
           (_cw_inObjs[0].getVerb()));

        try
           {
           BusObj[] _cw_outObjs = DtpMapService.runMap(
              "Sub_SaOrderLineItem_to_CwOrderLineItem",
              "CwMap",
              _cw_inObjs,
              cwExecCtx);
           _cw_outObjs[0].setVerb(_cw_inObjs[0].getVerb());
           ObjOrder.setWithCreate("OrderLineItem", _cw_outObjs[0]);
           }

        catch (MapNotFoundException me)
           {
           logError(5502,
              " Sub_SaOrderLineItem_to_CwOrderLineItem ");
           throw new MapFailureException ("Submap not found");
           }
        }
     }

  // Start of the child relationship code
  BusObjArray temp = (BusObjArray)ObjOrder.get("OrderLineItem");
  try
     {
     IdentityRelationship.maintainCompositeRelationship(
        "OrdrLine",
        "SAPOrln",
        ObjSAP_Order,
        temp,
        cwExecCtx);
     }

  catch RelationshipRuntimeException re
     {
     logError(re.toString());
     }

  // This call to updateMyChildren() assumes the existence of the
  // OrdrOrln parent/child relationship between the SAP_Order
  // (parent) and SAP_OrderItem (child)
  IdentityRelationship.updateMyChildren(
     "OrdrOrln",
     "SAOrders",
     ObjSAP_Order,
     "SAOrdrLn",
```

```
      "LineItem",
      "OrdrLine",
      "SAPOrln",
      cwExecCtx);

   // End of the child relationship code
   }
}
```

Personalizzazione della mappa secondaria

Nella mappa per l'oggetto business child (la mappa secondaria), aggiungere il codice di mappatura all'attributo chiave dell'oggetto child. Il solo codice da aggiungere è un richiamo al `setVerb()` metodo per impostare il verbo dall'oggetto child al verbo dell'oggetto parent. Per ulteriori informazioni, consultare "Impostazione del verbo dell'oggetto business di destinazione" a pagina 39.

Nota: Quando la chiave principale dell'oggetto child richiede il metodo `maintainCompositeRelationship()`, effettuare il richiamo nella mappa parent, appena dopo la fine del loop `for` per il richiamo della mappa secondaria. Nella mappa secondaria, il codice per la chiave principale dell'oggetto di destinazione deve contenere la seguente riga:

```
// maintainCompositeRelationship()
è richiamato nella mappa parent.
```

Gestione delle istanze child

Activity Editor fornisce i blocchi di funzione in Tabella 102 per gestire le istanze dell'oggetto child che appartengono a un parent in una relazione di identità.

Tabella 102. Blocchi di funzione per la gestione delle istanze del child

Blocco di funzione	Descrizione
Generale/API/Rapporto identità/Aggiungi a Secondari personali	Aggiunge le istanze della relazione child alle tabelle di relazione parent/child
Generale/API/Relazione identità/Elimina Secondari personali	Elimina le istanze della relazione child alle tabelle di relazione parent/child
Generale/API/Relazione identità/Aggiorna Secondari personali	Elimina o aggiunge le istanze di relazioni child dalle tabella di relazione parent/child.

Nota: L'utilizzo più comune dei blocchi di funzione in Tabella 102 è di conservare gli oggetti business child nelle relazioni personalizzate che intessano le relazioni di identità composite.

I blocchi di funzione in Tabella 102 suppongono che l'oggetto business parent inoltrato è un dopo immagine, ovvero l'immagine dell'oggetto business *dopo* che l'operazione del verbo è stata effettuata. Ad esempio, se un oggetto business dispone di un verbo Aggiorna con l'aggiornamento causato dall'aggiunta di nuovi oggetti child, tali nuovi oggetti child già esistono nell'oggetto business. In modo simile, se un oggetto business dispone di un verbo Aggiorna con l'aggiornamento causato dall'eliminazione degli oggetti child, l'oggetto business ha già questi oggetti child eliminati.

Le sezioni di seguito riportate descrivono i passi per la gestione delle istanze child:

- "Creazione di una definizione di relazione parent/child" a pagina 299

- "Gestione degli aggiornamenti all'oggetto business parent"

Creazione di una definizione di relazione parent/child

Una *relazione parent/child* è una relazione da 1-a-molti tra gli oggetti business parent (1) e child (molti). Una relazione parent/child interessa i seguenti partecipanti:

- Un partecipante contenente l'attributo di chiave dell'oggetto business parent
- Un partecipante contenente la chiave dell'oggetto business child

Le tabelle di relazione per una relazione parent/child abilita i blocchi di funzione in Tabella 102 per la traccia degli oggetti business child associati ad un determinato oggetto business parent.

Passi per la creazione della definizione della relazione parent/child

Per creare una definizione di relazione per una relazione parent/child, effettuare i seguenti passi in Relationship Designer Express:

1. Creare una definizione del partecipante il cui tipo di partecipante è l'oggetto business parent.
2. Impostare l'attributo del partecipante sull'attributo di chiave dell'oggetto business parent:

   Espandere l'oggetto business parent e selezionare l'attributo chiave.
3. Creare una definizione del partecipante il cui tipo di partecipante è l'oggetto business child.
4. Impostare l'attributo del partecipante sulla chiave dell'attributo child:

   Espandere l'oggetto business child (non l'attributo child con l'oggetto parent) e selezionare l'attributo di chiave da questo oggetto child.

Nota: La relazione parent-child necessita di essere conservata solo se l'oggetto child *non* dispone di una chiave univoca, ovvero l'oggetto child esiste *solo* all'interno del contesto del relativo parent.

Per ulteriori informazioni, consultare "Definizione delle relazioni di identità" a pagina 257.

Gestione degli aggiornamenti all'oggetto business parent

Questa sezione fornisce informazioni per assicurare che gli oggetto child che partecipano ad una relazione di identità composita siano correttamente gestiti durante un Aggiornamento:

- "Confronto delle immagini prima e dopo"
- "Suggerimenti sull'utilizzo di Aggiorna i child" a pagina 300

Confronto delle immagini prima e dopo

La funzione di blocco Aggiorna i child aggiorna le tabelle di relazione per una relazione parent/child. Una relazione parent/child è necessaria per determinare se gli oggetti child sono stati aggiunti o eliminati da un oggetto business parent.

Per un determinato oggetto business parent, questo metodo assicura la corrispondenza delle seguenti immagini dell'oggetto business:

- Le informazioni prima dell'immagine sono contenute nelle tabella di relazione per la relazione parent/child.
- Le informazioni dopo immagine sono contenute nell'oggetto business parent.

Affinché la mappa rilevi che un oggetto business è stato eliminato, deve determinare quante istanze dell'oggetto business child di questo tipo aveva l'oggetto business partente prima dell'aggiornamento (prima dell'immagine) e confrontarle a quelle che al momento sono presenti nell'oggetto parent (dopo immagine). La mappa può utilizzare il blocco di funzione Aggiorna i child per effettuare questo confronto e ricercare ciò che è stato aggiunto o eliminato.

Quando Aggiorna i child confronta prima e dopo l'immagine, determina se rimuovere le istanze di relazione associate dalla tabella di relazioni per qualunque oggetto child che *non* è presente nell'oggetto business parent. Il metodo rimuove le istanze di relazione dalle seguenti tabelle di relazione:

- La tabella di relazione per il partecipante del child nella relazione parent/child
- La tabella di relazione per il partecipante nella relazione di identità composita contenente gli oggetti parent e child

Nota: Sebbene Aggiorna i child aggiunga anche le istanze alla tabella di relazioni per qualunque oggetto child che *è* presente nell'oggetto business parent (ma non nella tabella di relazioni child), non è necessario effettuare tale operazioni quando si esegue il richiamo nel contesto di una relazione di identità composita. Tutti i nuovi oggetti child per l'oggetto parent sono stati già aggiunti alle tabelle di relazione mediante il blocco di funzione Conserva relazione composita. Per ulteriori informazioni, consultare "Azioni di Generale/API/Relazione di identità/Conserva relazione composita" a pagina 291.

Suggerimenti sull'utilizzo di Aggiorna i child

Quando si utilizza il blocco di funzione Aggiorna i child per conservare le tabelle di relazioni per un oggetto child interessato in una relazione di identità composita, tenere presente i suggerimenti di seguito riportati:

- Assicurarsi di utilizzare il blocco di funzione Aggiorna i child *dopo* la funzione di blocco Conserva relazione composita e di aver impostato i verbi appropriati negli oggetti business child.
- Il blocco di funzione Aggiorna i child è necessario solo per tracciare gli oggetti child interessati nelle relazioni composite.

  Non è necessario utilizzare il blocco di funzione Aggiorna i child per tracciare gli oggetti child interessati in una relazione di identità semplice. Per ulteriori informazioni, consultare "Codifica di una relazione di identità semplice a livello di child" a pagina 288.
- Il blocco di funzione Aggiorna i child (come il blocco di funzione Conserva relazione composita) interessa solo le chiavi composite che si estendono solo ai due livelli nidificati: il parent e gli immediati child.

  In altre parole, il metodo non può gestire il caso in cui la chiave composita dell'oggetto child del child dipende dal valore dei relativi oggetti parent del parent. Ad esempio, se A è un oggetto business di livello superiore, B è il child di A e C è il child di B, i due metodo non supportano le definizioni del partecipante per l'oggetto child C che sono le seguenti:
  - Il tipo di partecipante A e gli attributi sono:

    ```
    attributo chiave di A: ID
    attributo chiave di B: B[0].ID
    attributo chiave di C: B[0].C[0].ID
    ```
  - Il tipo di partecipante A e gli attributi sono:

    ```
    attributo chiave di A: ID
    attributo chiave di C: B[0].C[0].ID
    ```

Per accedere ad un oggetto child del child, tali metodi supportano solo le definizioni del partecipante di seguito riportate:

– Il tipo di partecipante B e gli attributi sono:

```
attributo chiave di B: ID
attributo chiave di C: C[0].ID
```

– Il tipo di partecipante B e gli attributi sono:

```
attributo chiave di B: ID
primo attributo chiave di C: C[0].ID1
secondo attributo chiave di C: C[0].ID2
```

- La funzione di blocco Aggiorna i child gestisce le tabelle di relazioni parent/child per EVENT_DELIVERY e i contesti di richiamo SERVICE_CALL_RESPONSE *solo*.

  Esecuzione del blocco di funzione Aggiorna i child con un contesto di richiamo SERVICE_CALL_REQUEST o ACCESS_RESPONSE *non* produce eventuali modifiche a queste tabelle di relazioni.
- Inoltre, il blocco di funzione Aggiorna i child può essere utilizzato quando l'oggetto business child dispone di un ID univoco, ovvero l'oggetto child partecipa ad una relazione di identità semplice. In questo caso, è necessario definire la relazione parent/child (consultare "Creazione di una definizione di relazione parent/child" a pagina 299).

Impostazione del verbo

Questa sezione fornisce informazioni sull'impostazione del verbo di un oggetto business che partecipa alla mappa:

- "Impostazione condizionata del verbo di destinazione"
- "Impostazione del verbo del child di origine" a pagina 303

Per informazioni sull'impostazione del verbo dell'oggetto business di destinazione, consultare "Impostazione del verbo dell'oggetto business di destinazione" a pagina 39.

Impostazione condizionata del verbo di destinazione

In genere, si imposta il verbo di destinazione sul valore del verbo di origine definendo una trasformazione di spostamento. (Per ulteriori informazioni su questa azione, consultare "Impostazione del verbo dell'oggetto business di destinazione" a pagina 39.) Tuttavia, a volte l'applicazione di origine imposta il verbo dell'oggetto business in modo non abituale, ad esempio il verbo viene impostato su Aggiorna anche se l'evento è nuovo. Un altro esempio è quando il verbo è sempre impostato su Recupera. In tali situazioni, la mappa deve reimpostare il verbo di destinazione su quello che corrisponde all'evento corrente.

Se la chiave dell'oggetto business di origine partecipa ad una relazione, la mappa può eseguire una *ricerca statica* nella tabella di relazioni per determinare se l'oggetto business di origine esiste. Quindi, la mappa può impostare il verbo di destinazione su Aggiorna o su Crea in base alla voce corrispondente trovata nella tabella. Questa ricerca statica viene eseguita allo stesso modo dell'accesso ad una relazione di ricerca. Tabella 103 illustra il blocco di funzione per utilizzare ciascun tipo di ricerca statica.

Tabella 103. Verifica dell'esistente dell'oggetto business di origine

Tipo di oggetto business di origine	Tipo di mappa	Blocco di funzione
Specifico dell'applicazione	In entrata	Generale/API/Relazione/ Recupera istanza
Generico	In uscita	Generale/API/Relazione/ Recupera partecipanti

Esempio dei passi per la personalizzazione della mappa in entrata

L'esempio di seguito riportato illustra il modo in cui una mappa in entrata può impostare in modo condizionato il verbo di destinazione in base al risultato di una ricerca:

1. Nella mappa, definire una trasformazione Personalizzata tra l'oggetto business di origine e il verbo di destinazione.
2. Nell'attività di questa trasformazione personalizzata, eseguire i passi indicati di seguito. L'obiettivo di questa attività è di identificare il numero di istanze nel partecipante della relazione. Se non sono presenti istanze del partecipante alla relazione, il verbo dell'oggetto business di destinazione deve essere Crea, altrimenti dovrebbe essere Aggiorna.
   a. Definire l'attività, come illustrato in Figura 136, per identificare il numero di istanze nel partecipante della relazione.

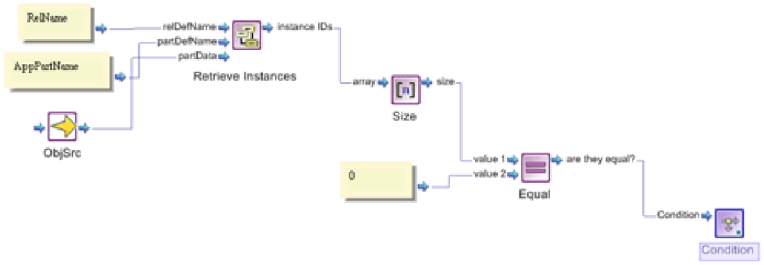


Figura 136. Identificazione del numero di istanze nel partecipante della relazione

   b. Fare doppio clic sul blocco di funzione della condizione nell'area di disegno per aprirla. Selezionare Azione true per definire l'azione da effettuare quando la condizione è true. Definire l'azione true come illustrato in Figura 137.

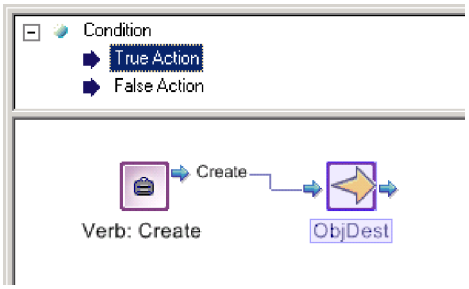


Figura 137. Definizione dell'azione true

   c. Selezionare l'azione false per definire l'azione da effettuare quando il numero di istanze del partecipante non è uguale a zero. Definire l'azione

false come illustrato in Figura 138.

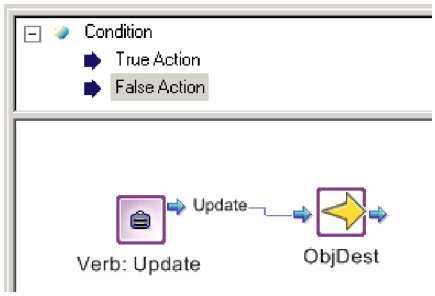


Figura 138. Definizione dell'azione false

Esempio dei passi per la personalizzazione della mappa in uscita

E' possibile utilizzare passi simili nella mappa in uscita per eseguire una ricerca statica in base alla chiave principale dell'oggetto generico. Per effettuare tale operazione, è necessario sostituire il blocco di funzione Generale/API relazione/Recupera istanze con il blocco di funzione Generale/API relazione/Recupera partecipanti. Di seguito sono riportati i passi:

1. Nella mappa, definire una trasformazione Personalizzata tra l'attributo chiave dell'oggetto business di origine e il verbo di destinazione.
2. Nell'attività di questa trasformazione personalizzata, eseguire i passi indicati di seguito. L'obiettivo di questa attività è di identificare il numero di partecipanti della relazione. Se non sono presenti istanze del partecipante alla relazione, il verbo dell'oggetto business di destinazione deve essere Crea, altrimenti dovrebbe essere Aggiorna.
   a. Definire l'attività, come illustrato in Figura 139, per identificare il numero di partecipanti della relazione.

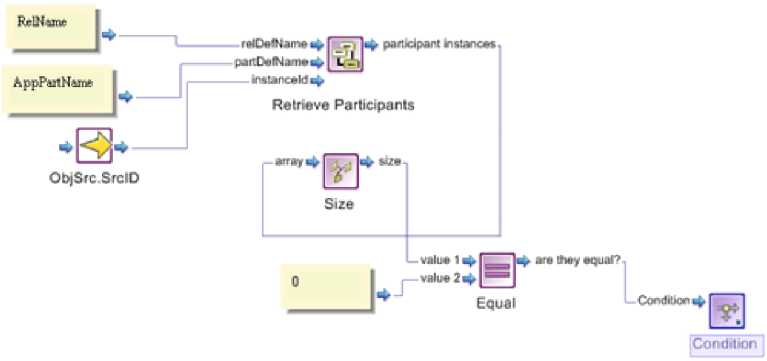


Figura 139. Identificazione del numero dei partecipanti della relazione

   b. Seguire i passi 2b e 2c, descritti in "Esempio dei passi per la personalizzazione della mappa in entrata" a pagina 302.

Impostazione del verbo del child di origine

Quando un oggetto business di origine parent dispone di oggetti business child, il valore del verbo child di origine è in genere uguale al verbo del parent. Pertanto, impostare l'oggetto child di origine definendo una trasformazione di spostamento dal verbo parent al verbo child. Tuttavia, se il verbo dell'oggetto parent è Aggiorna, l'aggiornamento potrebbe essere il risultato di eventuali modifiche

illustrate in Tabella 104.

Tabella 104. Aggiornamento di un oggetto business parent

Aggiorna attività	Verbo nell'oggetto child
Modifica di alcuni attributi non child nell'oggetto parent	Aggiorna
Modifica di alcuni attributi in un oggetto child	Aggiorna
Aggiunta di più oggetti child	Crea
Eliminazione degli oggetti child esistenti	Elimina

Tutte le modifiche in Tabella 104 vengono rappresentate da un verbo Aggiorna nell'oggetto parent. Tuttavia, non tutte le modifiche rappresentano un Aggiornamento all'oggetto child. Il valore del verbo del child di origine dipende dall'azione intrapresa dal verbo parent. Quando la chiave dell'oggetto child partecipa ad una relazione di identità (composita o semplice), il valore del verbo del child di origine non dipende da un solo verbo parent, ma anche dal contesto di richiamo. In tali casi, utilizzare il blocco di funzione Conserva verbo child per gestire l'impostazione del verbo dell'oggetto child di origine.

In questa sezione vengono fornite le seguenti informazioni relative all'utilizzo del blocco di funzione Conserva verbo child per conservare il verbo di un oggetto child di origine:

- "Determinazione dell'impostazione del verbo child"
- "Suggerimenti per l'utilizzo del blocco di funzione Conserva verbo child" a pagina 306

Determinazione dell'impostazione del verbo child

Il blocco di funzione Conserva verbo child deve assicurare che le impostazione del verbo degli oggetti child nell'oggetto business di origine siano appropriate, dato il verbo nell'oggetto di origine parent e il contesto di richiamo. Le azioni effettuate da questo metodo sono basate sul verbo nell'oggetto di origine parent e sul contesto di richiamo.

Contesti di richiamo EVENT_DELIVERY e ACCESS_REQUEST: Quaqndo un contesto di richiamo è `EVENT_DELIVERY` o `ACCESS_REQUEST`, la mappa che sta per essere chiamata è una mappa in entrata, ovvero trasforma un oggetto business specifico dell'applicazione in un oggetto business generico. La mappa in entrata riceve un oggetto business specifico dell'applicazione come input e restituisce un oggetto business generico come output. Per `EVENT_DELIEVERY` (o `ACCESS_REQUEST`), non sono presenti casi particolari da gestire quando si impostano i verbi child. Pertanto, il metodo `maintainChildVerb()` copia il verbo parent nel verbo child per tutti i valori del verbo, come illustrato in Tabella 105.

Tabella 105. Azioni per i contesti di richiamo EVENT_DELIVERY e ACCESS_REQUEST

Verbo di un oggetto business generico	Azione eseguita dal blocco di funzione Conserva verbo child
Crea Elimina AggiornaRecupera	Impostare i verbi di tutti gli oggetti child nell'oggetto di origine sul verbo nell'oggetto di origine parent. Questa azione sovrascrive eventuali verbi esistenti nell'oggetto child.

Contesto di richiamo SERVICE_CALL_REQUEST: Quando il contesto di richiamo è `SERVICE_CALL_REQUEST`, la mappa richiamata è una mappa in uscita, ovvero trasforma un oggetto business generico in un oggetto business specifico dell'applicazione. La mappa in uscita riceve un oggetto business generico come input e restituisce un oggetto business specifico dell'applicazione come output. Per `SERVICE_CALL_REQUEST`, il codice Java generato dal blocco di funzione Conserva

verbo child gestisce il caso particolare per un verbo Aggiorna: se la modifica all'oggetto parente è costituita dalla creazione di nuovi oggetti child, il blocco di funzione Conserva verbo child modifica il verbo in Crea per qualunque oggetto child che al momento non esiste nelle tabelle di relazioni, come illustrato in Tabella 106.

Tabella 106. Azioni per il contesto di richiamo SERVICE_CALL_REQUEST

Verbo di un oggetto business generico	**Azione eseguita dal blocco di funzione Conserva verbo child**
Crea Elimina Recupera Aggiorna	Impostare i verbi di tutti gli oggetti child nell'oggetto di origine sul verbo nell'oggetto di origine parent. Questa azione sovrascrive eventuali verbi esistenti nell'oggetto child. 1. Richiamare l'istanza di relazione dalla tabella di relazioni child per il valore di chiave dell'oggetto business generico. 2. Impostare il verbo dell'oggetto child in base alla ricerca della tabella: • Se un'istanza di relazioni per questo oggetto child esiste, impostare il verbo dell'oggetto child su Aggiorna. • Se un'istanza di relazioni per questo oggetto child *non* esiste, impostare il verbo dell'oggetto child su Crea.

Contesto di richiamo SERVICE_CALL_RESPONSE: Quando il contesto di richiamo è `SERVICE_CALL_RESPONSE`, la mappa che sta per essere richiamata è una mappa in entrata, ovvero trasforma un oggetto business specifico dell'applicazione in un oggetto business generico. La mappa in entrata riceve un oggetto business specifico dell'applicazione come input e restituisce un oggetto business generico come output.

Il comportamento del blocco di funzione Conserva verbo child è determinato dal secondo all'ultimo parametro del metodo. Questo parametro è il flag booleano *`to_Retrieve`*, il cui valore indica se l'applicazione reimposta o conserva i verbi degli oggetti child durante l'elaborazione di una richiesta di collaborazione, come illustrato in Tabella 107.

Tabella 107. Comportamento del connettore

Valore di to_Retrieve flag	**Comportamento del connettore**
`true`	Il connettore imposta i verbi dell'oggetto child su diversi valori da quelli nell'applicazione. Ad esempio, se un oggetto business entra nel connettore con un verbo parent Aggiorna e un verbo child Crea, il connettore potrebbe reimpostare tutti i verbi dell'oggetto child sui valori parent dopo che l'applicazione ha completato l'operazione. In questo caso, il verbo del child verrebbe modificato in Aggiorna.
`false`	Il connettore conserva i verbi dell'oggetto child. Ad esempio, se un oggetto business entra nel connettore con un verbo parent Aggiorna e un verbo child Crea, il connettore conserva tutti i verbi dell'oggetto child. In questo caso, il verbo del child sarebbe ancora Crea.

Nota: Il codice Java generato dal blocco di funzione Conserva verbo child utilizza il valore del parametro *`to_Retrieve`* *solo* quando elabora il contesto di richiamo `SERVICE_CALL_RESPONSE`.

Se l'argomento *`to_Retrieve`* è `true`, il blocco di funzione Conserva verbo esegue le attività illustrate in Tabella 108.

Tabella 108. Azioni per il contesto di richiamo SERVICE_CALL_RESPONSE

Verbo di un oggetto business generico	Azione eseguita dal blocco di funzione Conserva verbo child
Crea Elimina Recupera Aggiorna	Impostare i verbi di tutti gli oggetti child nell'oggetto di origine sul verbo nell'oggetto di origine parent. Questa azione sovrascrive eventuali verbi esistenti nell'oggetto child. 1. Ricercare ciascun oggetto child nella tabella di relazioni child. 2. Impostare il verbo dell'oggetto child in base alla ricerca della tabella: • Se un'istanza di relazioni per questo oggetto child esiste, impostare il verbo dell'oggetto child su Aggiorna. • Se un'istanza di relazioni per questo oggetto child *non* esiste, impostare il verbo dell'oggetto child su Crea.

Nota: Se non si è sicuri del comportamento dell'applicazione, impostare l'argomento *to_Retrieve* su `true`. Con un valore di flag `true`, le prestazioni potrebbero essere condizionate, poiché il codice Java generato dal blocco di funzione Conserva verbo child potrebbe eseguire una ricerca non necessaria. Tuttavia, è, in genere, più sicuro effettuare una ricerca non necessaria che disporre di un'impostazione del verbo non corretta nell'oggetto child.

Suggerimenti per l'utilizzo del blocco di funzione Conserva verbo child

Il blocco di funzione Conserva verbo child conserva il verbo degli oggetti child in un oggetto business di origine. Può gestire oggetti child che sono parte di una relazione di identità semplice o composita. Questo blocco di funzione assicura che le impostazioni del verbo siano appropriate dato il verbo nell'oggetto di origine parent e il contesto di richiamo.

Tenere presente i seguenti suggerimenti quando si utilizza il blocco di funzione Conserva verbo child:

- Dal secondo all'ultimo parametro del metodo è il flag booleano, *to_Retrieve*, che indica se l'applicazione reimposta o conserva i verbi dell'oggetto child.

  Per ulteriori informazioni sull'impostazione del flag *to_Retrieve*, consultare "Contesto di richiamo SERVICE_CALL_RESPONSE" a pagina 305.
- L'ultimo parametro del metodo è il flag booleano *is_Composite*, che indica se l'oggetto child è parte di una relazione di identità semplice o composita.

  L'attributo chiave di un oggetto business child può partecipare ad uno dei seguenti tipi di relazione di identità:
  - Come chiave univoca in una relazione di identità semplice

    Impostare il valore del flag *is_Composite* su `false`.
  - Come chiave non univoca della chiave composita in una relazione di identità composita, in questo caso l'altra parte della chiave composita è la chiave univoca nell'oggetto business parent.

    Impostare il valore del flag *is_Composite* su `true`.
- Verificare di utilizzare il blocco di funzione nell'attributo child della mappa del parent di origine *prima* di richiamare la mappa secondaria.

  Per gli oggetti child con cardinalità multipla, utilizzare il blocco di funzione Conserva verbo child appena *prima* dell'avvio del loop `for`. il metodo viene iterato mediante gli oggetti child per impostare i verbi child correttamente.

Esecuzione di ricerche della chiave esterna

Una *chiave esterna* è un attributo all'interno di un oggetto business contenente il valore di chiave o un altro oggetto business. Questo valore di chiave è considerato "esterno" per l'oggetto business di origine, poiché identifica qualche altro oggetto business. Per trasformare una chiave esterna in un oggetto business di origine, è necessario accedere alla tabella di relazioni associata all'oggetto business cui fa riferimento la chiave esterna (la tabella di relazioni esterna). Da questa tabella di relazioni esterna, è possibile ottenere il valore di chiave associato per la chiave esterna dell'oggetto business di destinazione.

La mappatura dell'API fornisce i metodi in Tabella 109 per eseguire le ricerche della chiave esterna.

Tabella 109. Blocchi di funzione per le ricerche della chiave esterna

Blocco di funzione	Descrizione
Generale/API/Relazione identità/Ricerca chiave esterna	Esegue una ricerca della chiave esterna, non trovando l'istanza di una relazione se la chiave esterna non esiste nella tabella di relazione esterna.
Generale/API/Relazione di identità/Riferimento incrociato della chiave esterna	Esegue una ricerca della chiave esterna, aggiungendo una nuova istanza di relazione nella tabella di relazione esterna se la chiave esterna non esiste.

Utilizzo del blocco di funzione Ricerca della chiave esterna

Il codice Java generato dal blocco di funzione Ricerca della chiave esterna esegue una ricerca in una tabella di relazione esterna per la chiave esterna dell'oggetto business di origine. Questo blocco di funzione effettua le azioni di seguito riportate:

1. Verifica che il partecipante specifico dell'applicazione contenga una chiave univoca, *non* una chiave composita.

   Determina il tipo di partecipante di un partecipante specifico dell'applicazione, che è l'oggetto business specifico dell'applicazione. In questo oggetto business, verifica che esista un solo attributo chiave. Se esiste più di un attributo chiave, il blocco di funzione Ricerca della chiave esterna non sa quale attributo chiave specifico dell'applicazione popolare con l'equivalente specifico dell'applicazione della chiave esterna dell'oggetto business generico. Pertanto, restituisce l'eccezione `RelationshipRuntimeException`.
2. Ricerca l'istanza di relazione nella tabella di relazione della chiave esterna che corrisponde al valore della chiave esterna in un oggetto business generico.
3. Ottiene il valore di chiave specifico dell'applicazione dall'istanza di relazione richiamata.
4. Copia il valore di chiave specifico dell'applicazione nella chiave esterna dell'oggetto business specifico dell'applicazione.

Il codice Java generato dal blocco di funzione Ricerca della chiave esterna esegue queste azioni sulla tabella di relazione esterna indipendentemente dal verbo nell'oggetto business di origine.

Utilizzo del blocco di funzione Riferimento incrociato della chiave esterna

Come per il blocco di funzione Ricerca della chiave esterna, il blocco di funzione Riferimento incrociato esegue una ricerca nella tabella di relazione esterna basata

sulla chiave esterna dell'oggetto business di origine. Tuttavia, il blocco di funzione Riferimento incrociato della chiave esterna fornisce funzioni aggiuntive, ovvero è possibile aggiungere una voce alla tabella di relazione esterna se la ricerca ha esito negativo. Le sezioni di seguito riportate illustrano il comportamento del blocco di funzione Riferimento incrociato della chiave esterna con ciascuno dei contesti di richiamo.

Contesti di richiamo EVENT_DELIVERY, ACCESS_REQUEST, e SERVICE_CALL_RESPONSE

Quando un contesto di richiamo è `EVENT_DELIVERY`, `ACCESS_REQUEST` o `SERVICE_CALL_RESPONSE`, la mappa richiamata è una mappa in entrata, ovvero trasforma un oggetto business specifico dell'applicazione in un oggetto business generico. La mappa in entrata riceve un oggetto business specifico dell'applicazione come input e restituisce un oggetto business generico come output. Pertanto, l'attività per il blocco di funzione Riferimento incrociato della chiave esterna è di ottenere dalla tabella di relazione esterna la chiave generica per un valore di chiave specifico dell'applicazione.

Per i contesti di richiamo `EVENT_DELIVERY`, `ACCESS_REQUEST` e `SERVICE_CALL_RESPONSE`, il blocco di funzione Riferimento incrociato della chiave esterna intraprende le azioni di seguito riportate:

1. Verifica che il partecipante generico contenga una chiave univoca, *non* una chiave composita.

   Determina il tipo di partecipante di un partecipante generico, che è l'oggetto business generico. In questo oggetto business, verifica che esista un solo attributo chiave. Se esiste più di un attributo chiave, il blocco di funzione Riferimento incrociato della chiave esterna non sa quale attributo generico popolare con l'equivalente generico della chiave esterna dell'oggetto business specifico dell'applicazione. Pertanto, restituisce l'eccezione `RelationshipRuntimeException`.
2. Ricerca l'istanza di relazione nella tabella di relazione della chiave esterna che corrisponde al valore della chiave esterna in un oggetto business specifico dell'applicazione. Tabella 110 illustra le azioni effettuate dal blocco di funzione Riferimento incrociato della chiave esterna sulla tabella di relazione in base al verbo dell'oggetto business specifico dell'applicazione.
3. Ottiene l'ID di istanza dall'istanza di relazione ripristinata.
4. Copia un ID di istanza nella chiave esterna di un oggetto business generico.

Tabella 110. Azioni per EVENT_DELIVERY, ACCESS_REQUEST e SERVICE_CALL_RESPONSE

Verbo dell'oggetto business specifico dell'applicazione	Azione eseguita per il blocco di funzione Riferimento incrociato della chiave esterna
Crea	Per i contesti di richiamo `EVENT_DELIVERY` e `ACCESS_REQUEST`, inserire una nuova voce di relazione nella tabella di relazione esterna per il valore di chiave dell'oggetto business specifico dell'applicazione. Per il contesto di richiamo `SERVICE_CALL_RESPONSE`, inserire nella tabella di relazione la nuova voce di relazione contenente il valore di chiave dell'oggetto business specifico dell'applicazione e il relativo ID istanza della relazione. Il metodo ottiene l'ID istanza della relazione dall'oggetto business della richiesta di origine nel contesto di esecuzione della mappa (`cwExecCtx`). Per ulteriori informazioni sul comportamento di `SERVICE_CALL_RESPONSE`, consultare "Contesto di richiamo SERVICE_CALL_RESPONSE" a pagina 284.

Tabella 110. Azioni per EVENT_DELIVERY, ACCESS_REQUEST e SERVICE_CALL_RESPONSE (Continua)

Verbo dell'oggetto business specifico dell'applicazione	Azione eseguita per il blocco di funzione Riferimento incrociato della chiave esterna
	Se una voce per questo valore di chiave già esiste, ripristinare quello esistente, *non* aggiungerne un altro alla tabella.
Aggiorna	Richiamare la voce della relazione dalla tabella di relazioni esterna per un determinato valore di chiave esterna dell'oggetto business specifico dell'applicazione. Se una voce per il valore di chiave esterna *non* esiste, inserire una nuova istanza di relazione nella tabella di relazioni per il valore di chiave esterna dell'oggetto business specifico dell'applicazione.
Recupera	Richiama la voce della relazione dalla tabella di relazioni esterna per un determinato valore di chiave esterna dell'oggetto business specifico dell'applicazione

Figura 140 illustra il modo in cui il blocco di funzione Riferimento incrociato della chiave esterna accede alla tabella di relazione esterna (per `App Obj C`) quando il contesto di richiamo è `EVENT_DELIVERY`, `ACCESS_REQUEST` o `SERVICE_CALL_RESPONSE` e il verbo per l'oggetto business specifico dell'applicazione (`App Obj A`) è Crea o Aggiorna.

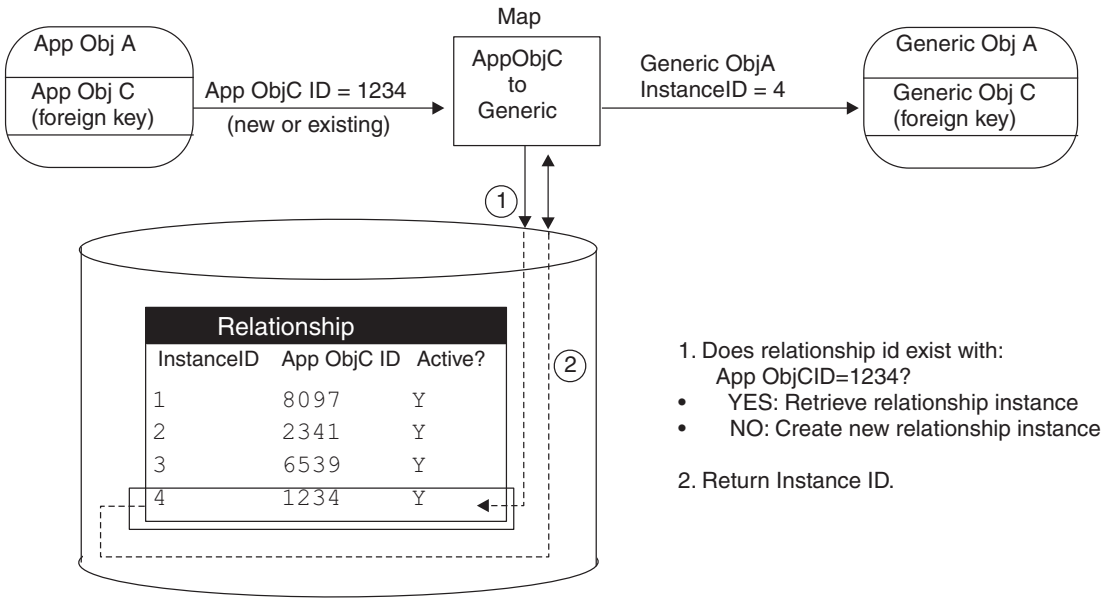


Figura 140. Ricerca della chiave esterna per un verbo Aggiorna o Crea

Nota: Il blocco di funzione Riferimento incrociato della chiave esterna aggiunge solo istanze di relazione alla tabella di relazione esterna nelle mappe in entrata.

Contesto di richiamo SERVICE_CALL_REQUEST e chiavi esterne

Quando il contesto di richiamo è `SERVICE_CALL_REQUEST`, la mappa richiamata è una mappa in uscita, ovvero trasforma un oggetto business generico in un oggetto business specifico dell'applicazione. La mappa in uscita riceve un oggetto business generico come input e restituisce un oggetto business specifico dell'applicazione

come output. Per il contesto di richiamo SERVICE_CALL_REQUEST, il blocco di funzione Riferimento incrociato della chiave esterna esegue le seguenti azioni:

1. Verifica che il partecipante specifico dell'applicazione contenga una chiave univoca, *non* una chiave composita.

   Determina il tipo di partecipante di un partecipante specifico dell'applicazione, che è l'oggetto business specifico dell'applicazione. In questo oggetto business, verifica che esista un solo attributo chiave. Se esiste più di un attributo chiave, il blocco di funzione Riferimento incrociato della chiave esterna non sa quale attributo chiave specifico dell'applicazione popolare con l'equivalente specifico dell'applicazione della chiave esterna dell'oggetto business generico. Pertanto, restituisce l'eccezione RelationshipRuntimeException.

2. Esegue l'attività illustrata in Tabella 111, in base al verbo dell'oggetto business specifico dell'applicazione.

   Il blocco di funzione Riferimento incrociato della chiave esterna ottiene dalla tabella di relazioni della chiave esterna un valore di chiave dell'oggetto business specifico dell'applicazione per un determinato ID di istanza della relazione *solo* se il verbo è Aggiorna, Elimina o Recupera. Il blocco di funzione Riferimento incrociato della chiave esterna *non* ottiene il valore di chiave specifico dell'applicazione per un verbo Crea.

Tabella 111 illustra l'azione effettuata dal blocco di funzione Riferimento incrociato della chiave esterna nella tabella di relazione esterna, in base al verbo dell'oggetto business generico.

Tabella 111. Azioni per il contesto di richiamo SERVICE_CALL_REQUEST e chiave esterna

Verbo di un oggetto business generico	**Azione eseguita per il blocco di funzione Riferimento incrociato della chiave esterna**
Crea	Non intraprendere alcuna azione. Il metodo scrive una nuova istanza di relazione nella tabella di relazioni esterna quando il contesto di richiamo è SERVICE_CALL_RESPONSE. Per ulteriori informazioni, consultare "Contesti di richiamo EVENT_DELIVERY, ACCESS_REQUEST, e SERVICE_CALL_RESPONSE" a pagina 308.
Aggiorna Elimina Recupera	1. Ottenere il valore di chiave dell'oggetto business generico (l'ID di istanza della relazione) dall'oggetto business della richiesta di origine dal contesto di esecuzione della mappa. 2. Richiamare l'istanza di relazione dalla tabella di relazioni esterna per il valore di chiave dell'oggetto business generico. Se un'istanza di relazione per questo valore di chiave *non* esiste, inoltrare un'eccezione RelationshipRuntimeException. Se non viene trovato alcun partecipante quando il verbo è Recupera, viene inoltrata un'eccezione CxMissingIDException. 3. Ottenere il valore di chiave specifico dell'applicazione dalla voce dell'istanza di relazione recuperata. 4. Copiare il valore di chiave specifico dell'applicazione nell'oggetto business specifico dell'applicazione.

Come illustrato in Tabella 111, quando il verbo è Crea, il blocco di funzione Riferimento incrociato della chiave esterna *non* scrive una nuova istanza di relazione nella tabella di relazione. Non viene eseguita questa operazione di scrittura, poiché non si dispone ancora del valore della chiave esterna specifica dell'applicazione corrispondente all'ID di istanza. Quando il connettore elabora l'oggetto business specifico dell'applicazione, viene notificata all'applicazione la necessità di inserire una nuova riga (o righe). Se questo inserimento viene eseguito

correttamente, l'applicazione invia una notifica al connettore, che crea l'oggetto business specifico dell'applicazione appropriato con un verbo Crea e il valore della chiave dell'applicazione.

Nota: Per il contesto di richiamo SERVICE_CALL_REQUEST, il blocco di funzione Riferimento incrociato della chiave esterna gestisce la tabella di relazioni esterna nello stesso modo in cui il blocco di funzione Conserva relazione di identità semplice gestisce una tabella di relazione.

Contesto di richiamo ACCESS_RESPONSE e chiavi esterne

Quando il contesto di richiamo è ACCESS_RESPONSE, la mappa richiamata è una mappa in uscita, ovvero trasforma un oggetto business generico in un oggetto business specifico dell'applicazione. La mappa in uscita riceve un oggetto business generico come input e restituisce un oggetto business specifico dell'applicazione come output. Pertanto, l'attività per il blocco di funzione Riferimento incrociato della chiave esterna è di ottenere dalla tabella di relazione esterna la chiave specifica dell'applicazione per un valore di chiave generico.

Per il contesto di richiamo ACCESS_RESPONSE, il blocco di funzione Riferimento incrociato della chiave esterna esegue le seguenti azioni:

1. Verifica che il partecipante specifico dell'applicazione contenga una chiave univoca, *non* una chiave composita.

   Determina il tipo di partecipante di un partecipante specifico dell'applicazione, che è l'oggetto business specifico dell'applicazione. In questo oggetto business, verifica che esista un solo attributo chiave. Se esiste più di un attributo chiave, il blocco di funzione Riferimento incrociato della chiave esterna non sa quale attributo chiave specifico dell'applicazione popolare con l'equivalente specifico dell'applicazione della chiave esterna dell'oggetto business generico. Pertanto, restituisce l'eccezione RelationshipRuntimeException.
2. Ricerca l'istanza di relazione nella tabella di relazione della chiave esterna che corrisponde al valore della chiave esterna in un oggetto business generico.
3. Ottiene il valore di chiave specifico dell'applicazione dall'istanza di relazione richiamata.
4. Copia il valore di chiave specifico dell'applicazione nella chiave esterna dell'oggetto business specifico dell'applicazione.

Il blocco di funzione Riferimento incrociato della chiave esterna effettua queste azioni sulla tabella di relazione esterna indipendentemente dal verbo nell'oggetto business generico.

Suggerimenti per l'utilizzo dei blocchi di funzione Riferimento incrociato della chiave esterna e Ricerca della chiave esterna

Tenere presente i seguenti suggerimenti quando si utilizzano i blocchi di funzione Riferimento della chiave esterna e Ricerca della chiave esterna:

- Inserire il richiamo ai blocchi di funzione Ricerca della chiave esterna o Riferimento incrociato della chiave esterna nel passo di trasformazione per l'attributo della chiave esterna per l'oggetto business di destinazione.
- I blocchi di funzione Ricerca della chiave esterna e Riferimento incrociato della chiave esterna *non* supportano le chiavi composte come chiavi esterne.
- Dopo aver utilizzato il blocco di funzione Ricerca della chiave esterna, verificare che l'attributo della chiave esterna *non* contenga un valore null. Un valore della chiave esterna null indica che il blocco di funzione Ricerca della chiave esterna non è riuscito a trovare il valore della chiave esterna per la chiave esterna

nell'oggetto business di origine. Per indicare questa condizione, registrare il numero messaggio 5007 o 5008 (rispettivamente, se l'errore della mappa è stato forzato o meno) e, facoltativamente, inoltrare l'eccezione `MapFailureException` per arrestare la mappa.

Non è necessaria questa verifica dopo aver utilizzato il blocco di funzione Riferimento incrociato della chiave esterna, poiché questo blocco di funzione aggiunge automaticamente una voce alla tabella di relazione esterna se il valore di chiave specifico dell'applicazione non esiste.

- Se qualunque degli attributi dell'oggetto child richiede l'utilizzo del blocco di funzione Riferimento incrociato della chiave esterna o Ricerca della chiave esterna, (ma non il blocco di funzione Conserva relazione di identità semplice o Conserva relazione composita), è possibile impostare il verbo di un oggetto child di origine definendo una trasformazione Sposta da verbo dell'oggetto parent di origine al verbo dell'oggetto business child. Effettuare il richiamo *nel* loop `for`, proprio appena prima del richiamo del metodo `runMap()`.

Conservazione delle relazioni personalizzate

La mappatura API fornisce i metodi per gestire le operazioni sulle tabelle di relazione per alcune relazioni di base, come ad esempio le relazioni di identità e di ricerca. Per le altre relazioni, è possibile creare relazioni personalizzate programmando l'aggiunta e l'eliminazione dei partecipanti.

Come programmare il codice di trasformazione per un attributo di relazione dipende da vari fattori che variano in base all'esecuzione della mappa. In genere, il codice viene strutturato come una serie di casi che gestiscono ciascuna delle situazioni possibili che potrebbero verificarsi. Di seguito sono riportati i fattori da considerare:

- Il verbo associato all'oggetto business di origine: Crea, Recupera, Aggiorna o Elimina. Ad esempio, se il verbo è Crea, la mappa, in genere, crea una nuova istanza di relazione o aggiunge una nuova istanza del partecipante ad un'istanza di relazione esistente. Il codice dovrebbe gestire ciascun verbo supportato dall'oggetto business specifico dell'applicazione e dal relativo connettore.
- Il contesto di richiamo associato all'istanza della mappa.I contesti di richiamo indicano l'obiettivo dell'esecuzione della mappa corrente e possono condizionare il modo di gestione di ciascuno dei casi del verbo. Ad esempio, se il verbo è Crea e il contesto di richiamo è`EVENT_DELIVERY`, in genere, viene creata una nuova istanza di relazione, se il contesto di richiamo è `SERVICE_CALL_RESPONSE`, in genere viene aggiunto un partecipante all'istanza di relazione. Per ulteriori informazioni sui contesti di richiamo, consultare "Contesti di esecuzione delle mappe" a pagina 200.
- La logica business contenuta nella collaborazione associata alla mappa. Ad esempio, se la collaborazione gestisce la sincronizzazione dei dati tra le due applicazioni, lo sviluppatore della mappa deve coordinarsi con lo sviluppatore di collaborazioni per determinare la logica business che è contenuto nella collaborazione e quale è gestita nella mappa.

Le classi`Relationship` e `IdentityRelationship` contengono metodi per la creazione di istanze di relazione e per l'aggiunta, l'eliminazione e l'aggiornamento delle istanze del partecipante. La classe del `Participant` fornisce i metodi "set" e "get" per specificare e recuperare varie proprietà di istanze del partecipante.

Le sezioni di seguito riportate forniscono informazioni sulla gestione di una relazione personalizzata:

- "Creazione di una nuova istanza di relazione"
- "Creazione di istanze del partecipante"
- "Eliminazione delle istanza del partecipante"

Creazione di una nuova istanza di relazione

Per creare una nuova istanza di relazione, utilizzare `create()` o i metodi `addMyChildren()`. Entrambi i metodi creano un'istanza di relazione con una nuova istanza del partecipante.

L'esempio più comune della creazione di una nuova istanza di relazione è quando il verbo è Crea e il contesto di richiamo è `EVENT_DELIVERY` (o `ACCESS_REQUEST`). In questo caso. un'applicazione genera un evento "crea" per cui è stata sottoscritta una collaborazione e la mappa trasforma un oggetto specifico dell'applicazione in un oggetto business generico. Per creare una nuova istanza di relazione, è necessario fornire il nome di definizione della relazione, il nome di definizione del partecipante che si sta aggiungendo e i dati associati al partecipante.

Il codice di seguito riportato crea una nuova istanza di relazione denominata `CustIden` dopo che un cliente viene aggiunto ad un'applicazione Clarify. La definizione del partecipante che rappresenta Clarify è denominata `ClarCust`:

```
instanceId = Relationship.create
("CustIden","ClarCust",appBusObj);
```

Il valore restituito `instanceId` è un ID di istanza della nuova relazione di istanza.

Creazione di istanze del partecipante

Per creare una nuova istanza del partecipante per la relazione, utilizzare il metodo `addParticipant()` . In genere, si crea una nuova istanza del partecipante quando:

- Il verbo dell'oggetto business di origine è Crea e il contesto di richiamo è `EVENT_DELIVERY`. In questo caso, si crea una nuova istanza di relazione insieme ad una nuova istanza del partecipante.
- Il verbo dell'oggetto business di origine è Crea e il contesto di richiamo è `SERVICE_CALL_RESPONSE`. In questo caso, aggiungere il nuovo partecipante ad un'istanza di relazione esistente.

Il seguente codice aggiunge una nuova istanza del partecipante ad una relazione denominata `CustIden`. Quest'ultimo suppone un verbo Crea e un contesto di richiamo `SERVICE_CALL_RESPONSE`. La variabile `relID` contiene l'ID dell'istanza di relazione per ricevere la nuova istanza del partecipante.

```
instanceId = Relationship.addParticipant("CustIden","SAPCust",
   relID, appBusObj);
```

Eliminazione delle istanza del partecipante

Per eliminare un'istanza del partecipante da un'istanza di relazione, utilizzare uno dei metodi della classe `Relationship` elencati in Tabella 112.

Tabella 112. Metodi che eliminano un'istanza del partecipante

Metodo di relazione	Descrizione
`deleteParticipant()`	Elimina *tutte* le istanze del partecipante che corrispondono al nome di definizione della relazione, al nome di definizione del partecipante e ai dati del partecipante specificati.

Tabella 112. Metodi che eliminano un'istanza del partecipante (Continua)

Metodo di relazione	Descrizione
`deleteParticipantByInstance()`	Elimina *un* partecipante da una determinata istanza di relazione specificata.
`deactivateParticipant()`	Identica a `deleteParticipant()`, ma lascia un record del partecipante nelle tabelle di relazione.
`deactivateParticipantByInstance()`	Identica a `deleteParticipantByInstance()`, ma lascia un record del partecipante nelle tabelle di relazione.

In genere, si eliminano le istanze del partecipante quando l'oggetto business di origine dispone di un verbo Elimina e il contesto di richiamo è `EVENT_DELIVERY` (o `ACCESS_REQUEST`) o `SERVICE_CALL_RESPONSE`.

Esempio: il seguente codice suppone un contesto di richiamo `SERVICE_CALL_RESPONSE` e un verbo Elimina. Elimina un'istanza del partecipante che rappresenta un cliente Clarify dalla relazione `CustIden`.

```
Relationship.deleteParticipant
("CustIden","ClarCust",appBusObj);
```

Scrittura di un codice di relazione protetto

Suggerimento: è necessario utilizzare gli standard di codifica protetti di seguito riportati per gli attributi che richiedono una gestione della relazione:

- Assicurarsi sempre che l'attributo di origine non sia null prima di richiamare un metodo della Mappatura API.
- Inserire sempre il richiamo al metodo della Mappatura API all'interno del blocco `try/catch` e visualizzare il messaggio di errore appropriato nella sezione `catch`.

Verifica degli attributi di origine null

Prima di richiamare uno dei seguenti metodi di Mappatura API in Tabella 113, assicurarsi che l'attributo di origine *non* sia null. Se l'attributo è null, registrare l'errore e *non* richiamare il metodo.

Tabella 113. Gestione degli attributi di origine null

Mappatura del metodo API	Numero errore da registrare	Arrestare l'esecuzione della mappa?
`maintainSimpleIdentityRelationship()`	5000	Sì
`foreignKeyXref()` `foreignKeyLookup()`	5003	La specifica della mappatura dovrebbe specificare se l'esecuzione della mappa deve essere arrestata.

Per arrestare l'esecuzione della mappa, inoltrare `l'eccezione MapFailureException`.

Gestione delle eccezioni dal metodo di mappatura API

Per assicurarsi che le eccezioni inoltrate dai metodi Mappatura API in Tabella 114 siano recapitati, inserire il richiamo al metodo della Mappatura API all'interno `try/`del blocco `catch`, quindi registrare il messaggio di errore appropriato all'interno della sezione `catch`.

Tabella 114. Gestione delle eccezioni dai metodi di Mappatura API

Mappatura del metodo API	Eccezione da recapitare	Numero errore da registrare
`maintainSimpleIdentity Relationship()`	`RelationshipRuntimeException`	5001
`maintainComposite Relationship()`		
	`CxMissingIDException`	5002
`foreignKeyLookup()`	`RelationshipRuntimeException`	5007 or 5008
`foreignKeyXref()`	`RelationshipRuntimeException`	5009

Esempio: il seguente frammento di codice comprende un richiamo al metodo `maintainSimpleIdentityRelationship()` che recapita le eccezioni `RelationshipRuntimeException` e `CxMissingIDException`, registra i messaggi informativi per visualizzare il testo dell'errore generato dal server e arresta l'esecuzione della mappa inoltrando `MapFailureException`:

```
try
   {
   // API call
   IdentityRelationship.maintainSimpleIdentityRelationship(...);
   }

catch (RelationshipRuntimeException re)
   {
   logError(5001);
   logInfo(re.toString());
   throw new MapFailureException("RelationshipRuntimeException");
   }

catch (CxMissingIDException ce)
   {
   logError(5002);
   logInfo(ce.toString());
   throw new MapFailureException("RelationshipRuntimeException");
   }
```

Esempio: il seguente frammento di codice illustra la gestione dell'eccezione per il metodo `foreignKeyLookup()` che recapita l'eccezione `RelationshipRuntimeException`, registra il messaggio informativo per visualizzare il testo dell'errore generato dal server, quindi verifica l'attributo di destinazione per controllare se è stato mappato correttamente, in caso contrario, il frammento visualizza un errore 5007 se la mappa deve arrestare l'esecuzione o il messaggio 5008 se può continuare l'esecuzione della mappa:

```
try
   {
   // API call
   IdentityRelationship.foreignKeyLookup(...);
   }

catch (RelationshipRuntimeException re)
   {
   logInfo(re.toString());
   }

if (ObjDest.isNull("DestAttr")
   {
   logError(5007, "DestAttrName", "SrcAttrName", "RelationshipName",
      "ParticipantName", strInitiator);
   throw new MapFailureException("foreignKeyLookup() failed");
```

```
   }
If the map execution is to be continued, use the following if statement:
if (ObjDest.isNull("DestAttr")
   {
   logError(5008, "DestAttrName", "SrcAttrName", "RelationshipName",
      "ParticipantName", strInitiator);
   }
```

Esempio: il seguente frammento del codice illustra la gestione dell'eccezione per il metodo `foreignKeyXref()` che recapita `RelationshipRuntimeException`, registra un messaggio informativo per visualizzare il testo dell'errore generato dal server, quindi verifica l'attributo di destinazione per controllare se è stato mappato correttamente, in caso contrario, il frammento visualizza un errore con il messaggio 5009 e arresta l'esecuzione della mappa inoltrando `MapFailureException`:

```
try
   {
   // API call
   IdentityRelationship.foreignKeyXref(...);
   }

catch (RelationshipRuntimeException re)
   {
   logInfo(re.toString());
   }

if (ObjDest.isNull("DestAttr")
   {
   logError(5009, "DestAttrName", "SrcAttrName", "RelationshipName",
      "ParticipantName", strInitiator);
   throw new MapFailureException("foreignKeyXref() failed");
   }
```

Esecuzione delle query nel database di relazione

Quando si utilizzano le relazioni, potrebbe essere necessario ottenere informazioni sulla definizione della relazione. Le informazioni sulla relazione vengono memorizzate in tabelle particolari nel database di relazioni. Per ottenere informazioni su una relazione, effettuare una query delle relative tabelle. Una *query* è una richiesta, in genere in forma di istruzione SQL (Structured Query Language), che viene inviata al database per l'esecuzione.

Per eseguire le query nel database di relazione:

1. Aprire una connessione al database di relazione per ottenere un oggetto `DtpConnection`.
2. Mediante l'oggetto `DtpConnection`, eseguire le query e gestire le transazioni nel database di relazione.

La connessione viene chiusa automaticamente quando la mappa termina l'esecuzione.

Importante: L'utilizzo della classe `DtpConnection` e dei relativi metodi per stabilire una connessione ad un database di relazioni è supportato *solo* per la compatibilità pregressa. Questi *metodi non approvati* non generano errori, ma se ne sconsiglia l'utilizzo e la migrazione del codice esistente in nuovi metodi. I metodi non approvati potrebbero essere eliminati in un futuro rilascio. Nello sviluppo della nuova mappa, utilizzare la classe `CwDBStoredProcedureParam` ed i relativi metodi per

ottenere una connessione al database ed eseguire query SQL. Per ulteriori informazioni, consultare “Esecuzione delle query nel database” a pagina 215.

Apertura di una connessione

Per effettuare una query al database di relazione, è necessario aprire prima una connessione a questo database con `getRelConnection()` il metodo della classe `BaseDLM`. Per identificare il database di relazioni da aprire, specificare il nome della definizione di relazione di cui si desidera effettuare la query. Il repository tiene traccia della posizione delle tabelle di relazione per ciascuna definizione di relazione. Per ulteriori informazioni, consultare “Impostazioni avanzate per le definizioni di relazioni” a pagina 262.

Esempio: il seguente richiamo a `getRelConnection()` apre il database di relazione che contiene le tabelle di relazione per la relazione `SapCust`:

```
DtpConnection connection = getRelConnection("SapCust");
```

Questo richiamo restituisce un oggetto `DtpConnection` nella variabile della `connessione`, che è possibile utilizzare per accedere al database di relazione.

Esecuzione della query

Il `executeSQL()` metodo invia la query corrente al database di relazione per l'esecuzione. Questa sezione illustra l'esecuzione dei seguenti tipi di query SQL:

- Query che restituiscono dati dalle tabelle di relazione (SELECT)
- Query che modificano le tabelle di relazione (INSERT, UPDATE, DELETE)
- Query che eseguono le procedure memorizzate

Query che restituiscono dati (SELECT)

L'istruzione SQL SELECT esegue la query di una o più tabelle per i dati. Per inviare un istruzione SELECT al database di relazione per l'esecuzione, specificare una rappresentazione della stringa di SELECT come argomento per il metodo `executeSQL()`.

Nota: Il metodo `DtpConnection.executeSQL()` è supportato solo per compatibilità pregressa. *Non* utilizzare questo metodo per lo sviluppo di un nuovo codice, invece utilizzare il metodo `executeSQL()` della classe `CwDBConnection`.

Esempio: il seguente richiamo a `executeSQL()` invia SELECT di un valore di colonna dalla tabella `RelRT_T`:

```
connection.executeSQL(
   "select data from RelRT_T where INSTANCEID = 2");
```

Nota: Nel codice precedente, la variabile `connection`è un oggetto `DtpConnection` ottenuto da un richiamo precedente del metodo `getRelConnection()`.

Inoltre, è possibile inviare un'istruzione SELECT che ha dei parametri all'interno utilizzando il secondo formato del metodo `executeSQL()`.

Esempio: il seguente richiamo a `executeSQL()` esegue la stessa attività dell'esempio precedente, ma tale richiamo inoltra l'ID istanza come parametro all'istruzione SELECT:

```
Vector argValues = new Vector();

String instance_id = "2";
```

```
argValues.addElement( instance_id );
connection.executeSQL(
   "select data from RelRT_T where INSTANCEID = ?", argValues);
```

L'istruzione SELECT restituisce dati dalle tabelle di relazione come righe. Ciascuna riga è una riga dalla tabella di relazione specificata che corrisponde alle condizioni in SELECT. Ciascuna riga contiene i valori per le colonne specificate dall'istruzione SELECT. E' possibile visualizzare i dati restituiti come vettori a bidimensionali di queste righe e colonne.

Suggerimenti: La sintassi dell'istruzione SELECT deve essere valida per un determinato database di relazione cui si accede. Per la sintassi esatta dell'istruzione SELECT, consultare la documentazione del database.

Per accedere ai dati restituiti, eseguire i passi riportati:

1. Ottenere una riga di dati
2. Ottenere i valori di colonna, uno per uno.

Tabella 115 illustra i metodi della classe `DtpConnection` che forniscono l'accesso alle righe dei dati di query restituiti.

Tabella 115. DtpConnection metodi per l'accesso alle righe

Attività per l'accesso alle righe	Metodo DtpConnection
Verifica dell'esistenza di una riga.	`hasMoreRows()`
Ottenere una riga di dati	`nextRow()`

Controllare il loop attraverso le righe restituite con il metodo `hasMoreRows()`. Terminare il loop della riga quando `hasMoreRows()` restituisce il valore `false`. Per ottenere una riga di dati, utilizzare il metodo `nextRow()`. Questo metodo restituisce i valori di colonna selezionati come elementi in un oggetto `Vettore` Java. Quindi, è possibile utilizzare una `Classe enumerazione` per accedere singolarmente ai valori di colonna. Entrambe le classi `Vector` e `Enumeration` si trovano nel pacchetto `java.util`. Per i metodi Java per l'accesso alle colonne di una riga di query restituita, consultare Tabella 71 a pagina 217.

Nota: Il meccanismo di accesso alle righe dalla query è uguale a quello per la classe `DtpConnection` non approvata e alla relativa sostituzione, la classe `CwDBConnection`. Per ulteriori informazioni, consultare "Esecuzione di query statiche che restituiscono dati (SELECT)" a pagina 216.

Esempio: il seguente esempio del codice ottiene un'istanza della classe `DtpConnection`, che è una connessione al database di relazione che memorizza la definizione della relazione `sampleRelationshipName`. Quindi, esegue un'istruzione SELECT che restituisce solo una colonna con un unico valore di stringa "`CrossWorlds`":

```
Vector theRow = null;
Enumeration theRowEnum = null;
String theColumn1 = null;
DtpConnection connectn = null;

try
   {
   connectn = getRelConnection("sampleRelationshipName");
   }
```

```
catch(DtpConnectionException e)
   {
   System.out.println(e.getMessage());
   }

// Test for a resulting single column, single row, result set
// specified condition
try
   {
   connectn.executeSQL(
      "select data from RelRT_T where INSTANCEID = 2");

   // Loop through each row
   while(connectn.hasMoreRows())
      {
      theRow = connectn.nextRow();
      int length = 0;
      if ((length = theRow.size())!= 1)
         {
         return "Expected result set size = 1," +
            " Actual result state size = " + length;
         }

      // Get each column
      theRowEnum = theRow.elements();
      if(theRowEnum.hasMoreElements())
         {
         // Get the value
         theColumn1 = (String)theRowEnum.nextElement();
         if(theColumn1.equals("CrossWorlds")==false)
            {
            return "Expected result = CrossWorlds,"
               + " Resulting result = " + theColumn1;
            }
         }
      }
   }

catch(DtpConnectionException e)
   {
   System.out.println(e.getMessage());
   }
```

Nota: L'istruzione SELECT *non* modifica il contenuto del database di relazione. Pertanto, *non* è necessario eseguire la gestione della transazione per le istruzioni SELECT.

Query che modificano le tabelle di relazione

Le istruzioni SQL che modificano una relazione comprendono:

- INSERT aggiunge nuove righe ad una tabella di relazione.
- UPDATE modifica le righe esistenti di una tabella di relazione.
- DELETE rimuove le righe da una tabella di relazione.

Per inviare una di queste istruzioni al database di relazione per l'esecuzione, specificare una rappresentazione di una stringa dell'istruzione come argomento al metodo `executeSQL()`.

Nota: Il metodo `DtpConnection.executeSQL()` è supportato solo per compatibilità pregressa. *Non* utilizzare questo metodo per lo sviluppo di un nuovo codice, invece utilizzare il metodo `executeSQL()` della classe `CwDBConnection`.

Esempio: il seguente richiamo a `executeSQL()` invia un'istruzione INSERT di una riga nella tabella `RelRT_T`:

```
connection.executeSQL("insert into RelRT_T values
(1, 3, 6)");
```

Nota: Nel codice precedente, la variabile `connection`è un oggetto `DtpConnection` ottenuto da un richiamo precedente del metodo `getRelConnection()`.

Per un'istruzione UPDATE o INSERT, è possibile determinare il numero di righe nella tabella di relazione che è stato modificato o aggiunto con il metodo `getUpdateCount()` .

Poiché le istruzioni INSERT, UPDATE e DELETE modificano il contenuto del database di relazione, è *necessario* seguire la gestione della transazione per tali istruzioni. Una *transazione* è costituita da una serie di passi operativi eseguiti come unità. Tutte le istruzioni SQL eseguite all'interno di una transazione riescono o non riescono come unità. Tabella 116 illustra i metodi nella classe `DtpConnection` che forniscono il supporto per le transazioni per le query SQL.

Tabella 116. DtpConnection metodi per la gestione delle transazioni

Attività di gestione della transazione	Metodo DtpConnection
Iniziare una transazione.	`beginTran()`
Terminare una transazione, effettuando il commit (salvataggio) di tutte le modifiche durante la transazione al database.	`commit()`
Determinare se una transazione è al momento attiva.	`inTransaction()`
Terminare la transazione, eseguendo il rollback (backout) di tutte le modifiche effettuate durante la transazione.	`rollBack()`

Per contrassegnare l'inizio di una transazione nel database di relazione, utilizzare il metodo `beginTran()`. Eseguire tutte le istruzioni SQL che devono riuscire o non riuscire come unità *tra* questo richiamo in `beginTran()` e la fine della transazione. E' possibile terminare la transazione in due modi:

- Richiamare `commit()` per terminare correttamente la transazione. Tutte le modifiche effettuate dalle istruzioni SQL vengono *salvate* nel database di relazione.
- Richiamare `rollBack()` per terminare correttamente la transazione. Viene effettuato il *backout* di tutte le modifiche effettuate dalle istruzioni SQL nel database di relazione.

E' possibile scegliere le condizioni che causano l'errore di una transazione. Verificare la condizione e richiamare `rollBack()` se si verifica una condizione di non riuscita. Altrimenti, richiamare `commit()` per terminare correttamente la transazione.

```
DtpConnection connection = getRelConnection("SapCust");

// begin a transaction
connection.beginTran();

// insert a row
connection.executeSQL("insert...");

// commit the transaction
```

```
connection.commit();

// release the database connection
releaseRelConnection(true);
```

Per determinare se una transazione è al momento attiva, utilizzare il metodo `inTransaction()`.

Query che richiamano le procedure memorizzate

Una procedura memorizzata è una procedura definita dall'utente che contiene istruzioni SQL e una logica condizionale. Le procedure memorizzate vengono memorizzate in un database. Quando si crea una nuova relazione, Relationship Designer Express crea una procedura memorizzata per conservare ciascuna tabella di relazione.

Tabella 117 illustra i metodi nella classe `DtpConnection` che richiama un procedura memorizzata. Tali metodi sono supportati solo per la compatibilità pregressa: *Non* utilizzare tali metodi per lo sviluppo di un nuovo codice, invece utilizzare i metodi `executeSQL()` e `executeStoredProcedure()` della classe `CwDBConnection`.

Tabella 117. DtpConnection metodi per il richiamo di una procedura memorizzata

Come richiamare la procedura memorizzata	Metodo DtpConnection	Utilizzare
Inviare un'istruzione CALL che esegue la procedura memorizzata nel database di relazione.	`executeSQL()`	Per richiamare una procedura memorizzata che *non* dispone di parametri OUT
Specificare il nome della procedura memorizzata e un vettore dei relativi parametri per creare un richiamo di procedura, che viene inviato al database di relazione per l'esecuzione.	`execStoredProcedure()`	Per richiamare una qualunque procedura memorizzata, compresa quella con i parametri OUT

Nota: E' possibile utilizzare i metodi JDBC per eseguire direttamente una procedura memorizzata. Tuttavia, l'interfaccia fornita dalla Mappatura API è più semplice e riutilizza le risorse del database e può incrementare l'efficienza dell'esecuzione. E' necessario utilizzare la Mappatura API per eseguire le procedure memorizzate.

Come illustrato in Tabella 117, la scelta del metodo da utilizzare per richiamare una procedura memorizzata dipende da:

- Se la procedura fornisce qualche parametro OUT

  Un parametro OUT è un parametro attraverso cui la procedura memorizzata restituisce un valore al codice di richiamo. Se la procedura memorizzata utilizza un parametro OUT, è necessario utilizzare `execStoredProcedure()` per richiamare la procedura memorizzata.
- Il numero di volte in cui viene richiamata la procedura memorizzata

  Il metodo `execStoredProcedure()` precompila la procedura memorizzata. Pertanto se si richiama la stessa procedura memorizzata più di una volta (ad esempio, in un loop), l'utilizzo di `execStoredProcedure()` può essere più rapido di `executeSQL()`, poiché il database di relazione può riutilizzare la versione precompilata.

Le seguenti sezioni descrivono il modo in cui utilizzare i metodi `executeSQL()` e `execStoredProcedure()` per richiamare una procedura memorizzata.

Richiamo delle procedure memorizzate con executeSQL(): Per richiamare una procedura memorizzata con il metodo `executeSQL()`, specificare come argomento al metodo `executeSQL()` una rappresentazione di stringa dell'istruzione CALL che comprende la procedura memorizzata e ogni argomento.

Esempio: il seguente richiamo a `executeSQL()` invia un'istruzione CALL per eseguire la procedura memorizzata `setOrderCurrDate()`:

```
connection.executeSQL("call setOrderCurrDate(345698)");
```

Nota: Nel codice precedente, la variabile `connection`è un oggetto `DtpConnection` ottenuto da un richiamo precedente del metodo `getRelConnection()`.

E' possibile eseguire la procedura memorizzata `setOrderCurrDate()`, poiché il singolo argomento si trova in un parametro IN, ovvero il valore è inviato solo alla procedura memorizzata. La procedura memorizzata non dispone di parametri OUT.

Nota: E' possibile utilizzare la forma `executeSQL()` che accetta un vettore di parametro per inoltrare i valori del parametro. Tuttavia, *non è possibile* utilizzare `executeSQL()` per richiamare una procedura memorizzata che utilizza un parametro OUT. Per eseguire tale procedura memorizzata, *è necessario* utilizzare `execStoredProcedure()`.

Utilizzare un blocco PL/SQL anonimo se si pianifica di richiamare gli oggetti PL/SQL memorizzati Oracle mediante ODBC utilizzando il metodo `DtpConnection executeSQL`. Di seguito è riportato un formato accettabile (il nome della procedura memorizzata è `myproc`):

```
executeSQL("begin myproc(...); end;");
```

Richiamo delle procedure memorizzate con execStoredProcedure(): Per richiamare una procedura memorizzata con il metodo `execStoredProcedure()`:

1. Specificare il nome della procedura memorizzata da eseguire come stringa:
2. Creare un vettore di parametro `Vector` degli oggetti `UserStoredProcedureParam`, che fornisce le informazioni sul parametro (come ad esempio il nome, il tipo e il valore di ciascun parametro).

Un parametro è un valore che è possibile inviare all'interno o all'esterno della procedura memorizzata. Il tipo di parametro in/out determina il modo in cui la procedura memorizzata utilizza il valore del parametro:

- Un parametro IN è *solo per input*: la procedura memorizzata accetta il valore del parametro come input ma non utilizza il parametro per restituire una valore al codice di richiamo.
- Un parametro OUT è *solo per output*: la procedura memorizzata non interpreta il valore del parametro come input ma utilizza il parametro per restituire un valore al codice di richiamo.
- Un parametro IN/OUT è per *input e output*: la procedura memorizzata accetta il valore del parametro come input e utilizza il parametro per restituire una valore al codice di richiamo.

Un oggetto `UserStoredProcedureParam` descrive un parametro singolo per una procedura memorizzata. Tabella 118 illustra le informazioni sul parametro che un oggetto `UserStoredProcedureParam` contiene oltre ai metodi per richiamare e impostare queste informazioni sul parametro.

Tabella 118. Informazioni sul parametro nell'oggetto UserStoredProcedureParam

Informazioni sul parametro	metodo UserStoredProcedureParam
Nome del parametro	`getParamName(), setParamName()`
Valore del parametro	`getParamValue(), setParamValue()`
Tipo di dati del parametro:	
• Come oggetto `Java`	`getParamDataTypeJavaObj(), setParamDataTypeJavaObj()`
• Come tipo di dati JDBC	`getParamDataTypeJDBC(), setParamDataTypeJDBC()`
Tipo in/out parametro	`getParamIOType(), setParamIOType()`
Posizione indice parametro	`getParamIndex(), setParamIndex()`

Passi per l'inoltro dei parametri ad una procedura memorizzata: per inoltrare parametri ad una procedura memorizzata, effettuare i seguenti passi:

1. Creare un oggetto `UserStoredProcedureParam` per conservare le informazioni sul parametro.

   Utilizzare `UserStoredProcedureParam()` il constructor per creare un nuovo oggetto `UserStoredProcedureParam`. Da questo constructor, inoltrare le seguenti informazioni sul parametro per inizializzare l'oggetto:

   - L'indice del parametro indica la posizione all'interno della dichiarazione della procedura memorizzata per questo parametro.
   - Il tipo di dati del parametro è, in genere, il nome dell'`Oggetto` Java che conserva il valore del parametro.
   - Il valore del parametro è un `Oggetto` Java che contiene il valore da assegnare al parametro. Per un parametro OUT, questo valore può essere fittizio, ma il tipo di `Oggetto` deve corrispondere al tipo di dati del parametro OUT nella dichiarazione della procedura memorizzata.
   - Il tipo in/out di parametro specifica se il parametro è IN, INOUT o OUT.
   - Il nome del parametro associa un nome stringa con il parametro.
2. Ripetere il passo 1 per ciascun parametro della procedura memorizzata.
3. Creare un oggetto `Vector` con sufficienti elementi per conservare tutti i parametri della procedura memorizzata.
4. Aggiungere l'oggetto `UserStoredProcedureParam` inizializzato all'oggetto `Vector` del parametro.

   Utilizzare il metodo `addElement()` della classe `Vector` per aggiungere l'oggetto `UserStoredProcedureParam`.
5. Una volta creati tutti gli oggetti `UserStoredProcedureParam` ed aggiunti al vettore del parametro `Vector`, inoltrare questo vettore del parametro come secondo argomento al metodo `execStoredProcedure()`.

   Il metodo `execStoredProcedure()` invia la procedura memorizzata ed i relativi parametri al database di relazione per l'esecuzione.

Nota: Il primo argomento per `execStoredProcedure()` è il nome della procedura memorizzata da eseguire.

Esempio: si supponga di disporre della procedura memorizzata `get_empno()` definita nel modo seguente:

```
create or replace procedure get_empno(emp_id IN number,
      emp_number OUT number) as
   begin
```

```
   select emp_no into emp_number
   from emp
   where emp_id = 1;
end;
```

Questa procedura memorizzata dispone di due parametri:

- Il primo parametro `emp_id` è un parametro IN.

  Pertanto, è necessario inizializzare il relativo oggetto `UserStoredProcedureParam` con un tipo in/out "IN", oltre al tipo di dati appropriato nella procedura memorizzata. Poiché `emp_id` è dichiarato come tipo SQL NUMBER (che conserva un valore intero), il tipo di dati e il valore del parametro devono essere costituiti da un `Oggetto` Java che conserva i valori interi: `Integer`.
- Il secondo parametro `emp_number` è OUT.

  Per questo parametro, creare un oggetto *vuoto* `UserStoredProcedureParam` da inviare alla procedura memorizzata. Inizializzare l'oggetto con un tipo in/out "OUT", oltre al tipo di dati e nome appropriati. Tuttavia, fornire un valore fittizio per questo parametro. Una volta completata l'esecuzione della procedura memorizzata, è possibile ottenere il valore restituito da questo parametro OUT con il metodo `getParamValue()`.

Esempio: il seguente esempio esegue la procedura memorizzata `get_empno()` con il metodo `execStoredProcedure()`:

```
DtpConnection connectn = null;
try
   {
   // Get database connection
   connectn = getRelConnection("Customer");

   // Create parameter Vector
   Vector paramData = new Vector(2);

   // Construct the procedure name
   String sProcName = "get_empno";

   // Create IN parameter
   UserStoredProcedureParam arg_in = new UserStoredProcedureParam(
      1, "Integer", new Integer(6), "IN", "arg_in");

   // Create dummy argument for OUT parameter
   UserStoredProcedureParam arg_out = new UserStoredProcedureParam(
      2, "Integer", new Integer(0), "OUT", "arg_out");

   // Add these two parameters to the parameter Vector
   paramData.addElement(arg_in);
   paramData.addElement(arg_out);

   // Run get_empno() stored procedure
   connectn.execStoredProcedure(sProcName, paramData);

   // Get the result from the OUT parameter
   arg_out = (UserStoredProcedureParam) paramData.elementAt(1);
   Integer emp_number = (Integer) arg_out.getParamValue();
   }
```

Suggerimenti: L'oggetto `Vector` è un vettore basato su zero mentre gli oggetti `UserStoredProcedureParam` sono indicizzati come come vettore su base uno. Nel codice precedente, il parametro OUT viene creato con un valore di indice 2 nel constructor `UserStoredProcedureParam()`, poiché il vettore di questo parametro è basato su uno. Tuttavia, per accedere al valore di questo parametro OUT dal vettore del

parametro `Vector`, il richiamo `elementAt()` specifica un valore di indice 1, poiché questo vettore `Vector` è basato su zero.

Una procedura memorizzata elabora i relativi parametri come tipi di dati SQL. Poiché i tipi di dati SQL e Java *non* sono identici, la Mappatura API deve convertire un valore del parametro tra questi due tipi di dati. Per un parametro IN, la Mappatura API converte il valore del parametro da un `Oggetto` Java al relativo tipo di dati SQL. Per un parametro OUT, la Mappatura API converte il valore del parametro dal tipo di dati SQL ad un `Oggetto` Java. La Mappatura API converte il valore di un parametro tra questi due tipi di dati utilizzando due strati di mappatura del tipo di dati:

- Dal tipo Java al tipo JDBC
- Dal tipo JDBC al tipo di dati SQL

La Mappatura API utilizza internamente il tipo di dati JDBC per conservare il valore del parametro inviato e inoltrato dalla procedura memorizzata. JDBC definisce una serie di identificativi di tipi SQL nella classe `java.sql.Types`. Questi tipi rappresentano i tipi SQL più comunemente utilizzati. Inoltre, JDBC fornisce una mappatura standard dai tipi JDBC ai tipi di dati Java. Ad esempio, un JDBC INTEGER viene normalmente mappato ad un tipo `int` Java.

Per mappare un parametro IN (o INOUT) da un oggetto Java all'equivalente JDBC, la Mappatura API utilizza le mappature in Tabella 119.

Tabella 119. Mappature da un oggetto Java a un equivalente tipo di dati JDBC

Da un oggetto Java	Al tipo di dati JDBC
`Stringa`	CHAR, VARCHAR o LONGVARCHAR
`java.math.BigDecimal`	NUMERIC
`Booleano`	BIT
`Integer`	INTEGER
`Long`	BIGINT
`Float`	REAL
`Double`	DOUBLE
`byte[]`	BINARY, VARBINARY o LONGVARBINARY
`java.sql.Date`	DATE
`java.sql.Time`	TIME
`java.sql.Timestamp`	TIMESTAMP
`Clob`	CLOB
`Blob`	BLOB
`Array`	ARRAY
`Struct`	STRUCT
`Ref`	REF

Per mappare un parametro OUT (o INOUT) pda un tipo di dati JDBC al relativo equivalente dell'oggetto Java, la Mappatura API utilizza le mappature in Tabella 120.

Tabella 120. Mappature dal tipo di dati JDBC all'oggetto Java

Dal tipo di dati JDBC	All'oggetto Java
CHAR, VARCHAR, LONGVARCHAR	`Stringa`
NUMERIC, DECIMAL	`java.math.BigDecimal`
BIT	`Booleano`
TINYINT	`Integer`
SMALLINT	`Integer`

Tabella 120. Mappature dal tipo di dati JDBC all'oggetto Java (Continua)

Dal tipo di dati JDBC	All'oggetto Java
INTEGER	`Integer`
BIGINT	`Long`
REAL	`Float`
FLOAT, DOUBLE	`Double`
BINARY, VARBINARY o LONGVARBINARY	`byte[]`
DATE	`java.sql.Date`
TIME	`java.sql.Time`
TIMESTAMP	`java.sql.Timestamp`
CLOB	`Clob`
BLOB	`Blob`
ARRAY	`Array`
STRUCT	`Struct`
REF	`Ref`

Pertanto, ogni oggetto `UserStoredProcedureParam` contiene due rappresentazioni del relativo tipo di dati, come illustrato in Tabella 121.

Tabella 121. Tipi di dati del parametro

Tipo di dati del parametro	Descrizione	metodo UserStoredProcedureParam
L'`Oggetto` Java	Il tipo di dati che il codice di trasformazione della mappa utilizza per conservare i valori del parametro	`getParamDataTypeJavaObj()`, `setParamDataTypeJavaObj()`
Tipo di dati JDBC	IL tipo di dati che la Mappatura API utilizza internamente per conservare il valore del parametro	`getParamDataTypeJDBC()`, `setParamDataTypeJDBC()`

E' possibile utilizzare i metodi `UserStoredProcedureParam` in Tabella 121 per accedere alle forme del parametro del tipo di dati. Tuttavia, è necessario utilizzare il tipo di dati `Object` Java (come ad esempio `Integer`, `String` o `Float`) per i seguenti motivi:

- Per i parametri IN (e INOUT), *è necessario* fornire il valore del parametro come `Oggetto Java`. Pertanto, fornire il tipo di dati del parametro come `Oggetto` Java è più coerente.
- Il metodo `execStoredProcedure()` invia i parametri in un vettore del parametro `Vector`. L'oggetto `Vector` può contenere solo elementi che sono `Oggetti` Java.

Il parametro `emp_id` di `get_empno()` è dichiarato con il tipo di dati SQL, NUMBER, che contiene un valore intero. Pertanto, nell'esempio del codice avviato per 324, il richiamo per `UserStoredProcedureParam()` per il parametro `emp_id` (parametro con la posizione di indice 1) imposta il relativo valore su 6 con il relativo terzo argomento:

```
new Integer(6)
```

Inoltre, questo richiamo imposta il tipo di parametro sullo stesso tipo di `Oggetto` Java con il relativo secondo argomento:

```
"Integer"
```

Chiusura di una connessione

La connessione al database di relazione è rilasciata quando l'esecuzione della mappa è terminata. Se la mappa viene eseguita correttamente, viene eseguito il commit di tutte le transazioni, se il commit non è già stato eseguito esplicitamente. Se l'esecuzione della mappa ha esito negativo (ad esempio, se viene restituita un'eccezione non gestita con un'istruzione `catch`), viene eseguito il rollback di tutte le transazioni se non è già stato eseguito esplicitamente.

Caricamento e scaricamento delle relazioni

Con `l'utilità repos_copy`, è possibile caricare e scaricare le definizioni della relazione specificate in nel repository.

Nota: Inoltre, è possibile utilizzare `repos_copy` per caricare e scaricare le definizioni della mappa nel repository. Per ulteriori informazioni, consultare "Importazione ed esportazione di mappe da InterChange Server Express" a pagina 85.

Scaricamento della definizione di una relazione

Con l'utilità `repos_copy`, è possibile scaricare le definizioni della relazione specificate nel repository con l'opzione `-e`. Un *file repository della relazione* è un file creato dall'utilità `repos_copy` quando estrae una definizione di relazione dal repository in un file di testo `.jar`).

Esempio: il seguente comando `repos_copy` scarica la definizione di relazione `StateLk` dal repository di InterChange Server Express denominato `WebSphereICS` in un file di repository della relazione:

```
repos_copy -eRelationship:StateLk -oRL_StateLookup.jar
-sWebSphereICS -uadmin -pnull
```

Attenzione: Una relazione *non* è un'entità di prima classe. Pertanto, il relativo spazio dei nomi è separato dalle entità di prima classe. Mentre le entità non di prima classe dispongono dello stesso nome, una relazione può disporre dello stesso nome di un'entità di prima classe (come ad esempio un oggetto business o una collaborazione). Tuttavia, se una definizione di relazione dispone di un nome che corrisponde a una *qualunque* entità di prima classe esistente, non è possibile utilizzare l'opzione `-e` di `repos_copy` per scaricare o caricare la definizione della relazione. E' possibile caricare e scaricare l'intero repository, che comprende le definizioni di relazione.

E' possibile copiare varie definizioni di relazioni in un file di repository della relazione.

Esempio: per copiare entrambe le definizioni di relazione `StateLk` e `CustLkUp`, utilizzare il seguente comando `repos_copy`:

```
repos_copy –eRelationship:StateLk+Relationship:CustLkUp
–oRL_Lookup_Relationships.jar –sWebSphereICS -uadmin -
pnull
```

Caricamento della definizione di una relazione

Inoltre, è possibile utilizzare `repos_copy` per caricare una definizione di relazione nel repository da un file di repository della relazione.

Esempio: il seguente comando repos_copy carica la definizione della relazione StateLk nel repository di InterChange Server Express denominato WebSphereICS:

```
repos_copy -iRL_StateLookup.jar
-sWebSphereICS -uadmin -pnull
```

L'utilità repos_copy effettua le seguenti convalide quando carica una definizione di relazione:

- Convalida l'URL di database della definizione di relazione caricata.
- Convalida che eventuali oggetti dipendenti per la definizione di relazione già esistano nel repository.

Se repos_copy non è in grado di eseguire entrambe queste convalide, non può caricare la definizione di relazione. Tuttavia, repos_copy fornisce delle particolari opzioni della riga comandi per eliminare o limitare tali convalide, come illustrato nella seguente sezione.

Convalida dell'URL del database

L'utilità repos_copy dispone dell'opzione -r per assistere durante il caricamento delle definizioni di relazione in un repository. L'opzione -r istruisce repos_copy ad aggiungere definizioni di relazione al repository senza creare i relativi schemi di runtime. Quando repos_copy effettua il backup di un intero repository (con l'opzione -o), alcune delle informazioni nel file di testo del repository risultante descrivono le definizioni della relazione. Quindi, se si utilizza repos_copy (senza l'opzione -r) per caricare un repository diverso con il contenuto di questo file di testo del repository, repos_copy potrebbe generare errori del seguente formato quando tenta di caricare le definizioni della relazione:

```
Errore del server: si è verificato un errore durante la convalida
delle informazioni di connessione al database di runtime per
la definizione della relazione Customer. L'URL del database
utilizzato è: jdbc:weblogic:mssqlserver4:Cwrelns312@CWDEV:1433.
Il nome di login del database utilizzato è: crossworlds.
Il tipo di database utilizzato è: W55s/wPE/l4=1.
Motivo: SqlServer.
```

La causa di questo errore è il tentativo da parte di repos_copydi convalidare l'URL per il database di relazione. Parte della definizione della relazione è l'URL del database del database di relazione.

Se repos_copy non è in grado di ricercare il database di relazione, genera un errore ed esegue il rollback del carico del repository. Se si sta per effettuare il backup e il ripristino sullo stesso InterChange Server Express (con gli stessi database di relazione), non è necessario includere l'opzione -r. La convalida dell'URL del database di relazione riesce, poiché gli URL di database possono essere localizzati. Pertanto, il repository viene caricato (comprese le definizioni delle relazioni) correttamente.

Tuttavia, nel processo di importazione di una migrazione quando si spostano i dati del repository da una macchina all'altra, l'opzione -r può essere utile. Se si esegue il comando repos_copy in un ambiente che non è in grado di localizzare eventuali database di relazione esistenti nei dati del repository, repos_copy genera un errore di convalida. Per annullare la convalida, includere l'opzione -r di repos_copy quando si carica il repository. Annullando questa convalida, repos_copy può aggiungere correttamente le definizioni di relazione al repository. Utilizza il database del repository corrente come posizione per il database di relazione. E' possibile utilizzare Relationship Designer Express per modificare l'URL del database per indicare una posizione appropriata di ciascuna relazione di database.

Esempio: il seguente comando repos_copy carica la definizione di relazione StateLk nel repository, annullando la convalida del relativo URL del database:

```
repos_copy -rStateLk -iRL_StateLookup.txt -sWebSphereICS -uadmin
-pnull
```

Convalida di oggetti dipendenti

Per impostazione predefinita, repos_copy convalida se tutti gli oggetti dipendenti esistono quando carica una definizione della relazione. Ad esempio, verifica che tutti gli oggetti business interessati nella relazione esistano nel repository. Se tutti gli oggetti dipendenti *non* esistono, repos_copy genera un errore ed esegue il rollback del caricamento del repository. Nella finestra comandi repos_copy, viene visualizzato il seguente messaggio:

```
Alcuni dei partecipanti per le relazioni mancano.
Per ulteriori informazioni, fare riferimento la file di log InterChange Server.
```

Parte 3. Riferimento della mappatura API

Capitolo 9. Classe BaseDLM

I metodi riportati in questo capitolo agiscono su oggetti della classe istanze mappa. Sono definite nella classe definita BaseDLM di IBM WebSphere Business Integration Server Express. La classe BaseDLM è un classe base per tutte le istanze mappa. Tutte le mappe create sono classi secondarie di BaseDLM e ne ereditano i metodi. La classe BaseDLM contiene metodi di utilità per la gestione degli errori, per il debug nelle mappe e per la connessione ad un database. Tutti i metodi di questa classe possono essere richiamati senza fare riferimento al nome della classe.

La Tabella 122 riepiloga i metodi della classe BaseDLM.

Tabella 122. Sommario metodi BaseDLM

Metodo	Descrizione	Pagina
getDBConnection()	Stabilisce una connessione con un database e restituisce un oggetto CwDBConnection.	333
getName()	Richiama il nome delle mappe corrente.	335
getRelConnection()	Stabilisce una connessione con un database relazionale e restituisce un oggetto DtpConnection.	336
implicitDBTransactionBracketing()	Richiama il modello di programmazione della transazione che l'istanza mappa utilizza per tutte le connessioni che ottiene.	337
isTraceEnabled()	Confronta il livello di traccia specificato con il livello di traccia corrente della mappa.	338
logError(), logInfo(), logWarning()	Invia un messaggio di errore, informativo, o di avviso al file log di InterChange Server Express.	338
raiseException()	Genera un eccezione.	340
releaseRelConnection()	Rilascia la connessione con un database relazionale.	341
trace()	Genera un messaggio di traccia.	342

getDBConnection()

Stabilisce una connessione con un database e restituisce un oggetto CwDBConnection.

Sintassi

```
CwDBConnection getDBConnection(String connectionPoolName)
CwDBConnection getDBConnection(String connectionPoolName,
   boolean implicitTransaction)
```

Parametri

connectionPoolName

Il nome di un pool di connessioni valido. Il metodo collega al database la cui connessione è contenuta in questo determinato pool di connessioni.

implicitTransaction
Un valore booleano che indica il modello di programmazione della transazione da utilizzare con il database associato alla connessione. I valori validi sono:

`true`	Il database utilizza parentesi di transazione implicite
`false`	Il database utilizza parentesi di transazione esplicite

Valori restituiti

Restituisce un oggetto `CwDBConnection`.

Eccezioni

`CwDBConnectionFactoryException` – Se si verifica un errore durante il tentativo di connessione al database.

Note

Il metodo `getDBConnection()` ottiene una connessione dal pool di connessioni specificato in *`connectionPoolName`*. Questa connessione permette di effettuare query ed aggiornamenti sul database ed essa associato. Tutte le connessioni in un determinato pool di connessioni sono associate allo stesso database. Il metodo restituisce un oggetto `CwDBConnection` tramite il quale è possibile eseguire query e gestire transazioni sul database. Consultare i metodi nella classe `CwDBConnection` per ulteriori informazioni.

Per valore predefinito tutte le connessioni utilizzano parentesi di transazione implicite come modello di programmazione delle transazioni. Per specificare un modello di programmazione della transazione *per una determinata connessione*, fornire un valore booleano per indicare il modello di programmazione della transazione desiderato come l'argomento facoltativo *`implicitTransaction`* del metodo `getDBConnection()`. La seguente chiamata `getDBConnection()` specifica parentesi di transazione esplicite per la connessione ottenuta dal pool di connessioni `ConnPool`:

```
conn = getDBConnection("ConnPool",false);
```

La connessione viene rilasciata quando l'istanza mappa termina l'esecuzione. Si può chiudere questa connessione in modo esplicito con il metodo `release()`. Si può determinare se una connessione è stata rilasciata con il metodo `isActive()`.

Esempi

L'esempio che segue stabilisce una connessione con un database associato alle connessioni incluse nel pool `CustConnPool`. Utilizza quindi una transazione implicita per inserire ed aggiornare righe in una tabella del database.

```
CwDBConnection connection = getDBConnection("CustConnPool");

// Insert a row
connection.executeSQL("insert...");

// Update rows...
connection.executeSQL("update...");
```

Siccome la chiamata precedente a `getDBConnection()`*non* comprende il secondo argomento facoltativo, questa connessione utilizza parentesi di transazione implicite come modello di programmazione della transazione (a meno che il

modello di programmazione della transazione non venga sovrascritto nella finestra di dialogo Proprietà mappa). Per cui, non si specificano le delimitazioni della transazione esplicita con `beginTransaction()`, `commit()`, e `rollback()`. In realtà, il tentativo di effettuare una chiamata di questi metodi di transazione con parentesi di transazione implicite genera una eccezione `CwDBTransactionException`.

Nota: Si può controllare il modello di programmazione della transazione corrente con il metodo `implicitDBTransactionBracketing()`.

L'esempio che segue stabilisce inoltre una connessione ad un database associato alle connessioni del pool `CustConnPool`. Esso specifica l'uso delle parentesi di transazione esplicite per questa connessione. Quindi, utilizza la transazione esplicita per contenere gli inserimenti e gli aggiornamenti delle righe della tabella del database.

```
CwDBConnection connection = getDBConnection("CustConnPool", false);

// Begin a transaction
connection.beginTransaction();

// Insert a row
connection.executeSQL("insert...");

// Update rows...
connection.executeSQL("update...");

// Commit the transaction
connection.commit();

// Release the connection
connection.release();
```

La chiamata che precede `getDBConnection()` include l'argomento facoltativo *`implicitTransaction`* per impostare il modello di programmazione della transazione con parentesi della transazione esplicite. Quindi, questo esempio utilizza le chiamate di transazione esplicite per indicare i confini della transazione. Se questo metodi di transazione venissero omessi, InterChange Server Express gestirebbe la transazione come implicita.

Consultare inoltre

`Capitolo 13, "Classe CwDBConnection", implicitDBTransactionBracketing(), isActive(), release()`

getName()

Richiama il nome della mappa corrente.

Sintassi

```
String getName()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

Nessuno.

Esempi

L'esempio che segue ottiene il nome della mappa corrente e registra un messaggio di informazione:

```
String mapName = getName();
logInfo(mapName + " is starting");
```

---

getRelConnection()

Stabilisce una connessione ad un database relazionale e restituisce un oggetto `DtpConnection`.

Sintassi

```
DtpConnection getRelConnection(String relDefName)
```

Parametri

relDefName Un nome della definizione di relazione. Il metodo connette ad un database contenente le tabelle di relazioni di questo database relazionale.

Valori restituiti

Restituisce un oggetto `DtpConnection`.

Eccezioni

`DtpConnectionException` – Se si verifica un errore durante il tentativo di connessione al database.

Note

Questo metodo stabilisce una connessione con un database che contiene le tabelle di relazioni utilizzate utilizzate dalla relazione *relDefName*, e permette di effettuare query ed aggiornamenti su un database relazionale. Il metodo restituisce un oggetto `DtpConnection` attraverso cui è possibile eseguire query e gestire transazioni. Consultare i metodi nella classe `DtpConnection` per ulteriori informazioni.

La connessione viene rilasciata quando la mappa termina l'esecuzione. Si può chiudere questa connessione in modo esplicito con il metodo `releaseRelConnection()`.

Esempi

L'esempio che segue stabilisce una connessione con un database contenente le tabelle delle relazioni per la relazione `SapCust`. Quindi utilizza una transazione per eseguire una query per l'inserimento di righe nella tabella nella relazione `SapCust`.

```
DtpConnection connection = getRelConnection("SapCust");

// begin a transaction
connection.beginTran();

// insert a row
connection.executeSQL("insert...");
```

```
// update rows...
connection.executeSQL("update...");

// commit the transaction
connection.commit();
```

Consultare inoltre

getDBConnection(), Capitolo 15, "Classe DtpConnection", releaseRelConnection()

---

implicitDBTransactionBracketing()

Richiama il modello di programmazione della transazione che l'istanza mappa utilizza per tutte le connessioni che ottiene.

Sintassi

```
boolean implicitDBTransactionBracketing()
```

Parametri

Nessuno.

Valori restituiti

Un valore `booleano` che indica il modello di programmazione della transazione da utilizzare in tutte le connessioni del database.

Note

Il metodo `implicitDBTransactionBracketing()` restituisce un valore booleano che indica quale modello di programmazione delle transazione l'istanza mappa assume venga utilizzato da *tutte* le connessioni ottenute, come segue:

- Il valore `true` indica che tutte le connessioni utilizzano parentesi di transazione *implicite*.
- Il valore `false` indica che tutte le connessioni utilizzano parentesi di transazione *esplicite*.

Il metodo è utile prima dell'ottenimento di una connessione per verificare se il modello di programmazione della transazione è appropriato per la connessione.

Nota: Si può sovrascrivere il modello di programmazione della transazione per una determinata connessione con il metodo `getDBConnection()`.

Esempi

L'esempio che segue assicura che l'istanza mappa utilizza parentesi di transazione esplicite per il database associato alla connessione `conn`:

```
if (implicitDBTransactionBracketing())
   CwDBConnection conn = getDBConnection("ConnPool", false);
```

Consultare inoltre

getDBConnection()

isTraceEnabled()

Confronta il livello di traccia specificato con il livello di traccia corrente della mappa.

Sintassi

```
Boolean isTraceEnabled(int traceLevel)
```

Parametri

traceLevel Il livello di traccia da confrontare con il livello di traccia corrente.

Valori restituiti

Restituisce `true` se il livello di traccia corrente del sistema è impostato sul livello di traccia specificato; restituisce `false` se i due livelli di traccia sono differenti.

Note

Il metodo `isTraceEnabled()` è utile per determinare se registrare un messaggio di traccia o meno. Siccome l'attività di traccia può ridurre le prestazioni, questo metodo è utile nella fase di sviluppo di un progetto.

Esempi

```
if ( isTraceEnabled(3) )
   {
   trace("Print this level-3 trace message");
   }
```

logError(), logInfo(), logWarning()

Invia un messaggio di errore, informativo, o di avviso al file di log di InterChange Server Express .

Sintassi

```
void logError(String message)
void logError(int messageNum)
void logError(int messageNum, String param [,...])
void logError(int messageNum, Object[] paramArray)

void logInfo(String message)
void logInfo(int messageNum)
void logInfo(int messageNum, String param [,...])
void logInfo(int messageNum, Object[] paramArray)

void logWarning(String message)
void logWarning(int messageNum)
void logWarning(int messageNum, String param [,...])
void logWarning(int messageNum, Object[] paramArray)
```

Parametri

message Il testo del messaggio.

messageNum Il numero di un messaggio nel file testo messaggio.

param Un parametro singolo. Ci possono essere fino a cinque parametri separati da virgole. Vengono risolti in modo sequenziale in un parametro nel testo del messaggio.

paramArray Un vettore di parametri.

Valori restituiti

Nessuno.

Eccezioni

Nessuno.

Note

Questo metodo invia un messaggio al destinatario della registrazione del InterChange Server Express . Il destinatario della registrazione può essere un file, una finestra oppure entrambi.

Per valore predefinito, il destinatario della registrazione è il file `InterchangeSystem.log`. Si può modificare il destinatario della registrazione inserendo un valore per il parametro `LOG_FILE` nel file di configurazione `InterchangeSystem.cfg`. Il valore del parametro può essere un nome file, `STDOUT` (che scrive il log nella finestra comandi del server), o entrambi.

All'interno di ogni set di metodi:

- Il primo modulo è autonomo e include tutto il testo necessario per la generazione di un messaggio.
- Il secondo modulo genera un messaggio che non contiene parametri.
- Il terzo modulo contiene un numero di messaggio ed un set di valori di parametri.
- Il quarto modulo è un vettore di parametri.

Tutti i moduli del metodo che ricevono un parametro *messageNum* richiedono l'uso di un file di messaggi che viene indicizzato in base al numero del messaggio. Per informazioni su come impostare un file di testo dei messaggi, consultare Capitolo 27, "File di messaggi", a pagina 517.

Esempi

L'esempio che segue registra un messaggio informativo, utilizzando `getString()` per ottenere un valore di attributo da registrare nel messaggio.

```
logInfo("Item shipped. CustomerID: "
   + fromCustomerBusObj.getString("CustomerID"));
```

L'esempio seguente registra un messaggio di errore il cui testo è contenuto nel file di messaggio della mappa. Il messaggio, che è il decimo del file di messaggi, prende due parametri: il cognome del cliente (attributo `LName`) ed il nome del cliente (attributo `FName`).

```
logError(10, customer.get("LName"), customer.get("FName");
```

L'esempio seguente registra un messaggio di errore utilizzando un vettore di parametri. A scopo descrittivo, l'esempio utilizza un vettore di soli due parametri. L'esempio dichiara il vettore `args` , che contiene due elementi, l'ID del cliente ed il nome del cliente. Il metodo `logError()` quindi registra l'errore, utilizzando il numero di messaggio 12 ed i valori nel vettore `args`.

```
Object[] args = {
      fromCustomerBusObj.getString("CustomerID"),
      fromCustomerBusObj.getString("CustomerName");
}

logError(12, args);
```

Consultare inoltre

`trace()`

---

raiseException()

Genera un'eccezione.

Sintassi

```
void raiseException(String exceptionType, String message)

void raiseException(String exceptionType, int messageNum,
   String parameter[,...])

void raiseException(RunTimeEntityException exception)
```

Parametri

exceptionType Una delle seguenti costanti definite di IBM WebSphere Business Integration Server Express:

`AnyException`	Qualsiasi tipo di eccezione
`AttributeException`	Problema di accesso attributo. Ad esempio, la collaborazione chiamata `getDouble()` su un attributo String o chiamata `getString()` su un attributo non esistente.
`JavaException`	Problemi con il codice Java che non è parte di IBM WebSphere Business Integration Server Express API.
`ObjectException`	L'oggetto business passato ad un metodo non era valido oppure è stato acceduto un oggetto nullo.
`OperationException`	Una chiamata di servizio è stata impostata in modo errato e non può essere inviata.
`ServiceCallException`	Chiamata servizio non riuscita. Ad esempio, un connettore o un'applicazione non sono disponibili.
`SystemException`	Qualsiasi errore interno nel sistema IBM WebSphere Business Integration Server Express.

message Una stringa di testo che include il messaggio di eccezione nella chiamata del metodo.

messageNum Il riferimento ad un messaggio numerato nel file di messaggi della mappa.

parameters Un valore per il parametro nel messaggio stesso. Ci possono essere fino a cinque parametri nella chiamata del metodo.

exception Il nome di una variabile dell'oggetto dell'eccezione.

Valori restituiti

Nessuno.

Note

Il metodo `raiseException()` ha tre moduli:

- Il primo modulo del metodo crea una nuova eccezione, passando un tipo di eccezione ed una stringa. Utilizzato per includere un messaggio nella chiamata metodo stessa.
- Il secondo modulo crea una nuova eccezione, passando un tipo di eccezione ed il riferimento ad un messaggio nel file di messaggi della mappa. La chiamata metodo può contenere fino a cinque parametri separati da virgole.
- Il terzo modulo genera un oggetto eccezione che la mappa ha gestito in precedenza. Ad esempio, un passo in cui è prevista una trasformazione potrebbe ricevere un'eccezione, assegnarla ad una variabile, e proseguire. Alla fine il passo di trasformazione genera l'eccezione.

Nota: Tutti i moduli del metodo che ricevono un parametro *messageNum* richiedono l'uso di un file di messaggi che viene indicizzato in base al numero del messaggio. Per informazioni su come impostare un file di testo dei messaggi, consultare Capitolo 27, "File di messaggi", a pagina 517.

Esempi

L'esempio che segue utilizza il primo modulo del metodo per generare un'eccezione del tipo `ServiceCallException`. Il testo è incluso nella chiamata metodo.

```
raiseException(ServiceCallException,
   "Attempt to validate Customer failed.");
```

Il prossimo esempio genera un'eccezione del tipo `ServiceCallException`. Il messaggio contenuto nel file messaggi è il seguente:

```
23
Customer update failed for CustomerID={1} CustomerName={2}
```

Il metodo `raiseException()` richiama il messaggio, poi i valori dei parametri del messaggio dalla variabile `fromCustomer`, e li passa alla chiamata `raiseException()`.

```
raiseException(ServiceCallException, 23,
   fromCustomer.getString("CustomerID"),
   fromCustomer.getString("CustomerName"));
```

L'esempio finale genera un'eccezione gestita in precedenza. La variabile definita dal sistema `currentException` è un oggetto eccezione che contiene l'eccezione.

```
raiseException(currentException);
```

releaseRelConnection()

Rilascia la connessione con un database relazionale.

Sintassi

```
void releaseRelConnection(Boolean doCommit)
```

Parametri

doCommit L'indicatore segnala se questo metodo deve effettuare una chiamata al metodo `DtpConnection.commit()` prima di rilasciare la connessione al database.

Valori restituiti

Nessuno.

Eccezioni

`DtpConnectionException` – Se si verifica un errore durante il tentativo di rilascio della connessione al database oppure se i commit o rollback hanno avuto esito negativo.

Note

Il metodo `releaseRelConnection()` rilascia la connessione per questa mappa specifica. Effettua il commit oppure il rollback della transazione del database sulla base del valore del suo argomento *doCommit* , come segue:

- Se *doCommit* è `true`, `releaseRelConnection()` assume di essere stato chiamato dopo la conclusione positiva dell'operazione sul database ed è quindi non è rischioso effettuare il commit della transazione.
- Se *doCommit* è `false`, `releaseRelConnection()` assume di essere stato chiamato come conseguenza di una eccezione e quindi è necessario effettuare il rollback della transazione.

Dopo che`releaseRelConnection()` ha eseguito l'azione scelta sulla transazione del database, rilascia la connessione che il thread corrette utilizza in modo esclusivo.

Consultare inoltre

`getRelConnection()`, `release()`

---

trace()

Genera un messaggio di traccia.

Sintassi

```
void trace(String traceMsg)
void trace(int traceLevel, String traceMsg)
void trace(int traceLevel, int messageNum)
void trace(int traceLevel, int messageNum, String param [,...])
void trace(int traceLevel, int messageNum, Object[] paramArray)
```

Parametri

traceLevel Il livello di traccia che causa la generazione del messaggio.

traceMsg Una stringa scritta nel file di traccia.

messageNum Un numero che identifica un messaggio nel file di messaggi della mappa.

param Un parametro singolo. Si possono aggiungere ulteriori parametri, separati da virgole, fino ad un massimo di cinque.

paramArray Un vettore di parametri.

Note

Il metodo `trace()` genera un messaggio che la mappa stampa se le traccia è spenta. Questo metodo ha cinque moduli:

- Il primo modulo prende un messaggio stringa che viene visualizzato quando le traccia è attivata a qualsiasi livello.
- Il secondo modulo prende un livello di traccia ed un messaggio stringa che viene visualizzato quando la traccia è attivata ad un livello determinato o ad un livello superiore.
- Il terzo modulo prende un livello di traccia ed un numero che identifica un messaggio nel file di messaggi della mappa. L'intero testo del messaggio viene visualizzato nel file di messaggi e viene stampato senza parametri, quando è impostata la traccia ad un determinato livello o ad un livello superiore.
- Il quarto modulo prende un livello di traccia, un numero che identifica un messaggio nel file di messaggi della mappa, ed uno o più parametri da usare nel messaggio. Si possono inviare fino a cinque valori di parametri da usare con il messaggio separandoli con una virgola.
- Il quinto modulo prende un livello di traccia, un numero che identifica un messaggio nel file di messaggi della mappa, ed un vettore di valori di parametri.

Nota: Tutti i moduli del metodo che ricevono un parametro *messageNum* richiedono l'uso di un file di messaggi che viene indicizzato in base al numero del messaggio. Per informazioni su come impostare un file di testo dei messaggi, consultare Capitolo 27, "File di messaggi", a pagina 517.

Si può impostare un livello di traccia per una mappa nelle Proprietà della mappa.

Esempi

L'esempio che segue genera un messaggio di traccia a Livello 2 e fornisce il testo del messaggio:

```
trace (2, "Starting to trace at Level 2");
```

L'esempio seguente stampa il messaggio 201 del file di messaggi della mappa se il livello di traccia è 2 o superiore. Il messaggio ha due parametri, un nome ed un anno, per il quali questa chiamata metodo passa valori.

```
trace(2, 201, "DAVID", "1961");
```

Consultare inoltre

`logError()`, `logInfo()`, `logWarning()`

Capitolo 10. Classe BusObj

I metodi riportati in questo capitolo agiscono sugli oggetti della classe `BusObj`.

Nota: La classe `BusObj` viene utilizzata sia per lo sviluppo di collaborazioni che per la mappatura. Consultare la sezione Note per i problemi relativi all'utilizzo dei singoli metodi.

Le prime due sezioni di questo capitolo descrivono le eccezioni elencate per questi metodi e come definire attributi ed oggetti business secondari in un oggetto business gerarchico. Le sezioni restanti descrivono i metodi elencati nella Tabella 123.

Tabella 123. Sommario metodi BusObj

Metodo	Descrizione	Pagina
`copy()`	Copia tutti i valori degli attributi dall'oggetto business di input in questo.	347
`duplicate()`	Crea un oggetto business (oggetto `BusObj`) esattamente come questo.	348
`equalKeys()`	Confronta i valori dell'attributo chiave di questo oggetto business con quelli dell'oggetto business di input.	349
`equals()`	Confronta i valori dell'attributo di questo oggetto business con quelli dell'oggetto business di input, compresi gli oggetti business secondari.	349
`equalsShallow()`	Confronta i valori dell'attributo di questo oggetto business con quelli dell'oggetto business di input, ad esclusione degli oggetti business secondari.	350
`exists()`	Testa esistenza di un attribuito Oggetto Business con un specificato nome.	351
`getBoolean()`, `getDouble()`, `getFloat()`, `getInt()`, `getLong()`, `get()`, `getBusObj()`, `getBusObjArray()`, `getLongText()`, `getString()`	Richiama il valore di un singolo attributo da un oggetto business.	351
`getLocale()`	Richiama la locale dei dati dell'oggetto business.	353
`getType()`	Richiama il nome della definizione dell'oggetto business su cui si basa questo oggetto business.	353
`getVerb()`	Richiama le istruzioni di questo oggetto business.	354
`isBlank()`	Controlla se il valore di un attributo è una stringa vuota.	354
`isKey()`	Controlla se un attributo dell'oggetto business è definito come attributo chiave.	355
`isNull()`	Controlla se il valore di un attributo di un oggetto business è nullo.	355
`isRequired()`	Controlla se un attributo dell'oggetto business è definito come attributo obbligatorio.	356

Tabella 123. Sommario metodi BusObj (Continua)

Metodo	Descrizione	Pagina
`keysToString()`	Richiama i valori di un attributo chiave primario dell'oggetto business come una stringa.	357
`set()`	Imposta un attributo di un oggetto business ad un specifico valore di un determinato tipo di dati.	357
`setContent()`	Imposta i contenuti di questo oggetto business in un altro oggetto business.	359
`setDefaultAttrValues()`	Imposta tutti i valori predefiniti degli attributi.	360
`setKeys()`	Imposta i valori degli attributi chiave di questo oggetto business ai valori degli attributi chiave di un altro oggetto business.	360
`setLocale()`	Imposta la locale dell'oggetto business corrente.	360
`setVerb()`	Imposta l'istruzione di un oggetto business.	361
`setVerbWithCreate()`	Crea l'istanza dell'oggetto business secondario e ne imposta l'istruzione.	361
`setWithCreate()`	Imposta l'attributo dell'oggetto business ad un determinato valore di un particolare tipo di dati, creando un oggetto per il valere se questo non esiste.	362
`toString()`	Restituisce i valori di tutti gli attributi di un oggetto business come stringa.	363
`validData()`	Controlla se un valore specificato è di tipo corretto per l'attributo specificato.	363

Eccezioni e tipi di eccezioni

I metodi per i quali le eccezioni o i tipi di eccezione sono elencati restituiscono l'eccezione `CollaborationException`. Alcuni metodi hanno elencate sia le eccezioni che i tipi di eccezione. Entrambi fanno riferimento ad un oggetto `CollaborationException` e differiscono come segue:

- Una *Eccezione* è una classe che è stata resa classe secondaria dell'oggetto `CollaborationException`. Se esiste una eccezione classe secondaria, la si può utilizzare nella mappatura per determinare in modo più preciso le cause del problema.
- Un *Tipo di eccezione* è una parte di dati in un oggetto `CollaborationException`. Gli sviluppatori di collaborazioni utilizzano questo tipo di eccezioni per individuare le eccezioni attraverso l'interfaccia utente del Designer. Inoltre, tutti gli utenti di `BusObj` possono utilizzare questo campo per determinare la causa di un errore se non è stata restituita alcuna classe eccezione più dettagliata di `CollaborationException`.

Sintassi per l'esplorazione di oggetti business gerarchici

Durante la scrittura di codice che richiede l'esplorazione di oggetti business gerarchici, si rende necessario utilizzare la sintassi che permette di specificare gli attributi in elementi di vettori di oggetti business, che sono elementi di vettori di oggetti business secondari, ed altre complessità simili. Questo capitolo specifica la sintassi da utilizzare.

La definizione di un attributo può essere:

```
[[attributeName[index].]...]attributeName
```

La sintassi arriva ad uno dei seguenti formati:

```
attributeName
attributeName[index].attributeName
attributeName[index]... .attributeName
```

Nota: Non utilizzare il punto (.) quando si crea un nome di un attributo di un oggetto business. Se l'attributo di un oggetto business ha un punto nel proprio nome, una mappa di IBM WebSphere Business Integration Server Express interpreta il punto come un operatore punto di Java e ne assegna significati particolari. Ad esempio, "nome.attributo" sarà interpretato come se "attributo" fosse un campo oppure un metodo dell'oggetto "nome".

Definizione di un attributo di tipo base

L'esempio che segue utilizza il metodo `busObj.get()` per richiamare un attributo di tipo base chiamato `OrderID` dall'oggetto business `orderObj`.

```
orderObj.get("OrderID");
```

Definizione di un attributo in un oggetto business secondario

L'esempio che segue assume che `orderObj` sia un oggetto business gerarchico. Un dei suoi attributi è `CustomerInfo`, un oggetto business secondario a cardinalità singola. L'esempio richiama il nome cliente dall'attributo `CustomerName` di `CustomerInfo`.

```
orderObj.get("CustomerInfo.CustomerName");
```

Definizione di un attributo in un secondario di un oggetto business secondario

Se esiste un catena di oggetti business secondari, in cui `CustomerInfo` è secondario rispetto a`orderObj` e `AddressInfo` è secondario rispetto a `CustomerInfo`, è possibile richiamare l'informazione della città da `AddressInfo` come segue:

```
orderObj.get("CustomerInfo.AddressInfo.City");
```

Definizione di un attributo in un elemento di un vettore di oggetti business secondari

Si può anche fare riferimento ad un oggetto business secondario in un vettore specificandone l'indice nel vettore. Il primo elemento nel vettore inizia sempre con lo zero. L'esempio che segue richiama il valore dell'attributo Quantity del terzo oggetto business secondario di un vettore.

```
orderObj.get("LineItem[2].Quantity");
```

copy()

Copia tutti i valori dell'attributo dall'oggetto businessdi input in questo.

Sintassi

```
void copy(BusObj inputBusObj)
```

Parametri

inputBusObj Il nome dell'oggetto business i cui valori degli attributi sono copiati nell'oggetto business corrente.

Note

Il metodo `copy()` copia l'intero oggetto business, compresi tutti gli oggetti business secondari e i vettori di oggetti business secondari. Il metodo non imposta riferimenti all'oggetto copiato. Al contrario clona gli attributi, in pratica crea copie separate degli attributi.

Esempi

L'esempio che segue copia i valori contenuti in `sourceCustomer` in `destCustomer`.

```
destCustomer.copy(sourceCustomer);
```

L'esempio che segue crea tre oggetti business (`myBusObj`, `myBusObj2`, e `mysettingBusObj`) ed imposta l'attributo `attr1` di `myBusObj` con il valore contenuto in `mysettingBusObj`. Quindi clona tutti gli attributi di `myBusObj` in `myBusObj2`.

```
BusObj myBusObj = new BusObj();
BusObj myBusObj2 = new BusObj();

BusObj mySettingBusObj = new BusObj();

myBusObj.set("attr1", mySettingBusObj);
myBusObj2.copy(myBusObj);
```

Dopo l'esecuzione di questo frammento, `myBusObj.attr1` e `myBusObj2.attr1` vengono *entrambi* impostati nell'oggetto business `mySettingBusObj`. Se però, `mySettingBusObj` viene modificato, si modifica anche `myBusObj.attr1` ma non `myBusObj2.attr1`. I valori degli attributi di `myBusObj2` sono stati clonati perché sono stati impostati con `copy()`. Quindi, il valore di `attr1` in `myBusObj2` è ancora il valore originale `mySettingBusObj.attr1`*precedente* alla modifica.

---

duplicate()

Crea un oggetto business (oggetto `BusObj`) esattamente uguale a questo.

Sintassi

```
BusObj duplicate()
```

Valori restituiti

L'oggetto business duplicato.

Eccezioni

`CollaborationException`—Il metodo `duplicate()` può impostare i seguenti tipi di eccezione per questa eccezione: `ObjectException`.

Note

questo metodo crea un clone del oggetto business e lo restituisce. Si deve assegnare in modo esplicito il valore di restituzione di questa chiamata metodo ad una variabile dichiarata del tipo `BusObj`.

Esempi

L'esempio che segue duplica `sourceCustomer` in modo da creare `destCustomer`.

```
BusObj destCustomer = sourceCustomer.duplicate();
```

equalKeys()

Confronta i valori chiave dell'oggetto business con quelli nell'oggetto business di input.

Sintassi

```
boolean equalKeys(BusObjinputBusObj)
```

Parametri

inputBusObj L'oggetto business da confrontare con questo oggetto business.

Valori restituiti

Restituisce `true` se i valori di tutti gli attributi chiave sono uguali. Restituisce `false` se non lo sono.

Eccezioni

`CollaborationException`—Il metodo `equalKeys()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `ObjectException` – Impostata se l'argomento dell'oggetto business non è valido.

Consultare inoltre

`equalsShallow()`, `equals()`

Note

Questo metodo effettua un confronto superficiale, non confronta le chiavi negli oggetti business secondari.

Esempi

L'esempio che segue confronta i valori chiave di `order2` con quelli di `order1`.

```
boolean areEqual = order1.equalKeys(order2);
```

equals()

Confronta i valori dell'attributo di questo oggetto business con quelli nell'oggetto business in input, compresi gli oggetti business secondari.

Sintassi

```
-boolean equals(Object inputBusObj)
```

Parametri

inputBusObj Un oggetto business da confrontare con questo oggetto business.

Valori restituiti

Restituisce `true` se i valori di tutti gli attributi sono uguali. Altrimenti restituisce `false`.

Eccezioni

`CollaborationException`—Il metodo `equals()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `ObjectException` – Impostata se l'argomento dell'oggetto business non è valido.

Note

Questo metodo confronta i valori dell'attributo di questo oggetto business con quelli dell'oggetto business di input. Se gli oggetti business sono gerarchici, il confronto comprende anche tutti gli attributi degli oggetti business secondari.

Nota: L'immissione dell'oggetto business come `Oggetto` assicura che questo metodo `equals()` ricopra il metodo `Object.equals()`.

In questo confronto, il valore null viene considerato equivalente a qualsiasi altro valore con cui viene confrontato e non impedisce la restituzione di `true`.

Consultare inoltre

`equalsShallow()`, `equalKeys()`

Esempi

L'esempio che segue confronta tutti gli attributi di `order2` con tutti quelli di `order1` ed assegna il risultato del confronto alla variabile `areEqual`. Nel caso vi fosse un oggetto business secondario verrebbero confrontati anche i suoi attributi.

```
boolean areEqual = order1.equals(order2);
```

---

equalsShallow()

Confronta i valori dell'attributo di questo oggetto business con quelli nell'oggetto business in input, esclusi gli oggetti business secondari.

Sintassi

```
boolean equalsShallow(BusObj inputBusObj)
```

Parametri

inputBusObj Un oggetto business da confrontare con questo oggetto business.

Valori restituiti

Restituisce `true` se i valori di tutti gli attributi sono uguali. Altrimenti restituisce `false`.

Eccezioni

`CollaborationException`—Il metodo `equalsShallow()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `ObjectException` – Impostata se l'argomento dell'oggetto business non è valido.

Consultare inoltre

`equals()`, `equalKeys()`

Esempi

L'esempio che segue confronta gli attributi di `order2` con quelli di `order1`, esclusi gli attributi degli oggetti business secondari se ve ne dovessero essere.

```
boolean areEqual = order1.equalsShallow(order2);
```

---

exists()

Verifica la presenza di un attributo di un oggetto business con un determinato nome.

Sintassi

```
boolean exists(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori restituiti

restituisce `true` se l'attributo esiste, altrimenti restituisce `false`.

Esempi

Nell'esempio che segue si verifica se l'oggetto business `order` ha un attributo chiamato `Notes`.

```
boolean notesAreHere = order.exists("Notes");
```

---

getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()

Richiama il valore di un singolo attributo di un oggetto business.

Sintassi

```
Object get(String attribute)
Object get(int position)

boolean getBoolean(String attribute)
double getDouble(String attribute)
float getFloat(String attribute)
int getInt(String attribute)
long getLong(String attribute)
Object get(String attribute)
BusObj getBusObj(String attribute)
BusObjArray getBusObjArray(String attribute)
String getLongText(String attribute)
String getString(String attribute)
```

Parametri

attribute Il nome di un attributo.

position Un numero intero che definisce la posizione ordinale di un attributo nell'elenco attributi di un oggetto business.

Valori restituiti

Il valore dell'attributo specificato.

Eccezioni

`CollaborationException`—I metodi `get` possono impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException` - Impostata se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere causata se la collaborazione richiama `getDouble()` su un attributo String che non è formato da cifre oppure richiama `getString()` su un attributo inesistente.

Note

Il metodo `get()` richiama un valore dell'attributo dall'oggetto business corrente. Questo restituisce un copia del valore dell'attributo. *Non* restituisce un riferimento all'oggetto di questo attributo nel oggetto business di origine. Pertanto, qualsiasi modifica del valore dell'attributo nell'oggetto business di origine *non* si riflette sul valore che è stato restituito da `get()`. Ogni volta che questo metodo viene richiamato, esso restituisce una nuova copia (clone) dell'attributo.

Il metodo `get()` contempla i seguenti moduli:

- Il primo modulo restituisce un valore del tipo definito nel nome del metodo. Ad esempio, `getBoolean()` restituisce un valore boolean, `getBusObj()` restituisce un valore BusObj, `getDouble()` restituisce un valore double, e così via. Per quanto, `getLongText()` restituisce un oggetto String poiché il tipo longtext di InterChange Server Express è un oggetto String con nessuna dimensione massima. Utilizzare questi moduli per richiamare attributi con basi specifiche oppure tipi di dati definiti da InterChange Server Express.

  Questi metodi permettono di accedere al valore di un attributo specificando il *nome* dell'attributo.
- Il secondo modulo, `get()` richiama il valore di un attributo di *qualsiasi* tipo. Si può effettuare il cast del valore restituito al valore appropriato al tipo di attributo.

  Questo metodo permette di accedere al valore di un attributo specificando *sia* il *nome* del attributo che la *posizione* nell'indice dell'attributo all'in terno dell'elenco attributi dell'oggetto business.

Esempi

L'esempio che segue illustra come `get()` restituisca una copia (clone) del valore dell'attributo invece del il riferimento all'oggetto:

```
BusObj mySettingBusObj = new BusObj();
BusObj myBusObj = new BusObj();

myBusObj.set("attr1", mySettingBusObj);

BusObj Extract = myBusObj.get("attr1");
```

Dopo l'esecuzione di questo frammento, se si modifica l'oggetto business `Extract`, `mySettingBusObj` *non* cambierà perché la chiamata `get()` ha restituito una copia dell'attributo `attr1`.

L'esempio che segue utilizza `getBusObj()` per richiamare un oggetto business secondario contenente l'indirizzo di un cliente dall'oggetto business `customer` ed assegnarlo alla variabile `address`.

```
BusObj address = customer.getBusObj("Address");
```

L'esempio che segue utilizza `getString()` per richiamare il valore dell'attributo `CustomerName`. La variabile dell'oggetto business è `sourceCustomer`.

```
String customerName = sourceCustomer.getString("CustomerName");
```

L'esempio che segue utilizza `getInt()` per richiamare i valori `Quantity` da due oggetti business le cui variabili sono `item1` e `item2`. L'esempio quindi calcola la somma di entrambe le quantità.

```
int sumQuantity = item1.getInt("Quantity") + item2.getInt("Quantity");
```

L'esempio seguente richiama l'attributo `Item` dalla variabile dell'oggetto business `order`. L'attributo `Item` è un vettore di oggetti business.

```
BusObjArray items = order.getBusObjArray("Item");
```

L'esempio che segue richiama il valore dell'attributo `CustID` dall'oggetto business di origine ed imposta il valore di `Customer` dell'oggetto business di destinazione in modo che corrisponda.

```
destination.set("Customer", source.get("CustID"));
```

L'esempio che segue accede il valore dell'attributo utilizzandone la posizione delle orinate all'interno dell'elenco attributi:

```
for i=0; i<maxAttrCount; i++)
{
        String strValue = (String)myBusObj.get(i);
    ...
```

getLocale()

Richiama la locale associata ai dati dell'oggetto business.

Sintassi

```
java.util.Locale getLocale()
```

Parametri

Nessuno.

Valori restituiti

Un oggetto Java Locale che contiene informazioni relative alla locale dell'oggetto business. L'oggetto Locale deve essere un'istanza della classe `java.util.Locale`.

Note

Il metodo `getLocale()` restituisce la locale associata ai dati dell'oggetto business. La locale è spesso diversa da quella della collaborazione in cui essa è eseguita.

Consultare inoltre

getLocale() (classe `BaseCollaboration`), setLocale()

getType()

Richiama il nome della definizione dell'oggetto business su cui si basa questo oggetto business.

Sintassi

```
String getType()
```

Valori restituiti

Il nome di una definizione di un oggetto business.

Note

Il tipo di oggetto business, per quanto riguarda questo metodo, è il nome della definizione dell'oggetto business da cui è stato creato l'oggetto business.

Restituzioni

L'esempio che segue richiama il tipo di un oggetto business chiamato `sourceShipTo`.

```
String typeName = sourceShipTo.getType();
```

L'esempio che segue copia un evento di attivazione in un nuovo oggetto business del tipo appropriato.

```
BusObj source = new BusObj(triggeringBusObj.getType());
```

---

getVerb()

Richiama l'istruzione di questo oggetto business.

Sintassi

```
String getVerb()
```

Valori restituiti

Il nome di un istruzione, quale Create, Retrieve, Update, o Delete.

Note

Nello sviluppo di una collaborazione, questo metodo è utile in ambienti dove vengono gestiti più tipi di eventi in ingresso. Il primo nodo azione in uno scenario richiama `getVerb()`. La transizione in uscita si collega a questo nodo azione e controlla i contenuti della stringa restituita, in modo che ogni collegamento di transizione in uscita diventa il punto di partenza di un percorso di esecuzione che gestisce una delle possibili istruzioni.

Esempi

Nell'esempio che segue si ottiene l'istruzione dall'oggetto business chiamato `orderEvent`, e la si assegna ad una variabile chiamata `orderVerb`.

```
String orderVerb = orderEvent.getVerb();
```

---

isBlank()

Verifica se il valore di un attributo è una stringa vuota.

Sintassi

```
boolean isBlank(String attribute)
```

Parametri

attribute Il nome di un attributo.

Restituzioni

Restituisce `true` se il valore dell'attributo è una stringa vuota, altrimenti restituisce `false`.

Note

Una stringa vuota è paragonabile ad una stringa `""`. È diversa da una stringa nulla, la cui presenza viene rilevata dal metodo `isNull()`.

Se una collaborazione deve richiamare il valore di un attributo, per poi utilizzarlo in qualche modo, può effettuare una chiamata `isBlank()` e `isNull()` per controllare se ha un valore prima di richiamarlo.

Esempi

L'esempio che segue controlla se l'attributo `Material` dell'oggetto business `sourcePaperClip` è una stringa vuota.

```
boolean key = sourcePaperClip.isBlank("Material");
```

---

isKey()

Controlla se l'attributo dell'oggetto business è definito come attributo chiave.

Sintassi

```
boolean isKey(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori restituiti

Restituisce `true` se l'attributo è un attributo chiave, altrimenti restituisce `false`.

Esempi

L'esempio che segue determina se l'attributo `CustID` dell'oggetto business `customer` è un attributo chiave.

```
boolean keyAttr = (customer.isKey("CustID"));
```

---

isNull()

Verifica se il valore di di un attributo di un oggetto business è nullo.

Sintassi

```
boolean isNull(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori restituiti

Restituisce `true` se il valore dell'attributo è nullo, altrimenti restituisce `false`.

Note

Un valore nullo indica nessun valore, a differenza di una stringa vuota che viene rilevata da una chiamata `isBlank()`. Controlla un oggetto con `isNull()` prima di utilizzarlo, perché se l'oggetto fosse nullo, l'operazione potrebbe avere esito negativo.

Un attributo può essere nullo nelle seguenti circostanze:

- Il valore dell'attributo è stato esplicitamente impostato su nullo.

  Un valore di attributo può essere impostato su nullo con il metodo `set()`.
- Il valore dell'attributo non è stato mai impostato.

  Nella creazione di un'istanza di un nuovo oggetto business, tutte i valori di un attributo sono impostati su nullo. Se il valore dell'attributo non è stato impostato tra la sua creazione ed il momento della chiamata `isNull()` è ancora nullo.
- Il valore nullo è stato inserito durante la mappatura.

  Quando una collaborazione elabora un oggetto business ricevuto da un connettore, il processo di mappatura potrebbe avere inserito il valore nullo. Il processo di mappatura converte l'oggetto business specifico dell'applicazione ricevuto dal connettore in un oggetto business generico gestito dalla collaborazione. La mappa assegna un valore nullo ad ogni attributo nell'oggetto business generico che non ha un corrispondente nell'oggetto specifico dell'applicazione.

Suggerimenti: Effettuare sempre una chiamata `isNull()` prima di eseguire un'operazione su un attributo che è un oggetto business secondario o un vettore di oggetti business secondari, poiché Java non consente operazioni su oggetti nulli.

Esempi

L'esempio che segue controlla se l'attributo `Material` dell'oggetto business `sourcePaperClip` è nullo.

```
boolean key = sourcePaperClip.isNull("Material");
```

L'esempio che segue controlla se l'attributo `CustAddr` dell'oggetto business `contract1` è nullo prima di richiamarlo. Il richiamo dell'attributo prosegue unicamente se il controllo `isNull()` restituisce false, dimostrando che l'attributo non è nullo.

```
if (! contract1.isNull("CustAddr"))
   {
   BusObj customerAddress = contract1.getBusObj("CustAddr");
   //fare qualcosa con l'oggetto business "customerAddress"
   }
```

isRequired()

Controlla se l'attributo dell'oggetto business è definito come attributoobbligatorio.

Sintassi

```
boolean isRequired(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori restituiti

Restituisce `true` se il valore dell'attributo è obbligatorio, altrimenti restituisce `false`.

Note

Se un attributo è definito come obbligatorio, deve avere un valore e non deve essere nullo.

Esempi

L'esempio che segue registra un'avvertenza se un attributo obbligatorio ha un valore nullo.

```
if ( (customer.isRequired("Address"))
      && (customerBusObj.isNull("Address")) )
   {
   logWarning(12, "Address is required and cannot be null.");
   }
else
   {
   //do something else
   }
```

keysToString()

Richiama il valore di un attributo chiave primario dell'oggetto business come stringa.

Sintassi

```
String keysToString()
```

Valori restituiti

Un oggetto stringa contenente tutti i valori chiave in un oggetto business, concatenati ed ordinati in base al valore ordinale dell'attributo.

Note

Il risultato in uscita di questo metodo contiene il nome dell'attributo ed il suo valore. I valori multipli sono valori di attributi chiave primari, concatenati e separati da spazi. Ad esempio, se c'è un solo attributo chiave primario, SS#, l'output potrebbe essere il seguente:

```
SS#=100408394
```

Se gli attributi chiave primari sono FirstName e LastName, l'output potrebbe essere il seguente:

```
FirstName=Nina LastName=Silk
```

Esempi

L'esempio che segue restituisce i valori degli attributi chiave dell'oggetto business rappresentati dalla variabile `fromOrder`.

```
String keyValues = fromOrder.keysToString();
```

set()

Imposta un attributo di un oggetto business ad un valore specifico di un determinato tipo di dati.

Sintassi

```
void set(String attribute, Object value)
void set(int position, Object value)

void set(String attribute, boolean value)
void set(String attribute, double value)
void set(String attribute, float value)
void set(String attribute, int value)
void set(String attribute, long value)
void set(String attribute, Object value)
void set(String attribute, String value)
```

Parametri

attribute	Il nome dell'attributo da impostare.
position	Un numero intero che definisce la posizione ordinale di un attributo nell'elenco attributi di un oggetto business.
value	Il valore di un attributo.

Eccezioni

CollaborationException—Il metodo set() può impostare i seguenti tipi di eccezione per questa eccezione:

- AttributeException—Impostata se si verifica un problema di accesso all'attributo.

Note

Il metodo set() imposta un valore dell'attributo nell'oggetto business corrente. Questo metodo imposta un riferimento oggetto per il parametro *value* quando assegna il valore all'attributo. *Non* clona il valore dell'attributo dall'oggetto business di origine. Pertanto, qualsiasi modifica di *value* nell'oggetto business di origine si riflette anche sull'attributo nell'oggetto business che effettua la chiamata set().

Il metodo set() contempla i seguenti moduli:

- Il primo modulo restituisce un valore del tipo definito dal metodo del secondo tipo di parametro. Ad esempio, set(String *attribute*, boolean *value*) imposta un attributo con un valore booleano, set(String *attribute*, double *value*) imposta un attributo con un valore double, e così via. Utilizzare questo modulo per impostare attributi con basi specifiche oppure tipi di dati definiti da InterChange Server Express.

  Questi metodi permettono di accedere al valore di un attributo specificando il *nome* dell'attributo.
- Il secondo modulo imposta il valore di un attributo di *qualsiasi* tipo. È possibile inviare qualsiasi tipo di dati come valore dell'attributo poiché il parametro valore attributo è di tipo Oggetto. Ad esempio, per impostare un attributo che sia un oggetto BusObj oppure LongText, utilizzare questo modulo del metodo e passare l'oggetto BusObj oppure LongText come valore dell'attributo.

  Questo modulo del metodo set() consente di accedere al valore di un attributo specificando *sia* il *nome* dell'attributo che la sua *posizione* d'indice nell'elenco degli attributi dell'oggetto business.

Esempi

L'esempio che segue imposta l'attributo LName in toCustomer al valore Smith.

```
toCustomer.set("LName", "Smith");
```

L'esempio che segue mostra come set() assegni un riferimento oggetto piuttosto che clonare il valore:

```
BusObj BusObj myBusObj = new BusObj();
BusObj mySettingBusObj = new BusObj();

myBusObj.set("attr1", mySettingBusObj);
```

Dopo l'esecuzione di questo frammento, l'attributo attr1 di myBusObj è impostato sull'oggetto business mySettingBusObj. Se mySettingBusObj viene modificato, myBusObj.attr1 viene modificato alla stessa maniera poiché set() effettua un riferimento all'oggetto mySettingBusObj quando imposta l'attributo attr1, *non* crea una copia statica di mySettingBusObj.

L'esempio che segue imposta il valore dell'attributo utilizzandone la posizione ordinale all'interno dell'elenco attributi:

```
for i=0; i<maxAttrCount; i++)
{
        myBusObj.set(i, strValue);
    ...
```

setContent()

Imposta i contenuti di questo oggetto business in un altro oggetto business. Dove entrambi gli oggetti *condividono* il contenuto.

Sintassi

```
void setContent(BusObj BusObj)
```

Note

L'utilizzo di setContent() è simile a quello di copy() poiché entrambe le funzioni copiano il contenuto da un oggetto business ad un altro. Ma setContent() *condivide* il contenuto di entrambi gli oggetti business. Ad esempio, se il contenuto di BusObjA è condiviso con BusObjB, ed il contenuto di BusObjA viene modificato, anche il contenuto di BusObjB cambia nello stesso momento.

Parametri

BusObj L'oggetto business i cui valori sono utilizzati per impostare i valori di questo oggetto business.

Eccezioni

CollaborationException—Il metodo setContent() può impostare uno dei seguenti tipi di eccezione per questa eccezione:

- AttributeException –Impostata se si verifica un problema di accesso all'attributo.
- ObjectException – Impostata se l'argomento dell'oggetto business non è valido.

Esempi

L'esempio che segue imposta i contenuti della variabile dell'istanza per l'oggetto di outputObjOutput1 a quelli dell'oggetto business rDstBO[0].

```
ObjOutput1.setContent(rDstBO[0]);
```

setDefaultAttrValues()

Imposta tutti gli attributi ai propri valori predefiniti.

Sintassi

```
void setDefaultAttrValues()
```

Note

Un definizione di un oggetto business che può contenere i valori predefiniti degli attributi. Questo metodo imposta i valori degli attributi di questo oggetto business ai valori specificati come predefiniti nella definizione.

Esempi

L'esempio seguente imposta i valori dell'oggetto business `PaperClip` a quelli predefiniti.

```
PaperClip.setDefaultAttrValues();
```

setKeys()

Imposta i valori degli attributi chiave di questo oggetto business ai valori degli attributi chiave di un altro oggetto business.

Sintassi

```
void setKeys(BusObj inputBusObj)
```

Parametri

inputBusObj L'oggetto business i cui valori sono utilizzati per impostare i valori di un altro oggetto business.

Eccezioni

`CollaborationException`—Il metodo `setKeys()` può impostare uno dei seguenti tipi di eccezione per questa eccezione:

- `AttributeException` –Impostata se si verifica un problema di accesso all'attributo.
- `ObjectException` – Impostata se l'argomento dell'oggetto business non è valido.

Esempi

L'esempio seguente imposta i valori chiave nell'oggetto business `helpdeskCustomer` come quelli dell'oggetto business `ERPCustomer` .

```
helpdeskCustomer.setKeys(ERPCustomer);
```

setLocale()

Imposta la locale dell'oggetto business corrente.

Sintassi

```
void setLocale(java.util.Locale locale
```

Parametri

locale L'oggetto Java Locale che contiene le informazioni riguardanti la locale da assegnare all'oggetto business. L'oggetto Locale deve essere un'istanza della classe `java.util.Locale`.

Valori restituiti

Nessuno.

Note

Il metodo `setLocale()` assegna una locale ai dati associati all'oggetto business. La locale potrebbe essere diversa da quella della collaborazione in cui la collaborazione viene eseguita.

Consultare inoltre

getLocale()

---

setVerb()

Imposta l'istruzione di un oggetto business.

Sintassi

```
void setVerb(String verb)
```

Parametri

istruzione L'istruzione dell'oggetto business.

Note

Il metodo `setVerb()` è utilizzato solo nella mappatura.

Nota: *Non* utilizzare questo metodo nello sviluppo di una collaborazione, dove si deve impostare un'istruzione per un oggetto business in uscita in modo interattivo, riempiendo le proprietà della chiamata di servizio.

Esempi

L'esempio che segue imposta l'istruzione Delete sull'oggetto business contactAddress.

```
contactAddress.setVerb("Delete");
```

---

setVerbWithCreate()

Crea l'istanza dell'oggetto business secondario e ne imposta l'istruzione.

Sintassi

```
void setVerbWithCreate(String attributeName, String verb)
```

Parametri

attributeName Il nome dell'oggetto business secondario creato.

verb L'istruzione da impostare.

Eccezioni

`CollaborationException`—Il metodo `setVerbWithCreate()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException`—Impostata se si verifica un problema di accesso all'attributo.

Note

Se l'attributo definito nel parametro *attributeName* è del tipo `BusObj` ed è nullo, la nuova istanza dell'oggetto business secondario viene creata e la sua istruzione viene impostata con il valore del parametro verb. Se l'istanza di questo oggetto business secondario esiste già, viene impostata solo la sua istruzione. Se l'oggetto business secondario è a cardinalità multipla, il parametro *attributeName* deve definire un indice.

Esempi

Nell'esempio che segue viene creata una istanza dell'oggetto business secondario childBO ed impostata l'istruzione Create:

```
myBO.setVerbWithCreate("childBO", "Create");
```

---

setWithCreate()

Imposta un attributo di un oggetto business ad un valore specifico di un determinato tipo di dati, creando un oggetto per il valore se non ne esiste già uno.

Sintassi

```
void setWithCreate(String attributeName, BusObj busObj)
void setWithCreate(String attributeName, BusObjArray busObjArray)
void setWithCreate(String attributeName, Object value)
```

Parametri

attributeName	Il nome dell'attributo da impostare.
oggbus	L'oggetto business da inserire nell'attributo di destinazione.
busObjArray	Il vettore di oggetti business da inserire nell'attributo di destinazione.
value	L'oggetto da inserire nell'attributo di destinazione. Questo oggetto deve essere di uno dei seguenti tipi: `BusObj`, `BusObjArray`, `Object`.

Eccezioni

`CollaborationException`—Il metodo `setWithCreate()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException`—Impostata se si verifica un problema di accesso all'attributo.

Note

Se l'oggetto fornito è un `BusObj` e l'attributo di destinazione contiene un oggetto business a cardinalità multipla, `BusObj` viene accodato a `BusObjArray` come ultimo elemento. Se l'attributo di destinazione contiene un `BusObj`, questo oggetto business va a coprire il valore precedente.

Esempi

L'esempio seguente imposta un attributo chiamato `ChildAttrAttr` ad un valore di 5. L'attributo viene trovato in un oggetto business contenuto in un attributo di myBO, `ChildAttr`. Se l'oggetto business `childAttr` non esiste al momento della chiamata, questo chiamata del metodo lo crea.

```
myBO.setWithCreate("childAttr.childAttrAttr", "5");
```

---

toString()

Restituisce i valori di tutti gli attributi in un oggetto business come una stringa.

Sintassi

```
String toString()
```

Valori restituiti

Un oggetto `String` contenente tutti i valori dell'attributo in un oggetto business.

Note

La stringa risultante da una chiamata di questo metodo è simile a quella contenuta nell'esempio che segue:

```
Name: GenEmployee
Verb: Create
Type: AfterImage
Attributes: (Name, Type, Value)

LastName:String, Davis
FirstName:String, Miles
SS#:String, 041-33-8989
Salary:Float, 15.00
ObjectEventId:String, MyConnector_922323619411_1
```

Esempi

L'esempio che segue restituisce una stringa contenente i valori dell'attributo della variabile dell'oggetto business `fromOrder`.

```
String values = fromOrder.toString();
```

---

validData()

Controlla se il valore definito e di un tipo valido per uno specifico attributo.

Sintassi

```
boolean validData(String attributeName, Object value)
boolean validData(String attributeName, BusObj value)
boolean validData(String attributeName, BusObjArray value)
boolean validData(String attributeName, String value)
boolean validData(String attributeName, long value)
boolean validData(String attributeName, int value)
boolean validData(String attributeName, double value)
boolean validData(String attributeName, float value)
boolean validData(String attributeName, boolean value)
```

Parametri

attributeName L'attributo.

value Il valore.

Restituzioni

true o false (restituzione booleana)

Note

Controlla la compatibilità dei valori immessi nell'attributo di destinazione (come specificato in *attributeName*). I criteri sono i seguenti:

per tipi primitivi (String, long, int, double, float, boolean)	i valori devono essere convertiti al tipo di dati dell'attributo
per un `BusObj`	il valore deve avere lo stesso tipo dell'attributo di destinazione
per un `BusObjArray`	il valore deve puntare ad un `BusObj` o un `BusObjArray` con lo stesso (definizione oggetto business) tipo dell'attributo
per un `Object`	il valore deve essere del tipo `String` , `BusObj`, o `BusObjArray`. Vengono applicate le regole di validazione corrispondenti.

Metodi obsoleti

Alcuni metodi nella classe `BusObj` erano supportati nelle versioni precedenti ma non lo sono più. Questi *metodi obsoleti* non genereranno errori, ma CrossWorlds raccomanda di evitarne l'uso e di migrare il codice esistente ai nuovi metodi. I metodi obsoleti potrebbero essere eliminati in un futuro release.

La Tabella 124 elenca i metodi obsoleti per la classe `BusObj`. Se non si è mai utilizzato Map Designer Express in precedenza, è possibile ignorare questa sezione.

Tabella 124. Metodi obsoleti, Classe BusObj

Metodo precedente	**Metodo sostitutivo**
`getCount()`	`BusObjArray.size()`
`getKeys()`	`keysToString()`
`getValues()`	`toString()`
`not`	operatore NOT Java standard, `"!"`
`set(BusObj inputBusObj)`	`copy()`
Tutti i metodi che hanno ricevuto come argomento di input un oggetto business secondario oppure un vettore di oggetti business secondari	Ottenere un ID interno per l'oggetto business secondario o per il vettore di oggetti business ed utilizzare i metodi della classe `BusObj` or `BusObjArray`

Il metodo `setVerb()`, che era elencato in precedenza come obsoleto, è stato adesso ripristinato per essere utilizzato nella mappatura. Da non utilizzare in una collaborazione.

Capitolo 11. Classe BusObjArray class

I metodi riportati in questo capitolo agiscono su oggetti della classe BusObjArray definita in Business Integration Server Express. La classe BusObjArray incapsula un vettore di oggetti business. In un oggetto business gerarchico,un attributo è una riferimento ad un vettore di oggetti business secondari quando la sua cardinalità è pari a n. Operazioni su una classe BusObjArray possono restituire sia un oggetto BusObjArray che un effettivo vettore di oggetti business.

Nota: La classe BusObjArray viene utilizza sia per lo sviluppo di collaborazioni che per la mappatura. Consultare la sezione Note per i problemi relativi all'utilizzo dei singoli metodi.

La Tabella 125 elenca i metodi della classe BusObjArray.

Tabella 125. Sommario metodi BusObjArray

Metodo	Descrizione	Pagina
addElement()	Aggiunge un oggetto business a questo vettore oggetto business.	366
duplicate()	Crea un vettore di oggetti business (oggetto BusObjArray) esattamente come questo.	366
elementAt()	Richiama un singolo oggetto business specificandone la posizione in questo vettore di oggetti business.	367
equals()	Confronta un altro vettore di oggetti business con questo.	367
getElements()	Ottiene il contenuto del Vettore Oggetto Business.	368
getLastIndex()	Ottieni ultimo indice dal Vettore Oggetto Business.	368
max()	Ottiene il massimo valore attribuito a tutti gli elementi del Vettore Oggetto Business.	368
maxBusObjArray()	Restituisce gli oggetti business con il massimo valore per l'attributo specificato, come vettore di oggetti business (oggetto BusObjArray).	369
maxBusObjs()	Restituisce gli oggetti business con il valore massimo per l'attributo specificato, come vettore di oggetti BusObj.	370
min()	Richiama il valore minimo dell'attributo specificato tra gli oggetti business di questo vettore.	371
minBusObjArray()	Restituisce gli oggetti business con il valore minimo per l'attributo specificato, come oggetti BusObjArray.	372
minBusObjs()	Restituisce gli oggetti business con il valore minimo per l'attributo specificato, come vettore di oggetti BusObj.	373
removeAllElements()	Rimuove tutti gli elementi da questo vettore di oggetti business.	374
removeElement()	Rimuove un elemento dell'oggetto business da un vettore di oggetti business.	374
removeElementAt()	Rimuove un elemento ad una particolare posizione del vettore oggetto business.	375

Tabella 125. Sommario metodi BusObjArray (Continua)

Metodo	Descrizione	Pagina
`setElementAt()`	Imposta il valore di un oggetto business in un Vettore Oggetto Business.	375
`size()`	Restituisce il numero di elementi dal Vettore Oggetto Business.	376
`sum()`	Aggiunge i valori dell'attributo specificato di ogni Oggetto Business al Vettore Oggetto Business.	376
`swap()`	Inverte la posizione di due Oggetti Business in questo Vettore Oggetto Business. Tenere presente che il primo elemento del vettore è zero (0), il secondo è 1, il terzo è 2, e così via.	377
`toString()`	Richiama i valori in questo Vettore Oggetto Business come singola stringa.	377

Nota: Consultare "Eccezioni e tipi di eccezioni" a pagina 346 per un importante chiarimento sulla gestione delle eccezioni con questa classe. La sezione si riferisce unicamente alle eccezioni in `BusObjArray` e `BusObj`.

addElement()

Aggiunge un oggetto business a questo vettore di oggetti business.

Sintassi

```
void addElement(BusObj element)
```

Parametri

element Un oggetto business da aggiungere a questo vettore.

Eccezioni

`CollaborationException`—Il metodo `addElement()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException` –Impostata se l'attributo non è valido.

Esempi

L'esempio che segue utilizza il metodo `getBusObjArray()` per richiamare un vettore di oggetti business chiamato `itemList` dall'oggetto business `order`. Il vettore viene assegnato a `items`, e quindi un nuovo oggetto business viene aggiunto ad `items`.

```
BusObjArray items = order.getBusObjArray("itemList");
items.addElement(new BusObj("oneItem"));
```

duplicate()

Crea un vettore di oggetti business (oggetto `BusObjArray`) esattamente uguale a questo.

Sintassi

```
BusObjArray duplicate()
```

Valori restituiti

Un vettore di oggetti business.

Esempi

L'esempio che segue duplica i vettori delle `voci`, creando `newItems`.

```
BusObjArray newItems = items.duplicate();
```

elementAt()

Richiama un singolo oggetto business specificandone la posizione in questo vettore di oggetti business.

Sintassi

```
BusObj elementAt(int index)
```

Parametri

indice L'elemento del vettore da richiamare. Il primo elemento del vettore è zero (0), il secondo è 1, il terzo è 2, e così via.

Eccezioni

`CollaborationException`—Il metodo `elementAt()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException` –Impostata se l'attributo non è valido.

Esempi

L'esempio che segue richiama l'undicesimo oggetto business nel vettore delle `voci` e lo assegna alla variabile `Item` .

```
BusObj Item = items.elementAt(10);
```

equals()

Confronta un altro vettore di oggetti business con questo.

Sintassi

```
boolean equals(BusObjArray inputBusObjArray)
```

Parametri

inputBusObjArray
Un vettore oggetti business da confrontare con questo vettore oggetti business.

Note

Il confronto di due vettori oggetti business controlla il numero di elementi ed il valore dei loro attributi.

Esempi

L'esempio seguente utilizza `equals()` per definire un loop condizionato, la cui parte interna non viene mostrata.

```
if (items.equals(newItems))
   {
...
   }
```

getElements()

Richiama i contenuti di questo vettore di oggetti business.

Sintassi

```
BusObj[] getElements()
```

Eccezioni

`CollaborationException`—Il metodo `getElements()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `ObjectException` – Impostata se uno degli elementi non è valido.

Esempi

L'esempio che segue stampa gli elementi del vettore di voci.

```
BusObj[] elements = items.getElements();
for (int i=0, i<elements.length; i++)
   {
   trace(1, elements[i].toString());
   }
```

getLastIndex()

Richiama l'ultimo indice disponibile da un vettore di oggetti business.

Sintassi

```
int getLastIndex()
```

Restituzioni

L'ultimo puntamento all'ultimo elemento in questo `BusObjArray`.

Note

In precedenza, per questa operazione veniva utilizzato il metodo `size()`. L'utente utilizzava il metodo `size()` del vettore di oggetti business per richiamare l'ultimo indice disponibile in un BusObjArray. Se però il BusObjArray conteneva degli intervalli questo approccio restituiva dati non corretti.

Come tutti i vettori Java, `BusObjArray` è un vettore relativo a zero. Questo significa che il metodo `size()` restituirà un valore maggiorato di 1 rispetto al metodo `getLastIndex()`.

Esempi

L'esempio che segue richiama l'ultimo indice nel vettore di oggetti business.

```
int lastElementIndex = items.getLastIndex();
```

max()

Richiama il valore massimo dell'attributo specificato tra tutti gli elementi di questo vettore di oggetti business.

Sintassi

```
String max(String attr)
```

Parametri

attr Un variabile che fa riferimento ad un attributo dell'oggetto business. L'attributo deve essere di uno dei seguenti tipi:`String`, `LongText`, `Integer`, `Float`, e `Double`.

Restituzioni

Il valore massimo dell'attributo specificato è sotto forma di stringa, oppure è nullo se il valore di quell'attributo è nullo per tutti gli elementi di questo `BusObjArray`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è valido nell'oggetto business immesso.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è tra quelli supportati elencati nella sezione note.

Tutte le eccezioni riportate in precedenza sono sono classi secondarie rispetto `CollaborationException`. Il metodo `max()` può impostare il seguente tipi di eccezione per questa eccezione: `AttributeException`.

Note

Il metodo `max()` cerca il valore massimo dell'attributo specificato negli oggetti business presenti in `BusObjArray`. Ad esempio, se si utilizzano tre oggetti impiegati, per l'attributo "Salario" di tipo "Float," verrà restituita la stringa con il salario maggiore.

Se il valore dell'attributo specificato per un elemento in `BusObjArray` risulta nullo, esso viene ignorato. Se il valore dell'attributo specificato fosse nullo per tutti gli elementi, viene restituito null.

Quando l'attributo è di tipo String, `max()` restituisce il valore dell'attributo contenente la stringa più lunga.

Esempi

```
String maxSalary = items.max("Salary");
```

---

maxBusObjArray()

Restituisce gli oggetti business con il valore massimo per l'attributo specificato, come vettore di oggetti business (oggetto `BusObjArray`).

Sintassi

```
BusObjArray maxBusObjArray(String attr)
```

Parametri

attr Una variabile `String`, `LongText`, `Integer`, `Float`, o `Double` che fa riferimento ad un attributo in un oggetto business contenuto in un vettore oggetti business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObjArray` oppure `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è valido nell'oggetto business immesso.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è tra quelli supportati elencati nella sezione note.

Tutte le eccezioni riportate in precedenza sono sono classi secondarie rispetto `CollaborationException`. Il metodo `maxBusObjArray()` può impostare il seguente tipi di eccezione per questa eccezione: `AttributeException`.

Note

Il metodo `maxBusObjArray()` individua uno o più oggetti business con il valore massimo dell'attributo specificato, e restituisce questi oggetti business in un oggetto `BusObjArray`.

Ad esempio, supponendo che questo sia un vettore di oggetti business contenente oggetti business Impiegato e che l'argomento di input sia l'attributo Salario, di tipo Float. Il metodo determina il valore massimo dell'attributo Salario in tutti gli oggetti business Impiegato, e restituisce l'oggetto business contenente questo valore. Se più oggetti business dovessero contenere lo stesso valore massimo per l'attributo Salario, il metodo li restituisce tutti.

Un oggetto business con il valore dell'attributo specificato nullo viene ignorato. Se il valore è nullo in tutti gli oggetti business di questo vettore, viene restituito null.

Quando l'attributo è di tipo String, il metodo restituisce la stringa più lunga.

Esempi

```
BusObjArray boarrayWithMaxSalary = items.maxBusObjArray("Salary");
```

---

maxBusObjs()

Restituisce gli oggetti business con il valore massimo per l'attributo specificato, come vettore di oggetti `BusObj`.

Sintassi

```
BusObj[] maxBusObjs(String attr)
```

Parametri

attr Una variabile `String`, `LongText`, `Integer`, `Float`, o `Double` che fa riferimento ad un attributo nell'oggetto business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObj[]` oppure `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è valido nell'oggetto business immesso.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è tra quelli supportati elencati nella sezione note.

Tutte le eccezioni riportate in precedenza sono sono classi secondarie rispetto `CollaborationException`. Il metodo `maxBusObjs()` può impostare il seguente tipi di eccezione per questa eccezione: `AttributeException`.

Note

Il metodo `maxBusObjs()` individua uno o più oggetti business con il valore massimo dell'attributo specificato, e restituisce questi oggetti business come vettore di oggetti `BusObj`.

Ad esempio, supponendo che questo sia un vettore di oggetti business contenente oggetti business Impiegato e che l'argomento di input sia l'attributo Salario, di tipo Float. Il metodo determina il valore massimo dell'attributo Salario in tutti gli oggetti business Impiegato, e restituisce l'oggetto business contenente questo valore. Se più oggetti business dovessero contenere lo stesso valore massimo per l'attributo Salario, il metodo li restituisce tutti.

Un oggetto business con il valore dell'attributo specificato nullo viene ignorato. Se il valore è nullo in tutti gli oggetti business di questo vettore, viene restituito null.

Quando l'attributo è di tipo String, il metodo restituisce la stringa più lunga.

Esempi

```
BusObj[] bosWithMaxSalary = items.maxBusObjs("Salary");
```

---

min()

Richiama il valore minimo dell'attributo specificato tra gli oggetti business di questo vettore.

Sintassi

```
String min(String attr)
```

Parametri

attr Una variabile `String`, `LongText`, `Integer`, `Float`, o `Double` che fa riferimento ad un attributo nell'oggetto business.

Restituzioni

Il valore minimo dell'attributo specificato è sotto forma di stringa, oppure è nullo se il valore di quell'attributo è nullo per tutti gli elementi di questo `BusObjArray`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è valido nell'oggetto business immesso.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è tra quelli supportati elencati nella sezione note.

Tutte le eccezioni riportate in precedenza sono sono classi secondarie rispetto `CollaborationException`. Il metodo `min()` può impostare il seguente tipi di eccezione per questa eccezione: `AttributeException`.

Note

Il metodo `min()` cerca il valore minimo dell'attributo specificato negli oggetti business presenti in questo vettore e di oggetti business.

Ad esempio, supponendo che questo sia un vettore di oggetti business contenente oggetti business Impiegato e che l'argomento di input sia l'attributo Salario, di tipo Float. Il metodo determina il valore minimo dell'attributo Salario in tutti gli oggetti business Impiegato, e restituisce l'oggetto business contenente questo valore. Se più oggetti business dovessero contenere lo stesso valore minimo per l'attributo Salario, il metodo li restituisce tutti.

Un oggetto business con il valore dell'attributo specificato nullo viene ignorato. Se il valore è nullo in tutti gli oggetti business di questo vettore, viene restituito null.

Quando l'attributo è di tipo String, il metodo restituisce la stringa più breve.

Esempi

```
String minSalary = items.min("Salary");
```

---

minBusObjArray()

Restituisce gli oggetti business con il valore minimo dell'attributo specificato, come un oggetto `BusObjArray`.

Sintassi

```
BusObjArray minBusObjArray(String attr)
```

Parametri

attr Una variabile `String`, `LongText`, `Integer`, `Float`, o `Double` che fa riferimento ad un attributo nell'oggetto business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObjArray` oppure `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è valido nell'oggetto business immesso.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è tra quelli supportati elencati nella sezione note.

Tutte le eccezioni riportate in precedenza sono sono classi secondarie rispetto `CollaborationException`. Il metodo `minBusObjArray()` può impostare il seguente tipi di eccezione per questa eccezione: `AttributeException`.

Note

Il metodo `minBusObjArray()` individua uno o più oggetti business con il valore minimo dell'attributo specificato, e restituisce questi oggetti business in un oggetto `BusObjArray`.

Ad esempio, supponendo che questo sia un vettore di oggetti business contenente oggetti business Impiegato e che l'argomento di input sia l'attributo Salario, di tipo Float. Il metodo determina il valore minimo dell'attributo Salario in tutti gli oggetti business Impiegato, e restituisce l'oggetto business contenente questo valore. Se più oggetti business dovessero contenere lo stesso valore minimo per l'attributo Salario, il metodo li restituisce tutti.

Un oggetto business con il valore dell'attributo specificato nullo viene ignorato. Se il valore è nullo in tutti gli oggetti business di questo vettore, viene restituito null.

Quando l'attributo è di tipo String, il metodo restituisce la stringa più breve.

Esempi

```
BusObjArray boarrayWithMinSalary = items.minBusObjArray("Salary");
```

---

minBusObjs()

Restituisce gli oggetti business con il valore minimo per l'attributo specificato, come vettore di oggetti `BusObj`.

Sintassi

```
BusObj[] minBusObjs(String attr)
```

Parametri

attr Una variabile `String`, `LongText`, `Integer`, `Float`, o `Double` che fa riferimento ad un attributo nell'oggetto business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObj[]` oppure `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è valido nell'oggetto business immesso.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è tra quelli supportati elencati nella sezione note.

Tutte le eccezioni riportate in precedenza sono sono classi secondarie rispetto `CollaborationException`. Il metodo `minBusObjs()` può impostare il seguente tipi di eccezione per questa eccezione: `AttributeException`.

Note

Il metodo `minBusObjs()` individua uno o più oggetti business con il valore massimo dell'attributo specificato, e restituisce questi oggetti business come vettore di oggetti `BusObj`.

Ad esempio, supponendo che questo sia un vettore di oggetti business contenente oggetti business Impiegato e che l'argomento di input sia l'attributo Salario, di tipo Float. Il metodo determina il valore minimo dell'attributo Salario in tutti gli oggetti business Impiegato, e restituisce l'oggetto business contenente questo valore. Se più oggetti business dovessero contenere lo stesso valore minimo per l'attributo Salario, il metodo li restituisce tutti.

Un oggetto business con il valore dell'attributo specificato nullo viene ignorato. Se il valore è nullo in tutti gli oggetti business di questo vettore, viene restituito null.

Quando l'attributo è di tipo String, il metodo restituisce la stringa più breve.

Esempi

```
BusObj[] bosWithMinSalary = items.minBusObjs("Salary");
```

removeAllElements()

Rimuove tutti gli elementi da questo vettore di oggetti business.

Sintassi

```
void removeAllElements()
```

Esempi

L'esempio che segue rimuove tutti gli elementi delle `voci` del vettore.

```
items.removeAllElements();
```

removeElement()

Rimuove un elemento dell'oggetto business da un vettore di oggetti business.

Sintassi

```
void removeElement(BusObj element)
```

Parametri

elementReference
Un variabile che fa riferimento ad un elemento del vettore.

Eccezioni

`CollaborationException`—Il metodo `removeElement()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException` –Impostata se l'attributo non è valido.

Note

Dopo l'eliminazione di un elemento dal vettore, questo si ridimensiona, modificando l'indice degli elementi esistenti.

Esempi

L'esempio che segue elimina l'elemento `Child1` delle `voci` del vettore di oggetti business.

```
items.removeElement(Child1);
```

removeElementAt()

Rimuove un elemento da una determinata posizione in questo vettore di oggetti business.

Sintassi

```
void removeElementAt(int index)
```

Note

Dopo l'eliminazione di un elemento dal vettore, questo si ridimensiona, modificando l'indice degli elementi esistenti.

Parametri

index L'indice dell'elemento.

Eccezioni

`CollaborationException`—Il metodo `removeElementAt()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException` –Impostata se l'attributo non è valido.

Esempi

L'esempio che segue elimina il sesto oggetto business nelle `voci` del vettore.

```
items.removeElementAt(5);
```

setElementAt()

Imposta il valore di un oggetto business in un vettore di oggetti business.

Sintassi

```
void setElementAt (int index, BusObj element)
```

Parametri

index Un numero intero che rappresenta la posizione d'indice. Il primo elemento del vettore è zero (0), il secondo è 1, il terzo è 2, e così via.

inputBusObj L'oggetto business contenente i valori da assegnare all'elemento del vettore.

Eccezioni

`CollaborationException`—Il metodo `setElementAt()` può impostare i seguenti tipi di eccezione per questa eccezione:

- `AttributeException` –Impostata se l'attributo non è valido.

Note

Questo metodo imposta i valori dell'oggetto business in una posizione del vettore specificata a quelli dell'oggetto business di in input.

Esempi

L'esempio che segue crea un nuovo oggetto business di tipo Item e lo aggiunge al vettore `items`, come quarto elemento.

```
items.setElementAt(5, new BusObj("Item"));
```

---

size()

Restituisce il numero di elementi in questo vettore di oggetti business.

Sintassi

```
int size()
```

Note

Come tutti i vettori Java, BusObjArray è un vettore relativo a zero. Questo significa che il metodo `size()` restituirà un valore maggiorato di 1 rispetto al metodo `getLastIndex()`.

Esempi

L'esempio che segue restituisce il numero di elementi nel vettore `items`.

```
int size = items.size();
```

---

sum()

Aggiunge i valori dell'attributo specificato a tutti gli oggetti business di questo vettore di oggetti business.

Sintassi

```
double sum(String attrName)
```

Parametri

attr Un variabile che fa riferimento ad un attributo dell'oggetto business. L'attributo deve essere di tipo Integer, Float, o Double.

Restituzioni

La somma dell'attributo specificato dall'elenco di oggetti business.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è valido nell'oggetto business immesso.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è tra quelli supportati elencati nella sezione note.

Tutte le eccezioni riportate in precedenza sono sono classi secondarie rispetto `CollaborationException`. Il metodo `sum()` può impostare il seguente tipi di eccezione per questa eccezione: `AttributeException`.

Esempi

```
double sumSalary = items.sum("Salary");
```

swap()

Inverte le posizioni di due oggetti business in questo vettore di oggetti business. Tenere presente che il primo elemento del vettore è zero (0), il secondo è 1, il terzo è 2, e così via.

Sintassi

```
void swap(int index1, int index2)
```

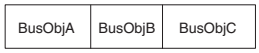
Parametri

index1 La posizione nel vettore di uno degli elementi che si vuole invertire.

index2 La posizione nel vettore dell'altro degli elementi che si vuole invertire.

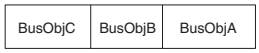
Esempi

L'esempio che segue utilizza `swap()` per invertire le posizioni di BusObjA e BusObjC nel vettore che segue:




```
swap(0,2);
```

Il risultato della chiamata swap() è il seguente vettore:



toString()

Richiama i valori in questo vettore di oggetti business come una singola stringa.

Sintassi

```
String toString()
```

Esempi

L'esempio che segue utilizza `toString()` per richiamare i contenuti del vettore di oggetti business `items` e quindi utilizza `logInfo()` per scriverli nel file di log.

```
logInfo(items.toString());
```

Capitolo 12. Classe CwBidiEngine

La classe CxBidiEngine fornisce metodi per il trasferimento degli oggetti business e stringhe da un formato bidirezionale all'altro.

Tabella 126 riepiloga i metodi nella classe CxBidiEngine.

Tabella 126. CwBidiEngine riepilogo metodo

Metodo	Descrizione	Pagina
`BiDiBOTransformation()`	Trasforma gli oggetti business di tipo BusinessObject da un formato bidirezionale all'altro.	379
`BiDiBusObjTransformation()`	Trasforma gli oggetti business di tipo BusObj da un formato bidirezionale ad un altro formato.	380
`BiDiStringTransformation()`	Trasforma le stringhe da un formato bidirezionale all'altro.	381

BiDiBOTransformation()

Il metodo BiDiTransformation() trasforma gli oggetti business di tipo BusinessObject da un formato bidirezionale in un altro formato. Usare questo metodo quando si sviluppano controllori, connettori e mappe.

Sintassi

```
BusinessObject BiDiBOTransformation(BusinessObject boIn, String formatIn,
   String formatOut, boolean replace)
```

Parametri

boIn L'oggetto business da trasformare. L'oggetto deve essere di tipo BusinessObject.

formatIn Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di immissione. Consultare Tabella 127 a pagina 380 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

formatOut Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare Tabella 127 a pagina 380 per i valori validi della stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

replace Un valore che indica se l'oggetto business di immissione deve essere sostituito. Il valore valido è `true` o `false`.

Valori di ritorno

Il valore di ritorno è un oggetto business trasformato. Se il metodo non riesce, restituisce un valore nullo.

Eccezioni

Nessuno.

Esempi

Consultare l'esempio in "BiDiStringTransformation()" a pagina 381.

BiDiBusObjTransformation()

Il metodo BiDiBusObjTransformation() trasforma gli oggetti business di tipo BusObj da un formato bidirezionale all'altro. Usare questo metodo all'interno della collaborazione.

Sintassi

```
BusObj BiDiBusObjTransformation(BusObj busObjIn, String formatIn,
   String formatOut, boolean replace)
```

Parametri

busObjIn L'oggetto business da trasformare. L'oggetto deve essere di tipo BusObj.

formatIn Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di immissione. Consultare Tabella 127 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

formatOut Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare Tabella 127 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

replace Un valore che indica se l'oggetto business di immissione deve essere sostituito. Il valore valido è `true` o `false`.

Tabella 127. Valori per le stringhe di formato

Posizione lettera	Scopo	Valori	Descrizione	Predefinito
1	Tipo	I	Implicito (Logico)	I
		V	Visivo	
2	Direzione	S	Sinistra a destra	S
		R	Destra a sinistra	
3	Inversione simmetrica	Y	Inversione simmetrica attiva	Y
		N	Inversione simmetrica non attiva	
4	Struttura	Y	Il testo è strutturato	N
		N	Testo non strutturato	
5	Struttura numerica	H	Hindi	N
		C	Contestuale	
		N	Nominale	

Valori di ritorno

Il valore di ritorno è un oggetto business trasformato. Se il metodo non riesce, restituisce un valore nullo.

Eccezioni

Nessuno.

Esempi

Questo esempio trasforma InputBOBusObj dal formato bidirezionale standard di Windows al formato bidirezionale visivo.

```
BusObj dummyBusObj = null;
dummyBusObj = CwBidiEngine.BiDiBusObjTransformation(
                                          InputBOBusObj,
                                    "ILYNN",
                                    "VLYNN",true);
```

BiDiStringTransformation()

Il metodo BiDiStringTransformation() trasforma le stringhe da un formato bidirezionale all'altro.

Sintassi

```
BiDiStringTransformation(String strIn, String formatIn, String formatOut
```

Parametri

strIn La stringa da trasformare.

formatIn Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di immissione. Consultare Tabella 128 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

formatOut Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare Tabella 128 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

Tabella 128. Valori per le stringhe di formato

Posizione lettera	Scopo	Valori	Descrizione	Predefinito
1	Tipo	I	Implicito (Logico)	I
		V	Visivo	
2	Direzione	S	Sinistra a destra	S
		R	Destra a sinistra	
3	Inversione simmetrica	Y	Inversione simmetrica attiva	Y
		N	Inversione simmetrica non attiva	
4	Struttura	Y	Il testo è strutturato	N
		N	Testo non strutturato	
5	Struttura numerica	H	Hindi	N
		C	Contestuale	
		N	Nominale	

Valori di ritorno

Il valore di ritorno è un oggetto stringa trasformato.

Eccezioni

Nessuno.

Esempi

Il seguente esempio applica il metodo BiDiStringTransformation() ai valori dell'attributo di un oggetto business.

```
for (int i = 0; i < bo.getAttrCount();i++) {
 intAttrType = bo.getAttributeType(i);
 Object attrValue = bo.getAttrValue(i);
 String attrName = bo.getAttrName(i);

 if (attrValue != null {
    // Si gestisce solo l'attributo Stringa o Testo lungo e non
    // l'attributo ObjectEventId
    if (((attrType == CxObjectAttrType.STRING)
           || (attrType == CxObjectAttrType.LONGTEXT))
           && (!(attrName.equals(OBJECT_EVENT_ID)))) {
             String strOut = BidiStringTransformation(attrValue.toString(),
             bo.setAttrValue(i, strOut);
    } else if (attrType == CxObjectAttrType.OBJECT) {
        CxObjectAttr attrDesc = bo.getAttrDesc(i);
        if (attrDesc.getCardinality().equals(CxObjectAttr.CARD_Single)) {
           BiDiTransformation((BusinessObject) attrValue, "ILYNN",
                                                          "VLYNN",
                                                          true);
       } else {
           // multiple cardinality
           CxObjectContainer cont = (CxObjectContainer) attrValue;
           int objCount = cont.getObjectCount();
           for (int j = 0; j < objCount; j++) {
               BiDiBOTransformation((BusinessObject) (cont.getObject(j)),
                                       "ILYNN",
                                       "VLYNN",
                                       true);
       }
    }
 }
```

Capitolo 13. Classe CwDBConnection

La classe `CwDBConnection` fornisce metodi per l'esecuzione di query SQL in un database. Le query vengono effettuate tramite una connessione, che si ottiene da un pool di connessioni. Per creare l'istanza di questa classe, si deve effettuare la chiamata `getDBConnection()` nella classe `BaseDLM`. Tutte le mappe sono derivate o sono classi secondarie di `BaseDLM` così hanno accesso a `getDBConnection()`.

La Tabella 126 riepiloga i metodi della classe `CwDBConnection`.

Tabella 129. Sommario metodi CwDBConnection

Metodo	Descrizione	Pagina
`beginTransaction()`	Inizia una transazione esplicita per la connessione corrente.	383
`commit()`	Sincronizza la transazione attiva associata con la connessione corrente.	384
`executeSQL()`	Esegue una query SQL statica specificando la relativa sintassi ed un vettore di parametri facoltativo.	386
`executePreparedSQL()`	Esegue una query SQL preparata specificando la relativa sintassi ed un vettore di parametri facoltativo.	385
`executeStoredProcedure()`	Esegue una procedura memorizzata SQL specificandone il nome ed il vettore di parametri.	388
`getUpdateCount()`	Restituisce il numero di righe che sono state modificate nell'ultima operazione di scrittura sul database.	389
`hasMoreRows()`	Determina se il risultato della query ha più righe da elaborare.	390
`inTransaction()`	Determina se una transazione è in avanzamento nella connessione corrente.	390
`isActive()`	Determina se la connessione corrente è attiva.	391
`nextRow()`	Richiama la riga successiva rispetto al risultato della query.	391
`release()`	Rilascia l'uso della connessione corrente, restituendola al suo pool di connessione.	392
`rollBack()`	Esegue il rollback della transazione attiva associata con la connessione corrente.	392

beginTransaction()

Inizia una transazione esplicita per la connessione corrente.

Sintassi

```
void beginTransaction()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore database.

Note

Il metodo `beginTransaction()` contrassegna l'inizio di una nuova transazione esplicita nella connessione corrente. I metodi `beginTransaction()`, `commit()` e `rollBack()` insieme permettono la gestione dei confini della transazione per un transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE, o UPDATE, ed una procedura memorizzata che comprende una di queste istruzioni SQL.

Se *non* si utilizza `beginTransaction()` per specificare l'inizio di una transazione esplicita, il database esegue ogni singola istruzione SQL come transazioni separate.

Importante: Utilizzare `beginTransaction()` solo se la connessione utilizza parentesi di transazione esplicite. Se la connessione utilizza parentesi di transazione implicite, l'utilizzo di `beginTransaction()` potrebbe generare l'eccezione `CwDBTransactionException`.

Prima di avviare una transazione esplicita, si deve creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` nella classe `BaseDLM`. Assicurarsi che questa transazione utilizzi parentesi di transazione esplicite.

Esempi

L'esempio che segue utilizza una transazione per eseguire una query allo scopo di inserire righe in una tabella del database associato con le connessioni contenute in `CustDBConnPool`.

```
CwDBConnection connection = getDBConnection("CustDBConnPool", false);

// Begin a transaction
connection.beginTransaction();

// Insert a row
connection.executeSQL("insert...");

// Commit the transaction
connection.commit();

// Release the connection
connection.release();
```

Consultare inoltre

`commit()`, `getDBConnection()`, `inTransaction()`, `rollBack()`

---

commit()

Sincronizza la transazione attiva associata con la connessione corrente.

Sintassi

```
void commit()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore database.

Note

Il metodo `commit()` termina la transazione attiva sincronizzando qualsiasi cambiamento effettuato sul database associato con la connessione corrente. I metodi `beginTransaction()`, `commit()` e `rollBack()` insieme permettono la gestione dei confini della transazione per un transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE, o UPDATE, ed una procedura memorizzata che comprende una di queste istruzioni SQL.

Importante: Utilizzare `commit()` solo se la connessione utilizza parentesi di transazione esplicite. Se la connessione utilizza parentesi di transazione implicite, l'utilizzo di `commit()` potrebbe generare l'eccezione `CwDBTransactionException`. Se non si termina la transazione esplicita con `commit()` (o `rollback()`) prima che venga rilasciata la connessione, InterChange Server Express termina in modo implicito la transazione in base all'esito della mappa. Se la mappa ha esito positivo, InterChange Server Express sincronizza questa transazione del database. Se la mappa *non* ha esito positivo, InterChange Server Express effettua il rollback in modo implicito della transazione del database. Indipendentemente dal successo della mappa, InterChange Server Express registra un avviso.

Prima di avviare una transazione esplicita, si deve creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` nella classe `BaseDLM`. Assicurarsi che questa transazione utilizzi parentesi di transazione esplicite.

Esempi

Per un esempio di sincronizzazione di una transazione, consultare l'esempio di `beginTransaction()`.

Consultare inoltre

`beginTransaction()`, `getDBConnection()`, `inTransaction()`, `rollBack()`

---

executePreparedSQL()

Esegue una query SQL preparata specificando la relativa sintassi ed un vettore di parametri facoltativo.

Sintassi

```
void executePreparedSQL(String query)
void executePreparedSQL(String query, Vector queryParameters)
```

Parametri

query Una stringa che rappresenta una query SQL da eseguire nel database.

queryParameters
Un oggetto `Vettore` di argomenti da passare ai parametri nella query SQL.

Valori restituiti

Nessuno.

Eccezioni

`CwDBSQLException` – Se si verifica un errore database.

Note

Il metodo `executePreparedSQL()` invia la stringa *query*, specificata come un'istruzione SQL preparata, al database associato con la connessione corrente. La prima volta che si esegue, questa query viene inviata come una stringa al database, che compila la stringa in un modulo eseguibile (chiamato istruzione preparata), esegue l'istruzione SQL e restituisce l'istruzione preparata a `executePreparedSQL()`. Il metodo `executePreparedSQL()` salva questa istruzione preparata in memoria. Utilizzare `executePreparedSQL()` per istruzioni SQL che si devono eseguire più volte. Il metodo `executeSQL()` *non* salva l'istruzione preparata e quindi è utile per istruzioni che si eseguono una sola volta.

Importante: Prima di avviare una query con `executePreparedSQL()`, si deve ottenere una connessione al database desiderato generando un oggetto `CwDBConnection` con il metodo `getDBConnection()` nella classe `BaseDLM`.

Le istruzioni SQL che è possibile eseguire, essendo in possesso delle necessarie autorizzazioni nel database, comprendono le seguenti:

- L'istruzione SELECT per richiedere dati da una o più tabelle del database.
  Utilizzare i metodi `hasMoreRows()` e `nextRow()` per accedere ai dati richiamati.
- Istruzioni SQL che modificano dati nel database
  - INSERT
  - DELETE
  - UPDATE

  Se la connessione utilizza parentesi di transazione esplicite, è necessario avviare esplicitamente ogni transazione con `beginTransaction()` e terminarla sia con `commit()` che `rollback()`.
- L'istruzione CALL per eseguire una procedura memorizzata preparata, con l'unica limitazione che la procedura memorizzata *non* può utilizzare i parametri OUT.

  Per eseguire procedure memorizzate con parametri OUT, utilizzare il metodo `executeStoredProcedure()`.

Consultare inoltre

`beginTransaction()`, `commit()`, `executeSQL()`, `executeStoredProcedure()`, `getDBConnection()`, `hasMoreRows()`, `nextRow()`, `rollBack()`

---

executeSQL()

Esegue una query SQL statica specificando la relativa sintassi ed un vettore di parametri facoltativo.

Sintassi

```
void executeSQL(String query)
void executeSQL(String query, Vector queryParameters)
```

Parametri

query Una stringa che rappresenta una query SQL da eseguire nel database.

queryParameters
Un oggetto `Vettore` di argomenti da passare ai parametri nella query SQL.

Valori restituiti

Nessuno.

Eccezioni

`CwDBSQLException` – Se si verifica un errore database.

Note

Il metodo `executeSQL()` invia la stringa *query*, specificata come un'istruzione SQL statica, al database associato con la connessione corrente. La query è inviata come stringa al database, che compila la stringa in un modulo eseguibile ed esegue l'istruzione SQL, senza salvare il modulo eseguibile. Utilizzare `executeSQL()` per istruzioni SQL che si devono eseguire un unica volta. Il metodo `executePreparedSQL()` salva il modulo eseguibile (chiamato istruzione preparata) ed è quindi utili per query che si devono eseguire più volte.

Importante: Prima di avviare una query con `executeSQL()`, si deve ottenere una connessione al database desiderato generando un oggetto `CwDBConnection` con il metodo `getDBConnection()` nella classe `BaseDLM`.

Le istruzioni SQL che è possibile eseguire, essendo in possesso delle necessarie autorizzazioni nel database, comprendono le seguenti:

- L'istruzione SELECT per richiedere dati da una o più tabelle del database.

  Utilizzare i metodi `hasMoreRows()` e `nextRow()` per accedere ai dati richiamati.
- Istruzioni SQL che modificano dati nel database
  - INSERT
  - DELETE
  - UPDATE

  Se la connessione utilizza parentesi di transazione esplicite, è necessario avviare esplicitamente ogni transazione con `beginTransaction()` e terminarla sia con `commit()` che `rollback()`.
- L'istruzione CALL per eseguire staticamente una procedura memorizzata, con l'unica limitazione che la procedura memorizzata *non* può utilizzare i parametri OUT.

  Per eseguire procedure memorizzate con parametri OUT, utilizzare il metodo `executeStoredProcedure()`.

Esempi

L'esempio che segue esegue una query per l'inserimento di righe in un database contabile la cui connessione è contenuta nel pool di connessioni `AccntConnPool`.

```
CwDBConnection connection = getDBConnection("AccntConnPool");

// Begin a transaction
connection.beginTransaction();

// Insert a row
connection.executeSQL("insert...");

// Commit the transaction
connection.commit();

// Release the database connection
connection.release();
```

Per esempio di codice completi che selezionano codice da una tabella di relazioni, consultare

Consultare inoltre

`executePreparedSQL()`, `executeStoredProcedure()`, `getDBConnection()`, `hasMoreRows()`, `nextRow()`

---

executeStoredProcedure()

Esegue una procedura memorizzata SQL specificandone il nome ed il vettore di parametri.

Sintassi

```
void executeStoredProcedure(String storedProcedure,
                  Vector storedProcParameters)
```

Parametri

storedProcedure
: Il nome della procedura memorizzata SQL da eseguire nel database.

storedProcParameters
: Un oggetto `Vettore` di parametri da passare alla procedura memorizzata. Ogni parametro è un'istanza della classe `CwDBStoredProcedureParam`.

Valori restituiti

Nessuno.

Eccezioni

`CwDBSQLException` – Se si verifica un errore database.

Note

Il metodo `executeStoredProcedure()` invia una chiamata alla *storedProcedure* specificata nel database associato con la connessione corrente. Questo metodo invia una chiamata alla procedura memorizzata come un'istruzione SQL preparata, ciò significa che la prima volta che si esegue, questa chiamata alla procedura memorizzata viene inviata come una stringa al database, che compila la stringa in un modulo eseguibile (chiamato istruzione preparata), esegue l'istruzione SQL e restituisce l'istruzione preparata a `executeStoredProcedure()`. Il metodo `executeStoredProcedure()` salva questa istruzione preparata in memoria.

Importante: Prima di eseguire una procedura memorizzata con `executeStoredProcedure()`, si deve creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` nella classe `BaseDLM`.

Per gestire qualsiasi dato che la procedura memorizzata restituisce, utilizzare i metodi `hasMoreRows()` e `nextRow()` .

Si possono inoltre utilizzare i metodi `executeSQL()` o `executePreparedSQL()` per eseguire una procedura memorizzata, fin quando questa procedura memorizzata *non* contiene parametri OUT. Se la procedura memorizzata contiene parametri OUT, si *deve* utilizzare `executeStoredProcedure()` per eseguirla. A differenza di quanto accade con `executeSQL()` o `executePreparedSQL()`, non è necessario passare l'intera istruzione SQL per eseguire una procedura memorizzata. Con `executeStoredProcedure()`, si deve passare unicamente il nome della procedura memorizzata ed il vettore di parametri `Vector` dell'oggetto `CwDBStoredProcedureParam`. Il metodo `executeStoredProcedure()` può determinare il numero di parametri dal vettore *`storedProcParameters`* e compilare l'istruzione di chiamata per la procedura memorizzata.

Consultare inoltre

`executePreparedSQL()`, `executeSQL()`, `getDBConnection()`, `hasMoreRows()`, `nextRow()`

---

getUpdateCount()

Restituisce il numero di righe che sono state modificate nell'ultima operazione di scrittura sul database.

Sintassi

```
int getUpdateCount()
```

Parametri

Nessuno.

Valori restituiti

Restituisce un `int` che rappresenta il numero di righe che sono state modificate nell'ultima operazione di scrittura sul database.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore database.

Note

Il metodo `getUpdateCount()` indica quante righe sono state modificate nell'operazione di aggiornamento più recente sul database associato con la connessione corrente. Questo metodo è utile quando, dopo l'invio di un istruzione UPDATE o INSERT al database, si vuole determinare il numero di righe modificate dall'istruzione SQL.

Importante: Prima di utilizzare questo metodo, si deve creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` della classe `BaseDLM` ed inviare una query che aggiorni il database sia con il metodo `executeSQL()` che con il metodo `executePreparedSQL()` della classe `CwDBConnection`.

Consultare inoltre

`executePreparedSQL()`, `executeSQL()`, `getDBConnection()`

---

hasMoreRows()

Determina se il risultato della query ha più righe da elaborare .

Sintassi

```
boolean hasMoreRows()
```

Parametri

Nessuno.

Valori restituiti

Restituisce `true` se ci sono più righe.

Eccezioni

`CwDBSQLException` – Se si verifica un errore database.

Note

Il metodo `hasMoreRows()` determina se il risultato della query, associata con la connessione corrente, ha più righe da elaborare. Utilizzare questo metodo per richiamare i risultati dalla query che restituisce dati. Tale query include un'istruzione SELECT e una procedura memorizzata. Solo una query alla volta può essere associata con la connessione. Quindi, se si esegue un'altra query prima che `hasMoreRows()` restituisca `false`, si perdono i dati dalla query iniziale.

Consultare inoltre

`executePreparedSQL()`, `executeSQL()`, `nextRow()`

---

inTransaction()

Determina se una transazione è in avanzamento nella corrente connessione.

Sintassi

```
boolean inTransaction()
```

Parametri

Nessuno.

Valori restituiti

Restituisce `true` se una transazione è attualmente attiva nella connessione corrente, altrimenti restituisce `false`.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore database.

Note

Il metodo `inTransaction()` restituisce un valore booleano che indica se la connessione corrente ha una transazione attiva, una transazione avviata ma non terminata.

Importante: Prima di avviare una transazione, si deve creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` nella classe `BaseDLM`.

Consultare inoltre

`beginTransaction()`, `commit()`, `getDBConnection()`, `rollBack()`

---

isActive()

Determina se la connessione corrente è attiva.

Sintassi

```
boolean isActive()
```

Parametri

Nessuno.

Valori restituiti

Restituisce `true` se la connessione corrente è attiva; restituisce `false` se la connessione è stata rilasciata.

Eccezioni

Nessuno.

Consultare inoltre

`getDBConnection()`, `release()`

---

nextRow()

Richiama la riga successiva dal risultato della query.

Sintassi

```
Vector nextRow()
```

Parametri

Nessuno.

Valori restituiti

Restituisce la riga successiva rispetto al risultato della query come oggetto `Vettore`.

Eccezioni

`CwDBSQLException` – Se si verifica un errore database.

Note

Il metodo `nextRow()` restituisce una riga di dati dal risultato della query associata con la connessione corrente. Utilizzare questo metodo per richiamare i risultati

dalla query che restituisce dati. Tale query include un'istruzione SELECT e una procedura memorizzata. Solo una query alla volta può essere associata con la connessione. Quindi, se si esegue un'altra query prima che `nextRow()` restituisca l'ultima riga di dati, si perdono i risultati della query iniziale.

Consultare inoltre

`hasMoreRows()`, `executePreparedSQL()`, `executeSQL()`, `executeStoredProcedure()`

release()

Rilascia l'uso della connessione corrente restituendola al suo pool di connessioni.

Sintassi

```
void release()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

`CwDBConnectionException`

Note

Il metodo `release()` rilascia in modo esplicito l'uso della connessione corrente dall'istanza mappa. Una volta rilasciata, la connessione ritorna al suo pool di connessioni, dove è disponibile per altri componenti (mappe o collaborazioni) che richiedono una connessione al database associato. Se non si rilascia la connessione in modo esplicito, l'istanza mappa la rilascia implicitamente al termine dell'esecuzione della mappa corrente. Quindi, *non* è possibile salvare una connessione in una variabile statica ed utilizzarla nuovamente.

Attenzione: *Non* utilizzare il metodo `release()` se una transazione è attiva. Con le parentesi di transazione implicite, InterChange Server Express non termina la transazione del database fin quando non determina l'esito della mappa. Quindi, l'utilizzo di questo metodo su una connessione che usa le parentesi di transazione implicite genera un'eccezione `CwDBTransactionException`. Se non si gestisce questa eccezione in modo esplicito, si genera un rollback automatico della transazione attiva. Si può utilizzare il metodo `inTransaction()` per determinare se la transazione è attiva.

Consultare inoltre

`getDBConnection()`, `inTransaction()`, `isActive()`

rollBack()

Esegue il rollback della transazione attiva associata con la connessione corrente.

Sintassi

```
void rollBack()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore database.

Note

Il metodo `rollback()` termina la transazione attiva annullando cambiamento effettuato sul database associato con la connessione corrente. I metodi `beginTransaction()`, `commit()` e `rollBack()` insieme permettono la gestione dei confini della transazione per un transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE, o UPDATE, ed una procedura memorizzata che comprende una di queste istruzioni SQL. Se il rollback ha esito negativo, `rollback()` restituisce l'eccezione`CwDBTransactionException` e registra un errore.

Importante: Utilizzare `rollback()` solo se la connessione utilizza parentesi di transazione esplicite. Se la connessione utilizza parentesi di transazione implicite, l'utilizzo di `rollback()` potrebbe generare l'eccezione `CwDBTransactionException`. Se non si termina la transazione esplicita con `rollback()` (o `commit()`) prima che venga rilasciata la connessione, InterChange Server Express termina in modo implicito la transazione in base all'esito della mappa. Se la mappa ha esito positivo, InterChange Server Express sincronizza questa transazione del database. Se la mappa *non* ha esito positivo, InterChange Server Express effettua il rollback in modo implicito della transazione del database. Indipendentemente dal successo della mappa, InterChange Server Express registra un avviso.

Prima di avviare una transazione esplicita, si deve creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` nella classe `BaseDLM`. Assicurarsi che questa transazione utilizzi parentesi di transazione esplicite.

Consultare inoltre

`beginTransaction()`, `commit()`, `getDBConnection()`, `inTransaction()`

Capitolo 14. Classe CwDBStoredProcedureParam

Un oggetto CwDBStoredProcedureParam descrive un singolo parametro per una procedura memorizzata. La Tabella 130 riepiloga i metodi della classe CwDBStoredProcedureParam.

Tabella 130. Sommario metodi CwDBStoredProcedureParam

Metodo	Descrizione	Pagina
CwDBStoredProcedureParam()	Crea una nuova istanza CwDBStoredProcedureParam contenente le informazioni sull'argomento per il parametro di una procedura memorizzata.	395
getParamType()	Richiama il tipo in/out del parametro di procedura memorizzata corrente come costante di numero intero.	396
getValue()	Richiama il valore del parametro di procedura memorizzata corrente.	397

CwDBStoredProcedureParam()

Crea una nuova istanza CwDBStoredProcedureParam contenente le informazioni sull'argomento per il parametro di una procedura memorizzata.

Sintassi

```
CwDBStoredProcedureParam(int paramType, String paramValue);

CwDBStoredProcedureParam(int paramType, int paramValue);
CwDBStoredProcedureParam(int paramType, Integer paramValue);
CwDBStoredProcedureParam(int paramType, Long
paramValue);

CwDBStoredProcedureParam(int paramType, double paramValue);
CwDBStoredProcedureParam(int paramType, Double paramValue);
CwDBStoredProcedureParam(int paramType, float paramValue);
CwDBStoredProcedureParam(int paramType, Float paramValue);
CwDBStoredProcedureParam(int paramType, BigDecimal paramValue);

CwDBStoredProcedureParam(int paramType, boolean paramValue);
CwDBStoredProcedureParam(int paramType, Boolean paramValue);

CwDBStoredProcedureParam(int paramType, java.sql.Date paramValue);
CwDBStoredProcedureParam(int paramType, java.sql.Time paramValue);
CwDBStoredProcedureParam(int paramType, java.sql.Timestamp paramValue);

CwDBStoredProcedureParam(int paramType, java.sql.Blob paramValue);
CwDBStoredProcedureParam(int paramType, java.sql.Clob paramValue);

CwDBStoredProcedureParam(int paramType, byte[] paramValue);
CwDBStoredProcedureParam(int paramType, Array paramValue);
CwDBStoredProcedureParam(int paramType, Struct paramValue);
```

Parametri

paramType Il tipo di parametro di in/out del parametro della procedura memorizzata associata.

paramValue Il valore dell'argomento da inviare alla procedura memorizzata. Questo valore è di uno dei seguenti tipi di dati Java

Valori restituiti

Restituisce un nuovo oggetto `CwDBStoredProcedureParam` che contiene le informazioni dell'argomento, per un argomento nella dichiarazione della procedura memorizzata.

Eccezioni

Nessuno.

Note

Il costruttore `CwDBStoredProcedureParam()` crea un'istanza `CwDBStoredProcedureParam` per descrivere un parametro di una procedura memorizzata. Le informazioni del parametro comprendono le seguenti:

- Il tipo in/out del parametro

  Il primo argomento del costruttore inizializza questo tipo di parametro di in/out. Per un elenco di parametri di in/out validi, consultare Tabella 131.
- Il valore del parametro

  Il secondo argomento del costruttore inizializza questo parametro. La classe `CwDBStoredProcedureParam` fornisce un modulo del proprio costruttore per ogni tipo di dati del valore del parametro che supporta.

Si fornisce un `Vettore` Java di parametri della procedura memorizzata al metodo `executeStoredProcedure()`, che crea una chiamata ad una procedura memorizzata sulla base del nome della procedura memorizzata ed del vettore dei parametri, ed invia questa chiamata al database associato con la connessione corrente.

Consultare inoltre

`executeStoredProcedure()`

---

getParamType()

Recupera il tipo in/out del parametro di procedura memorizzata corrente come costante di numero intero.

Sintassi

```
int getParamType()
```

Parametri

Nessuno.

Valori restituiti

Restituisce il tipo di in/out del parametro associato `CwDBStoredProcedureParam`.

Eccezioni

Nessuno.

Note

Il metodo `getParamType()` restituisce il tipo del parametro di in/out del parametro della procedura memorizzata corrente. Il tipo del parametro di in/out indica come la procedura memorizzata utilizza il parametro. La classe `CwDBStoredProcedureParam` rappresenta ogni tipo di in/out come una costante, come mostrato nella Tabella 131.

Tabella 131. Tipi di In/Out del parametro

Tipo di in/out del parametro	Descrizione	Costante del tipo In/Out
Parametro IN	Un parametro IN è *solo di input*, quindi, la procedura memorizzata accetta il suo valore come input ma *non* utilizza il parametro per restituire il valore.	`PARAM_IN`
Parametro OUT	Un parametro OUT è *solo di output*, quindi, la procedura memorizzata *non* ne legge il valore in input, ma utilizza il parametro per restituire il valore.	`PARAM_OUT`
Parametro INOUT	Un parametro INOUT è di *input ed output*, quindi, la procedura memorizzata accetta il suo valore come input ed utilizza il parametro anche per restituire il valore.	`PARAM_INOUT`

Consultare inoltre

`CwDBStoredProcedureParam()`, `getValue()`

getValue()

Recupera il valore del parametro di procedura memorizzata corrente.

Sintassi

```
Object getValue()
```

Parametri

Nessuno.

Valori restituiti

restituisce il valore del parametro `CwDBStoredProcedureParam` associato come `Oggetto` Java.

Eccezioni

Nessuno.

Note

Il metodo `getValue()` restituisce il valore del parametro come `Oggetto` Java (quale `Integer`, `Double` , oppure `String`). Il valore restituito ad un parametro OUT è il is the JDBC `NULL`, `getParamValue()` restituisce la costante`null`.

Consultare inoltre

`CwDBStoredProcedureParam()`, `getParamType()`

Capitolo 15. Classe DtpConnection

La classe `DtpConnection` è parte del Data Transformation Package (DTP). Fornisce metodi per l'esecuzione di query SQL su database relazionali. Per creare l'istanza di questa classe, si deve effettuare la chiamata `getRelConnection()` nella classe `BaseDLM`. Tutte le mappe sono derivate o sono classi secondarie di `BaseDLM`, così hanno accesso a `getRelConnection()`.

Importante: La classe `DtpConnection` ed i suoi metodi sono supportate *solo* per la compatibilità con le versioni precedenti. Questi *metodi obsoleti* non generano errori, ma è il caso di evitarli e migrare il codice esistente ai nuovi metodi. I metodi obsoleti potrebbero essere eliminati in un futuro release. Nello sviluppo di una nuova mappa, utilizzare la classe `CwDBConnection` ed i relativi metodi per stabilire una connessione con un database.

La Tabella 132 riepiloga i metodi della classe `DtpConnection`.

Tabella 132. Sommario metodi DtpConnection

Metodo	Descrizione	Pagina
`beginTran()`	Avvia una transazione SQL per il database relazionale.	399
`commit()`	Sincronizza la transazione corrente nel database relazionale.	400
`executeSQL()`	Esegue una query SQL nel database relazionale specificando un'istruzione CALL.	401
`execStoredProcedure()`	Esegue una procedura memorizzata SQL nel database relazionale specificandone il nome ed il vettore di parametri.	402
`getUpdateCount()`	Restituisce il numero di righe che sono state modificate nell'ultima operazione di scrittura sul database relazionale.	403
`hasMoreRows()`	Determina se il risultato della query ha più righe da elaborare.	403
`inTransaction()`	Determina se una transazione è in avanzamento nel database relazionale.	404
`nextRow()`	Richiama la riga successiva nel vettore del risultato della query.	404
`rollBack()`	Effettua il rollback della transazione corrente nel database relazionale.	405

beginTran()

Avvia una transazione SQL per il database relazionale.

Sintassi

```
void beginTran()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

I metodi `beginTran()`, `commit()` e `rollBack()` insieme forniscono supporto alla transazione per le query SQL.

Prima di avviare una transazione, si deve creare un oggetto `DtpConnection` con il metodo `getRelConnection()` nella classe `BaseDLM`.

Esempi

L'esempio che segue utilizza una transazione per eseguire una query allo scopo di inserire righe in una tabella nella relazione `SapCust`.

```
DtpConnection connection = getRelConnection("SapCust");

// begin a transaction
connection.beginTran();

// insert a row
connection.executeSQL("insert...");

// commit the transaction
connection.commit();
```

Consultare inoltre

`commit()`, `getRelConnection()`, `inTransaction()`, `rollBack()`

---

commit()

Sincronizza la transazione corrente nel database relazionale.

Sintassi

```
void commit()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

I metodi `beginTran()`, `commit()` e `rollBack()` insieme forniscono supporto alla transazione per le query SQL.

Prima di avviare una transazione, si deve creare un oggetto `DtpConnection` con il metodo `getRelConnection()` nella classe `BaseDLM`.

Esempi

L'esempio che segue utilizza una transazione per eseguire una query allo scopo di inserire righe in una tabella nella relazione SapCust.

```
DtpConnection connection = getRelConnection("SapCust");

// begin a transaction
connection.beginTran();

// insert a row
connection.executeSQL("insert...");

// commit the transaction
connection.commit();
```

Consultare inoltre

`beginTran()`, `getRelConnection()`, `inTransaction()`, `rollBack()`

---

executeSQL()

Esegue una query SQL nel database relazionale specificando istruzione CALL.

Sintassi

```
void executeSQL(String query)
void executeSQL(String query, Vector queryParameters)
```

Parametri

query La query SQL da eseguire nel database relazionale.

queryParameters
Un oggetto Vettore di argomenti da passare ai parametri nella query SQL.

Valori restituiti

Nessuno.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

Prima di eseguire una query con `executeSQL()`, si deve creare un oggetto `DtpConnection` con il metodo `getRelConnection()` nella classe `BaseDLM`.

Le istruzioni SQL che è possibile eseguire comprendono INSERT, SELECT, DELETE e UPDATE. Si possono inoltre eseguire procedure memorizzate con l'unica limitazione che tali procedure *non* possono utilizzare i parametri OUT. Per eseguire procedure memorizzate con parametri OUT, utilizzare il metodo `execStoredProcedure()`.

Esempi

L'esempio che segue esegue una query allo scopo di inserire righe in una tabella nella relazione `SapCust`.

```
DtpConnection connection = getRelConnection("SapCust");

// begin a transaction
```

```
connection.beginTran();

// insert a row
connection.executeSQL("insert...");

// commit the transaction
connection.commit();

// release the database connection
releaseRelConnection(true);
```

Consultare inoltre

`execStoredProcedure()`, `getRelConnection()`, `hasMoreRows()`, `nextRow()`

---

execStoredProcedure()

Esegue una procedura memorizzata SQL nel database relazionale specificandone il nome ed il vettore di parametri.

Sintassi

```
void execStoredProcedure(String storedProcedure,
   Vector storedProcParameters)
```

Parametri

storedProcedure
: Il nome della procedura memorizzata SQL da eseguire nel database relazionale.

storedProcParameters
: Un oggetto `Vettore` di parametri da passare alla procedura memorizzata. Ogni parametro è un'istanza della classe `UserStoredProcedureParam`.

Valori restituiti

Nessuno.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

Prima di eseguire una procedura memorizzata con `execStoredProcedure()`, si deve creare un oggetto `DtpConnection` con il metodo `getRelConnection()` nella classe `BaseDLM`.

Si possono inoltre utilizzare il metodo `executeSQL()` per eseguire una procedura memorizzata, fin quando questa procedura memorizzata non contiene parametri OUT. Se la procedura memorizzata contiene parametri OUT, si *deve* utilizzare `execStoredProcedure()` per eseguirla. A differenza di quanto accade con `executeSQL()`, non è necessario passare l'intera istruzione SQL per eseguire una procedura memorizzata. Con `execStoredProcedure()`, si deve passare unicamente il nome della procedura memorizzata ed il vettore di parametri `Vector` dell'oggetto `UserStoredProcedureParam`. Il metodo `execStoredProcedure()` può determinare il numero di parametri dal vettore *`storedProcParameters`* e compilare l'istruzione di chiamata per la procedura memorizzata.

Consultare inoltre

`executeSQL()`, `getRelConnection()`, `hasMoreRows()`, `nextRow()`

getUpdateCount()

Restituisce il numero di righe che sono state modificate nell'ultima operazione di scrittura sul database relazionale.

Sintassi

```
int getUpdateCount()
```

Parametri

Nessuno.

Valori restituiti

Restituisce un `int` che rappresenta il numero di righe che sono state modificate nell'ultima operazione di scrittura sul database.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

Prima di utilizzare questo metodo, si deve creare un oggetto `DtpConnection` con il metodo `getRelConnection()` nella classe `BaseDLM`.

Questo metodo è utile quando, dopo l'invio di un istruzione UPDATE o INSERT al database relazionale, si vuole determinare il numero di righe modificate dall'istruzione SQL.

Consultare inoltre

`executeSQL()`, `getRelConnection()`

hasMoreRows()

Determina se il risultato della query ha più righe da elaborare .

Sintassi

```
boolean hasMoreRows()
```

Parametri

Nessuno.

Valori restituiti

Restituisce `true` se ci sono più righe.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

Il metodo `hasMoreRows()` determina se la query, associata con il database relazionale corrente, ha più righe da elaborare. Utilizzare questo metodo per richiamare i risultati dalla query che restituisce dati. Tale query include un'istruzione SELECT e una procedura memorizzata. Solo una query alla volta può essere associata con la connessione. Quindi, se si esegue un'altra query prima che `hasMoreRows()` restituisca `false`, si perdono i dati dalla query iniziale.

Consultare inoltre

`nextRow()`, `executeSQL()`, `getUpdateCount()`

---

inTransaction()

Determina se una transazione è in avanzamento nel database relazionale .

Sintassi

```
boolean inTransaction()
```

Parametri

Nessuno.

Valori restituiti

Restituisce `"True"` se una transazione è in esecuzione.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

Prima di avviare una transazione, si deve creare un oggetto `DtpConnection` con il metodo `getRelConnection()` nella classe `BaseDLM`.

Consultare inoltre

`beginTran()`, `commit()`, `getRelConnection()`, `rollBack()`

---

nextRow()

Richiama la riga successiva nel vettore del risultato della query.

Sintassi

```
Vector nextRow()
```

Parametri

Nessuno.

Valori restituiti

Restituisce la riga successiva rispetto al risultato della query come un oggetto `Vettore`.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

Il metodo `nextRow()` restituisce una riga di dati della query associata con il database relazionale. Utilizzare questo metodo per richiamare i risultati dalla query che restituisce dati. Tale query include un'istruzione SELECT e una procedura memorizzata. Solo una query alla volta può essere associata con la connessione. Quindi, se si esegue un'altra query prima che `nextRow()` restituisca l'ultima riga di dati, si perdono i dati della query iniziale.

Consultare inoltre

`hasMoreRows()`, `executeSQL()`, `getUpdateCount()`

---

rollBack()

Effettua il rollback della transazione corrente nel database relazionale.

Sintassi

```
void rollBack()
```

Parametri

Nessuno.

Valori restituiti

Nessuno.

Eccezioni

`DtpConnectionException` – Se si verifica un errore database.

Note

I metodi `beginTran()`, `commit()` e `rollBack()` insieme forniscono supporto alla transazione per le query SQL.

Prima di avviare una transazione, si deve creare un oggetto `DtpConnection` con il metodo `getRelConnection()` nella classe `BaseDLM`.

Consultare inoltre

`beginTran()`, `commit()`, `getRelConnection()`, `inTransaction()`

Capitolo 16. Classe DtpDataConversion

Una delle attività più comuni nella mappatura di oggetti business è la conversione di valori di attributi da un tipo di dati ad un altro, un processo denominato *conversione dati.* Il metodo `DtpDataConversion` permette di effettuare la conversione dati in modo semplice.

Le classi tipo di dati nel pacchetto `java.lang` contengono alcuni metodi di conversione, ma non sono supportate tutte le possibili conversioni. La classe `DtpDataConversion` concentra diversi metodi di conversione in un unica classe e supporta le conversioni più comuni che si eseguono nelle mappe. I metodi `getType()` e `isOKToConvert()` rendono semplice determinare se è possibile effettuare una determinata conversione.

Tutti i metodi in questa classe sono dichiarati come statici. La Tabella 133 riepiloga i metodi della classe `DtpDataConversion`.

Tabella 133. Sommario metodi DtpDataConversion

Metodo	Descrizione	Pagina
`getType()`	Determina il tipo dati di un valore.	407
`isOKToConvert()`	Determina se è possibile effettuare la conversione da un tipo dati ad un altro.	409
`toBoolean()`	Converte un oggetto Java in un oggetto `Booleano`.	410
`toDouble()`	Converte un oggetto o un tipo dati primitivo in un oggetto `Double`.	411
`toFloat()`	Converte un oggetto o un tipo dati primitivo in un oggetto `Float`.	411
`toInteger()`	Converte un oggetto o un tipo dati primitivo in un oggetto `Integer`.	412
`toPrimitiveBoolean()`	Converte un oggetto String o Boolean in un tipo dati `boolean` primitivo.	413
`toPrimitiveDouble()`	Converte un oggetto o un tipo dati primitivo in un tipo dati `double` primitivo.	413
`toPrimitiveFloat()`	Converte un oggetto o un tipo dati primitivo in un tipo dati `float` primitivo.	414
`toPrimitiveInt()`	Converte un oggetto o un tipo dati primitivo in un tipo dati `int` primitivo.	415
`toString()`	Converte un oggetto o un tipo dati primitivo in un oggetto String.	416

getType()

Determina il tipo di dati di un valore.

Sintassi

```
int getType(Object objectData)
int getType(int integerData)
int getType(float floatData)
int getType(double doubleData)
int getType(boolean booleanData)
```

Parametri

objectData Qualsiasi oggetto Java.

integerData Qualsiasi variabile `int` primitiva.

floatData Qualsiasi variabile `float` primitiva.

doubleData Qualsiasi variabile `double` primitiva.

booleanData Qualsiasi variabile `boolean` primitiva.

Valori restituiti

Restituisce un numero intero rappresentante il tipo di dati del valore del parametro. È possibile interpretare il valore di ritorno confrontandolo con una di queste costanti che sono dichiarate come statiche e finali nella classe `DtpDataConversion`:

INTEGER_TYPE I dati hanno un valore `int` primitivo oppure un oggetto `Integer`.

STRING_TYPE I dati sono un oggetto`String`.

FLOAT_TYPE I dati hanno un valore `float` primitivo oppure un oggetto `Float`.

DOUBLE_TYPE I dati hanno un valore `double` primitivo oppure un oggetto `Double`.

BOOL_TYPE I dati hanno un valore `boolean` primitivo oppure un oggetto `Boolean`.

DATE_TYPE I dati sono un oggetto `Data`.

LONGTEXT_TYPE I dati sono un oggetto`Longtext`.

UNKNOWN_TYPE I dati sono di tipo sconosciuto.

Eccezioni

Nessuno.

Note

È possibile utilizzare i valori restituiti da `getType()` nel metodo `OKToConvert()` per determinare se è possibile una conversione tra i due tipi di dati.

Esempi

```
int conversionStatus = DtpDataConversion.isOKToConvert(
   DtpDataConversion.getType(srcObject),
   DtpDataConversion.getType(destObject));

switch(conversionStatus)
   {
   case DtpDataConversion.OKTOCONVERT:
      // go ahead and convert
      break;
   case DtpDataConversion.POTENTIALDATALOSS:
      // convert, then check value
      break;
   case DtpDataConversion.CANNOTCONVERT:
      // return an error
      break;
}
```

Consultare inoltre

`isOKToConvert()`

isOKToConvert()

Determina se è possibile effettuare la conversione da un tipo dati ad un altro.

Sintassi

```
int isOKToConvert(int srcDatatype, int destDataType)
int isOKToConvert(String srcDataTypeStr, String destDataTypeStr)
```

Parametri

srcDataType Un numero intero restituito da `getType()`, che rappresenta il tipo di dati del valore origine che si vuole convertire.

destDataType Un numero intero restituito da `getType()`, che rappresenta il tipo di dati in cui si vuole convertire il valore origine.

srcDataTypeStr
Una stringa contenente il nome del tipo di dati per il valore origine che si vuole convertire. I valori possibile: `Boolean`, boolean, `Double`, `double`, `Float`, `float`, `Integer`, `int`, e `String`.

destDataTypeStr
Una stringa contenente il nome del tipo di dati in cui si vuole convertire il valore origine. I valori possibile: `Boolean`, boolean, `Double`, `double`, `Float`, `float`, `Integer`, `int`, e `String`.

Valori restituiti

Restituisce un numero intero che specifica se è possibile convertire un valore del tipo di dati di origine in un valore del tipo di dati di destinazione. È possibile interpretare il valore di ritorno confrontandolo con una di queste costanti che sono dichiarate come statiche e finali nella classe `DtpDataConversion`:

`OKTOCONVERT` È possibile convertire il tipo di dati di origine in quelli di destinazione.

`POTENTIALDATALOSS`
La conversione è possibile, ma c'è il rischio di perdita di dati se l'origine contiene caratteri non recuperabili o deve essere troncata per adattarsi al tipo di dati di destinazione.

`CANNOTCONVERT` Non è possibile convertire il tipo di dati di origine in quelli di destinazione.

Eccezioni

Nessuno.

Note

Il metodo restituisce un numero intero che rappresenta il tipo di dati del valore passato come parametro. Si utilizza il primo modulo di `isOKToConvert()` assieme con `getType()` per determinare se una conversione dati tra due attributi è possibile. Se la chiamata metodo `isOKToConvert()`, utilizza `getType()` su entrambi gli attributi di origine e di destinazione per generare i parametri *srcDataType* e *destDataType* .

Il secondo modulo del metodo accetta valori String che contengono nomi di tipi di dati dei dati di origine e di destinazione. Utilizzare questo modulo se si vuole sapere che tipi di dati sono, e si vuole verificare se è possibile una conversione.

La Tabella 134 mostra le possibili conversioni per ogni combinazione di tipi di dati di origine e di destinazione. Nella tabella:

- OK significa che la conversione è possibile senza perdita di dati.
- PD significa che la conversione è possibile, ma c'è il rischio di perdita di dati se l'origine contiene caratteri non recuperabili o deve essere troncata per adattarsi al tipo di dati di destinazione.
- NO significa che la conversione non è possibile.

Tabella 134. Possibili conversioni tra tipi di dati

	DESTINAZIONE						
ORIGINE	**int, Integer**	**String**	**float, Float**	**double, Double**	**booleano Boolean**	**Date**	**Longtext**
int Integer	OK	OK	OK	OK	NO	NO	OK
String	PD[1]	OK	PD[1]	PD[1]	PD[2]	PD	OK
float, Float	PD[3]	OK	OK	OK	NO	NO	OK
double, Double	PD[3]	OK	PD[3]	OK	NO	NO	OK
boolean, Boolean	NO	OK	NO	NO	OK	NO	OK
Date	NO	OK	NO	NO	NO	OK	OK
Longtext	PD[1]	PD[3]	PD[1]	PD[1]	PD[2]	PD	OK

[1]Durante la conversione di un valore String o Longtext in un qualsiasi tipo numerico, il valore String o Longtext deve contenere unicamente numeri e decimali. È necessario rimuovere qualsiasi altro carattere, come simboli, dai valori String o Longtext prima della conversione. Altrimenti verrà restituita una eccezione `DtpIncompatibleFormatException`.

[1]Durante la conversione di un valore String o Longtext in un Boolean, il valore String o Longtext deve essere `"true"` o `"false"`. Qualsiasi stringa che non sia `"true"` (maiuscole/minuscole non sono considerate) verrà considerata false.

[3]Poiché il tipo di dati di origine supporta un maggiore precisione di quelli di destinazione, il valore potrebbe essere troncato.

Esempi

```
if (DtpDataConversion.isOKToConvert(getType(mySource),
      getType(myDest))== DtpDataConversion.OKTOCONVERT)
   // map these attributes
else
   // skip these attributes
```

Consultare inoltre

`getType()`

toBoolean()

Converte un oggetto Java in un oggetto `Boolean`.

Sintassi

```
Boolean toBoolean(Object objectData)
Boolean toBoolean(boolean booleanData)
```

Parametri

objectData Un oggetto Java che si desidera converitre in `Boolean`. L'unico oggetto attualmente supportato è `String`.

booleanData Qualsiasi variabile `boolean` primitiva.

Valori restituiti

Restituisce un oggetto `Boolean`.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in Boolean.

Esempi

```
Boolean MyBooleanObj = DtpDataConversion.toBoolean(MyStringObj);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toPrimitiveBoolean()`

---

toDouble()

Converte un oggetto o un tipo di dati primitivo in oggetto `Double`.

Sintassi

```
Double toDouble(Object objectData)
Double toDouble(int integerData)
Double toDouble(float floatData)
Double toDouble(double doubleData)
```

Parametri

objectData Un oggetto Java. Gli oggetti attualmente supportati sono: `Float`, `Integer`, e `String`.

integerData Qualsiasi variabile `int` primitiva.

floatData Qualsiasi variabile `float` primitiva.

doubleData Qualsiasi variabile `double` primitiva.

Valori restituiti

Restituisce un oggetto `Double`.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in Double.

Esempi

```
Double myDoubleObj = DtpDataConversion.toDouble(myInteger);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toPrimitiveDouble()`

---

toFloat()

Converte un oggetto o un tipo di dati primitivo in oggetto `Float`.

Sintassi

```
Float toFloat(Object objectData)
Float toFloat(int integerData)
Float toFloat(float floatData)
Float toFloat(double doubleData)
```

Parametri

objectData	Un oggetto Java. Gli oggetti attualmente supportati sono: `Double`, `Integer`, e `String`.
integerData	Qualsiasi variabile `int` primitiva.
floatData	Qualsiasi variabile `float` primitiva.
doubleData	Qualsiasi variabile `double` primitiva.

Valori restituiti

Restituisce un oggetto `Float`.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in Float.

Esempi

```
Float myFloatObj = DtpDataConversion.toFloat(myInteger);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toPrimitiveFloat()`

---

toInteger()

Converte un oggetto o un tipo di dati primitivo in oggetto `Integer`.

Sintassi

```
Integer toInteger(Object objectData)
Integer toInteger(int integerData)
Integer toInteger(float floatData)
Integer toInteger(double doubleData)
```

Parametri

objectData	Un oggetto Java. Gli oggetti attualmente supportati sono: `Double`, `Float`, e `String`.
integerData	Qualsiasi variabile `int` primitiva.
floatData	Qualsiasi variabile `float` primitiva.
doubleData	Qualsiasi variabile `double` primitiva.

Valori restituiti

Restituisce un oggetto `Integer`.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in Integer.

Esempi

```
Integer myIntegerObj = DtpDataConversion.toInteger(myFloat);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toPrimitiveInt()`

---

toPrimitiveBoolean()

Converte un oggetto String o Boolean in un tipo di dati `boolean` primitivo.

Sintassi

```
boolean toPrimitiveBoolean(Object objectData)
```

Parametri

objectData Un oggetto `String` o `Boolean` che si vuole convertire in tipo di dati `boolean` primitivo.

Valori restituiti

Restituisce un oggetto `boolean` primitivo.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in boolean.

Note

Questo metodo può gestire unicamente oggetti String e Boolean.

Esempi

```
boolean MyBoolean = DtpDataConversion.toPrimitiveBoolean(MyStringObj);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toBoolean()`

Per ulteriori informazioni, consultare Java Language Specification.

---

toPrimitiveDouble()

Converte un oggetto oppure un tipo di dati primitivo in un tipo di dati `double` primitivo.

Sintassi

```
double toPrimitiveDouble(Object objectData)
double toPrimitiveDouble(int integerData)
double toPrimitiveDouble(float floatData)
```

Parametri

objectData Un oggetto Java. Gli oggetti attualmente supportati sono: `Double`, `Float`, `Integer`, e `String`.

integerData Qualsiasi variabile `int` primitiva.

floatData Qualsiasi variabile `float` primitiva.

Valori restituiti

Restituisce un oggetto `double` primitivo.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in double.

Note

Questo metodo gestisce unicamente oggetti String, Integer, Float, e Double ed il valore di ritorno non può essere uguale al valore di input.

Questo metodo ha la stessa limitazione del tipo double in Java.

Il più alto numero positivo finito letterale `double` è `1.79769313486231570e+308`. Il più basso numero positivo diverso da zero letterale di tipo `double` è `4.94065645841246544e-324`, con 15 cifre decimali significative (nell'ordine di `999,999,999,999.99`, nel raggio di miliardi).

Esempi

```
double myDouble = DtpDataConversion.toPrimitiveDouble(myObject);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toDouble()`

Per ulteriori informazioni, consultare Java Language Specification.

---

toPrimitiveFloat()

Converte un oggetto oppure un tipo di dati primitivo in un tipo di dati `float` primitivo.

Sintassi

```
float toPrimitiveFloat(Object objectData)
float toPrimitiveFloat(int integerData)
float toPrimitiveFloat(double doubleData)
```

Parametri

objectData Un oggetto Java. Gli oggetti attualmente supportati sono: `Double`, `Float`, `Integer`, e `String`.

integerData Qualsiasi variabile `int` primitiva.

doubleData Qualsiasi variabile `double` primitiva.

Valori restituiti

Restituisce un oggetto `float` primitivo.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in float.

Note

Questo metodo gestisce unicamente oggetti String, Integer, Float, e Double e ci sarà una perdita di dati quando il tipo input è Double.

Questo metodo ha la stessa limitazione del tipo float in Java.

Il più alto numero positivo finito letterale `float` è `3.40282347e+38f`. Il numero positivo finito più basso diverso da zero letterale di tipo `float` è `1.40239846e-45f`, con 6-7 cifre significative. È bene non utilizzare valori superiori a `99,999.99` con questo tipo dati.

Esempi

```
float myFloat = DtpDataConversion.toPrimitiveFloat(myInteger);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toFloat()`

Per ulteriori informazioni, consultare Java Language Specification.

---

toPrimitiveInt()

Converte un oggetto oppure un tipo di dati primitivo in un tipo di dati `int` primitivo.

Sintassi

```
int toPrimitiveInteger(Object objectData)
int toPrimitiveInteger(float floatData)
int toPrimitiveInteger(double doubleData)
```

Parametri

objectData	Un oggetto Java. Gli oggetti attualmente supportati sono: `Double`, `Float`, `Integer`, e `String`.
floatData	Qualsiasi variabile `float` primitiva.
doubleData	Qualsiasi variabile `double` primitiva.

Valori restituiti

Restituisce un oggetto `int` primitivo.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in integer.

Note

Questo metodo gestisce unicamente oggetti String, Integer, Float, e Double e ci sarà una perdita di dati quando il tipo input è Float o Double.

Questo metodo ha la stessa limitazione del tipo int in Java.

Il numeri positivi più alti esadecimali o ottali letterali del tipo int rispettivamente sono `0xffffffff` e `017777777777`, che sono uguali a `2147483647 (231-1)`. I numeri negativi esadecimali ed ottali più alti letterali di tipo int sono rispettivamente `0x80000000e 020000000000`, ognuno dei quali rappresenta un valore decimale `-2147483648 (-231)`. I letterali esadecimali ed ottali `0xffffffff` e `037777777777`, rappresentano il valore decimale `-1`.

Esempi

```
int myInt = DtpDataConversion.toPrimitiveInt(myObject);
```

Consultare inoltre

`getType()`, `isOKToConvert()`, `toInteger()`

Per ulteriori informazioni, consultare Java Language Specification.

---

toString()

Converte un oggetto oppure un tipo di dati primitivo in un oggetto String.

Sintassi

```
String toString(Object objectData)
String toString(int integerData)
String toString(float floatData)
String toString(double doubleData)
```

Parametri

objectData	Un oggetto Java. Gli oggetti attualmente supportati sono: `Double`, `Float`, e `Integer`.
integerData	Qualsiasi variabile `int` primitiva.
floatData	Qualsiasi variabile `float` primitiva.
doubleData	Qualsiasi variabile `double` primitiva.

Valori restituiti

Restituisce un oggetto `String`.

Eccezioni

`DtpIncompatibleFormatException` – Se il tipo di dati di origine non è convertibile in String.

Esempi

```
String myString = DtpDataConversion.toString(myObject);
```

Consultare inoltre

`getType()`, `isOKToConvert()`

Capitolo 17. Classe DtpDate

La classe `DtpDate` confronta valori di data ed ora, ne imposta i formati, e restituisce i componenti di un valore ora e data.

I metodi statici (classi) operano sul nome della classe. I metodi statici prendono una serie di oggetti business e restituiscono le prime o le ultime date oppure l'oggetto business che contiene le prime o le ultime date.

I metodi di istanza operano su oggetti business. Si passa un valore data al costruttore `DtpDate` ed è quindi possibile manipolare l'oggetto data risultante. I metodi di istanza permettono di recuperare, formattare, e modificare i valori associati alla data. Si possono inoltre impostare i formati in cui si desidera gestire le date.

I metodi di conversione dati sono utili quando due applicazioni archiviano le date in formati diversi. ad esempio, il SAP potrebbe inviare date nel formato `26/8/1999 15:23:20` mentre Clarify potrebbe aver bisogno della data nel formato `26 Augusto, 1999 15:23:20`.

I valori passati alla classe `DtpDate` devono seguire queste regole:

Giorno	Un numero da 1 a 30. Se nella stringa data-ora non è presente un separatore tra il mese, l'anno e la data, e questa è in formato numerico, i singoli caratteri devono essere preceduti da uno zero (0), come 01
Mese	Un numero compreso tra 1 e 12, un nome come Gennaio o Febbraio, oppure una abbreviazione del mese (3 caratteri) come Gen o Feb. Se nella stringa data-ora non è presente un separatore tra il mese, l'anno e la data, e questa è in formato numerico, i singoli caratteri devono essere preceduti da uno zero (0), come 01
Anno	Un numero di 4 cifre.
Ora	Un valore nell'intervallo da 1 a 23, rappresentazione nel formato a 24 ore. AM o PM non sono permessi.
Minuti	Un numero nell'intervallo da 01 a 59.
Second.	Un numero nell'intervallo da 01 a 59.

La Tabella 135 riepiloga i metodi della classe `DtpDate`. Notare che i metodi statici e di istanza sono separati nella tabella ma sono in ordine alfabetico nel capitolo.

Tabella 135. Sommario metodi DtpDate

Metodo	Descrizione	Pagina
Costruttore		
`DtpDate()`	Analizza la data in base al formato specificato.	419
Metodi statici		
`getMaxDate()`	Da un elenco di oggetti business, restituisce la data più recente come oggetto `DtpDate`.	431
`getMinDate()`	Da un elenco di oggetti business, restituisce la data meno recente come oggetto `DtpDate`.	434
`getMaxDateBO()`	Da un elenco di oggetti business, restituisce quelli con la data più recente.	433
`getMinDateBO()`	Da un elenco di oggetti business, restituisce quelli con la data meno recente.	435

Tabella 135. Sommario metodi DtpDate (Continua)

Metodo	Descrizione	Pagina
Metodi di istanza		
`addDays()`	Aggiunge il numero di giorni specificato a questa data.	421
`addWeekdays()`	Aggiunge il numero di giorni feriali specificato a questa data.	422
`addYears()`	Aggiunge il numero di anni specificato a questa data.	422
`after()`	Controlla se questa data è successiva a quella fornita come parametro di input.	423
`before()`	Controlla se questa data è precedente a quella fornita come parametro di input.	424
`calcDays()`	Calcola il numero di giorni tra questa data ed un'altra.	424
`calcWeekdays()`	Calcola il numero di giorni feriali tra questa data ed un'altra.	425
`get12MonthNames()`	Restituisce la rappresentazione con nome breve corrente dei dodici mesi per questa data.	426
`get12ShortMonthNames()`	Restituisce la rappresentazione con nome completo corrente dei dodici mesi per questa data.	426
`get7DayNames()`	Restituisce i nomi correnti dei sette giorni della settimana per questa data.	427
`getCWDate()`	Formatta nuovamente la data nel formato data generico IBM.	427
`getDayOfMonth()`	Restituisce il giorno del mese per questa data.	427
`getDayOfWeek()`	Restituisce il giorno della settimana per questa data.	428
`getHours()`	Restituisce il valore ore per questa data.	428
`getIntDay()`	Restituisce il giorno della settimana in questa data come numero intero.	428
`getIntDayOfWeek()`	Restituisce il giorno della settimana per questa data.	429
`getIntMilliSeconds()`	Restituisce il valore millisecondi di questa data.	429
`getIntMinutes()`	Restituisce il valore minuti in questa data come numero intero.	429
`getIntMonth()`	Restituisce il mese in questa data come numero intero.	430
`getIntSeconds()`	Restituisce i secondi in questa data come numero intero.	430
`getIntYear()`	Restituisce l'anno in questa data come numero intero.	430
`getMSSince1970()`	Restituisce il numero in millisecondi tra il 1 Gennaio 1970 00:00:00 e questa data.	431
`getMinutes()`	Restituisce il valore minuti di questa data.	436
`getMonth()`	Restituisce la rappresentazione con nome completo del mese in questa data.	436
`getNumericMonth()`	Restituisce il valore del mese di questa data in formato numerico.	437
`getSeconds()`	Restituisce il valore dei secondi di questa data come stringa.	437
`getShortMonth()`	Restituisce la rappresentazione con nome breve del mese di questa data.	438
`getYear()`	Restituisce il valore anno di questa data.	438
`set12MonthNames()`	Modifica la rappresentazione con nome completo dei dodici mesi per questa data.	438

Tabella 135. Sommario metodi DtpDate (Continua)

Metodo	Descrizione	Pagina
`set12MonthNamesToDefault()`	Ripristina la rappresentazione con nome completo dei dodici mesi al valore predefinito per questa data.	439
`set12ShortMonthNames()`	Modifica la rappresentazione con nome breve dei dodici mesi per questa data.	439
`set12ShortMonthNamesToDefault()`	Ripristina la rappresentazione con nome breve dei dodici mesi al valore predefinito per questa data.	440
`set7DayNames()`	Modifica i nomi dei dei sette giorni della settimana per questa data.	440
`set7DayNamesToDefault()`	Ripristina i nomi dei giorni della settimana al valore predefinito per questa data.	441
`toString()`	Restituisce la data nel formato specificato o nel formato predefinito.	441

DtpDate()

Analizza la data in base al formato specificato.

Sintassi

```
public DtpDate()

public DtpDate(String dateTimeStr, String format)

public DtpDate(String dateTimeStr, String format, String[] monthNames,
   String[] shortMonthNames)

public DtpDate(long msSince1970, boolean isLocalTime)
```

Parametri

dateTimeStr La data-ora in formato stringa.

format Il formato della data. Consultare le note per i dettagli.

monthNames Un vettore di stringhe rappresentanti i nomi completi dei 12 mesi. Se nulla, il valore predefinito è Gennaio, Febbraio, Marzo, e così via.

shortMonthNames
Un vettore di stringhe rappresentanti i nomi brevi dei mesi. Se questo valore è nullo mentre *monthNames* non lo è, questo valore è composto dai primi tre caratteri con nome completo del mese, come Gen, Feb, Mar, Apr, e così via.

msSince1970 Il numero di millisecondi dal 1 Gennaio 1970 00:00:00.

isLocalTime Impostare su `true` se l'ora è già quella locale, altrimenti su `false`.

Valori restituiti

Nessuno

Eccezioni

`DtpDateException` - Quando il costruttore incontra errori di analisi. Ciò può accadere se la data non è nel formato specificato.

Note

Il primo modulo del costruttore non prende alcun parametro. Assegna la data corrente del sistema al nuovo oggetto `DtpDate`. Non emette `DtpDateException`.

Il secondo e terzo modulo del costruttore analizzano la data sulla base del *formato* della data specificato ed estraggono i valori giorno, mese, anno, ora, minuti, e secondi. Questi valori possono essere recuperati e formattati nuovamente con altri metodi `DtpDate`.

Ad esempio, un mese può essere recuperato in uno dei seguenti formati:

- La rappresentazione con nome completo (formato predefinito): Gennaio, Febbraio, Marzo, Aprile, Maggio, Giugno, Luglio, Agosto, Settembre, Ottobre, Novembre, e Dicembre
- Il formato numerico: 1-12
- La rappresentazione con nome breve, che consiste nei primi tre caratteri del nome di ogni mese: Gen, Feb, Mar, Apr, Mag, Giu, Lug, Ago, Set, Ott, Nov, Dic

I dati recuperati non dipendono dal contesto dagli altri dati.

Si può cambiare la rappresentazione con nome breve o completo del mese nei seguenti modi:

- Rispettivamente con i metodi `set12MonthNames()` e `set12ShortMonthNames()`
- Passando la rappresentazione come parametro nel terzo modulo del costruttore `DtpDate()`

Il quarto modulo del costruttore prende il numero di millisecondi dal 1 Gennaio 1970 00:00:00. Molte applicazioni rappresentano la dato in questo modo.

Formato data

Nel *formato* data, la data precede sempre l'ora. L'ora è facoltativa. Se è mancanti in una stringa data-ora, le ore, i minuti ed i secondi hanno il valore predefinito 00.

Il formato data utilizza le seguenti lettere chiave sensibili al maiuscolo/minuscolo:

`D`	giorno
`M`	mese
`Y`	anno
`h`	ore
`m`	minuti
`s`	secondi

queste lettere chiave possono essere separate con un separatore quale "/" o "-".

Esempi

Nell'esempio seguente mostra il costruttore `DtpDate()` che crea i nuovi oggetti data `aDate`, `date2`, e `date3`:

```
Dtpdate aDate = new DtpDate("5/21/1997 15:23:01", "M/D/Y h:m:s");
DtpDate date2 = new DtpDate("05211997 152301", "MDY hms");
DtpDate date3 = new DtpDate("Jan 10, 1999 10:00:00", "M D, Y h:m:s");
```

Il seguente formato data dà come risultato che viene emessa la `DtpDateException`:

```
h:m:s D/M/Y
```

addDays()

Aggiunge il numero di giorni specificato a questa data.

Sintassi

```
public DtpDate addDays(int numberOfDays)
```

Parametri

numberOfDays Un numero intero. Se è un numero negativo, la nuova data sarà quella precedente del numero di giorni specificati in *numberOfDays* rispetto all'istanza corrente di `DtpDate`.

Valori restituiti

Un nuovo oggetto `DtpDate`.

Eccezioni

`DtpDateException`

Note

Il metodo `addDays()` aggiunge il numero di giorni specificato a questa data. Si possono utilizzare i metodi `get()` per recuperare le informazioni relative alla nuova data risultante. L'oggetto `DtpDate` restituito eredita tutte le proprietà dell'oggetto corrente `DtpDate`, come il nome del mese, il formato della data e così via.

La nuova data sarà regolata in modo da risultare valida. Ad esempio, aggiungendo cinque giorni al 29 Gennaio 1999 00:00:00 si otterrà il 3 Febbraio 1999 00:00:00, aggiungendo poi -30 giorni si otterrà il 30 Dicembre 1998 00:00:00.

L'aggiunta di giorni non incide sull'ora del giorno.

Esempi

```
try
   {
   DtpDate toDay = new DtpDate();
   DtpDate tomorrow = toDay.addDays(1);
   System.out.println("Tomorrow is "
      + tomorrow.getDayOfMonth() + "/"
      + tomorrow.getNumericMonth() + "/"
      + tomorrow.getYear() + " "
      + tomorrow.getHours() + ":"
      + tomorrow.getMinutes() + ":"
      + tomorrow.getSeconds());
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Consultare inoltre

`addWeekdays()`, `addYears()`

addWeekdays()

Aggiunge il numero di giorni feriali specificato a questa data.

Sintassi

```
public DtpDate addWeekdays(int numberOfWeekdays)
```

Parametri

numberOfWeekdays
Un numero intero. Se è un numero negativo, la nuova data sarà quella precedente del numero di giorni feriali specificato in *numberOfWeekdays* rispetto all'istanza corrente di `DtpDate`.

Valori restituiti

Un nuovo oggetto `DtpDate`.

Eccezioni

`DtpDateException`

Note

Il metodo `addWeekdays()` aggiunge il numero di giorni feriali specificato a questa data. Si possono quindi utilizzare i metodi get per recuperare le informazioni della nuova data risultante. L'oggetto `DtpDate` restituito eredita tutte le proprietà dell'oggetto corrente `DtpDate`, come il nome del mese, il formato della data e così via.

Solo i giorni Lunedì, Martedì, Mercoledì, Giovedì e Venerdì o i valori equivalenti, sono considerati giorni feriali. Lunedì viene considerato il primo giorno della settimana.

Esempi

```
try
   {
   DtpDate toDay = new DtpDate("8/2/1999 00:00:00", "M/D/Y h:m:s");
   DtpDate fiveWeekdaysLater = toDay.addWeekdays(5);
   // The new date should be 8/9/1999 00:00:00
   System.out.println("Next month is "
      + fiveWeekdaysLater.getDayOfMonth() + "/"
      + fiveWeekdaysLater.getNumericMonth() + "/"
      + fiveWeekdaysLater.getYear() + " "
      + fiveWeekdaysLater.getHours() + ":"
      + fiveWeekdaysLater.getMinutes() + ":"
      + fiveWeekdaysLater.getSeconds());
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Consultare inoltre

`addDays()`, `addYears()`

addYears()

Aggiunge il numero di anni specificato a questa data.

Sintassi

```
public DtpDate addYears(int numberOfYears)
```

Parametri

numberOfYears Un numero intero. Se è un numero negativo, la nuova data sarà quella precedente del numero di anni specificato in *numberOfYears* rispetto all'istanza corrente di `DtpDate`.

Valori restituiti

Un nuovo oggetto `DtpDate`.

Note

Il metodo `addYears()` aggiunge il numero di anni specificato a questa data. Si possono utilizzare i metodi `get()` per recuperare le informazioni relative alla nuova data risultante. L'oggetto `DtpDate` restituito eredita tutte le proprietà dell'oggetto corrente `DtpDate`, come il nome del mese, il formato della data e così via.

Esempi

```
DtpDate toDay = new DtpDate();
DtpDate lastYear= toDay.addYears(-1);
System.out.println("Next month is "
   + lastYear.getDayOfMonth() + "/"
   + lastYear.getNumericMonth() + "/"
   + lastYear.getYear() + " "
   + lastYear.getHours() + ":"
   + lastYear.getMinutes() + ":"
   + lastYear.getSeconds());
```

Consultare inoltre

`addDays()`, `addWeekdays()`

---

after()

Controlla se questa data è successiva a quella immessa come parametro di input.

Sintassi

```
public boolean after(DtpDate date)
```

Parametri

date La data da confrontare con questa data.

Valori restituiti

Restituisce `true` se questa data è successiva a quella immessa, e `false` se questa data è precedente.

Eccezioni

`DtpDateException`

Esempi

```
try
   {
   DtpDate toDay = new DtpDate();
   DtpDate tomorrow = yesterday.addDays(-1);
   // isAfter should be false.
   boolean isAfter = yesterday.after(today)
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Consultare inoltre

`before()`

---

before()

Controlla se questa data è precedente a quella immessa come parametro di input.

Sintassi

```
public boolean before(DtpDate date)
```

Parametri

date La data da confrontare con questa data.

Valori restituiti

Restituisce `true` se questa data è precedente a quella immessa, e `false` se questa data è successiva.

Eccezioni

`DtpDateException`

Esempi

```
try
   {
   DtpDate toDay = new DtpDate();
   DtpDate tomorrow = yesterday.addDays(-1);
   // isBefore should be true.
   boolean isBefore = yesterday.before(today)
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Consultare inoltre

`after()`

---

calcDays()

Calcola il numero di giorni tra questa data ed un'altra data.

Sintassi

```
public int calcDays(DtpDate date)
```

Parametri

date La data da confrontare con questa data.

Valori restituiti

Un numero intero che rappresenta il numero di giorni. Sempre un numero positivo.

Eccezioni

`DtpDateException`

Note

Il metodo `calcDays()` calcola la differenza in numero di giorni tra questa data ed un'altra. Il numero è sempre l'intero numero di giorni.

La differenza tra 19990615 00:30:59 e 19990615 23:59:59 è 0 giorni, mentre la differenza tra 19990615 23:59:59 e 19990616 00:01:01 è di 1 giorno.

Esempi

```
try
   {
   DtpDate toDay = new DtpDate();
   DtpDate tomorrow = toDay.addDays(1);
   int days = today.caldDays(tomorrow);
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Consultare inoltre

`calcWeekdays()`

---

calcWeekdays()

Calcola il numero di giorni feriali tra questa data ed un'altra data.

Sintassi

```
public int calcWeekdays(DtpDate date)
```

Parametri

date La data da confrontare con questa data.

Valori restituiti

Un numero intero che rappresenta il numero di giorni feriali. Sempre un numero positivo.

Eccezioni

`DtpDateException`

Note

Il metodo `calcWeekdays()` calcola il numero di giorni feriali tra questa data ed un'altra. La differenza tra sabato e Domenica è 0, e la differenza tra Venerdì e Lunedì è 1. Si assume che i giorni feriali vadano dal Lunedì al Venerdì e valori equivalenti. I giorni feriali non corrispondono ai giorni lavorativi, poiché una festività può capitare anche in un giorno feriale.

Esempi

```
try
   {
   DtpDate toDay = new DtpDate();
   DtpDate tomorrow = toDay.addDays(1);
   int days = today.caldWeekdays(tomorrow);
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Consultare inoltre

`calcDays()`

---

get12MonthNames()

Restituisce la rappresentazione con nome completo corrente dei dodici mesi per questa data.

Sintassi

```
public String[ ] get12MonthNames()
```

Valori restituiti

Un vettore di oggetti Stringa contenenti i nomi effettivi dei dodici mesi.

Esempi

```
DtpDate toDay = new DtpDate();
String[] toDay.get12MonthNames();
```

Consultare inoltre

`set12MonthNames()`, `set12MonthNamesToDefault()`

---

get12ShortMonthNames()

Restituisce la rappresentazione con nome breve corrente dei dodici mesi per questa data.

Sintassi

```
public String[ ] get12ShortMonthNames()
```

Valori restituiti

Un vettore di oggetti Stringa contenenti i nomi brevi effettivi dei dodici mesi.

Esempi

```
DtpDate toDay = new DtpDate();
String[] toDay.get12ShortMonthNames();
```

Consultare inoltre

`set12ShortMonthNames()`, `set12ShortMonthNamesToDefault()`

get7DayNames()

Restituisce i nomi correnti dei sette giorni della settimana per questa data.

Sintassi

```
public String[ ] get7DayNames()
```

Valori restituiti

Un vettore di oggetti Stringa contenenti i nomi effettivi dei sette giorni della settimana.

Esempi

```
DtpDate toDay = new DtpDate();
String[] toDay.get7DayNames();
```

Consultare inoltre

`set7DayNames()`, `set7DayNamesToDefault()`

getCWDate()

Formatta nuovamente questa data nel formato data generico IBM.

Sintassi

```
public String getCWDate()
```

Valori restituiti

Una stringa che rappresenta la data nel formato oggetto business generico di IBM WebSphere Business Integration Server Express. Il formato è YMD hms. Alcuni esempi sono:

- 19990615 150701
- 19990831 114122

Note

Il formato data generico IBM ha la seguente forma:

YYYYMMDD HHMMSS

Esempi

```
DtpDate toDay = new DtpDate();
String genericDate = toDay.getCWDate();
```

getDayOfMonth()

Restituisce il giorno del mese per questa data.

Sintassi

```
public String getDayOfMonth()
```

Valori restituiti

La stringa che rappresenta il giorno del mese, quale 01, 20, 30, e così via.

Esempi

```
DtpDate toDay = new DtpDate();
String dayOfMonth = toDay.getDayOfMonth();
```

Consultare inoltre

`getIntDay()`

---

getDayOfWeek()

Restituisce il giorno della settimana per questa data.

Sintassi

```
public String getDayOfWeek()
```

Valori restituiti

Un stringa che indica il giorno della settimana, quale Lunedì, Martedì e così via.

Esempi

```
DtpDate toDay = new DtpDate();
String dayOfWeek = toDay.getDayOfWeek();
```

Consultare inoltre

`getIntDayOfWeek()`

---

getHours()

Restituisce il valore delle ore per questa data.

Sintassi

```
public String getHours()
```

Valori restituiti

Una stringa che rappresenta il valore delle ore che sarà tra 00 e 23.

Esempi

```
DtpDate toDay = new DtpDate();
String hours = toDay.getHours();
```

---

getIntDay()

Restituisce il giorno del mese in questa data come numero intero.

Sintassi

```
public int getIntDay()
```

Valori restituiti

Un valore `intero` che rappresenta il giorno del mese.

Esempi

```
DtpDate toDay = new DtpDate();
int day = toDay.getIntDay();
```

Consultare inoltre

`getDayOfMonth()`

---

getIntDayOfWeek()

Restituisce il giorno della settimana in questa data come numero intero.

Sintassi

```
public int getIntDayOfWeek()
```

Valori restituiti

Un valore `intero` che rappresenta il giorno della settimana. I valori possibili sono 0 (Lunedì), 1 (Martedì), 2 (Mercoledì), 3 (Giovedì), 4 (Venerdì), 5 (Sabato), o 6 (Domenica).

Esempi

```
DtpDate toDay = new DtpDate();
int dayOfWeek = toDay.getIntDayOfWeek();
```

Consultare inoltre

`getDayOfWeek()`

---

getIntMilliSeconds()

Restituisce il valore millisecondi di questa data.

Sintassi

```
public int getIntMilliSeconds()
```

Valori restituiti

Un valore `intero` che rappresenta i millisecondi, l'intervallo è da 0 a 999.

Esempi

```
DtpDate toDay = new DtpDate();
int millisecs = toDay.getIntMilliSeconds();
```

---

getIntMinutes()

Restituisce il valore dei minuti di questa data come un numero intero.

Sintassi

```
public int getIntMinutes()
```

Valori restituiti

Un valore `intero` che rappresenta i minuti. L'intervallo va da 0-59.

Esempi

```
DtpDate toDay = new DtpDate();
int mins = toDay.getIntMinutes();
```

Consultare inoltre

`getMinutes()`

---

getIntMonth()

Restituisce il mese di questa data come un numero intero.

Sintassi

```
public int getIntMonth()
```

Valori restituiti

Un valore `intero` che rappresenta il mese. L'intervallo è 1 (Gennaio) - 12 (Dicembre).

Esempi

```
DtpDate toDay = new DtpDate();
int month = toDay.getIntMonth();
```

Consultare inoltre

`getMonth()`, `getNumericMonth()`

---

getIntSeconds()

Restituisce i secondi di questa data come un numero intero.

Sintassi

```
public int getIntSeconds()
```

Valori restituiti

Un valore `intero` che rappresenta i secondi. L'intervallo va da 0-59.

Esempi

```
DtpDate toDay = new DtpDate();
int secs = toDay.getIntSeconds();
```

Consultare inoltre

`getSeconds()`, `getMSSince1970()`

---

getIntYear()

Restituisce l'anno di questa data come un numero intero.

Sintassi

```
public int getIntYear()
```

Valori restituiti

Un valore `intero` che rappresenta l'anno.

Esempi

```
DtpDate toDay = new DtpDate();
int year = toDay.getIntYear();
```

Consultare inoltre

`getYear()`

---

getMSSince1970()

Restituisce il numero di millisecondi tra il 1 Gennaio 1970 00:00:00 e questa data.

Sintassi

```
public long getMSSince1970()
```

Valori restituiti

Un numero intero. Sarà negativo se la data è precedente al 1 Gennaio 1970 00:00:00.

Eccezioni

`DtpDateException`

Esempi

```
try
   {
   DtpDate toDay = new DtpDate();
   long ms = toDay.getMSSince1970();
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Consultare inoltre

`getSeconds()`

---

getMaxDate()

Da un elenco di oggetti business, restituisce la data più recente come oggetto `DtpDate`.

Sintassi

```
public static DtpDate getMaxDate(BusObjArray boList, String attr,
      String dateFormat)
```

Parametri

boList	Un elenco di oggetti business.
attr	L'attributo dell'oggetto business da usare quando si fa il confronto. L'attributo deve essere di tipo `Data`.
dateFormat	Il formato della data. Fare riferimento a `DtpDate()` per maggiori dettagli. Se è nullo, si assume che la data è il numero di millisecondi dal 1970.

Valori restituiti

Un oggetto `DtpDate` che contiene la data massima.

Eccezioni

`DtpIncompatibleBOTypeException` - Quando gli oggetti business nell'elenco non sono dello stesso tipo.

`DtpUnknownAttributeException` - Quando l'attributo specificato non è valido nell'oggetto business immesso.

`DtpUnsupportedAttributeTypeException` - Quando il tipo dell'attributo specificato non è tra quelli supportati elencati in precedenza.

Tutte queste eccezioni sono classi secondarie di `RunTimeEntityException`.

Note

Il metodo `getMaxDate()` scorre l'elenco degli oggetti business alla ricerca di quello con la data più recente, e la restituisce sotto forma di oggetto `DtpDate`.

Suggerimenti: Questo è un metodo statico.

Nella valutazione della data il 1 Gen 2004 000000 è successivo al 1 Gen 2002 000000, che è a sua volta successivo al 1 Gen 1999 000000

Si assume che le informazioni della data vengano archiviate nel nome dell'attributo passato nel metodo. Se un oggetto ha un'informazione della data nulla, viene ignorato. Se tutti gli oggetti hanno l'informazione della data nulla, viene restituito null.

Esempi

```
try
   {
   DtpDate maxDate = DtpDate.getMaxDate(bos, "Start Date",
      "D/M/Y h:m:s");
   }
catch ( RunTimeEntityException err )
   {
   System.out.println(err.getMessage());
   }
```

Consultare inoltre

`getMinDate()`, `getMaxDateBO()`

getMaxDateBO()

Da un elenco di oggetti business, restituisce quelli con la data più recente.

Sintassi

```
public static BusObj[] getMaxDateBO(BusObj[] boList, String attr,
      String dateFormat)
public static BusObj[] getMaxDateBO(BusObjArray boList, String attr,
      String dateFormat)
```

Parametri

boList Un elenco di oggetti business. Può essere sia un vettore di `BusObj` oppure un istanza di `BusObjArray`. L'oggetto business deve essere dello stesso tipo.

attr L'attributo dell'oggetto business con cui confrontare. L'attributo deve essere di tipo `Data`.

dateFormat Il formato della data. Fare riferimento a `DtpDate()` per maggiori dettagli. Se è nullo, si assume che la data è il numero di millisecondi dal 1970.

Valori restituiti

Un vettore di oggetti business aventi la data più recente.

Eccezioni

Queste tre eccezioni sono classi secondarie di `RunTimeEntityException`.

`DtpIncompatibleBOTypeException` - Quando gli oggetti business nell'elenco non sono dello stesso tipo.

`DtpUnknownAttributeException` - Quando l'attributo specificato non è valido nell'oggetto business immesso.

`DtpUnsupportedAttributeTypeException` - Quando il tipo dell'attributo specificato non è tra quelli supportati elencati in precedenza.

`DtpDateException` - Quando il formato della data non è valido.

Note

Il metodo `getMaxDateBO()` scorre l'elenco degli oggetti business alla ricerca di quello con la data più recente, e lo restituisce. Se più oggetti business hanno la stessa data massima, vengono restituiti tutti gli oggetti con questa data.

Suggerimenti: Questo è un metodo statico.

Nella valutazione della data meno recente il 1 Gen 2004 000000 è successivo al 1 Gen 2002 000000, che è a sua volta successivo al 1 Gen 1999 000000

Si assume che le informazioni della data vengano archiviate nel nome dell'attributo passato nel metodo. Se un oggetto ha un'informazione della data nulla, viene ignorato. Se tutti gli oggetti hanno l'informazione della data nulla, viene restituito null.

Esempi

```
try
   {
   BusObj[] max = DtpDate.getMaxDateBO(bos, "Start Date",
      "D/M/Y h:m:s");
   }
catch ( RunTimeEntityException err )
   {
   System.out.println(err.getMessage());
   }
```

Consultare inoltre

`getMaxDate()`, `getMinDateBO()`

---

getMinDate()

Da un elenco di oggetti business, restituisce la data meno recente come oggetto `DtpDate`.

Sintassi

```
public static DtpDate getMinDate(BusObjArray boList, String attr,
      String dateFormat)
```

Parametri

boList	Un elenco di oggetti business.
attr	L'attributo dell'oggetto business da usare quando si fa il confronto. L'attributo deve essere di tipo `Data`.
dateFormat	Il formato della data. Fare riferimento a `DtpDate()` per maggiori dettagli. Se è nullo, si assume che la data è il numero di millisecondi dal 1970.

Valori restituiti

Un oggetto `DtpDate` che contiene la data meno recente.

Eccezioni

`DtpIncompatibleBOTypeException` - Quando gli oggetti business nell'elenco non sono dello stesso tipo.

`DtpUnknownAttributeException` - Quando l'attributo specificato non è valido nell'oggetto business immesso.

`DtpUnsupportedAttributeTypeException` - Quando il tipo dell'attributo specificato non è tra quelli supportati elencati in precedenza.

Tutte queste eccezioni sono classi secondarie di `RunTimeEntityException`.

Note

Il metodo `getMinDate()` scorre l'elenco degli oggetti business alla ricerca di quello con la data meno recente, e la restituisce sotto forma di oggetto `DtpDate`.

Suggerimenti: Questo è un metodo statico.

Nella valutazione delle data il 1 Gen 1999 000000 è precedente al 1 Gen 2002 000000, che è a sua volta precedente al 1 Gen 2004 000000

Si assume che le informazioni della data vengano archiviate nel nome dell'attributo passato nel metodo. Se un oggetto ha un'informazione della data nulla, viene ignorato. Se tutti gli oggetti hanno l'informazione della data nulla, viene restituito null.

Esempi

```
try
   {
   DtpDate minDate = DtpDate.getMinDate(bos, "Start Date",
      "D/M/Y h:m:s");
   }
catch ( RunTimeEntityException err )
   {
   System.out.println(err.getMessage());
   }
```

Consultare inoltre

`getMaxDate()`, `getMinDateBO()`

---

getMinDateBO()

Da un elenco di oggetti business, restituisce quelli con la data meno recente.

Sintassi

```
public static BusObj[] getMinDateBO(BusObj[] boList, String attr,
      String dateFormat)
public static BusObj[] getMinDateBO(BusObjArray boList, String attr,
      String dateFormat)
```

Parametri

boList	Un elenco di oggetti business.
attr	L'attributo dell'oggetto business da usare quando si fa il confronto. L'attributo deve essere di tipo `Data`.
dateFormat	Il formato della data. Fare riferimento a `DtpDate()` per maggiori dettagli. Se è nullo, si assume che la data è il numero di millisecondi dal 1970.

Valori restituiti

Un vettore di oggetti business aventi la data.

Eccezioni

`DtpIncompatibleBOTypeException` - Quando gli oggetti business nell'elenco non sono dello stesso tipo.

`DtpUnknownAttributeException` - Quando l'attributo specificato non è valido nell'oggetto business immesso.

`DtpUnsupportedAttributeTypeException` - Quando il tipo dell'attributo specificato non è tra quelli supportati elencati in precedenza.

`DtpDateException` - Quando il formato della data non è valido.

Tutte queste eccezioni sono classi secondarie di `RunTimeEntityException`.

Note

Il metodo `getMinDateBO()` scorre l'elenco degli oggetti business alla ricerca di quello con la data meno recente, e la restituisce sotto forma di oggetto `DtpDate`.

Suggerimenti: Questo è un metodo statico.

Nella valutazione della data meno recente il 1 Gen 2004 000000 è successivo al 1 Gen 2002 000000, che è a sua volta successivo al 1 Gen 1999 000000

Si assume che le informazioni della data vengano archiviate nel nome dell'attributo passato nel metodo. Se un oggetto ha un'informazione della data nulla, viene ignorato. Se tutti gli oggetti hanno l'informazione della data nulla, viene restituito null.

Esempi

```
try
   {
   BusObj[] min = DtpDate.getMinDateBO(bos, "Start Date",
      "D/M/Y h:m:s");
   }
catch ( RunTimeEntityException err )
   {
   System.out.println(err.getMessage());
   }
```

Consultare inoltre

`getMinDate()`, `getMaxDateBO()`

---

getMinutes()

Restituisce il valore dei minuti di questa data.

Sintassi

```
public String getMinutes()
```

Valori restituiti

La stringa che rappresenta i minuti. Il valore restituito è compreso tra 1 e 59.

Consultare inoltre

`getIntMinutes()`

---

getMonth()

Restituisce la rappresentazione con nome completo del mese in questa data.

Sintassi

```
public String getMonth()
```

Valori restituiti

Il nome del mese, come Gennaio, Febbraio e così via.

Consultare inoltre

`getIntMonth()`, `getNumericMonth()`, `getShortMonth()`

getNumericMonth()

Restituisce il valore del mese di questa data in formato numerico.

Sintassi

```
public String getNumericMonth()
```

Valori restituiti

La stringa in formato numerico del mese, quale 01, 02, e così via.

Esempi

```
DtpDate toDay = new DtpDate();
System.out.println("Today is "
   + toDay.getDayOfMonth() + "/"
   + toDay.getNumericMonth() + "/"
   + toDay.getYear() + " "
   + toDay.getHours() + ":"
   + toDay.getMinutes() + ":"
   + toDay.getSeconds());
```

Consultare inoltre

`getIntMonth()`, `getMonth()`

getSeconds()

Restituisce i secondi di questa data come stringa.

Sintassi

```
public String getSeconds()
```

Valori restituiti

La stringa che rappresenta i secondi. Il valore restituito è compreso tra 1 e 59.

Esempi

```
DtpDate toDay = new DtpDate();
System.out.println("Today is "
   + toDay.getDayOfMonth() + "/"
   + toDay.getNumericMonth() + "/"
   + toDay.getYear() + " "
   + toDay.getHours() + ":"
   + toDay.getMinutes() + ":"
   + toDay.getSeconds());
```

Consultare inoltre

`getIntSeconds()`

getShortMonth()

Restituisce la rappresentazione con nome breve del mese da questa data.

Sintassi

```
public String getShortMonth()
```

Valori restituiti

Il nome del mese è nel formato breve, quale Gen, Feb e così via.

Esempi

```
DtpDate toDay = new DtpDate();
DtpDate lastYear= toDay.addYears(-1);
System.out.println("Next month is "
   + lastYear.getShortMonth() + " "
   + lastYear.getDayOfMonth() + ", "
   + lastYear.getYear() + " "
   + lastYear.getHours() + ":"
   + lastYear.getMinutes() + ":"
   + lastYear.getSeconds());
```

Consultare inoltre

`getMonth()`, `set12ShortMonthNames()`, `set12ShortMonthNamesToDefault()`

getYear()

Restituisce l'anno di questa data.

Sintassi

```
public String getYear()
```

Valori restituiti

La stringa che rappresenta l'anno. Il valore anno comprende il secolo. Ad esempio 1998 e 2004.

Esempi

```
DtpDate toDay = new DtpDate();
DtpDate lastYear= toDay.addYears(-1);
System.out.println("Next month is "
   + lastYear.getDayOfMonth() + "/"
   + lastYear.getNumericMonth() + "/"
   + lastYear.getYear() + " "
   + lastYear.getHours() + ":"
   + lastYear.getMinutes() + ":"
   + lastYear.getSeconds());
```

Consultare inoltre

`getIntYear()`

set12MonthNames()

Modifica la rappresentazione con nome completo dei nomi dei dodici mesi per questa data.

Sintassi

```
public void set12MonthNames(String[] monthNames,
      boolean resetShortMonth)
```

Parametri

monthNames Un vettore di stringhe contenente i nomi dei dodici mesi. Il primo elemento è il primo mese dell'anno, l'ultimo elemento è l'ultimo mese dell'anno.

resetShortMonthNames
Per valore predefinito, i nomi brevi dei mesi sono i primi tre caratteri del nome completo del mese. Se questo indicatore è impostato su `true`, i nomi brevi dei mesi si modificheranno sulla base dei nuovi nomi completi. Se è invece impostato su `false`, questo metodo non modificherà i nomi brevi dei mesi.

Valori restituiti

Nessuno.

Eccezioni

`DtpDateException` - Quando i nomi dei mesi immessi non sono 12.

Consultare inoltre

`get12MonthNames()`, `set12MonthNamesToDefault()`

---

set12MonthNamesToDefault()

Ripristina la rappresentazione con nome completo dei dodici mesi ai valori predefiniti per questa data.

Sintassi

```
public void set12MonthNamesToDefault()
```

Valori restituiti

Nessuno.

Note

I nomi predefiniti sono Gennaio, Febbraio, Marzo e così via.

Consultare inoltre

`get12MonthNames()`, `set12MonthNames()`

---

set12ShortMonthNames()

Modifica la rappresentazione con nome breve dei nomi dei dodici mesi per questa data.

Sintassi

```
public void set12ShortMonthNames(String[] shortMonths)
```

Parametri

shortMonths Un elenco di oggetti business.

Valori restituiti

Nessuno.

Eccezioni

`DtpDateException` - Quando i nomi dei mesi immessi non sono 12.

Consultare inoltre

`get12ShortMonthNames()`, `set12ShortMonthNamesToDefault()`

---

set12ShortMonthNamesToDefault()

Ripristina la rappresentazione con nome breve dei dodici mesi ai valori predefiniti per questa data.

Sintassi

```
public void set12ShortMonthNamesToDefault()
```

Valori restituiti

Nessuno

Note

I nomi brevi sono Gen, Feb, Mar e così via.

Consultare inoltre

`get12ShortMonthNames()`, `set12ShortMonthNames()`

---

set7DayNames()

Modifica i nomi dei sette giorni della settimana per questa data.

Sintassi

```
public void set7DayNames(String[] dayNames)
```

Parametri

dayNames Un vettore di stringhe contenente i sette giorni della settimana. Il primo elemento deve essere l'equivalente di Lunedì.

Valori restituiti

Nessuno.

Eccezioni

`DtpDateException` - Quando non sono stati specificati esattamente sette giorni.

Consultare inoltre

`get7DayNames()`, `set7DayNamesToDefault()`

set7DayNamesToDefault()

Ripristina i nomi dei sette giorni della settimana ai valori predefiniti per questa data.

Sintassi

```
public void set7DayNamesToDefault()
```

Valori restituiti

Nessuno.

Note

I nomi predefiniti sono Lunedì, Martedì, Mercoledì e così via.

Consultare inoltre

`get7DayNames()`, `set7DayNames()`

toString()

Restituisce la data in un formato specificato o predefinito.

Sintassi

```
public String toString()
public String toString(String format)
public String toString(String format boolean twelveHr)
```

Parametri

format Il formato della data. Fare riferimento a `DtpDate()` per maggiori dettagli.

twelveHr Un valore booleano che, se impostato su true, specifiche che il metodo restituisce l'ora nel formato a 12 ore piuttosto che a 24.

Valori restituiti

Un stringa contenente le informazioni della data, come:

```
19990930 053029 PM
```

A prescindere dal formato della posizione del mese, la stringa di output è sempre un numero intero di 2 caratteri (01 per Gennaio, 12 per Dicembre e così via).

Eccezioni

`DtpDateException` - Quando il formato della data non è valido.

Esempi

```
try
   {
   DtpDate toDay = new DtpDate();
   String date = toDay.toString("Y/M/D h:m:s");
   }
catch ( DtpDateException date_e )
   {
   System.out.println(date_e.getMessage());
   }
```

Capitolo 18. Classe DtpMapService

Una mappa secondaria è una mappa che viene chiamata dall'interno di un'altra mappa. La classe `DtpMapService` fornisce un metodo per l'esecuzione di mappe secondarie. La Tabella 136 riepiloga il metodo nella classe `DtpMapService`.

Tabella 136. Sommario metodi DtpMapService

Metodo	Descrizione	Pagina
`runMap()`	Esegue la mappa specificata.	443

runMap()

Esegue la mappa specificata.

Sintassi

```
BusObj[] runMap(String mapName, String mapType,
                BusObj[] srcBOs, cwExecCtx)
```

Parametri

mapName	Il nome della mappa da eseguire.
mapType	Il tipo della mappa da eseguire. Utilizza *solo* la seguente costante, definita nella classe `DtpMapService`: `CWMAPTYPE`–una mappa IBM WebSphere Business Integration Server Express
srcBOs	Un vettore di oggetti business che rappresentano gli oggetti business di origine per *mapName*.
`cwExecCtx`	Una variabile che contiene il contesto di esecuzione per la mappa corrente. Questa variabile è definita nel codice che Map Designer Express genera per ogni mappa.

Valori restituiti

Un vettore di oggetti business che rappresentano gli oggetti business di destinazione per *mapName*.

Eccezioni

`MapFailureException` – Se si verifica un errore mentre si cerca di eseguire *mapName*.

`MapNotFoundException` – Se non è stata trovata *mapName* nel repository.

`CxMissingIDException` – Consultare `maintainSimpleIdentityRelationship()`.

Note

Utilizzare il metodo `runMap()` per chiamare una mappa secondaria dall'interno di un'altra mappa. Per ulteriori informazioni sulla chiamata di mappe secondarie, consultare "Trasformazione con una mappa secondaria" a pagina 47.

Esempi

Il codice seguente chiama una mappa secondaria per mappare un oggetto business `Address` specifico dell'applicazione in un oggetto business `Address` generico:

```
// Create the BusObj Array
BusObj[] rSrcBOs = new BusObj[1];
rSrcBOs[0] = MyCustomerObj.MyAddressObj[0];

// Make the call to the map service
OutObjName = DtpMapService.runMap(MyAppAddressToGenAddress,
   DtpMapService.CWMAPTYPE,rSrcBOs,cwExecCtx);
```

Consultare inoltre

"Trasformazione con una mappa secondaria" a pagina 47

Capitolo 19. Classe DtpSplitString

La classe DtpSplitString permette di dividere o analizzare una stringa in token e richiamare i risultati. Questa classe è utile per la manipolazione di stringhe formattate come chiavi miste, date o numeri di telefono.

La classe DtpSplitString è simile alla classe StringTokenizer presente nel pacchetto java.util. Quando si lavora con mappe IBM WebSphere Business Integration Server Express, DtpSplitString ha i seguenti vantaggio rispetto a StringTokenizer:

- I token in un oggetto DtpSplitString sono indicizzati. Ciò rende facile l'estrazione di token specifici a cui si è interessati. Ad esempio, se si analizza un numero telefonico (quale 650-555-1111) in tre token utilizzando il trattino (-) come delimitatore, si può estrarre il prefisso referenziando l'elemento 0 e compilando ciò che rimane del numero telefonico concatenando elemento 1 ed elemento 2.
- Un oggetto DtpSplitString permette uno scorrimento bidirezionale dei token. Mentre si scorrono gli elementi utilizzando nextElement() e prevElement() tutti gli elementi rimangono a disposizione.

La Tabella 137 riepiloga i metodi della classe DtpSplitString.

Tabella 137. Sommario metodi DtpSplitString

Metodo	Descrizione	Pagina
DtpSplitString()	Costruisce una nuova istanza di DtpSplitString ed analizza una stringa in token.	445
elementAt()	Restituisce un elemento nell'oggetto DtpSplitString alla posizione specificata.	446
firstElement()	Restituisce un elemento nell'oggetto DtpSplitString alla posizione zero.	447
getElementCount()	Restituisce un numero intero contenente il numero totale di elementi.	447
getEnumeration()	Restituisce un'enumerazione di oggetti stringa dove ogni stringa è uno dei token analizzati.	448
lastElement()	Restituisce l'ultimo elemento nell'oggetto DtpSplitString.	448
nextElement()	Restituisce l'elemento successivo nell'oggetto DtpSplitString.	449
prevElement()	Restituisce l'elemento precedente nell'oggetto DtpSplitString.	449
reset()	Ripristina il numero della posizione corrente nell'oggetto DtpSplitString a zero.	450

DtpSplitString()

Costruisce una nuova istanza di DtpSplitString ed analizza una stringa in token.

Sintassi

```
DtpSplitString(String str, String delimiters)
```

Parametri

str La stringa da analizzare.

delimiters Una stringa contenente i delimitatori utilizzati in *str*. Ci potrà essere più di un delimitatore, ma ognuno di questi avrà la lunghezza di un carattere.

Note

`DtpSplitString()` analizza *str* in token, chiamati elementi, sulla base degli delimitatori specificati. Dopo la chiamata di `DtpSplitString()`, è possibile chiamare uno qualsiasi dei metodi della classe `DtpSplitString` per selezionare e richiamare elementi specifici.

Esempi

```
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");
```

---

elementAt()

Restituisce un elemento nell'oggetto `DtpSplitString` alla posizione specificata.

Sintassi

```
String elementAt(int nth)
```

Parametri

nth La posizione dell'elemento da estrarre dall'oggetto `DtpSplitString`. La posizione del primo elemento è zero.

Valori restituiti

Restituisce una `Stringa` contenente l'elemento alla posizione *nth*.

Eccezioni

`DtpNoElementAtPositionException` – Se si specifica una posizione non valida per *nth*.

Note

Gli elementi sono enumerati dal primo all'ultimo partendo da zero. Ad esempio, se i delimitatori sono virgole e spazi, l'elemento alla posizione due nella stringa, `"This,is a test"`è `"a"`.

Il metodo `elementAt()` restituisce l'elemento alla posizione specificata ma non cambia la posizione dell'elemento corrente.

Esempi

```
// Create a DtpSplitString object
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");

//This call returns "a"
public String MyString.elementAt(2);
```

Consultare inoltre

`getElementCount()`

firstElement()

Restituisce un elemento nell'oggetto `DtpSplitString` alla posizione zero.

Sintassi

```
String firstElement()
```

Valori restituiti

Restituisce una stringa contenente l'elemento alla posizione zero.

Eccezioni

`DtpNoElementAtPositionException` – Se non ci sono elementi.

Note

Gli elementi nell'oggetto `DtpSplitString`sono enumerati dal primo all'ultimo partendo da zero. Quindi, il primo elemento è alla posizione zero.

Il metodo `firstElement()` restituisce l'elemento alla posizione zero ma *non* cambia la posizione dell'elemento corrente.

Esempi

```
// Create a DtpSplitString object
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");

// This call returns the first element containing "This"
String anElement = MyString.firstElement();
```

Consultare inoltre

`lastElement()`

getElementCount()

Restituisce il numero totale di elementi nell'oggetto `DtpSplitString`.

Sintassi

```
int getElementCount()
```

Valori restituiti

Restituisce un numero intero contenente il numero totale di elementi.

Note

Gli elementi sono enumerati dal primo all'ultimo partendo da zero. Se `getElementCount()` restituisce 6, l'elemento con il numero più alto è il 5.

Esempi

```
// Create a DtpSplitString object
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");

// This call returns the integer 4
String numElements = MyString.getElementCount();
```

Consultare inoltre

`firstElement()`, `lastElement()`

getEnumeration()

Restituisce un'enumerazione di oggetti stringa dove ogni stringa è uno dei token analizzati.

Sintassi

```
Enumeration getEnumeration()
```

Valori restituiti

Restituisce un oggetto Enumerazione.

Note

Il metodo `getEnumeration()` fornisce un'alternativa per l'elaborazione dei token analizzati in un oggetto `DtpSplitString`. Per ulteriori informazioni sulle operazioni con oggetti Enumerazione, consultare il pacchetto `Java.Util`.

lastElement()

Restituisce l'ultimo elemento in un oggetto `DtpSplitString`.

Sintassi

```
String lastElement()
```

Valori restituiti

Restituisce una stringa contenente l'ultimo elemento.

Eccezioni

`DtpNoElementAtPositionException` – Se non ci sono elementi.

Note

Gli elementi sono enumerati dal primo all'ultimo partendo da zero. L'ultimo elemento è quello con il numero più alto. Il numero di posizione dell'ultimo elemento corrisponde a `getElementCount()-1`.

Il metodo `lastElement()` restituisce l'ultimo elemento ma non cambia la posizione dell'elemento corrente.

Esempi

```
// Create a DtpSplitString object
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");

// This call returns the last element, containing "test"
String anElement = MyString.lastElement();
```

Consultare inoltre

`firstElement()`, `getElementCount()`

nextElement()

Restituisce l'elemento successivo in un oggetto DtpSplitString.

Sintassi

```
String nextElement()
```

Valori restituiti

Restituisce una stringa contenente l'elemento successivo.

Eccezioni

DtpNoElementAtPositionException – Se non ci sono elementi.

Note

La prima volta che si chiama nextElement(), questo restituisce l'elemento alla posizione zero. Nelle successive chiamate metodo, nextElement() restituisce l'elemento alla posizione uno, due, tre, e così via. Si può utilizzare nextElement(), in combinazione con prevElement(), per scorrere gli elementi (token) in un oggetto DtpSplitString.

Esempi

```
// Create a DtpSplitString object
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");

// This call returns element 0 containing "This"
String firstElement = MyString.nextElement()

// This call returns element 1 containing "is"
String secondElement = MyString.nextElement()
```

Consultare inoltre

prevElement(), reset()

prevElement()

Restituisce l'elemento precedente in un oggetto DtpSplitString.

Sintassi

```
String prevElement()
```

Valori restituiti

Restituisce una stringa contenente l'elemento precedente.

Eccezioni

DtpNoElementAtPositionException – Se non c'è alcun elemento precedente.

Note

Si può utilizzare prevElement(), in combinazione con nextElement(), per scorrere gli elementi (token) in un oggetto DtpSplitString. La prima volta che si chiama nextElement(), la posizione dell'elemento è zero. Le chiamate successive di

nextElement()incrementano la posizione di uno. Il metodo prevElement() restituisce l'elemento precedente e diminuisce la posizione dell'elemento di uno.

Esempi

```
// Create a DtpSplitString object
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");

// This call returns element 0 containing "This"
String firstElement = MyString.nextElement()

// This call returns element 1 containing "is"
String secondElement = MyString.nextElement()

// This call returns element 0 containing "This"
String anotherElement = MyString.prevElement()
```

Consultare inoltre

nextElement()

---

reset()

Ripristina il numero della posizione corrente nell'oggetto DtpSplitString a zero.

Sintassi

```
void reset()
```

Valori restituiti

Nessuno.

Note

La posizione dell'elemento predefinita è zero. Ogni volta che si effettua una chiamata nextElement(), la posizione dell'elemento incrementa di uno. Il metodo prevElement() restituisce l'elemento precedente e diminuisce la posizione dell'elemento di uno. Si può usare reset()per ripristinare la posizione corrente a zero.

Esempi

```
// Create a DtpSplitString object
DtpSplitString MyString = new DtpSplitString("This,is a test",", ");

// This call returns element 0 containing "This"
String firstElement = MyString.nextElement()

// This call returns element 1 containing "is"
String secondElement = MyString.nextElement()

// Reset the position to zero
MyString.reset()

// This call returns element 0 containing "This"
String firstElement = MyString.nextElement()
```

Consultare inoltre

nextElement(), prevElement()

Capitolo 20. Classe DtpUtils

La classe `DtpUtils` esegue diverse operazioni di carattere generale.

La Tabella 138 riepiloga i metodi della classe `DtpUtils`.

Tabella 138. Sommario metodi DtpUtils

Metodo	Descrizione	Pagina
`padLeft()`	Riempie la stringa con il carattere specificato.	451
`padRight()`	Riempie la stringa con il carattere specificato.	451
`stringReplace()`	Sostituisce tutte le ricorrenze di un modello contenuto in una stringa con un altro modello.	451
`truncate()`	Tronca questo numero.	453

padLeft()

Riempie la stringa con il carattere specificato.

Sintassi

```
public static String padLeft(String src, char padWith, int totalLen)
```

Parametri

src La stringa da riempire.

padWith Il carattere utilizzato per il riempimento.

totalLen La nuova dimensione della stringa, un numero positivo. Se il valore è 0, più basso della dimensione della stringa, oppure negativo, viene restituita la stringa originale.

Valori restituiti

Una nuova stringa riempita.

Note

Riempie la stringa con un carattere specificato.

Esempi

La chiamata seguente restituisce 0000012345:

```
padLeft("12345", '0', 10);
```

La chiamata seguente restituisce 123456:

```
padLeft("123456", '0', 5);
```

padRight()

Riempie la stringa con il carattere specificato.

Sintassi

```
public static String padLeft(String src, char padWith, int totalLen)
```

Parametri

src	La stringa da riempire.
padWith	Il carattere utilizzato per il riempimento.
totalLen	La nuova dimensione della stringa, un numero positivo. Se il valore è 0, più basso della dimensione della stringa, oppure negativo, viene restituita la stringa originale.

Valori restituiti

Una nuova stringa riempita.

Note

Riempie la stringa con un carattere specificato.

Esempi

La chiamata seguente restituisce 1234500000:

```
padRight("12345", '0', 10);
```

La chiamata seguente restituisce 123456:

```
padRight("123456", '0', 5);
```

stringReplace()

Sostituisce tutte le ricorrenze di un modello contenuto in una stringa con un altro modello.

Sintassi

```
public static String stringReplace(String src, String oldpattern,
      String newPattern)
```

Parametri

src	La stringa da modificare.
oldPattern	Il carattere utilizzato per il riempimento.
newPattern	Il modello della stringa da utilizzare nella sostituzione.

Valori restituiti

Una nuova stringa con un nuovo modello.

Note

Il metodo sostituisce tutte le ricorrenze del valore specificato in *oldPattern*, con il valore specificato in *newPattern*. Per la sostituzione di un singolo carattere, utilizzare `replace()` nella classe `String` di Java. Se *oldPattern* non viene trovata, viene restituita la stringa originale, non modificata.

Esempi

Il risultato dell'esempio seguente è `youoyou and dad`.

```
stringReplace("momomom and dad", "mom", "you");
```

---

truncate()

Tronca questo numero.

Sintassi

```
public static double truncate(Object aNumber, int precision)
   throws DtpIncompatibleFormatException

public static double truncate(float aNumber, int precision)
public static double truncate(double aNumber, int precision)

public static int truncate(Object aNumber)
   throws DtpIncompatibleFormatException

public static int truncate(float aNumber)
public static int truncate(double aNumber)
```

Parametri

aNumber	Un numero. I tipi validi sono `String`, `float`, e `double`.
precision	Il numero di cifre alla destra dei decimali da rimuovere.

Valori restituiti

Un numero `double` o `int`.

Note

Questo metodo rimuove cifre da questo numero, a partire da destra.

I primi tre moduli dei metodi troncano il numero rimuovendo le cifre alla destra della posizione decimale, partendo da destra. Se il numero in input è intero, non verrà troncato. Il numero di tipo `Oggetto` deve essere `String`, `Double` o `Float`.

Gli ultimi tre moduli dei metodi troncano il numero rimuovendo tutte le cifre alla destra della posizione decimale, restituendo un valore intero.

Esempi

L'esempio che segue restituisce 123.45:

```
truncate("123.4567", 2);
```

L'esempio che segue restituisce 123:

```
truncate(123.456, 4)
```

Capitolo 21. Classe IdentityRelationship

I metodi documentati in questo capitolo operano su oggetti della classe `IdentityRelationship`. Questi oggetti rappresentano istanze delle relazioni d'identità. La classe `IdentityRelationship` fornisce una funzionalità aggiuntiva necessaria quando si accede il database repository. Associa una serie di API esistenti in metodi che forniscono una facilità d'impiego per lo sviluppatore delle mappe.

Il codice sorgente per i metodi nella classe `IdentityRelationship` è fornito e può essere utilizzato come nell'ambiente IBM WebSphere Business Integration Server Express o personalizzato per adattarsi ad altri ambienti.

Tabella 139 elenca i metodi della classe `IdentityRelationship`.

Tabella 139. Riepilogo del metodo IdentityRelationship

Metodo	Descrizione	Pagina
`addMyChildren()`	Aggiunge le istanze secondarie specificate in una relazione principale/secondaria per una relazione identità.	456
`deleteMyChildren()`	Rimuove le istanze secondarie in una relazione principale/secondaria per una relazione identità appartenente alla struttura principale specificata.	457
`foreignKeyLookup()`	Controlla nella tabella delle relazioni esterna basata su una chiave esterna di un oggetto business sorgente, non riuscendo a trovare una istanza di relazione se la chiave esterna non esiste nella tabella delle relazioni esterna.	458
`foreignKeyXref()`	Controlla nella tabella delle relazioni nel database relazioni basato su una chiave esterna di un oggetto business sorgente, aggiungendo una nuova istanza di relazione nella tabella delle relazioni esterna se questa chiave esterna non esiste.	460
`maintainChildVerb()`	Imposta il verbo dell'oggetto business secondario basato sul contesto di esecuzione della mappa e il verbo di un oggetto business principale.	462
`maintainCompositeRelationship()`	Mantiene una relazione identità composta dalla mappa principale.	464
`maintainSimpleIdentityRelationship()`	Mantiene una relazione di identità semplice composta da una mappa principale o secondaria.	467
`updateMyChildren()`	Aggiunge e cancella le istanze secondarie in una relazione principale/secondaria specificata di una Relazione Identità come necessario.	469

Nota: Tutti i metodi nella classe `IdentityRelationship` sono dichiarati come statici. E' possibile chiamare uno qualsiasi dei metodi in questa classe da un'istanza di relazione esistente o facendo riferimento alla classe `IdentityRelationship`: `IdentityRelationship.`*`metodo`*, dove *`metodo`* è il nome di un metodo in Tabella 139.

addMyChildren()

Aggiunge le istanze secondarie specificate in una relazione principale/secondaria per una relazione d'identità.

Sintassi

```
public static void addMyChildren(String parentChildRelDefName,
      String parentParticpntDefName, BusObj parentBusObj,
      String childParticpntDefName, Object childBusObjList,
      CxExecutionContext map_ctx)
```

Parametri

parentChildRelDefName
Il nome della definizione di relazione parent/child.

parentParticpntDefName
Il nome della definizione partecipante che rappresenta l'oggetto business principale nella relazione parent/child.

parentBusObj La variabile che contiene l'oggetto business principale.

childParticpntDefName
Il nome della definizione partecipante che rappresenta l'oggetto business secondario nella relazione parent/child.

childBusObjList
La variabile che contiene l'oggetto business secondario o oggetti da aggiungere alla relazione. Questo parametro può essere un unico oggetto business generico (`BusObj`) o una tabella di oggetti business generici (`BusObjArray`).

map_ctx Il contesto di esecuzione della mappa. Per inoltrare il contesto di esecuzione della mappa, utilizzare la `variabile cwExecCtx`, che Map Designer Express definisce per ogni mappa.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `addMyChildren()` aggiunge le istanze child in *childBusObjList* alle tabelle di relazione della definizione di relazione *parentChildRelDefName*. Questo metodo è utile in una relazione personalizzata che interessa un oggetto business principale con una chiave univoca. Quando ad un oggetto business principale vengono aggiunti nuovi oggetti secondari, utilizzare `addMyChildren()` per confrontare l'immagine seguente (in *parentBusObj*) con l'immagine precedente (informazioni nelle tabelle di relazione) in modo da determinare quali oggetti secondari sono nuovi nell'immagine seguente. Per ogni nuovo oggetto secondario, `addMyChildren()` aggiunge un'istanza secondaria alle tabelle di relazione per i partecipanti principali e secondari (*parentParticpntDefName* e *childParticpntDefName* rispettivamente). Se l'oggetto business principale non è presente nella tabella di relazione, `addMyChildren()` inserisce un'istanza di relazione per questo oggetto principale.

Il metodo addMyChildren() richiede la definizione di una relazione principale/secondaria con Relationship Designer Express. Per informazioni su come creare questo tipo di relazione, consultare “Creazione di una definizione di relazione parent/child” a pagina 299.

Vedere anche

deleteMyChildren(), updateMyChildren()

“Gestione delle istanze child” a pagina 298.

deleteMyChildren()

Rimuove le istanze secondarie specificate in una relazione principale/secondaria per una relazione identità appartenente all’oggetto principale specificato.

Sintassi

```
void deleteMyChildren(String parentChildRelDefName,
      String parentParticpntDefName, BusObj parentBusObj,
      String childParticpntDefName, Object childBusObjList,
      CxExecutionContext map_ctx)

void deleteMyChildren(String parentChildRefDefName,
      String parentParticpntDefName, BusObj parentBusObj,
      String childParticpntDefName, CxExecutionContext map_ctx)
```

Parametri

parentChildRelDefName
: Il nome della definizione di relazione parent/child.

parentParticpntDefName
: Il nome della definizione partecipante che rappresenta l’oggetto business principale nella relazione parent/child.

parentBusObj
: La variabile che contiene l’oggetto business principale.

childParticpntDefName
: Il nome della definizione partecipante che rappresenta l’oggetto business secondario nella relazione parent/child.

childBusObjList
: La variabile che contiene l’oggetto business secondario o oggetti da eliminare dalla relazione. Questo parametro può essere un unico oggetto business generico (BusObj) o una tabella di oggetti business generici (BusObjArray).

map_ctx
: Il contesto di esecuzione della mappa. Per inoltrare il contesto di esecuzione della mappa, utilizzare la variabile cwExecCtx, che Map Designer Express definisce per ogni mappa.

Valori di ritorno

Nessuno.

Eccezioni

RelationshipRuntimeException

Note

Il metodo `deleteMyChildren()` elimina le istanze secondarie da una definizione di relazione *parentChildRelDefName* principale/secondaria. Supporta i seguenti moduli:

- Il primo modulo del metodo rimuove dalle tabelle di relazione per i partecipanti principali e secondari quelle istanze secondarie che corrispondono a ciascuno degli oggetti business in *childBusObjList*. Individua un'istanza secondaria da eliminare mettendo in corrispondenza il valore e nome dell'oggetto secondario, così come il valore e nome dell'oggetto principale.
- Il secondo modulo del metodo rimuove dalle tabelle di relazione per i partecipanti principali e secondari tutte le istanze secondarie dell'oggetto principale *parentBusObj*. Individua l'istanza secondaria da eliminare mettendo in corrispondenza il valore e nome dell'oggetto principale.

Questo metodo è utile in una relazione personalizzata che interessa un oggetto business principale con una chiave univoca. Quando un oggetto business principale presenta degli oggetti secondari rimossi, utilizzare `deleteMyChildren()` per confrontare l'immagine seguente (in *parentBusObj*) con l'immagine precedente (informazioni nelle tabelle di relazione) in modo da determinare quali oggetti secondari sono stati rimossi dall'immagine seguente. Per ogni oggetto secondario, `deleteMyChildren()` rimuove l'istanza secondaria corrispondente dalle tabelle di relazione per i partecipanti principali e secondari (*parentParticpntDefName* e *childParticpntDefName* rispettivamente).

Il metodo `deleteMyChildren()` richiede la definizione di una relazione principale/secondaria con Relationship Designer Express. Per informazioni su come creare questo tipo di relazione, consultare "Creazione di una definizione di relazione parent/child" a pagina 299.

Vedere anche

`addMyChildren()`, `updateMyChildren()`

"Gestione delle istanze child" a pagina 298

foreignKeyLookup()

Controlla nella tabella di relazione esterna basata sulla chiave esterna dell'oggetto business sorgente, non riuscendo a trovare un'istanza di relazione se la chiave esterna non è presente nella tabella di relazione esterna.

Sintassi

```
public static void foreignKeyLookup(String relDefName,
      String appParticpntDefName, BusObj
 appSpecificBusObj,       String appForeignAttr,
       BusObj genericBusObj,
String genForeignAttr, CwExecutionContext map_ctx)
```

Parametri

relDefName Il nome della semplice relazione d'identità che gestisce l'oggetto business esterno.

appParticpntDefName
Il nome della definizione partecipante che rappresenta l'oggetto

business specifico dell'applicazione nella relazione d'identità semplice. Il tipo di partecipante è l'oggetto business esterno specifico dell'applicazione.

appSpecificBusObj
La variabile che contiene l'oggetto business specifico dell'applicazione, contenente il riferimento all'oggetto business esterno.

appForeignAttr
Il nome dell'attributo nell'oggetto business specifico dell'applicazione contenente un valore chiave per l'oggetto business esterno.

genericBusObj La variabile contenente l'oggetto business generico in cui o da cui l'attributo *appSpecificObject* viene mappato.

genForeignAttr
Il nome dell'attributo nell'oggetto business generico contenente il riferimento generico a un oggetto business esterno.

map_ctx Il contesto di esecuzione della mappa. Per inoltrare il contesto di esecuzione della mappa, utilizzare la `variabile cwExecCtx`, che Map Designer Express definisce per ogni mappa.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `foreignKeyLookup()` esegue un controllo della chiave esterna nella tabella di relazione per il partecipante *AppParticpntDefName*; cioè, controlla la tabella di relazione esterna per un'istanza di relazione che corrisponde al valore nella chiave esterna dell'oggetto business *appSpecificBusObj*. Se tale controllo ha esito negativo, il metodo `foreignKeyLookup()` imposta semplicemente su null la chiave esterna nell'oggetto business di destinazione; *non* inserisce una riga nella tabella di relazione esterna (come effettua il metodo `foreignKeyXref()`). Questo metodo può essere utilizzato in entrambe le mappe in entrata e in uscita.

Esempi

Nell'oggetto `Clarify_PartRequest` per `Requisition`, il campo `VendorId` è un controllo chiave esterna (Purchasing non chiama Vendor Wrapper). Non viene qui utilizzato il metodo `foreignKeyXref()` perché non si intende inserire una riga se il controllo ha esito negativo.

```
if (ObjCustomerRole.isNull("RoleId"))
   {
   logError(5003, "OrderAssociatedCustomers.RoleId");
   // throw new MapFailureException("OrderAssociatedCustomers.RoleId
   // is null");
   }

try
   {
   IdentityRelationship.foreignKeyLookup("Customer", "SAPCust",
         ObjSAP_OrderPartners, "PartnerId", ObjCustomerRole,
         "RoleId", cwExecCtx);
```

```
   }

catch (RelationshipRuntimeException re)
   {
   logWarning(re.getMessage());
   }
if (ObjSAP_OrderPartners.get("PartnerId") == null)
   {
   logError(5007, "SAP_OrderPartners.PartnerId",
      "OrderAssociatedCustomers.RoleId", "Customer", "SAPCust",
      strInitiator);
   throw new MapFailureException("ForeignKeyLookup failed");
   }
```

Vedere anche

`foreignKeyXref()`

"Esecuzione di ricerche della chiave esterna" a pagina 307

---

foreignKeyXref()

Controlla nella tabella di relazione del database di relazione basato sulla chiave esterna dell'oggetto business sorgente, aggiungendo una nuova istanza di relazione alla tabella di relazione esterna se la chiave esterna non esiste.

Sintassi

```
public static void foreignKeyXref(String relDefName,
      String appParticpntDefName, String genParticpntDefName,
      BusObj appSpecificBusObj, String appForeignAttr,
      BusObj genericBusObj, String genForeignAttr,
      CxExecutionContext map_ctx)
```

Parametri

relDefName Il nome della relazione d'identità semplice che gestisce l'oggetto business esterno.

appParticpntDefName
Il nome della definizione partecipante per l'oggetto business specifico dell'applicazione nella relazione d'identità semplice. Il tipo di partecipante è l'oggetto business esterno specifico dell'applicazione.

genParticpntDefName
Il nome della definizione partecipante per l'oggetto business generico nella relazione d'identità semplice. Il tipo di partecipante è l'oggetto business generico esterno.

appSpecificBusbj
L'oggetto business specifico dell'applicazione contenente il riferimento all'oggetto esterno.

appForeignAttr
Il nome dell'attributo nell'oggetto business specifico dell'applicazione contenente un valore chiave per l'oggetto business esterno.

genericObject L'oggetto business generico in cui o da cui l'attributo *appSpecificObject* viene mappato.

genForeignAttr
Il nome dell'attributo nell'oggetto business generico contenente il riferimento generico a un oggetto business esterno.

map_ctx Il contesto di esecuzione della mappa. Per inoltrare il contesto di esecuzione della mappa, utilizzare la `variabile cwExecCtx`, che Map Designer Express definisce per ogni mappa.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `foreignKeyXref()` esegue un controllo della chiave esterna nella tabella di relazione per il partecipante *AppParticpntDefName*; cioè, controlla la tabella di relazione esterna per un'istanza di relazione che corrisponde al valore nella chiave esterna dell'oggetto business *appSpecificBusObj*. Se tale controllo ha esito negativo, il metodo `foreignKeyXref()` aggiunge una nuova istanza di relazione per la chiave specifica dell'applicazione alla tabella di relazione esterna; *non* imposta semplicemente su null la chiave esterna nell'oggetto business di destinazione (come effettua il metodo `foreignKeyLookup()`). Questo metodo può essere utilizzato in entrambe le mappe in entrata e in uscita.

Il metodo `foreignKeyXref()` effettua le seguenti operazioni di convalida sugli argomenti inseriti:

- Convalidare il nome della definizione di relazione *relDefName*.
- Convalidare il nome della definizione del partecipante *particpntDefName* per l'oggetto business specifico dell'applicazione.
- Accertarsi che la relazione *relDefName* sia una relazione d'identità. Inoltre, la definizione del partecipante in *relDefName* che rappresenta l'oggetto business generico deve essere definita come `IBM WebSphere Business Integration Server Express`. Per ulteriori informazioni su come specificare queste impostazioni, consultare "Definizione delle relazioni di identità" a pagina 257.

Se si verifica un errore con una qualsiasi convalida, `foreignKeyXref()` invia l'eccezione `RelationshipRuntimeException`.

Una volta convalidati gli argomenti, l'azione avviata da `foreignKeyXref()` dipende dalle seguenti informazioni:

- Il contesto di chiamata nel contesto di esecuzione della mappa, inserito come parte dell'argomento *map_ctx* (`cwExecCtx`)
- L'istruzione nell'oggetto business sorgente
  - L'oggetto business specifico dell'applicazione (*appSpecificBusObj*) per contesti di chiamata `EVENT_DELIVERY` (o `ACCESS_REQUEST`) e `SERVICE_CALL_RESPONSE`
  - L'oggetto business generico (*genericBusObj*) per contesti di chiamata `SERVICE_CALL_REQUEST` e `ACCESS_RESPONSE`

Il metodo `foreignKeyXref()` gestisce tutte le aggiunte base delle istanze di relazione nella tabella di relazione esterna per l'associazione corretta del contesto di chiamata e dell'istruzione. Per maggiori informazioni sulle operazioni eseguite

da foreignKeyXref(), consultare "Utilizzo del blocco di funzione Riferimento incrociato della chiave esterna" a pagina 307. Tabella 110 e Tabella 111 forniscono le procedure per ciascuno dei contesti di chiamata.

Esempi

Nella mappa Clarify_SFAQuote a Order, il campo CustomerId è un controllo chiave esterna (Sales Order Processing Collab chiama Customer Wrapper).

```
if (ObjSAP_OrderLineItem.get("SAP_OrderLineObjectIdentifier[0]")
      != null)
   {
   if (ObjSAP_OrderLineItem.getString(
"SAP_OrderLineObjectIdentifier[0].ObjectQualifier").equals("002"))
      {
      BusObj temp = ObjSAP_OrderLineItem.getBusObj(
         "SAP_OrderLineObjectIdentifier[0]");
      if (temp.isNull("ItemId"))
         {
         logWarning(5003,
   "SAP_OrderLineItem.SAP_OrderLineObjectIdentifier[1].ItemId");
         }
      else
         {
         try
            {
            IdentityRelationship.foreignKeyXref(
               "Item",
               "SAPMbasc",
               "CWItba",
               temp,
               "ItemId",
               ObjOrderLineItem,
               "ItemId",
               cwExecCtx);
            }
         catch (RelationshipRuntimeException re)
            {
            logWarning(re.getMessage());
            }

         if (ObjOrderLineItem.get("ItemId") == null )
            {
            logError(5009, "OrderLineItem.ItemId",
         "SAP_OrderLineItem.SAP_OrderLineObjectIdentifier.ItemId",
               "Item",
               "SAPMbasc",
               strInitiator);

            throw new MapFailureException("ForeignKeyXref() failed");
            }
         }
      }
   }
```

Vedere anche

foreignKeyLookup()

"Esecuzione di ricerche della chiave esterna" a pagina 307

maintainChildVerb()

Imposta l'istruzione dell'oggetto business secondario basata sul contesto di esecuzione della mappa e l'istruzione dell'oggetto business principale.

Sintassi

```
public static void maintainChildVerb (String relDefName,
      String appSpecificParticpntName,
String genericParticpntName,
      BusObj appSpecificObj,
String  appSpecificChildObj,
      BusObj genericObj,
String  genericChildObj,
      CxExecutionContext map_ctx,
boolean to_Retrieve,
boolean is_Composite)
```

Parametri

relDefName Il nome della relazione d'identità che gestisce l'oggetto business secondario.

appSpecificParticpntName
Il nome della definizione di partecipante specifico dell'applicazione.

genericParticpntName
Il nome della definizione di partecipante generico.

appSpecificObj
L'oggetto specifico dell'applicazione che contiene l'oggetto secondario.

appSpecificChildObj
Il nome dell'oggetto business secondario dell'applicazione.

genericObj L'oggetto business generico in cui o da cui l'attributo *appSpecificObject* viene mappato.

genericChildObj
Il nome dell'oggetto business secondario generico.

ctx Il contesto di esecuzione.

to_Retrieve L'indicatore per la logica SERVICE_CALL_RESPONSE. Quando la condizione è vera (true), aggiornare le istruzioni degli oggetti business secondari. Se falsa, non eseguire alcuna operazione.

isComposite L'indicatore che segnala se il partecipante secondario utilizza le chiavi composite. Se la condizione è vera (true), le chiavi vengono utilizzate; se falsa, non vengono utilizzate.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`—Consultare la sezione Note per maggiori informazioni sul momento di invio di questa eccezione

`ClassCastException`

Note

Il metodo `maintainChildVerb()` esegue le seguenti operazioni di convalida su argomenti inseriti:

- Convalidare il nome della definizione di relazione *relDefName*.

- Convalidare il nome delle definizioni partecipante per l'oggetto business specifico dell'applicazione (*appSpecificParticpntName*) e per l'oggetto business generico (*genericParticpntName*).
- Accertarsi che gli oggetti business specifici dell'applicazione (*appSpecificObject*) e generici (*genericObject*) *non* siano nulli.
- Accertarsi che la relazione *relDefName* sia una relazione d'identità. Inoltre, la definizione del partecipante in *relDefName* che rappresenta l'oggetto business generico deve essere definita come IBM WebSphere Business Integration Server Express. Per ulteriori informazioni su come specificare queste impostazioni, consultare "Definizione delle relazioni di identità" a pagina 257.

Se si verifica un errore con una qualsiasi convalida, maintainChildVerb() invia l'eccezione RelationshipRuntimeException.

Una volta convalidati gli argomenti, l'azione avviata da maintainChildVerb() dipende dalle seguenti informazioni:

- Il contesto di chiamata nel contesto di esecuzione della mappa, inserito come parte dell'argomento *map_ctx* (cwExecCtx)
- L'istruzione nell'oggetto business sorgente
  - L'oggetto business specifico dell'applicazione (*appSpecificObj*) per contesti di chiamata EVENT_DELIVERY (o ACCESS_REQUEST) e SERVICE_CALL_RESPONSE
  - L'oggetto business generico (*genericObj*) per il contesto di chiamata SERVICE_CALL_REQUEST

Per maggiori informazioni sulle operazioni eseguite da maintainChildVerb(), consultare "Determinazione dell'impostazione del verbo child" a pagina 304. Da Tabella 105 a Tabella 108 si possono consultare le azioni di ciascuno dei contesti di chiamata.

Questo metodo viene chiamato in fase di trasformazione di un attributo secondario di un oggetto principale. Tale oggetto secondario può partecipare:

- In fase di trasformazione per un attributo chiave di una mappa secondaria che trasforma gli oggetti business secondari se quest'ultimi sono associati utilizzando una chiave univoca

Generalmente viene utilizzato maintainChildVerb() per impostare l'istruzione di un oggetto secondario che partecipa a una relazione d'identità composita (maintainCompositeRelationship()). Tuttavia, è anche possibile richiamare questo attributo per impostare l'istruzione di un oggetto secondario che partecipa a una relazione d'identità semplice (maintainSimpleIdentityRelationship()).

Esempi

Per un esempio riguardante maintainChildVerb(), consultare "Personalizzazione delle regole della mappa per una relazione di identità composita" a pagina 293.

Vedere anche

maintainCompositeRelationship(), maintainSimpleIdentityRelationship()

"Impostazione del verbo del child di origine" a pagina 303

maintainCompositeRelationship()

Memorizza una una relazione identità composta dalla mappa principale.

Sintassi

```
public static void maintainCompositeRelationship(String relDefName,
      String particpntDefName, BusObj appSpecificBusObj,
      Object genericBusObjList, CxExecutionContext map_ctx)
```

Parametri

relDefName Il nome della relazione d'identità composita (come definito in Relationship Designer Express) in cui l'attributo principale partecipa.

particpntDefName
Il nome del partecipante che include la chiave composita. Questo partecipante è sempre specifico dell'applicazione.

appSpecificBusObj
La variabile che contiene l'oggetto business specifico dell'applicazione utilizzato in questa mappa. Si tratta dell'oggetto business principale.

genericBusObjList
La variabile contenente l'oggetto business generico o gli oggetti utilizzati in questa mappa, ogni oggetto business generico è un oggetto business secondario contenuto di un oggetto principale generico. Questo parametro può essere un unico oggetto business generico (`BusObj`) o una tabella di oggetti business generici (`BusObjArray`).

map_ctx Il contesto di esecuzione della mappa. Per inoltrare il contesto di esecuzione della mappa, utilizzare la `variabile cwExecCtx`, che Map Designer Express definisce per ogni mappa.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`

`CxMissingIDException`
Se un partecipante non esiste nelle tabelle di relazione durante l'esecuzione di una mappa con istruzione Retrieve e contesto di chiamata `SERVICE_CALL_REQUEST`. Il connettore invia un messaggio di "richiesta chiamata di servizio non riuscita" alla funzione di collaborazione senza inviare l'oggetto business all'applicazione.

Note

Il metodo `maintainCompositeRelationship()` conserva la tabella di relazione associata al partecipante *particpntDefName* della relazione d'identità composita *relDefName*. Questo metodo conserva una relazione il cui partecipante utilizza le chiavi da molteplici oggetti business a diversi livelli (una chiave composita).

Nota: Il metodo `maintainCompositeRelationship()` *non* può gestire la condizione dove la chiave composita secondaria dipende dai rispettivi elementi principali. Per ulteriori informazioni, consultare "Azioni di Generale/API/Relazione di identità/Conserva relazione composita" a pagina 291.

Questo metodo si ripete in tutti gli oggetti business secondari dell'oggetto business principale *appSpecificObj*, conservando le istanze di relazione nella tabella di relazione del partecipante *partDefName*. Il metodo ottiene gli ID delle istanze di relazione dall'array di oggetti business generici che riceve (*genericObjs*). Per ogni istanza secondaria, `maintainCompositeRelationship()` chiama il metodo `maintainSimpleIdentityRelationship()` per eseguire la gestione relazione-tabella reale. L'operazione eseguita da `maintainSimpleIdentityRelationship()` dipende dalle seguenti informazioni:

- Il contesto di chiamata nel contesto di esecuzione della mappa, inserito come parte dell'argomento *map_ctx* (`cwExecCtx`)
- L'istruzione nell'oggetto business sorgente, che può essere:
  - L'oggetto business specifico dell'applicazione (*appSpecificBusObj*) per contesti di chiamata `EVENT_DELIVERY` (o `ACCESS_REQUEST`) e `SERVICE_CALL_RESPONSE`
  - L'oggetto business generico (un elemento dell'array *genericBusObjList*) per contesti di chiamata `SERVICE_CALL_REQUEST` e `ACCESS_RESPONSE`

Per maggiori informazioni sulle operazioni eseguite da `maintainSimpleIdentityRelationship()`, consultare "Accesso alle tabelle di relazione di identità" a pagina 278. Da Tabella 95 a Tabella 99 si possono consultare le azioni di ciascuno dei contesti di chiamata.

Utilizzare `maintainCompositeRelationship()` con i metodi `maintainChildVerb()` e `updateMyChildren()` per conservare una relazione composita. Per ulteriori informazioni, consultare "Personalizzazione delle regole della mappa per una relazione di identità composita" a pagina 293.

Esempi

```
// Di seguito è riportato un esempio di un frammento codice in una mappa principale. Memorizza
// la tabella di relazione per tutte le istanze di un tipo di oggetto secondario per
// quest'oggetto principale specifico dell'applicazione.

BusObjArray secondLevel2 =
      (BusObjArray)ObjFirstLevelBusObj2.get("MultiCardChild");

IdentityRelationship.maintainCompositeRelationship(
   "CmposRel",
   "AppSpPrt",
   ObjFirstLevelBusObj2,
   secondLevel2,
   cwExecCtx);

IdentityRelationship.updateMyChildren(
   "PCRel",
   "Parent",
   ObjFirstLevelBusObj2,
   "Child",
   "MultiCardChild",
   "CmposRel",
   "AppSpPrt",
   cwExecCtx);
```

Per maggiori esempi riguardanti `maintainCompositeRelationship()`, consultare "Personalizzazione delle regole della mappa per una relazione di identità composita" a pagina 293.

Vedere anche

`updateMyChildren()`, `maintainChildVerb()`, `maintainSimpleIdentityRelationship()`

"Utilizzo delle relazioni di identità composite" a pagina 290

---

maintainSimpleIdentityRelationship()

Memorizza una relazione di identità semplice composta da una mappa principale o secondaria.

Sintassi

```
public static void maintainSimpleIdentityRelationship(
      String relDefName, String particpntDefName,
      BusObj appSpecificBusObj, BusObj genericBusObj,
      CxExecutionContext map_ctx)
```

Parametri

relDefName Il nome della relazione d'identità semplice (come definito in Relationship Designer Express) in cui quest'attività partecipa.

particpntDefName
Il nome della definizione partecipante che rappresenta l'oggetto business specifico dell'applicazione.

appSpecificBusObj
La variabile che contiene l'oggetto business specifico dell'applicazione utilizzato in questa mappa.

genericBusObj La variabile che contiene l'oggetto business generico utilizzato in questa mappa.

map_ctx Il contesto di esecuzione della mappa. Per inoltrare il contesto di esecuzione della mappa, utilizzare la `variabile cwExecCtx`, che Map Designer Express definisce per ogni mappa.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`
Consultare la sezione Note per maggiori informazioni sul momento di invio di questa eccezione.

`CxMissingIDException`
Se un partecipante non esiste nelle tabelle di relazione durante l'esecuzione di una mappa con istruzione Retrieve e contesto di chiamata `SERVICE_CALL_REQUEST`. Il connettore invia un messaggio di "richiesta chiamata di servizio non riuscita" alla funzione di collaborazione senza inviare l'oggetto business all'applicazione.

Note

Il metodo `maintainSimpleIdentityRelationship()` conserva la tabella di relazione associata al partecipante *particpntDefName* della relazione d'identità semplice *relDefName*. Questo metodo conserva una relazione il cui partecipante utilizza chiave univoche provenienti da molteplici oggetti business allo stesso livello.

Il metodo maintainSimpleIdentityRelationship() esegue le seguenti operazioni di convalida su argomenti inseriti:

- Convalidare il nome della definizione di relazione *relDefName*.
- Convalidare il nome della definizione del partecipante *particpntDefName* per l'oggetto business specifico dell'applicazione.
- Accertarsi che gli oggetti business specifici dell'applicazione (*appSpecificBusObj*) e generici (*genericBusObj*) *non* siano nulli.
- Accertarsi che la relazione *relDefName* sia una relazione d'identità. Inoltre, la definizione del partecipante in *relDefName* che rappresenta l'oggetto business generico deve essere definita come IBM WebSphere Business Integration Server Express. Per ulteriori informazioni su come specificare queste impostazioni, consultare "Definizione delle relazioni di identità" a pagina 257.
- Accertarsi che il contesto di chiamata sia valido (consultare Tabella 94 per un elenco di contesti di chiamata validi).
- Accertarsi che l'istruzione dell'oggetto business specifico dell'applicazione sia supportata. Deve essere una delle seguenti: Crea, Aggiorna, Elimina, Richiama.

Se si verifica un errore con una qualsiasi convalida, maintainSimpleIdentityRelationship() invia l'eccezione RelationshipRuntimeException.

Una volta convalidati gli argomenti, l'azione avviata da maintainSimpleIdentityRelationship() dipende dalle seguenti informazioni:

- Il contesto di chiamata nel contesto di esecuzione della mappa, inserito come parte dell'argomento *map_ctx* (cwExecCtx)
- L'istruzione nell'oggetto business sorgente
  - L'oggetto business specifico dell'applicazione (*appSpecificBusObj*) per contesti di chiamata EVENT_DELIVERY (o ACCESS_REQUEST) e SERVICE_CALL_RESPONSE
  - L'oggetto business generico (*genericBusObj*) per contesti di chiamata SERVICE_CALL_REQUEST e ACCESS_RESPONSE

Il metodo maintainSimpleIdentityRelationship() gestisce tutte le operazioni di aggiunta ed eliminazione base dei partecipanti e delle istanze di relazione per ogni combinazione di contesto di chiamata e istruzione. Per maggiori informazioni sulle operazioni eseguite da maintainSimpleIdentityRelationship(), consultare "Accesso alle tabelle di relazione di identità" a pagina 278. Da Tabella 95 a Tabella 99 si possono consultare le azioni di ciascuno dei contesti di chiamata.

E' possibile chiamare questo metodo in uno dei seguenti casi:

- In fase di trasformazione per un attributo chiave di un oggetto principale
- In fase di trasformazione per un attributo chiave di una mappa secondaria che trasforma gli oggetti business secondari se quest'ultimi sono associati utilizzando una chiave univoca

Utilizzare maintainSimpleIdentityRelationship() con il metodo maintainChildVerb() per conservare una relazione d'identità semplice. Per ulteriori informazioni, consultare "Definizione delle regole di trasformazione per una relazione di identità semplice" a pagina 288.

Esempi

Il seguente esempio conserva la relazione d'identità semplice tra gli oggetti business `Clarify_BusOrg` e `Customer` generici in una mappa `Clarify_BusOrg-a-Customer` in entrata:

```
IdentityRelationship.maintainSimpleIdentityRelationship(
   "CustIdentity",
   "ClarBusOrg",
   ObjClarify_BusOrg,
   ObjCustomer,
   cxExecCtx);
```

Per maggiori esempi riguardanti `maintainSimpleIdentityRelationship()`, consultare "Definizione delle regole di trasformazione per una relazione di identità semplice" a pagina 288.

Vedere anche

`maintainChildVerb()`

"Utilizzo di relazioni di identità semplici" a pagina 277

---

updateMyChildren()

Aggiunge ed elimina le istanze secondarie in una relazione principale/secondaria specificata di una relazione d'identità.

Sintassi

```
void updateMyChildren(String parentChildRelDefName,
      String parentParticpntDef, BusObj parentBusObj,
      String childParticpntDef, String childAttrName,
      String childIdentityRelDefName,
      String childIdentityParticpntDefName,
      CxExecutionContext map_ctx)
```

Parametri

parentChildRelDefName
: Il nome della definizione di relazione parent/child.

parentParticpntDefName
: Il nome della definizione partecipante che rappresenta l'oggetto business principale nella relazione parent/child.

parentBusObj
: La variabile che contiene l'oggetto business principale.

childParticpntDefName
: Il nome della definizione partecipante che rappresenta l'oggetto business secondario nella relazione parent/child.

childAttrName
: Il nome dell'attributo nell'oggetto business principale il cui tipo è il nome oggetto secondario che partecipa alla relazione principale/secondaria. Ad esempio, in una relazione cliente-indirizzo, se l'oggetto principale contiene un attributo `Address1`, che è un oggetto business secondario del tipo `Address`, il nome attributo *childAttrName* è `Address1`.

childIdentityRelDefName
: Il nome della relazione d'identità in cui partecipa l'oggetto business secondario.

childIdentityParticpntDefName
Il nome della definizione partecipante che rappresenta l'oggetto business secondario nella relazione di identità.

map_ctx Il contesto di esecuzione della mappa. Per inoltrare il contesto di esecuzione della mappa, utilizzare la `variabile cwExecCtx`, che Map Designer Express definisce per ogni mappa.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`
Consultare la sezione Note per maggiori informazioni sul momento di invio di questa eccezione.

Note

Il metodo `updateMyChildren()` aggiorna le istanze secondaria nelle tabelle di relazione delle definizioni di relazione *parentChildRelDefName* e *childIdentityRelDefName*. Tale metodo è utile in una relazione d'identità quando un oggetto business principale viene aggiornato come risultato dell'aggiunta o rimozione degli oggetti secondari. Utilizzare `updateMyChildren()` per confrontare l'immagine successiva (in *parentBusObj*) con l'immagine precedente (informazioni nelle tabelle di relazione) in modo da determinare quali oggetti secondari sono nuovi o eliminati nell'immagine successiva.

Nota: Il metodo `updateMyChildren()` *non* può gestire la condizione dove la chiave composita secondaria dipende dai rispettivi elementi principali. Per ulteriori informazioni, consultare "Suggerimenti sull'utilizzo di Aggiorna i child" a pagina 300.

Il metodo `updateMyChildren()` effettua le seguenti operazioni di convalida sugli argomenti inseriti:

- Convalidare il nome della definizione di relazione *parentChildrelDefName* (primo argomento).
- Accertarsi che la relazione *parentChildRelDefName* sia una relazione principale/secondaria e che *parentParticpntDefName* e *childParticpntDefName* facciano parte della definizione di relazione *parentChildRefDefName*.
- Accertarsi che la relazione *childIdentityRelDefName* sia una relazione d'identità. Inoltre, la definizione del partecipante in *childIdentityRelDefName* che rappresenta l'oggetto business generico deve essere definita come `IBM WebSphere Business Integration Server Express`. Per ulteriori informazioni su come specificare queste impostazioni, consultare "Definizione delle relazioni di identità" a pagina 257.
- Accertarsi che *childIdentityParticpntDefName* faccia parte della definizione di relazione *childIdentityRefDefName*

Se si verifica un errore con una qualsiasi convalida, `updateMyChildren()` invia l'eccezione `RelationshipRuntimeException`.

Una volta convalidati gli argomenti, il metodo `updateMyChildren()` aggiunge o elimina gli elementi secondari dall'elenco di oggetti business secondari che appartengono all'oggetto business principale specificato come appropriato. Tale

metodo esegue una delle seguenti operazioni nelle tabelle di relazione per partecipanti principali e secondari (*parentParticpntDefName* e *childParticpntDefName* rispettivamente):

- Per ogni nuovo oggetto secondario, `updateMyChildren()` aggiunge un'istanza secondaria.

  Questo metodo *non* effettua aggiunte alla tabella di relazione secondaria dal momento che tutti gli oggetti business che sono attualmente associati all'oggetto principale sono stati già conservati quando `maintainCompositeRelationship()` è stato chiamato.
- Per ogni oggetto secondario eliminato, `updateMyChildren()` rimuove l'istanza secondaria corrispondente.

  Tale metodo rimuove dalla tabella secondaria dei riferimenti incrociati in aggiunta alla tabella di relazione principale/secondaria.

Il metodo `updateMyChildren()` richiede la definizione di una relazione principale/secondaria con Relationship Designer Express. Per informazioni su come creare questo tipo di relazione, consultare "Creazione di una definizione di relazione parent/child" a pagina 299.

Nota: Se l'oggetto business secondario dispone di una chiave univoca, l'attributo del partecipante secondario è la chiave univoca dell'oggetto secondario. Se l'oggetto secondario non dispone di una chiave univoca, l'attributo del partecipante secondario è una chiave non univoca.

Esempi

Per osservare un esempio riguardante `updateMyChildren()` insieme al metodo `maintainCompositeRelationship()`, consultare la sezione Esempi di `maintainCompositeRelationship()`.

Per maggiori esempi riguardanti `updateMyChildren()`, consultare "Personalizzazione delle regole della mappa per una relazione di identità composita" a pagina 293.

Vedere anche

`addMyChildren()`, `deleteMyChildren()`, `maintainCompositeRelationship()`, `maintainSimpleIdentityRelationship()`

"Gestione degli aggiornamenti all'oggetto business parent" a pagina 299

Capitolo 22. Classe CxExecutionContext

La classe `CxExecutionContext`fornisce metodi che agiscono su un contesto di esecuzione globale, che è il possessore di informazioni sul contesto accessibili dall'utente associato ad un dato flusso. Attualmente, solo l'informazione di esecuzione specifica della mappa viene mostrata come contesto di esecuzione della mappa. Map Designer Express dichiara automaticamente una variabile speciale nel codice della mappa per accedere il contesto di esecuzione, `cwExecCtx`. Quando si chiama una mappa da una collaborazione, per quanto, si debba creare l'istanza del proprio contesto di esecuzione globale ed contesto di esecuzione mappa.

La Tabella 140 riepiloga i metodi della classe `CxExecutionContext`.

Tabella 140. Sommario metodi CxExecutionContext

Metodo	Descrizione	Pagina
`CxExecutionContext()`	Crea una nuova istanza di un contesto di esecuzione globale.	473
`getContext()`	Richiama il contesto di esecuzione specificato dal contesto di esecuzione globale.	474
`setContext()`	Imposta un particolare contesto di esecuzione in modo che sia parte del contesto di esecuzione globale.	474

Costanti statiche

La classe `CxExecutionContext` definisce le costanti statiche che la Tabella 141 mostra.

Tabella 141. Costanti statiche definite nella classe CxExecutionContext

Nome costante	Significato
`MAPCONTEXT`	Una costante di stringa per indicare che il contesto di esecuzione è specifico della mappa.

CxExecutionContext()

Crea una nuova istanza di un contesto di esecuzione globale.

Sintassi

```
CxExecutionContext()
```

Parametri

Nessuno

Valori di ritorno

Restituisce una nuova istanza del contesto di esecuzione globale.

Note

Il costruttore `CxExecutionContext()` restituisce un contesto di esecuzione globale, che è necessario per contenere il contesto di esecuzione della mappa prima di chiamare una mappa dalla collaborazione.

getContext()

Richiama il contesto esecuzione specificato dal contesto di esecuzione globale.

Sintassi

```
Object getContext(String contextName)
```

Parametri

contextName Il nome di un contesto di esecuzione da ottenere dal contesto di esecuzione globale.

Valori di ritorno

Restituisce un'istanza del contesto di esecuzione specificato.

Esempi

L'esempio che segue ottiene un contesto di esecuzione della mappa da un contesto di esecuzione globale.

```
(MapExeContext) mapExeContext = globalExeContext.getContext(
   CxExecutionContext.MAPCONTEXT);
```

setContext()

Imposta un particolare contesto di esecuzione in modo che sia parte del contesto di esecuzione globale.

Sintassi

```
void setContext(String contextName, Object context)
```

Parametri

contextName Il nome del contesto di esecuzione specifico da assegnare al contesto di esecuzione globale. Attualmente, `MAPCONTEXT` è l'unico valore valido.

context Un oggetto che contiene l'informazione per il contesto di esecuzione. Per contesti di esecuzione di mappa, questo `Oggetto` è di tipo `MapExeContext`.

Valori di ritorno

Nessuno

Note

È possibile impostare in modo esplicito il contesto di esecuzione della mappa se si vuole eseguire una mappa secondaria con trasformazioni relazionali. In questo caso si può decidere di creare un nuovo contesto, con il proprio contesto di chiamata (initiator) nel contesto della mappa così che la relazione agisce in modo corretto.

Attualmente, è possibile utilizzare CxExecutionContext() per impostare il contesto di esecuzione della mappa, e salvare un unica istanza MapExeContext() in CxExecutionContext(). Se si utilizza setContext()per impostare un nuovo MapExeContext()nel CxExecutionContext(), il precedente andrà perduto.

Esempi

L'esempio che segue mostra l'utilizzo di setContext():

```
         mapExeContext originalMapExeContext = (MapExeContext)
cwExecCtx.getContext(CxExecutionContext.MAPCONTEXT);
     MapExeContext newMapExeContext = new MapExeContext();
newMapExeContext.setInitiator(originalMapExeContext.getInitiator());
     newMapExeContext.setConnName(originalMapExeContext.getConnName());
     newMapExeContext.setLocale(originalMapExeContext.getLocale());
     cwExecCtx.setContext(CxExecutionContext.MAPCONTEXT,
newMapExeContext);
```

L'esempio che segue salva un contesto di esecuzione della mappa in un contesto di esecuzione globale:

```
globalExeContext.setContext(CxExecutionContext.MAPCONTEXT,
   mapExeContext);
```

L'esempio seguente mostra come impostare il contesto di esecuzione della mappa:

```
CxExecutionContext cwCtx = new CxExecutionContext();
MapExeContext mapCtx = new MapExeContext();
cwCtx.setContext(CxExecutionContext.MAPCONTEXT, mapCtx);

// do some work involving execution context
     cwExecCtx.setContext(CxExecutionContext.MAPCONTEXT,
originalMapExeContext);
```

Capitolo 23. Classe MapExeContext

La classe `MapExeContext` fornisce i metodi per l'esecuzione della query e l'impostazione dei diversi valori di runtime che sono attivi durante l'esecuzione della mappa.

Tabella 142 riepiloga i metodi della classe `MapExeContext`.

Tabella 142. Riepilogo del metodo MapExeContext

Metodo	Descrizione	Pagina
`getConnName()`	Richiama il nome del connettore associato all'istanza della mappa corrente.	477
`getInitiator()`	Richiama il contesto di chiamata associato all'istanza della mappa corrente.	477
`getLocale()`	Richiama la locale associata al contesto di esecuzione della mappa.	479
`getOriginalRequestBO()`	Richiama l'oggetto business della richiesta originale associato all'istanza della mappa corrente.	479
`setConnName()`	Imposta il nome del connettore associato all'istanza della mappa corrente.	480
`setInitiator()`	Imposta il contesto di chiamata associato all'istanza della mappa corrente.	481
`setLocale()`	Imposta la locale associata al contesto di esecuzione della mappa.	481

getConnName()

Richiama il nome del connettore associato all'istanza della mappa corrente.

Sintassi

```
String getConnName()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce una stringa contenente il nome del connettore.

Eccezioni

Nessuno.

Vedere anche

`setConnName()`

getInitiator()

Richiama il contesto di chiamata associato all'istanza della mappa corrente.

Sintassi

```
String getInitiator()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce una variabile di costante statica rappresentante il contesto di chiamata per l'esecuzione dell'istanza mappa corrente. I contesti di chiamata sono espressi da uno dei seguenti valori:

`EVENT_DELIVERY`
: Gli oggetti business sorgenti di cui si esegue la mappatura vengono inviati da un'applicazione a InterChange Server Express tramite un connettore.

`ACCESS_REQUEST`
: Gli oggetti sorgenti di cui si esegue la mappatura vengono inviati da un'applicazione a InterChange Server Express tramite un client di accesso.

`SERVICE_CALL_REQUEST`
: Gli oggetti sorgenti di cui si esegue la mappatura vengono inviati da InterChange Server Express ad un'applicazione tramite un connettore.

`SERVICE_CALL_RESPONSE`
: Gli oggetti sorgenti di cui si esegue la mappatura vengono restituiti a InterChange Server Express da un'applicazione tramite un connettore, in seguito a una corretta richiesta di chiamata assistenza.

`SERVICE_CALL_FAILURE`
: Gli oggetti sorgenti di cui si esegue la mappatura vengono restituiti a InterChange Server Express da un'applicazione tramite un connettore, in seguito a una richiesta di chiamata assistenza non riuscita.

`ACCESS_RESPONSE`
: Gli oggetti sorgenti di cui si esegue la mappatura vengono restituiti da InterChange Server Express all'applicazione tramite un client di accesso.

Eccezioni

Nessuno.

Note

Il contesto di chiamata fa parte del contesto di esecuzione della mappa. Per maggiori informazioni su come vengono utilizzati i contesti di chiamata nelle mappe, consultare "Contesti di esecuzione delle mappe" a pagina 200.

Esempi

Nell'esempio riportato di seguito, confrontare il programma di avvio runtime della mappa con le costanti definite nella classe MapExeContext:

```
cwMapCtx =
(MapExeContext)cwExecCtx.getContext (CxExecutionContext.MAPCONTEXT);
```

```
String sInitiator = null;
sInitiator = cwMapCtx.getInitiator();
     logInfo("**************Initiator =  + sInitiator);
```

Vedere anche

`getOriginalRequestBO()`, `setInitiator()`

---

getLocale()

Richiama la locale associata al contesto di esecuzione della mappa.

Sintassi

```
Locale getLocale()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce un oggetto Locale contenente la lingua e il codice paese per il contesto di esecuzione della mappa.

Eccezioni

Nessuno.

Note

Questo metodo deve essere eseguito sulla variabile mappa del tipo `MapExeContext`, che assume il nome `cwMapCtx` quando generato dal sistema o che viene denominato dall'utente quando si chiama una mappa in un ambiente che non genera in modo automatico il codice mappa (come in una collaborazione).

Esempi

L'esempio riportato di seguito richiama la locale del contesto di esecuzione della mappa in una variabile e poi ne esegue la notifica in un'istruzione di traccia:

```
Locale mapLocale = cwMapCtx.getLocale();
String mapLocaleToString = mapLocale.toString();
trace(3, "THE MAP LOCALE IS: " + mapLocaleToString);
```

Vedere anche

`setLocale()`

---

getOriginalRequestBO()

Richiama l'oggetto business della richiesta originale associato all'istanza della mappa corrente.

Sintassi

```
BusObj getOriginalRequestBO()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce l'oggetto business della richiesta originale per la mappa, come mostrato nella seguente tabella:

Contesti di chiamata	Oggetto business della richiesta originale
EVENT_DELIVERY ACCESS_REQUEST	Oggetto business specifico dell'applicazione proveniente dall'applicazione
SERVICE_CALL_REQUEST SERVICE_CALL_FAILURE	L'oggetto business generico interrotto da InterChange Server Express
SERVICE_CALL_RESPONSE	L'oggetto business generico interrotto da SERVICE_CALL_REQUEST
ACCESS_RESPONSE	Oggetto business specifico dell'applicazione proveniente dalla richiesta di accesso iniziale

Eccezioni

Nessuno.

Note

L'oggetto business della richiesta originale fa parte del contesto di esecuzione della mappa. Il metodo `getOriginalRequestBO()` restituisce l'oggetto business della richiesta originale, che dipende dal contesto di chiamata della mappa. Per maggiori informazioni su come viene utilizzato questo oggetto business nelle mappe, consultare "Oggetti business della richiesta originale" a pagina 202.

Vedere anche

`getInitiator()`

setConnName()

Imposta il nome del connettore associato all'istanza della mappa corrente.

Sintassi

```
void setConnName(String connectorName)
```

Parametri

connectorName Nome del connettore

Valori di ritorno

Nessuno.

Eccezioni

Nessuno.

Note

Il controller del connettore specificato deve essere eseguito in InterChange Server Express.

Vedere anche

`getConnName()`

setInitiator()

Imposta il contesto di chiamata associato all'istanza della mappa corrente.

Sintassi

```
void setInitiator(String callingContext)
```

Parametri

callingContext

Una stringa contenente uno dei seguenti valori:

EVENT_DELIVERY	Gli oggetti sorgenti di cui si esegue la mappatura vengono inviati da un'applicazione, tramite un connettore, a InterChange Server Express.
ACCESS_REQUEST	Gli oggetti sorgenti di cui si esegue la mappatura vengono inviati da un'applicazione a InterChange Server Express tramite un client di accesso.
SERVICE_CALL_REQUEST	Gli oggetti sorgenti di cui si esegue la mappatura vengono inviati da InterChange Server Express ad un'applicazione tramite un connettore.
SERVICE_CALL_RESPONSE	Gli oggetti sorgenti di cui si esegue la mappatura vengono restituiti a InterChange Server Express da un'applicazione tramite un connettore, in seguito a una corretta richiesta di chiamata assistenza.
SERVICE_CALL_FAILURE	Gli oggetti sorgenti di cui si esegue la mappatura vengono restituiti a InterChange Server Express da un'applicazione tramite un connettore, in seguito a una richiesta di chiamata assistenza non riuscita.
ACCESS_RESPONSE	Gli oggetti sorgenti di cui si esegue la mappatura vengono restituiti da InterChange Server Express all'applicazione tramite un client di accesso.

Valori di ritorno

Nessuno.

Eccezioni

Nessuno.

Note

Il contesto di chiamata fa parte del contesto di esecuzione della mappa. Il contesto di chiamata indica la direzione in cui viene eseguita la mappatura dell'oggetto business sorgente. Per maggiori informazioni su come vengono utilizzati i contesti di chiamata nelle mappe, consultare "Contesti di esecuzione delle mappe" a pagina 200.

Vedere anche

`getInitiator()`

---

setLocale()

Imposta la locale associata al contesto di esecuzione della mappa.

Sintassi

```
void setLocale(Locale newLocale)
```

Parametri

newLocale Il nuovo oggetto `Locale` su cui impostare il contesto di esecuzione della mappa.

Valori di ritorno

Nessuno.

Eccezioni

Nessuno.

Note

Questo metodo deve essere eseguito sulla variabile mappa del tipo `MapExeContext`, che assume il nome `cwMapCtx` quando generato dal sistema o che viene denominato dall'utente quando si chiama una mappa in un ambiente che non genera in modo automatico il codice mappa (come in una collaborazione).

La locale dell'oggetto business prodotta da una mappa è condizionata dalla locale del contesto di esecuzione della mappa. Se si modifica la locale del contesto di esecuzione della mappa come parte della logica delle mappe, la nuova locale viene copiata nell'oggetto business. Tale operazione viene eseguita quando la logica modificabile dall'utente è terminata (cioè, quando sono terminate le trasformazioni visibili nel diagramma di Map Designer Express). E' possibile utilizzare tale API per modificare l'oggetto business in una locale diversa da quella che aveva quando è stato inserito nella mappa.

Esempi

Il codice riportato di seguito definisce un nuovo oggetto `Locale`, imposta il contesto di esecuzione della mappa su quel nuovo valore Locale e quindi notifica la locale del contesto di esecuzione della mappa:

```
Locale newLocale = new Locale("ja", "JP");
cwMapCtx.setLocale(newLocale);
trace(3, "THE MAP LOCALE IS NOW: " + cwMapCtx.getLocale().toString());
```

Vedere anche

`getLocale()`

Metodi obsoleti

Alcuni metodi nella classe `MapExeContext` erano supportati nelle versioni precedenti ma non sono più supportati. Tali *metodi obsoleti* non genereranno errori, ma CrossWorlds ne sconsiglia l'utilizzo e consiglia di migrare il codice esistente ai nuovi metodi. I metodi non approvati potrebbero essere eliminati in un futuro release.

Tabella 143 elenca il metodo obsoleto per la classe `MapExeContext`. Se Map Designer Express non è stato utilizzato prima, ignorare questa sezione.

Tabella 143. Metodo obsoleto, classse MapExeContext

Metodo precedente	Metodo sostitutivo
`getGenericBO()`	`getOriginalRequestBO()`

Capitolo 24. Classe Participant

I metodi documentati in questo capitolo operano su oggetti della classe Participant. Le istanze Participant sono utilizzate nelle istanze di relazione. Ogni istanza Participant contiene le seguenti informazioni:

- nome della definizione di relazione
- ID istanza di relazione
- nome della definizione di partecipante
- dati da associare al partecipante.

La classe Participant fornisce dei metodi per l'impostazione e la ricerca di ciascuno di questi valori per un dato partecipante.

Tabella 144 riepiloga i metodi della classe Participant.

Tabella 144. Riepilogo del metodo Participant

Metodo	Descrizione	Pagina
Participant()	Crea una nuova istanza Participant.	485
getBusObj(), getString(), getLong(), getInt(), getDouble(), getFloat(), getBoolean()	Richiama i dati associati all'istanza Participant.	487
getInstanceId()	Richiama l'ID istanza della relazione in cui sta partecipando l'istanza Participant.	487
getParticipantDefinition()	Richiama il nome della definizione partecipante da cui viene creata l'istanza Participant.	488
getRelationshipDefinition()	Richiama il nome della definizione di relazione in cui l'istanza Participant sta partecipando.	488
set()	Imposta i dati associati all'istanza Participant.	489
setInstanceId()	Imposta l'ID istanza della relazione in cui sta partecipando l'istanza Participant.	489
setParticipantDefinition()	Imposta il nome della definizione partecipante da cui viene creata l'istanza Participant.	490
setRelationshipDefinition()	Imposta la definizione di relazione in cui sta partecipando l'istanza Participant.	490

Participant()

Crea una nuova istanza Participant.

Sintassi

Per aggiungere una nuova istanza partecipante ad una esistente in una istanza di relazione:

```
Participant(String relDefName,String partDefName,
int instanceId,BusObj partData)
Participant(String relDefName,String partDefName,
int instanceId,String partData)
Participant(String relDefName,String partDefName,
int instanceId,long partData)
Participant(String relDefName,String partDefName,
int instanceId,int partData)
Participant(String relDefName,String partDefName,
int instanceId,double partData)
Participant(String relDefName,String partDefName,
int instanceId,float partData)
Participant(String relDefName,String partDefName,
int instanceId,boolean partData)
```

Per creare una nuova istanza partecipante con nessuna istanza di relazione:

```
Participant(String relDefName,String partDefName, BusObj partData)
Participant(String relDefName,String partDefName, String partData)
Participant(String relDefName,String partDefName, long partData)
Participant(String relDefName,String partDefName, int partData)
Participant(String relDefName,String partDefName, double partData)
Participant(String relDefName,String partDefName, float partData)
Participant(String relDefName,String partDefName, boolean partData)
```

Parametri

relDefName Nome della definizione di relazione

partDefName Nome della definizione partecipante che descrive il partecipante.

instanceId L'ID per l'istanza di relazione che riceve la nuova istanza partecipante.

participantData
Dati da associare all'istanza partecipante. Può essere uno dei seguenti tipi di dati: `BusObj`, `String`, `long`, `int`, `double`, `float`, `boolean`.

Valori di ritorno

Restituisce la nuova istanza partecipante.

Eccezioni

`RelationshipRuntimeException` – Consultare "Gestione delle eccezioni" a pagina 196.

Note

Questo metodo è il costruttore classe `Participant`. Utilizza i seguenti moduli:

- Il primo modulo del costruttore aggiunge una nuova istanza partecipante all'istanza di relazione identificata da *instanceId*.
- Il secondo modulo crea una nuova istanza partecipante con nessuna istanza di relazione associata. E' possibile utilizzare quest'istanza partecipante come argomento per `IdentityRelationship.addMyChildren()` o `Relationship.create()` per creare una nuova istanza di relazione. Con il metodo `Relationship.create()` l'assenza di un ID istanza di relazione è un requisito.

I dati da associare al parametro *participantData* dipende dal tipo di relazione:

- Per creare un'istanza partecipante per una relazione d'identità, utilizzare un oggetto business come parametro *participantData*.

- Per creare un partecipante per una relazione di controllo, utilizzare uno qualsiasi dei seguenti tipi di dati per il parametro *participantData*: `String`, `long`, `int`, `double`, `float`, `boolean`.

Esempi

```
// creare un'istanza partecipante con nessun ID istanza di relazione
participant p = new Participant(myRelDef,myPartDef,myBusObj);

// creare un'istanza di relazione
int relInstanceId = Relationship.addParticipant(p);
```

Vedere anche

`addMyChildren()`, Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249, “Trasformazione con una mappa secondaria” a pagina 47

---

getBusObj(), getString(), getLong(), getInt(), getDouble(), getFloat(), getBoolean()

Richiama i dati associati all’istanza `Participant`.

Sintassi

```
BusObj getBusObj()
String getString()
long getLong()
int getInt()
double getDouble()
float getFloat()
boolean getBoolean()
```

Valori di ritorno

Restituisce i dati associati a quest’istanza partecipante. Tale valore dati è del tipo incluso nel nome metodo. Ad esempio, `getBoolean()` restituisce un valore `boolean`, `getBusObj()` restituisce un valore `BusObj`, `getDouble()` restituisce un valore `double` e così via.

Eccezioni

`RelationshipRuntimeException` – Consultare “Gestione delle eccezioni” a pagina 196.

Vedere anche

`set()`, Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249, “Trasformazione con una mappa secondaria” a pagina 47

---

getInstanceId()

Richiama l’ID istanza di relazione della relazione in cui sta partecipando l’istanza `Participant`.

Sintassi

```
int getInstanceId()
```

Valori di ritorno

Restituisce un numero intero che rappresenta l'ID istanza dell'istanza di relazione in cui sta partecipando quest'istanza `Participant`. Se l'istanza `Participant` *non* è un membro di un'istanza di relazione, questo metodo restituisce la costante `INVALID_INSTANCE_ID`.

Eccezioni

`RelationshipRuntimeException` – Consultare "Gestione delle eccezioni" a pagina 196.

Vedere anche

`setInstanceId()`, Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249, "Trasformazione con una mappa secondaria" a pagina 47

---

getParticipantDefinition()

Richiama il nome di definizione partecipante da cui viene creata l'istanza `Participant`.

Sintassi

```
String getParticipantDefinition()
```

Valori di ritorno

Restituisce un valore `String` contenente il nome della definizione di partecipante associato a quest'istanza partecipante.

Eccezioni

`RelationshipRuntimeException` – Consultare "Gestione delle eccezioni" a pagina 196.

Vedere anche

`setParticipantDefinition()`, Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249, "Trasformazione con una mappa secondaria" a pagina 47

---

getRelationshipDefinition()

Richiama il nome della definizione di relazione in cui sta partecipando l'istanza `Participant`.

Sintassi

```
String getRelationshipDefinition()
```

Valori di ritorno

Restituisce un valore `String` contenente il nome della definizione di relazione in cui partecipa quest'istanza partecipante.

Eccezioni

`RelationshipRuntimeException` – Consultare "Gestione delle eccezioni" a pagina 196.

Vedere anche

`setRelationshipDefinition()`, Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249, “Trasformazione con una mappa secondaria” a pagina 47

---

set()

Imposta i dati associati all'istanza `Participant`.

Sintassi

```
void set(BusObj partData)
void set(String partData)
void set(long partData)
void set(int partData)
void set(double partData)
void set(float partData)
void set(boolean partData)
```

Parametri

partData	Dati da associare all'istanza `Participant`. Può essere uno dei seguenti tipi di dati: BusObj, String, long, int, double, float, boolean.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException` – Consultare “Gestione delle eccezioni” a pagina 196.

Note

Se si impostano i dati partecipante in modo da essere un oggetto business (tipo `BusObj`), la definizione di relazione e quella di partecipante devono essere già impostate. Se si impostano i dati del partecipante su un qualsiasi altro tipo di dati, non ha importanza quale impostazione si specifica prima.

Vedere anche

`getBusObj()`, `getString()`, `getLong()`, `getInt()`, `getDouble()`, `getFloat()`, `getBoolean()`, Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249, “Trasformazione con una mappa secondaria” a pagina 47

---

setInstanceId()

Imposta l'ID istanza della relazione in cui sta partecipando `l'istanza Participant`.

Sintassi

```
void setInstanceId(int id)
```

Parametri

id	L'ID istanza della relazione.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException` – Consultare “Gestione delle eccezioni” a pagina 196.

Note

`setInstanceId()` può essere utilizzato per rimuovere l'ID istanza di relazione quando si intende inoltrare un'istanza di partecipante come parametro a `Participant()` o ai metodi `create()`. In questo caso, si imposta l'ID istanza sulla costante `INVALID_INSTANCE_ID`.

Esempi

```
// rimuovere l'ID istanza di relazione
myParticipant.setInstanceId(Participant.INVALID_INSTANCE_ID);

// inviare l'istanza partecipante al metodo create()
int newRelId = create(myParticipant);
```

Vedere anche

`getInstanceId()`, Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249, “Trasformazione con una mappa secondaria” a pagina 47

---

setParticipantDefinition()

Imposta il nome di definizione partecipante da cui viene creata l'istanza `Participant`.

Sintassi

```
void setParticipantDefinition(String partDefName)
```

Parametri

partDefName Nome della definizione partecipante da cui viene creata l'istanza `Participant`.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException` – Consultare “Gestione delle eccezioni” a pagina 196.

Vedere anche

`setParticipantDefinition()`, Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249, “Trasformazione con una mappa secondaria” a pagina 47

---

setRelationshipDefinition()

Imposta la definizione di relazione in cui sta partecipando l'istanza `Participant`.

Sintassi

```
void setRelationshipDefinition(String relDefName)
```

Parametri

relDefName Nome della definizione di relazione

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException` – Consultare “Gestione delle eccezioni” a pagina 196.

Vedere anche

`getRelationshipDefinition()`, Capitolo 7, “Creazione delle definizioni di relazione”, a pagina 249, “Trasformazione con una mappa secondaria” a pagina 47

Capitolo 25. Classe Relationship

I metodi documentati in questo capitolo operano su oggetti della classe IBM WebSphere Business Integration Server Express `Relationship`. La classe `Relationship` fornisce i metodi per la gestione delle istanze runtime di relazioni definite *istanze di relazione*. Si utilizzano tipicamente questi metodi durante le operazioni di trasformazione di attributi per oggetti business che sono mappati come relazioni d'identità o ricerche statiche. Per maggiori informazioni sulla programmazione degli attributi di relazione utilizzando i metodi in questa classe, consultare "Trasformazione con una mappa secondaria" a pagina 47.

Gran parte dei metodi in questa classe supportano variazioni ai parametri specificati. Di solito le variazioni si attengono a queste direttive:

- Per identificare uno specifico partecipante in un'istanza di relazione, si specifica solitamente il nome di definizione della relazione, il nome di definizione del partecipante, l'ID istanza di relazione e l'oggetto business associato al partecipante.
- In alternativa, è possibile specificare un'istanza Participant contenente il nome di definizione della relazione, il nome di definizione del partecipante, l'ID istanza e l'oggetto business come propri attributi.
- Per alcune operazioni, è possibile omettere l'ID istanza di relazione (ad esempio, quando si crea una nuova relazione) o il nome dell'oggetto business.

Nella maggior parte dei casi, se si ha un'istanza Participant (ad esempio, quale risultato di una chiamata `retrieve()`), è più facile trasferirla come parametro ad un metodo della classe `Relationship` piuttosto che specificare ogni attributo singolarmente.

Tutti i metodi in questa classe sono dichiarati come statici. E' possibile richiamarli dalle istanze di relazione esistenti oppure specificando la classe `Relationship`.

Tabella 145 riepiloga i metodi nella classe `Relationship`.

Tabella 145. Riepilogo del metodo di relazione

Metodo	Descrizione	Pagina
Metodi statici		
`addParticipant()`	Aggiunge un nuovo partecipante ad un'istanza di relazione.	494
`create()`	Crea una nuova istanza di relazione.	496
`deactivateParticipant()`	Disattiva un partecipante da una o più istanze di relazione.	497
`deactivateParticipantByInstance()`	Disattiva un partecipante da un'istanza di relazione specifica.	498
`deleteParticipant()`	Rimuove un'istanza di partecipante da una o più istanze di relazione.	499
`deleteParticipantByInstance()`	Rimuove un partecipante da un'istanza di relazione specifica.	500
`getNewID()`	Restituisce il successivo ID istanza di relazione disponibile per una relazione, in base al nome della definizione di relazione.	501

Tabella 145. Riepilogo del metodo di relazione (Continua)

Metodo	Descrizione	Pagina
`retrieveInstances()`	Richiama solo gli ID dell'istanza di relazione per zero o più istanze di relazione che contengono un'istanza partecipante fornita.	502
`retrieveParticipants()`	Ottiene zero o più partecipanti da un'istanza di relazione.	504
`updateParticipant()`	Aggiorna un partecipante in una o più istanze di relazione.	505

addParticipant()

Aggiunge un nuovo partecipante a unaistanza di relazione.

Sintassi

Per aggiungere un nuovo partecipante ad un'istanza di relazione esistente:

```
int addParticipant
 (String relDefName,
String partDefName,
int instanceId,BusObj partData)

int addParticipant
 (String relDefName,
String partDefName,
 int instanceId,String partData)

int addParticipant
 (String relDefName,
String partDefName,int instanceId,
long partData)

int addParticipant
 (String relDefName,
String partDefName,int instanceId,
int partData)

int addParticipant
 (String relDefName,
String partDefName,
int instanceId,
double partData)

int addParticipant
 (String relDefName,
String partDefName,
int instanceId, float partData)

int addParticipant
 (String relDefName,
String partDefName,
int instanceId,
boolean partData)
```

Per aggiungere un partecipante ad una nuova istanza di relazione:

```
int addParticipant
 (String relDefName,
String partDefName,
      BusObj partData)
int addParticipant
 (String relDefName,
String partDefName,
```

```
      String partData)
int addParticipant
 (String relDefName,
String partDefName,
      long partData)
int addParticipant
 (String relDefName,
String partDefName,
      int partData)
int addParticipant
 (String relDefName,
String partDefName,
      double partData)
int addParticipant
 (String relDefName,
String partDefName,
      float partData)
int addParticipant
 (String relDefName,
String partDefName,
      boolean partData)
```

Per aggiungere un'istanza partecipante esistente ad un'istanza di relazione:

```
int addParticipant(Participant participant)
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante.
instanceId	ID dell'istanza di relazione che riceve il nuovo partecipante.
partData	Dati da associare al partecipante. Può essere uno dei seguenti tipi di dati: `BusObj`, `String`, `long`, `int`, `double`, `float`, `boolean`.
participant	Partecipante da aggiungere alla relazione.

Valori di ritorno

Restituisce un numero intero che rappresenta l'ID istanza della relazione che riceve il nuovo partecipante.

Eccezioni

`RelationshipRuntimeException` – Consultare "Gestione delle eccezioni" a pagina 196.

Note

Il primo modulo del metodo aggiunge un nuovo partecipante all'istanza di relazione specificata. Ogni variazione supporta un diverso tipo di dati per i dati da associare al partecipante.

Il secondo modulo, dal momento che non specifica un'istanza di relazione, crea una nuova istanza di relazione e aggiunge un nuovo partecipante. In questo caso, il valore di ritorno risulta l'ID istanza della nuova istanza di relazione creata. Ogni variazione supporta un diverso tipo di dati per i dati da associare al partecipante.

Il terzo modulo aggiunge l'istanza partecipante che si invia all'istanza di relazione specificata nell'istanza partecipante. Se l'istanza partecipante non ha alcun ID istanza di relazione, viene creata una nuova istanza di relazione e restituito il nuovo ID istanza.

Il metodo addParticipant() è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe Relationship.

Vedere anche

create()

---

create()

Crea un nuovo istanza di relazione.

Sintassi

```
int create(String relDefName, String partDefName, BusObj partData)
int create(String relDefName, String partDefName, String partData)
int create(String relDefName, String partDefName, long partData)
int create(String relDefName, String partDefName, int partData)
int create(String relDefName, String partDefName, double partData)
int create(String relDefName, String partDefName, float partData)
int create(String relDefName, String partDefName, boolean partData)
int create(Participant participant)
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Il nome della definizione partecipante.
partData	Dati da associare al partecipante. Può essere uno dei seguenti tipi di dati: BusObj, String, long, int, double, float, boolean.
participant	Primo partecipante nella relazione.

Valori di ritorno

Restituisce un numero intero che rappresenta l'ID istanza di relazione della nuova relazione.

Eccezioni

RelationshipRuntimeException

Note

Il metodo create() crea una nuova istanza di relazione con un'istanza partecipante della definizione partecipante *partDefName*. E' possibile specificare i dati per questa nuova istanza partecipante mediante l'argomento *partData*. In seguito alla chiamata di questo metodo, è possibile chiamare addMyChildren() per aggiungere ulteriori partecipanti all'istanza di relazione.

Nell'ultimo modulo del metodo, il parametro *participant* non può avere un ID istanza di relazione. Generalmente, le istanze partecipante hanno degli ID istanze di relazione. Poiché tale metodo crea una nuova istanza di relazione, occorre essere certi che l'istanza partecipante non abbia già un'istanza ad essa associata. Per fare ciò, utilizzare il metodo setInstanceId() (nella classe Participant) per impostare l'ID istanza sulla costante INVALID_INSTANCE_ID.

Il metodo create() è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe Relationship.

Vedere anche

`addMyChildren()`, `setInstanceId()`

---

deactivateParticipant()

Disabilita un partecipante da una o più istanze di relazione.

Sintassi

```
void deactivateParticipant(String relDefName,
String partDefName,
      BusObj partData)

void deactivateParticipant(String
relDefName,
String partDefName,
      String partData)

void deactivateParticipant(String relDefName,
String partDefName,
       long partData)

void deactivateParticipant(String relDefName,
String partDefName,
       int partData)

void deactivateParticipant(String relDefName,
String partDefName,
       double partData)

void deactivateParticipant(String relDefName,
String partDefName,
      float partData)

void deactivateParticipant(String relDefName,
String partDefName,
      boolean partData)

void deactivateParticipant(Participant participant)
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante.
partData	Dati associati al partecipante. Può essere uno dei seguenti tipi di dati: `BusObj`, `String`, `long`, `int`, `double`, `float`, `boolean`.
participant	Partecipante da disabilitare nella relazione.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `deactivateParticipant()` disabilita il partecipante da tutte le istanze di *relDefName* dove l'attributo *partData*è associato a *partDefName*. Tale metodo *non*

rimuove il partecipante dalle tabelle di relazioni. Utilizzare questo metodo per rimuovere un partecipante mentre si conserva un record della relativa esistenza nelle tabelle di relazioni.

Per visualizzare i partecipanti disabilitati, è possibile eseguire direttamente il query delle tabelle di relazioni. Per individuare i nomi delle tabelle ed accedere alle informazioni per una determinata relazione, aprire la definizione di relazione utilizzando Relationship Designer Express e scegliere Impostazioni avanzate dal menu Modifica. Consultare "Specifica delle impostazioni di relazione avanzate" a pagina 262 per maggiori informazioni su queste impostazioni.

Attenzione: Poiché `deactivateParticipant()` non rimuove realmente le righe di partecipante dalle tabelle di relazione, non utilizzare questo metodo come prassi per eliminare i partecipanti. In caso contrario le tabelle di relazione diventano troppo grandi in modo non necessario.

Il metodo `deactivateParticipant()` è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe `Relationship`.

Vedere anche

`deleteParticipant()`, `deactivateParticipantByInstance()`, Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249, "Trasformazione con una mappa secondaria" a pagina 47

---

deactivateParticipantByInstance()

Disabilita un partecipante da una specifica istanza di relazione.

Sintassi

```
void deactivateParticipantByInstance(String relDefName,
       String partDefName, int instanceId [, BusObj partData ] )

void deactivateParticipantByInstance(String relDefName,
       String partDefName, int instanceId [, String partData ] )

void deactivateParticipantByInstance(String relDefName,
       String partDefName, int instanceId [, long partData ] )

void deactivateParticipantByInstance(String relDefName,
       String partDefName, int instanceId [, int partData ] )

void deactivateParticipantByInstance(String relDefName,
       String partDefName, int instanceId [, double partData ] )

void deactivateParticipantByInstance(String relDefName,
      String partDefName, int instanceId [, float partData ] )

void deactivateParticipantByInstance(String relDefName,
      String partDefName, int instanceId [, boolean partData ] )
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante.
instanceId	L'ID dell'istanza di relazione a cui appartiene il partecipante.

partData Dati associati al partecipante. Può essere uno dei seguenti tipi di dati: BusObj, String, long, int, double, float, boolean. Si tratta di un parametro facoltativo

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException` – Consultare "Gestione delle eccezioni" a pagina 196.

Note

Il metodo `deactivateParticipantByInstance()` disabilita il partecipante specificato dall'istanza di relazione che quell'ID istanza di relazione *instanceID* identifica. Tuttavia, il metodo *non* rimuove il partecipante dalle tabelle di relazioni. Utilizzare questo metodo quando si desidera rimuovere un partecipante mentre si conserva un record della relativa esistenza nelle tabelle di relazioni.

Per visualizzare i partecipanti disabilitati, è possibile eseguire direttamente il query delle tabelle di relazioni. Per individuare i nomi delle tabelle ed accedere alle informazioni per una determinata relazione, aprire la definizione di relazione utilizzando Relationship Designer Express e scegliere Impostazioni avanzate dal menu Modifica. Consultare "Specifica delle impostazioni di relazione avanzate" a pagina 262 per maggiori informazioni su queste impostazioni.

Attenzione: Poiché `deactivateParticipantByInstance()` non rimuove realmente le righe di partecipante dalle tabelle di relazione, non utilizzare questo metodo come prassi per eliminare i partecipanti. In caso contrario le tabelle di relazione diventano troppo grandi in modo non necessario.

Il metodo `deactivateParticipantByInstance()` è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe `Relationship`.

Vedere anche

`deleteParticipant()`, `deactivateParticipant()`

deleteParticipant()

Rimuove un'istanza partecipante da una o più istanze di relazione.

Sintassi

```
void deleteParticipant(String relDefName, String partDefName,
BusObj partData)
void deleteParticipant(String relDefName, String partDefName,
String partData)
void deleteParticipant(String relDefName, String partDefName,
long partData)
void deleteParticipant(String relDefName, String partDefName,
int partData)
void deleteParticipant(String relDefName, String partDefName,
double partData)
void deleteParticipant(String relDefName, String partDefName,
float partData)
void deleteParticipant(String relDefName, String partDefName,
```

```
boolean partData)

void deleteParticipant(Participant participant)
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante.
partData	Dati associati al partecipante. Può essere uno dei seguenti tipi di dati: BusObj, String, long, int, double, float, boolean.
participant	Un'istanza Participant che rappresenta il partecipante da rimuovere dalla relazione.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `deleteParticipant()` elimina il partecipante specificato da tutte le istanze di *relDefName* dove *partData*è associato a *partDefName* e lo elimina dalle tabelle di relazione sottostanti.

Il metodo `deleteParticipant()` è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe `Relationship`.

Vedere anche

`deactivateParticipant()`, `deleteParticipantByInstance()`

---

deleteParticipantByInstance()

Rimuove un partecipante da una specifica istanza di relazione.

Sintassi

```
void deleteParticipantByInstance(String relDefName,
    String partDefName, int instanceId [, BusObj partData] )

void deleteParticipantByInstance(String relDefName,

String partDefName, int instanceId [, String partData] )

void deleteParticipantByInstance(String relDefName,

String partDefName, int instanceId [, long partData] )

void deleteParticipantByInstance(String relDefName,

String partDefName, int instanceId [, int partData] )

void deleteParticipantByInstance(String relDefName,

String partDefName, int instanceId [, double partData] )

void deleteParticipantByInstance(String relDefName,
```

```
String partDefName, int instanceId [, float partData] )

void deleteParticipantByInstance(String relDefName,
    String partDefName, int instanceId [, boolean partData] )
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante.
instanceId	L'ID dell'istanza di relazione a cui appartiene il partecipante.
partData	Dati associati al partecipante. Può essere uno dei seguenti tipi di dati: `BusObj`, `String`, `long`, `int`, `double`, `float`, `boolean`. Si tratta di un parametro facoltativo.

Valori di ritorno

Nessuno.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `deleteParticipantByInstance()` elimina un'istanza partecipante dalla relazione identificata da *instanceId* ID istanza di relazione. Il metodo rimuove il partecipante dall'istanza di relazione e dalle tabelle di relazione sottostanti.

Se viene fornito un parametro *partData* facoltativo, `deleteParticipantByInstance()` elimina l'istanza partecipante *solo* se *partData* è il dato associato alla definizione di partecipante *partDefName*.

L'ultimo modulo del metodo accetta un'istanza partecipante come unico parametro. L'istanza partecipante deve contenere il nome di definizione della relazione, il nome di definizione del partecipante e l'ID istanza oppure i dati partecipante.

Il metodo `deleteParticipantByInstance()` è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe `Relationship`.

Vedere anche

`deactivateParticipant()`

---

getNewID()

Restituisce il successivo ID istanza di relazione disponibile per una relazione, in base al nome della definizione di relazione.

Sintassi

```
public static int getNewID(String relDefName)
```

Parametri

relDefName	Nome della definizione di relazione

Valori di ritorno

Restituisce un ID istanza di relazione, in base al nome della definizione di relazione.

Eccezioni

`RelationshipRuntimeException`

Note

Poiché è possibile utilizzare l'ID istanza di relazione come ID generico delle relazioni d'identità tipiche di IBM WebSphere Business Integration Server Express, questo nuovo ID può essere utilizzato come ID generico per le relazioni "generic-to-generic".

---

retrieveInstances()

Richiama solo gli ID istanza di relazione per zero o più istanze di relazione che contengono una determinata istanza di partecipante.

Sintassi

```
int[] retrieveInstances(String relDefName,
String partDefName,
     BusObj partData)

int[] retrieveInstances(String relDefName,
String partDefName,
      String partData)

int[] retrieveInstances(String relDefName,
String partDefName,
      long partData)

int[] retrieveInstances(String relDefName,
String partDefName,
      int partData)

int[] retrieveInstances(String relDefName,
String partDefName,
      double partData)

int[] retrieveInstances(String relDefName,
String partDefName,
      float partData)

int[] retrieveInstances(String relDefName,
String partDefName,
     boolean partData)

int[] retrieveInstances(String relDefName,
String[] partDefList,
      BusObj partData)

int[] retrieveInstances(String relDefName,
String[] partDefList,
      String partData)

int[] retrieveInstances(String relDefName,
String[] partDefList,
       long partData)

int[] retrieveInstances(String relDefName,
String[] partDefList,
```

```
      int partData)

int[] retrieveInstances(String relDefName,
String[] partDefList,
      double partData)

int[] retrieveInstances(String relDefName,
String[] partDefList,
       float partData)

int[] retrieveInstances(String relDefName,
String[] partDefList,
      boolean partData)

int[] retrieveInstances(String relDefName, BusObj partData)
int[] retrieveInstances(String relDefName, String partData)
int[] retrieveInstances(String relDefName, long partData)
int[] retrieveInstances(String relDefName, int partData)
int[] retrieveInstances(String relDefName, double partData)
int[] retrieveInstances(String relDefName, float partData)
int[] retrieveInstances(String relDefName, boolean partData)
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante.
partDefList	Elenco delle definizioni di partecipante.
partData	Dati da associare al partecipante. Può essere uno dei seguenti tipi di dati: `BusObj`, `String`, `long`, `int`, `double`, `float`, `boolean`.

Valori di ritorno

Restituisce una tabella di numeri interi che rappresentano gli ID istanza delle relazioni che contengono il partecipante.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `retrieveInstances()` realizza una relazione di controllo in una mappa in entrata. Tale metodo riceve gli ID delle istanze di relazione dalle tabelle di relazione associate alle istanze di partecipante specificate (*partDefList* e *partData*oppure solo *partData*). Il metodo richiama *solo* quegli attributi che sono associati alla definizione di relazione *relDefName*. Il metodo *non* inserisce altri attributi nell'oggetto business. Gli attributi associati alla definizione di relazione sono generalmente gli attributi chiave e tutti gli altri che l'utente seleziona esplicitamente. Consultare Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249 per ulteriori informazioni sulle definizioni di relazione.

Se il metodo `retrieveInstances()` non trova un'istanza di relazione per i dati specificati, *non* causa un'eccezione. L'assenza di dati nella tabella di relazione non significa che il controllo non è stato eseguito correttamente. Se si desidera segnalare un'eccezione quando `retrieveInstances()` non trova un valore, è necessario controllare il valore degli ID istanze che il metodo restituisce e specificare esplicitamente un'eccezione `MapFailureException` se il valore è nullo.

Il metodo `retrieveInstances()` è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe `Relationship`.

Vedere anche

`addMyChildren()`, `deactivateParticipant()`, `deleteParticipant()`, `retrieveParticipants()`

"Personalizzazione delle trasformazioni della mappa per una relazione di ricerca" a pagina 275

---

retrieveParticipants()

Richiama zero o più partecipanti da una istanza di relazione.

Sintassi

```
Participant[] retrieveParticipants(String relDefName,
       String partDefName, int instanceId)

Participant[] retrieveParticipants(String relDefName,
       String[] partDefList, int instanceId)

Participant[] retrieveParticipants(String relDefName,
        int instanceId)
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante.
partDefList	Elenco delle definizioni di partecipante.
instanceId	L'ID dell'istanza di relazione a cui appartiene il partecipante.

Valori di ritorno

Restituisce una tabella di istanze Participant.

Eccezioni

`RelationshipRuntimeException`

Note

Il metodo `retrieveParticipants()` realizza una relazione di controllo in una mappa in uscita. Il metodo riceve le istanze di partecipante dalle tabelle di relazione associate all'ID istanza di relazione *instanceID* specificato. Il metodo richiama *solo* quegli attributi che sono associati alla definizione di relazione *relDefName*. Il metodo *non* inserisce altri attributi nell'oggetto business. Gli attributi associati alla definizione di relazione sono generalmente gli attributi chiave e tutti gli altri che l'utente seleziona esplicitamente. Consultare Capitolo 7, "Creazione delle definizioni di relazione", a pagina 249 per ulteriori informazioni sulle definizioni di relazione.

`retrieveParticipants()` causa l'eccezione `RelationshipRuntimeException` se riceve *instanceId di valore nullo*. Se non si è certi che il metodo `retrieveInstances()` abbia restituito un ID istanza corrispondente, controllare il valore di *instanceId* per un valore nullo *prima* della chiamata a `retrieveParticipants()`.

Il metodo retrieveParticipants() è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe Relationship.

Vedere anche

addMyChildren(), deactivateParticipant(), deleteParticipant(), retrieveInstances()

"Personalizzazione delle trasformazioni della mappa per una relazione di ricerca" a pagina 275

updateParticipant()

Aggiorna un partecipante in una o più istanze di relazione.

Sintassi

```
void updateParticipant(String relDefName, String partDefName,
BusObj partData)
```

Parametri

relDefName	Nome della definizione di relazione
partDefName	Nome della definizione di partecipante che partecipa alla relazione *relDefName*.
partData	Dati da associare al partecipante. Può essere uno dei seguenti tipi di dati: BusObj.

Valori di ritorno

Nessuno.

Eccezioni

RelationshipRuntimeException

Note

Il metodo updateParticipant() aggiorna *partData*nelle istanze di *relDefName* dove *partData*è associato con *partDefName*. Questo metodo aggiorna gli attributi senza chiave dell'oggetto business associato al partecipante specificato. Vengono aggiornati solo gli attributi associati alla definizione di relazione.

Il metodo updateParticipant() aggiorna tutte le istanza partecipante nella relazione *relDefName* che hanno:

- Una definizione partecipante *partDefName*
- Dei valori chiave che corrispondono ai valori chiave dell'oggetto business *partData*

Questo metodo aggiorna gli attributi senza chiave delle istanze partecipante con i valori nell'oggetto business partData. Vengono aggiornati solo gli attributi associati alla definizione di relazione.

Per modificare un attributo chiave o un tipo partecipante che *non* è un oggetto business (come String, long, int, double, float o boolean), è necessario prima

eliminare il partecipante utilizzando `deleteParticipant()` o `deactivateParticipant()` e quindi aggiungere un nuovo partecipante utilizzando `addMyChildren()`.

Il metodo `updateParticipant()` è un metodo di classe dichiarato come statico. E' possibile richiamare questo metodo da un'istanza di relazione esistente oppure facendo riferimento alla classe `Relationship`.

Vedere anche

`deleteParticipant()`, `deactivateParticipant()`, `addMyChildren()`

Metodi obsoleti

Alcuni metodi nella classe `Relationship`sono stati spostati nella classe `IdentityRelationship`. Tali *metodi non approvati* non genereranno errori, ma IBM ne sconsiglia l'utilizzo e consiglia di migrare il codice esistente ai nuovi metodi. I metodi non approvati potrebbero essere eliminati in un futuro release.

Tabella 146 elenca i metodi obsoleti per la classe `Relationship`.

Tabella 146. Metodi obsoleti, classe Relationship

Metodo precedente	Metodo sostitutivo
`addMyChildren()`	`addMyChildren()` nella classe `IdentityRelationship`
`deleteMyChildren()`	`deleteMyChildren()` nella classe `IdentityRelationship`
`maintainCompositeRelationship()`	`maintainCompositeRelationship()` nella classe `IdentityRelationship`
`maintainSimpleIdentityRelationship()`	`maintainSimpleIdentity Relationship()` nella classe `IdentityRelationship` class
`updateMyChildren()`	`updateMyChildren()` nella classe `IdentityRelationship`

Capitolo 26. Classe UserStoredProcedureParam

La classe UserStoredProcedureParam fornisce dei metodi per la gestione dei valori argomento di procedure memorizzate, che si possono eseguire sul database di relazione. Un oggetto UserStoredProcedureParam descrive un singolo parametro per una procedura memorizzata.

Importante: La classe UserStoredProcedureParam e i relativi metodi sono supportati *solo* per la compatibilità con le versioni precedenti. I *metodi obsoleti* non genereranno errori, ma è opportuno evitarne l'utilizzo e migrare il codice esistente ai nuovi metodi. I metodi non approvati potrebbero essere eliminati in un futuro release. Nello sviluppo di una nuova mappa, utilizzare la classe CwDBStoredProcedureParam ed i relativi metodi per fornire gli argomenti ad una procedure memorizzata.

Tabella 147 riepiloga i metodi della classe UserStoredProcedureParam.

Tabella 147. Riepilogo del metodo UserStoredProcedureParam

Metodo	Descrizione	Pagina
UserStoredProcedureParam()	Crea una nuova istanza di UserStoredProcedureParam contenente le informazioni sull'argomento per il parametro di una procedura memorizzata.	508
getParamDataTypeJavaObj()	Richiama il tipo dati di questo parametro della procedura memorizzata come Java Object, quale Integer, Double o String.	508
getParamDataTypeJDBC()	Richiama il tipo dati di questo parametro della procedura memorizzata come un tipo dati JDBC intero.	509
getParamIndex()	Richiama la posizione indice di questo parametro della procedura memorizzata.	510
getParamIOType()	Richiama il tipo di parametro in/out per questo parametro della procedura memorizzata.	510
getParamName()	Richiama il nome di questo parametro della procedura memorizzata.	511
getParamValue()	Richiama il valore di questo parametro della procedura memorizzata.	511
setParamDataTypeJavaObj()	Imposta il tipo dati come Java Object per questo parametro della procedura memorizzata.	512
setParamDataTypeJDBC()	Imposta il tipo dati come JDBC per questo parametro della procedura guidata.	512
setParamIndex()	Imposta la posizione indice di questo parametro della procedura memorizzata.	513
setParamIOType()	Imposta il tipo di parametro in/out per questo parametro della procedura memorizzata.	513
setParamName()	Imposta il nome di questo parametro della procedura memorizzata.	514
setParamValue()	Imposta il valore di questo parametro della procedura memorizzata.	514

UserStoredProcedureParam()

Crea una nuova istanza di `UserStoredProcedureParam` contenente le informazioni sull'argomento per il parametro di una procedura memorizzata.

Sintassi

```
UserStoredProcedureParam(int paramIndex, String paramType,
   Object paramValue, String ParamIOType, String paramName)
```

Parametri

paramIndex La posizione indice del parametro associato nella dichiarazione della procedura memorizzata. La numerazione dell'indice comincia con uno (1).

paramType Il tipo dati (come Java `Object`) del parametro associato.

paramValue Il valore argomento da inviare alla procedura memorizzata.

ParamIOType Il tipo in/out del parametro associato. I tipi validi sono: "IN" - il valore parametro è *solo input*. "INOUT" - il valore parametro è *input e output*. "OUT" - il valore parametro è *solo output*.

paramName Il nome dell'argomento, da utilizzare in istruzioni successive che realizzano la tabella `Vector`.

Valori di ritorno

Restituisce un nuovo oggetto `UserStoredProcedureParam` contenente le informazioni sull'argomento per l'argomento nella posizione *argIndex* della dichiarazione della procedura memorizzata.

Eccezioni

`DtpConnectionException` – Se un parametro non è valido.

getParamDataTypeJavaObj()

Richiama il tipo dati per questo parametro della procedura memorizzata come Java `Object`, quale `Integer`, `Double` o `String`.

Sintassi

```
String getParamDataTypeJavaObj()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il tipo dati del parametro `UserStoredProcedureParam` associato come Java `Object`.

Eccezioni

Nessuno.

Note

Un Java `Object` è una delle due rappresentazioni del tipo dati del parametro memorizzato nell'oggetto `UserStoredProcedureParam`. Utilizzare `getParamDataTypeJavaObj()` per ricevere il tipo dati Java `Object`; si consiglia l'utilizzo del tipo dati Java `Object` poiché:

- Per i parametri IN (e INOUT), è *necessario* fornire il valore del parametro come Java `Object`. Pertanto, fornendo il tipo dati del parametro come Java `Object` consente una maggiore congruenza.
- Il metodo `execStoredProcedure()` invia i parametri in una tabella di parametri `Vector`. L'oggetto `Vector` può contenere solo elementi che sono dei Java `Object`.

Vedere anche

`getParamDataTypeJDBC()`, `setParamDataTypeJavaObj()`

---

getParamDataTypeJDBC()

Richiama il tipo dati di questo parametro della procedura memorizzata come tipo dati JDBC intero.

Sintassi

```
int getParamDataTypeJDBC()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il tipo dati del parametro `UserStoredProcedureParam` associato come tipo dati JDBC.

Eccezioni

Nessuno.

Note

Il tipo dati JDBC è una delle due rappresentazioni del tipo dati del parametro memorizzato nell'oggetto `UserStoredProcedureParam`. I tipi di dati JDBC sono rappresentati da valori interi e includono quanto segue:

- `java.sql.Types.INTEGER`
- `java.sql.Types.VARCHAR`
- `java.sql.Types.DOUBLE`
- `java.sql.Types.DATE`

Questi tipi di dati sono definiti in `java.sql.Types`.

Consiglio: Utilizzare il tipo dati Java `Object` anziché il tipo dati JDBC. Tuttavia, la mappatura API utilizza il JDBC internamente, pertanto è possibile ottenerne il valore dall'oggetto `UserStoredProcedureParam` con il comando `getParamDataTypeJDBC()`.

Vedere anche

`getParamDataTypeJavaObj()`, `setParamDataTypeJDBC()`

getParamIndex()

Richiama la posizione indice di questo parametro della procedura memorizzata.

Sintassi

```
int getParamIndex()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce la posizione indice del parametro `UserStoredProcedureParam` associato.

Eccezioni

Nessuno.

Note

La posizione indice di un parametro della procedura memorizzata rappresenta la posizione nell'elenco dei parametri della dichiarazione relativa alla procedura memorizzata. Il primo parametro ha una posizione indice di uno (1). La posizione indice *non* fa riferimento a parametri letterali che potrebbero essere forniti alla procedura memorizzata.

Vedere anche

`setParamIndex()`

getParamIOType()

Richiama il tipo di parametro in/out per questo parametro della procedura memorizzata.

Sintassi

```
String getParamIOType()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il tipo in/out del parametro `UserStoredProcedureParam` associato.

Eccezioni

Nessuno.

Note

Il tipo di parametro in/out indica in che modo la procedura memorizzata utilizza il parametro. Può essere la rappresentazione stringa di una delle seguenti:

- Parametro IN

  Un parametro IN ha come valore *solo input*; cioè, la procedura memorizzata ne accetta il valore come input ma *non* utilizza il parametro per restituire un valore. L'attributo `getParamIOType()` restituisce il tipo di parametro in/out come "IN".

- Parametro INOUT

  Un parametro INOUT ha come valore *input e output*; cioè, la procedura memorizzata ne accetta il valore come input e utilizza anche il parametro per restituire un valore. L'attributo `getParamIOType()` restituisce il tipo di parametro in/out come "INOUT".
- Parametro OUT

  Un parametro OUT ha come valore *solo output*; cioè, la procedura memorizzata *non* legge il relativo valore come input ma utilizza il parametro per restituire un valore. L'attributo `getParamIOType()` restituisce il tipo di parametro in/out come "OUT".

Vedere anche

`setParamIOType()`

---

getParamName()

Richiama il nome di questo parametro della procedura memorizzata.

Sintassi

```
String getParamName()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il nome del parametro dall'oggetto `UserStoredProcedureParam` associato.

Eccezioni

Nessuno.

Note

Il nome del parametro viene fornito solo a scopo informativo. Viene utilizzato soltanto per messaggi di errore e per il debug. Non è richiesto il nome del parametro per accedere il parametro della procedura memorizzata poiché si accedono le procedure memorizzate mediante la relativa posizione indice nella dichiarazione della procedura memorizzata.

Vedere anche

`setParamName()`

---

getParamValue()

Richiama il valore di questo parametro della procedura memorizzata.

Sintassi

```
Object getParamValue()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il valore del parametro `UserStoredProcedureParam` associato come Java `Object`.

Eccezioni

Nessuno.

Note

Il metodo `getParamValue()` restituisce il valore del parametro come Java `Object` (cioè `Integer`, `Double` o `String`). Se il valore restituito ad un parametro OUT è JDBC `NULL`, `getParamValue()` restituisce la costante `null`.

Vedere anche

`setParamValue()`

---

setParamDataTypeJavaObj()

Imposta il tipo dati come Java `Object` per questo parametro della procedura memorizzata.

Sintassi

```
void setParamDataTypeJavaObj(String paramDataType)
```

Parametri

paramDataType Il tipo dati del parametro come Java `Object`.

Eccezioni

`DtpConnectionException` – Se il tipo dati di input non è supportato.

Note

Un Java `Object` è una delle due rappresentazioni del tipo dati del parametro memorizzato nell'oggetto `UserStoredProcedureParam`. Utilizare `setParamDataTypeJavaObj()` per impostare il tipo dati come Java `Object`. Si consiglia l'utilizzo del tipo dati Java `Object` poiché:

- Per i parametri IN (e INOUT), è *necessario* fornire il valore del parametro come Java `Object`. Pertanto, fornendo il tipo dati del parametro come Java `Object` consente una maggiore congruenza.
- Il metodo `execStoredProcedure()` invia i parametri in una tabella di parametri `Vector`. L'oggetto `Vector` può contenere solo elementi che sono dei Java `Object`.

Vedere anche

`getParamDataTypeJavaObj()`, `setParamDataTypeJDBC()`

---

setParamDataTypeJDBC()

Imposta il tipo dati come JDBC per questo parametro della procedura memorizzata.

Sintassi

```
void setParamDataTypeJDBC(int paramDataType)
```

Parametri

paramDataType Il tipo dati del parametro come tipo JDBC.

Eccezioni

`DtpConnectionException` – Se il tipo dati di input non è supportato.

Note

Ogni oggetto `UserStoredProcedureParam` contiene due rappresentazioni dei propri tipi di dati: Java `Object` e JDBC. Si consiglia l'utilizzo del tipo dati Java `Object` poiché:

- Per i parametri IN (e INOUT), è *necessario* fornire il valore del parametro come Java `Object`. Pertanto, fornendo il tipo dati del parametro come Java `Object` consente una maggiore congruenza.
- Il metodo `execStoredProcedure()` invia i parametri in una tabella di parametri `Vector`. L'oggetto `Vector` può contenere solo elementi che sono dei Java `Object`.

Vedere anche

`getParamDataTypeJDBC()`, `setParamDataTypeJavaObj()`

---

setParamIndex()

Imposta la posizione indice di questo parametro della procedura memorizzata.

Sintassi

```
void setParamIndex(int paramIndex)
```

Parametri

paramIndex La posizione indice del parametro della procedura memorizzata

Note

La posizione indice di un parametro della procedura memorizzata rappresenta la posizione nell'elenco dei parametri della dichiarazione relativa alla procedura memorizzata. Il primo parametro ha una posizione indice di uno (1). La posizione indice *non* fa riferimento a parametri letterali che potrebbero essere forniti alla procedura memorizzata.

Vedere anche

`getParamIndex()`

---

setParamIOType()

Imposta il tipo di parametro in/out di questo parametro della procedura memorizzata.

Sintassi

```
void setParamIOType(String paramIOType)
```

Parametri

paramIOType Il tipo I/O del parametro della procedura memorizzata

Note

Il tipo di parametro in/out indica in che modo la procedura memorizzata utilizza il parametro. Può essere uno dei seguenti:

- Parametro IN

  Un parametro IN ha come valore *solo input*; cioè, la procedura memorizzata ne accetta il valore come input ma *non* utilizza il parametro per restituire un valore. Per un parametro IN, impostare il tipo di parametro in/out su "IN".
- Parametro INOUT

  Un parametro INOUT ha come valore *input e output*; cioè, la procedura memorizzata ne accetta il valore come input e utilizza anche il parametro per restituire un valore. Per un parametro INOUT, impostare il tipo di parametro in/out su "INOUT".
- Parametro OUT

  Un parametro OUT ha come valore *solo output*; cioè, la procedura memorizzata *non* legge il relativo valore come input ma utilizza il parametro per restituire un valore. Per un parametro OUT, impostare il tipo di parametro in/out su "OUT".

Vedere anche

`getParamIOType()`

---

setParamName()

Imposta il nome di questo parametro della procedura memorizzata.

Sintassi

```
void setParamName(String paramName)
```

Parametri

paramName Il nome del parametro della procedura memorizzata

Note

Il nome del parametro viene fornito solo a scopo informativo. Viene utilizzato soltanto per messaggi di errore e per il debug. Non è richiesto il nome del parametro per accedere il parametro della procedura memorizzata poiché si accedono le procedure memorizzate mediante la relativa posizione indice nella dichiarazione della procedura memorizzata.

Vedere anche

`getParamName()`

---

setParamValue()

Imposta il valore di questo parametro della procedura memorizzata.

Sintassi

```
void setParamValue(Object paramValue)
```

Parametri

paramValue Il valore del parametro della procedura memorizzata. Il valore deve essere un Java `Object` (cioè `Integer`, `Double` o `String`).

Note

E' necessario impostare il valore del parametro come Java `Object`.

Vedere anche

`getParamValue()`

Capitolo 27. File di messaggi

Ciascuna mappa può avere un file di messaggi associato. Il *file di messaggi* contiene il testo per l'eccezione della mappa ed i messaggi di log. Un numero univoco identifica ciascun messaggio nel file dei messaggi. Il testo del messaggio può anche includere variabili di segnaposto, definiti *parametri*.

I metodi che creano i messaggi di mappa forniscono due alternative di creazione del testo del messaggio che appare ad un utente. La codifica della chiamata del metodo può:

- Includere il testo del messaggio.
- Contenere un riferimento al testo del messaggio contenuto in un file dei messaggi esterno.

E' solitamente una procedura migliore che una mappa faccia riferimento a un file di messaggi piuttosto che generi il testo in modo autonomo, al fine di facilitare la manutenzione, la gestione e il contesto di internazionalizzazione.

Questo capitolo descrive i file dei messaggi, il relativo utilizzo ed impostazione. Riporta i seguenti argomenti:

Ubicazione dei messaggi

Tutti i file dei messaggi sono ubicati nella seguente directory del prodotto IBM WebSphere Business Integration Server Express:

```
DLMs\messages
```

Nota: In questo documento, le barre rovesciate (\) vengono utilizzate come convenzione per i percorsi di directory. Per le installazioni in ambiente Linux, sostituire le barre (/) con le barre rovesciate (\). Tutti i nomi percorso del prodotto WebSphere Business Integration Server Express sono relativi alla directory del sistema in cui è installato WebSphere Business Integration Server Express

E' possibile utilizzare tre tipi di file dei messaggi per generare i messaggi per una mappa:

- Un file dei messaggi specifico per mappe, *mapName_locale*.txt in cui *mapName* corrisponde al nome della mappa e *locale* corrisponde alla locale in cui è definita la mappa.

  I messaggi delle mappe appaiono nella scheda Messaggi di Map Designer Express e sono memorizzati come parte della definizione delle mappe nell'archivio. Quando si compila la mappa, Map Designer Express estrae il contenuto dei messaggi e crea (o aggiorna) il file dei messaggi per l'uso in fase di runtime. Il nome del file dei messaggi ha il seguente formato:

MapName_locale.txt

Esempio: se la mappa LegacyAddress_to_CwAddress è creata in una locale Inglese negli Stati Uniti, Map Designer Express crea il file dei messaggi denominato LegacyAddress_to_CwAddress_en_US.txt e lo colloca nella directory ProjectName\Maps\Messages. Dopo la distribuzione della mappa in InterChange Server Express, sarà collocata nella directory DLMs\messages.

- Il file di messaggi UserMapMessages.txt

  E' possibile aggiungere, in questo file, nuovi numeri messaggio che rientrano in un'intervallo "sicuro", come definito da WebSphere Business Integration Server Express (consultare Tabella 148). Inoltre, è possibile utilizzare un numero di messaggio già definito nel file di messaggi generico di WebSphere Business Integration Server Express (CwMapMessages.txt, descritto successivamente) e modificare il testo del messaggio esistente con il testo di scelta dell'utente. Poiché la ricerca del file UserMapMessaages.txt viene eseguita *prima* del file di messaggi WebSphere Business Integration Server Express, le modifiche aggiunte sostituiscono quei messaggi.
- Il file di messaggi generico WebSphere Business Integration Server Express, CwMapMessages.txt (che WebSphere Business Integration Server Express fornisce).

  Se la mappa utilizzata *non* indica uno degli altri due file di messaggi, deve indicare il file qui indicato. Tabella 148 elenca i numeri messaggio assegnati da WebSphere InterChange Server e che sono contenuti nel file di messaggi generico.

  Attenzione: *Non* cambiare il contenuto del file di messaggi generico di WebSphere Business Integration Server Express CwMapMessages.txt! Effettuare le modifiche a un messaggio generico copiandolo nel file di messaggi UserMapMessage.txt per personalizzarlo.

Il campo di questi file va da quelli specifici per mappe a quelli a scopo generale. I messaggi utilizzabili da una qualsiasi mappa si trovano in un file generico, fornito da WebSphere Business Integration Server Express. Gli altri due file consentono di personalizzare i messaggi per le mappe utilizzate, se necessario.

Importante: All'avvio, InterChange Server Express legge i file UserMapMessages.txt e CwMapMessages.txt presenti in memoria. Se si effettuano modifiche al file UserMapMessages.txt, è *necessario* riavviare InterChange Server Express affinché queste modifiche siano disponibili alle mappe.

Tabella 148. Messaggi CwMapMessages.txt

Numero messaggio	Testo del messaggio	Utilizzo del messaggio
5000	Mappatura - Il valore della chiave primaria nell'oggetto sorgente è nullo. L'esecuzione della mappa viene arrestata.	Utilizzato se la chiave primaria dell'oggetto sorgente ha un valore nullo. La verifica chiave primaria sorgente = nullo deve essere sempre eseguita prima che vengano richiamati i metodi di relazione che si basano sulla chiave primaria dell'oggetto sorgente. Se la chiave ha valore nullo, viene visualizzato l'errore e l'esecuzione della mappa viene arrestata.

Tabella 148. Messaggi CwMapMessages.txt (Continua)

Numero messaggio	Testo del messaggio	Utilizzo del messaggio
5001	Mappatura - RelationshipRuntimeException. L'esecuzione della mappa viene arrestata.	Utilizzato se `RelationshipRuntimeException` viene rilevato in una delle seguenti condizioni: • Blocchi di funzione – Generale/API/Relazione identità/Conserva relazione identità semplice – Generale/APIs/Relazione identità/Conserva relazione composta • Mappatura API – `maintainSimpleIdentityRelationship()` – `maintainCompositeRelationship()`
5002	Mappatura - CxMissingIDException. L'esecuzione della mappa viene arrestata.	Utilizzato se `CxMissingIDException` viene rilevato in una delle seguenti condizioni: • Blocchi di funzione – Generale/API/Relazione identità/Conserva relazione identità semplice – Generale/API/Relazione identità/Conserva relazione composta • Mappatura API – `maintainSimpleIdentityRelationship()` – `maintainCompositeRelationship()`
5003	Mappatura - mancano i dati nell'attributo {1}.	Utilizzato quando l'attributo sorgente ha valore nullo prima dell'utilizzo del blocco funzione Controllo chiave esterna (`foreignKeyLookup()`) o Riferimento incrociato chiave esterna (`foreignKeyXref()`). La verifica attributo sorgente = nullo deve essere sempre eseguita prima che vengano richiamati questi metodi di relazione. Se la chiave ha valore nullo, viene visualizzato l'errore e l'esecuzione della mappa viene arrestata.
5007	Mappatura - ForeignKeyLookup() di '{1}' con valore sorgente '{2}' non riuscito per la relazione '{3}' e partecipante '{4}' su programma di avvio '{5}'. L'esecuzione della mappa viene arrestata.	Utilizzato se l'attributo di destinazione ha valore nullo dopo l'utilizzo del blocco funzione Controllo chiave esterna (`foreignKeyLookup()`). E' stata arrestata l'esecuzione della mappa.
5008	Mappatura - ForeignKeyLookup() di '{1}' con valore sorgente '{2}' non riuscito per la relazione '{3}' e partecipante '{4}' su programma di avvio '{5}'. L'esecuzione della mappa è continuata.	Utilizzato se l'attributo di destinazione ha valore nullo dopo l'utilizzo del blocco funzione Controllo chiave esterna (`foreignKeyLookup()`). La mappa continua l'esecuzione.
5009	Mappatura - ForeignKeyXref() di '{1}' con valore sorgente '{2}' non riuscito per la relazione '{3}' e partecipante '{4}' su programma di avvio '{5}'. L'esecuzione della mappa viene arrestata.	Utilizzato se l'attributo di destinazione ha valore nullo dopo l'utilizzo del blocco funzione Riferimento incrociato chiave esterna (`foreignKeyXref()`). E' stata arrestata l'esecuzione della mappa.

Quando in una mappa si fa riferimento a un numero messaggio, i file dei messaggi vengono ricercati nel seguente ordine:

1. Il file di messaggi specifico della mappa *mapName_locale*.txt in cui *mapName* corrisponde al nome della mappa.

2. Il file UserMapMessages.txt viene ricercato.
3. Viene ricercato il file di messaggi generico CwMapMessages.txt di WebSphere Business Integration Server Express.

Tabella 149 mostra degli esempi di codice che indicano condizioni in cui ciascuno dei messaggi nel file CwMapMessages.txt può essere utilizzato.

Tabella 149. Esempi di codice per messaggi CwMapMessages.txt

Numero messaggio	Esempio codice
5000	`ObjContract.setVerb(ObjSAP_Contract.getVerb()); if (ObjSAP_Contract.get("ContractId") == null) { logError(5000); throw new MapFailureException( "Data in the primary key is missing"); }`
5001	`try { IdentityRelationship.maintainSimpleIdentityRelationship("Contract", "SAPCntr", ObjSAP_Contract, ObjContract, cwExecCtx); } catch (RelationshipRuntimeException e1) { logError(5001); throw new MapFailureException("RelationshipRuntimeException"); } catch (CxMissingIDException e2) { logError(5002); throw new MapFailureException("CxMissingIDException"); }`
5002	Vedere l'esempio di codice riportato sopra.
5003	`if (ObjSAP_Contract.get("CustomerId") == null) { logError(5003, "CustomerId"); throw new MapFailureException("CustomerId is null"); }`
5007	`try { IdentityRelationship.foreignKeyLookup ("Customer", "OracCust", ObjOracle_OrderImport, "customer_id", ObjOrder, "CustomerId", cwExecCtx); } catch (RelationshipRuntimeException e) { logWarning(e.toString()); } if (ObjOracle_OrderImport.get("customer_id") == null) { logError(5007, "customer_id", "CustomerId", "Customer", "OracCust", strInitiator); throw new MapFailureException("foreignKeyLookup() failed."); }`
5008	`try { IdentityRelationship.foreignKeyLookup ("Customer", "OracCust", ObjOracle_OrderImport, "customer_id", ObjOrder, "CustomerId", cwExecCtx); } catch (RelationshipRuntimeException e) { logWarning(e.toString()); } if (ObjOracle_OrderImport.get("customer_id") == null) { logError(5008, "customer_id", "CustomerId", "Customer", "OracCust", strInitiator); }`
5009	`try { IdentityRelationship.foreignKeyXref ("Customer", "OracCust", "CWCust", ObjOracle_OrderImport, "customer_id", ObjOrder, "CustomerId", cwExecCtx); } catch (RelationshipRuntimeException e) { logWarning(e.toString()); } if (ObjOracle_OrderImport.get("customer_id") == null({ logError(5009, "customer_id", "CustomerId", "Customer", "OracCust", strInitiator); throw new MapFailureException("foreignKeyXref() failed."); }`

Formato per i messaggi di mappe

Per garantire la congruenza dei messaggi, WebSphere Business Integration Server Express ha sviluppato un formato messaggio. Questa sezione descrive quel formato, includendo:

- "Formato del messaggio"
- "Parametri del messaggio"
- "Commenti" a pagina 522

Nota: E' necessario modificare il file dei messaggi specifico per mappe dalla scheda Messaggio in Map Designer Express e non direttamente. Map Designer Express sovrascrive tutte le modifiche personalizzate nel file dei messaggi specifico per mappe con i messaggi salvati nella mappa. Tuttavia, per i file di messaggi `UserMapMessages.txt` e `CwMapMessages.txt`, modificarli direttamente fornisce maggiore sicurezza.

Formato del messaggio

Il formato di ogni messaggio è:

NumMessaggio
Messaggio

Il numero del messaggio (*NumMessaggio*) e il messaggio stesso (*Messaggio*) devono trovarsi su diverse righe, con un carattere di ritorno a capo alla fine di ciascuna riga.

Esempio: I messaggi di mappa possono includere un messaggio identificato con il numero 23, il cui testo include due variabili segnaposto, contrassegnate come {1} e {2}, come illustrato in Figura 141..

```
23
ID cliente {1} non può essere modificato: {2}
```

Figura 141. Messaggio di esempio

Parametri del messaggio

Quando la mappa richiama un metodo che visualizza un particolare messaggio, trasferisce il numero di identificazione del messaggio al metodo e potenzialmente i parametri aggiuntivi. Il metodo utilizza il numero di identificazione per individuare il messaggio corretto nel file di messaggi e inserisce i valori dei parametri aggiuntivi nelle variabili segnaposto del testo del messaggio.

Non è necessario scrivere messaggi separati per ciascuna situazione possibile. Utilizzare i parametri che rappresentano i valori che cambiano al runtime. L'utilizzo dei parametri consente ad ogni messaggio di essere utilizzato in più situazioni e aiuta a limitare le dimensioni del file di messaggi.

Un paramento viene sempre visualizzato come un numero compreso tra parentesi graffe: { *numero*}. Per ciascun parametro occorre aggiungere il messaggio, inserire il numero tra parentesi graffe all'interno del testo del messaggio, nel modo seguente:

`testo messaggio {`*numero*`} altro testo messaggio.`

Esempio: fare riferimento nuovamente al messaggio 23 in Figura 141. Quando la mappa intende visualizzare o registrare questo messaggio, trasferisce al metodo appropriato il numero di identificazione del messaggio (23) e due parametri aggiuntivi:

- Il parametro 1 diventa il numero ID cliente (6701)
- Il parametro 2 diventa una variabile `String` contenente ulteriore testo di descrizione, come `maggiore della lunghezza massima`.

Il metodo individua il messaggio corretto, sostituisce i valori del parametro per i segnaposti del messaggio e visualizza o registra il seguente messaggio:

```
ID cliente 6701 non può essere modificato: maggiore
della lunghezza massima
```

Poiché il testo del messaggio utilizza la descrizione della voce mancante e il relativo ID come parametri, invece di includerli come stringhe codificate in modo protetto, è possibile utilizzare lo stesso messaggio per ciascuna coppia di ID cliente e di testo descrittivo.

Commenti

Far precedere ogni riga di commento di un file di messaggio dal segno cancelletto (#).

Esempio: un commento potrebbe essere simile al seguente:

```
# File di messaggi per la mappa dell'oggetto business Address.
```

Consiglio: è buona norma iniziare il file con una serie di righe di commento per formare una breve intestazione. Includere nei dati dell'intestazione il nome della mappa e determinate informazioni come il programma di creazione del file e la data di creazione.

Conservazione dei file

In un sito utente, un amministratore potrebbe impostare una procedura per il filtraggio dei messaggi di mappa, inviando una notifica a chi può risolvere il problema mediante e-mail o cerca persone. Ciò vuol dire che i numeri errore e il significato associato ai numeri resta lo stesso dopo il primo release di una mappa.

Consiglio: è possibile modificare il testo associato a dei numeri di errore, ma occorre evitare di modificare il testo o riassegnare i numeri dell'errore. Se si modifica il significato associato ai numeri di errore, occorre tenere traccia della modifica e inviare una notifica agli utenti della mappa.

Operazioni che utilizzano i file di messaggi

I file di messaggi contengono testo di messaggi utilizzati in diversi tipi di operazioni. Tabella 9 a pagina 25 elenca i tipi di operazioni che utilizzano i file di messaggi ed i metodi della classe `BaseDLM` che eseguono queste operazioni.

Tabella 150. Operazioni di generazione dei messaggi

Operazione	Blocco funzioni	Metodo
Presenza di eccezioni	Generale/API/Mappe/Eccezione/ Presenza eccezione mappa	`raiseException()`

Tabella 150. Operazioni di generazione dei messaggi (Continua)

Operazione	Blocco funzioni	Metodo
creazione log	Generale/Registro e Traccia/Registro ID informazione	`logInfo()`
	Generale/Registro e Traccia/Registro ID errore	`logError()`
	Generale/Registro e Traccia/Registro ID avviso	`logWarning()`
traccia	Generale/Registro e Traccia/Traccia/Traccia a livello	`trace()`

In questa sezione vengono descritte le operazioni di creazione dei messaggi che influiscono sull'esecuzione della mappa.

Presenza di eccezioni

Il metodo `raiseException()` presenta diverse forme. Una sintassi comunemente utilizzata è la seguente:

```
raiseException(String exceptionType,
   int messageNum, String param[,...])
```

Con questa sintassi è possibile avere da uno o tre parametri *param* `String`. Pertanto, è possibile avere fino a cinque parametri separati da virgole in un'esecuzione di chiamata a `raiseException()`.

Questo esempio identifica una nuova eccezione utilizzando il numero messaggio 23 e trasferisce due parametri al messaggio, il valore dell'ID cliente e una stringa:

```
raiseException(AttributeException, 23,
   fromCustomer.getString("CustomerID"),
   "greater than maximum length");
```

Figura 141 mostra il testo del messaggio 23 come appare nel file dei messaggi.

Registrazione dei messaggi

Una mappa esegue la registrazione di un messaggio per una qualsiasi azione di interesse per un amministratore. Per eseguire il log (registrare) di un messaggio, una mappa utilizza i `logInfo()`, `logWarning()` e `logError()` della classe `BaseDLM`. Ciascun metodo è associato ad un diverso livello di severità del messaggio.

Livelli di severità

Per registrare un messaggio, è necessario richiamare il metodo associato al livello di severità del messaggio. Tabella 151 elenca i livelli di severità ed i relativi metodi associati.

Tabella 151. Livelli dei messaggi

Livello di gravità	Metodo	Descrizione
Informazioni	`logInfo()`	Solo informativo. L'utente non deve intraprendere azioni.
Attenzione	`logWarning()`	Fornisce informazioni su un problema. Non utilizzare questo livello per problemi che l'utente deve risolvere.
Errore	`logError()`	Indica un problema grave che l'utente deve esaminare.

Utilizzo di un file di messaggi

Ogni mappa ha almeno un file di messaggi ad essa associato. Se una mappa non utilizza messaggi personalizzati, i relativi messaggi provengono dal file di sistema dei messaggi della mappa, `CwMapMessages.txt`. Una mappa utilizza messaggi personalizzati perché dispone di un file di messaggi specifico per mappe (generato dai messaggi immessi nella scheda Messaggi di Map Designer Express). Per ulteriori informazioni, consultare "Ubicazione dei messaggi" a pagina 517.

Quando una mappa registra un errore, il testo del messaggio di errore proviene dal file di messaggi della mappa.

Esempio: l'esempio riportato di seguito registra un messaggio di errore il cui testo è contenuto nel file di messaggi della mappa. Il testo del messaggio di errore 10 viene visualizzato come segue all'interno del file di messaggi:

```
10
Credit report error for {1}, {2}.
```

Il codice di registrazione del messaggio è simile al seguente:

```
logError(10, customer.get("LName"), customer.get("FName");
```

Quando viene eseguito il metodo `logError()`, il testo del messaggio 10 viene scritto nel file di log, con il cognome e nome del cliente sostituiti per i parametri 1 e 2.

Esempio il messaggio registrato per un cliente di nome John Davidson è simile al seguente:

```
Credit report error for Davidson, John.
```

Regole per una corretta registrazione (log) dei messaggi

Durante la creazione dei messaggi, prestare attenzione al modo in cui viene utilizzata la funzione di registrazione da parte degli amministratori.

Assegnazione dei livelli di severità: E' importante essere precisi quando si assegnano i livelli di errore ai messaggi. La funzione di notifica mediante e-mail di sistemi IBM invia un messaggio ad una specifica persona, di solito l'amministratore, quando rileva la creazione di messaggi di errori generici o irreversibili. Gli amministratori utilizzano tale funzione di notifica mediante e-mail IBM e inoltre possono collegare la funzione al cerca persona in modo da inviare una pagina quando si verifica un errore. L'accuratezza nell'assegnazione dei livelli di errore ai messaggi consente di ridurre il numero di messaggi critici.

Revisione dei messaggi: E' possibile revisionare il testo di un messaggio in qualsiasi momento, apportando variazioni o inserendo del testo. Tuttavia, una volta assegnato un numero di messaggio ad un determinato tipo di errore, è importante non riassegnare lo stesso numero. Molti amministratori fanno uso di file script per filtrare i messaggi e questi script interfacciano con i numeri assegnati ai messaggi. Pertanto, è importante che i numeri presenti nei file di messaggi non cambino significato. In caso contrario, gli utenti possono perdere dei messaggi o ricevere messaggi non attesi.

Quando utilizzare i messaggi informativi: E' possibile utilizzare il metodo `logInfo()` per creare messaggi temporanei per il debug personale. Tuttavia, accertarsi di rimuovere queste chiamate ai metodi di debug quando si termina l'attività di sviluppo.

Non utilizzare il metodo `logInfo()` per documentare l'operazione regolare della collaborazione. In caso contrario, nei file di log dell'amministratore saranno inseriti messaggi non importanti. Invece, utilizzare il metodo `trace()` per fornire all'amministratore informazioni dettagliate sulle operazioni di debug.

Aggiunta di messaggi di traccia

E' possibile aggiungere dei messaggi di traccia alla mappa durante l'esecuzione dell'istanza mappa in modo da generare una descrizione dettagliata delle operazioni svolte. I messaggi di traccia sono utili per il debug personale e per la risoluzione dei problemi sul posto da parte degli amministratori.

I messaggi di traccia si differenziano da quelli di registrazione (log); i messaggi di traccia vengono eliminati come impostazione predefinita mentre i messaggi di registrazione non possono essere eliminati. I messaggi di traccia sono generalmente più dettagliati e vengono consultati solo in determinate circostanze, ad esempio quando una persona in modo intenzionale configura il livello di traccia della mappa a un numero maggiore di zero. E' possibile inviare i messaggi di traccia e di registrazione a diversi file.

I messaggi di traccia possono essere aggiunti a una mappa per notificare le operazioni specifiche di quella mappa. Esistono alcuni tipi di informazioni che la mappa può immettere nel file di traccia:

- I valori chiave di un oggetto business nella fase in cui la mappa inizia o termina una particolare operazione di trasformazione.
- La scelta di una particolare sezione del percorso di esecuzione.

Assegnazione dei livelli di traccia

Ogni messaggio di traccia deve essere associato al livello di traccia compreso tra 1 e 5. Solitamente il livello di traccia è associato a un livello di dettaglio: messaggi al livello 1 contengono tipicamente meno dettagli dei messaggi a livello 2 che contengono meno dettagli rispetto a quelli al livello 3 e così via. Pertanto, se si attiva la funzione di traccia al livello 1, si possono consultare dei messaggi che contengono meno dettagli rispetto ai messaggi al livello 5. Tuttavia, è possibile assegnare i livelli in qualsiasi modo si ritiene più utile.

Consigli: di seguito sono riportati alcuni suggerimenti:

- E' possibile assegnare lo stesso livello a tutti i messaggi di traccia.
- E' possibile assegnare dei livelli di traccia in base al livello del dettaglio.
- E' possibile assegnare i livelli dei messaggi in base all'oggetto business interessato: i messaggi di traccia di livello 1 relativi a un determinato oggetto business, i messaggi di traccia di livello 2 relativi a un altro oggetto business e così via.

Quando si attiva la funzione di traccia per un particolare livello, i messaggi associati al livello specificato e quelli associati a tutti i livelli inferiori vengono visualizzati. Ad esempio, la funzione di traccia a livello 2 visualizza i messaggi associati sia al livello 2 che al livello 1.

Suggerimenti: Accertarsi di prendere nota dei livelli di traccia nella documentazione utilizzata, in modo che gli utenti conoscano quale livello utilizzare quando necessitano della funzione di traccia.

Creazione di un messaggio di traccia

Esempio: di seguito è riportato un esempio di un messaggio e della chiamata metodo che crea il messaggio. Il messaggio viene visualizzato nel file di messaggi nel seguente modo:

```
20
Begin transformation on {1} attribute: value = {2}
```

La chiamata metodo riceve il valore dell'attributo `LName` e poi utilizza il valore per sostituire il parametro nel messaggio. Il codice appare nella mappa come segue e il messaggio viene visualizzato quando l'utente imposta la funzione di traccia al livello 3:

```
trace(3, 20, "LName", customer.get("LName"));
```

Impostazione del livello di traccia

Figura 142 mostra la scheda Generale della finestra di dialogo Proprietà mappa in Map Designer Express. Per informazioni su come visualizzare la finestra Proprietà mappa, consultare "Specifica delle informazioni sulla proprietà della mappa" a pagina 62.) Da notare che è possibile impostare il livello di traccia per i messaggi di traccia in questa finestra.

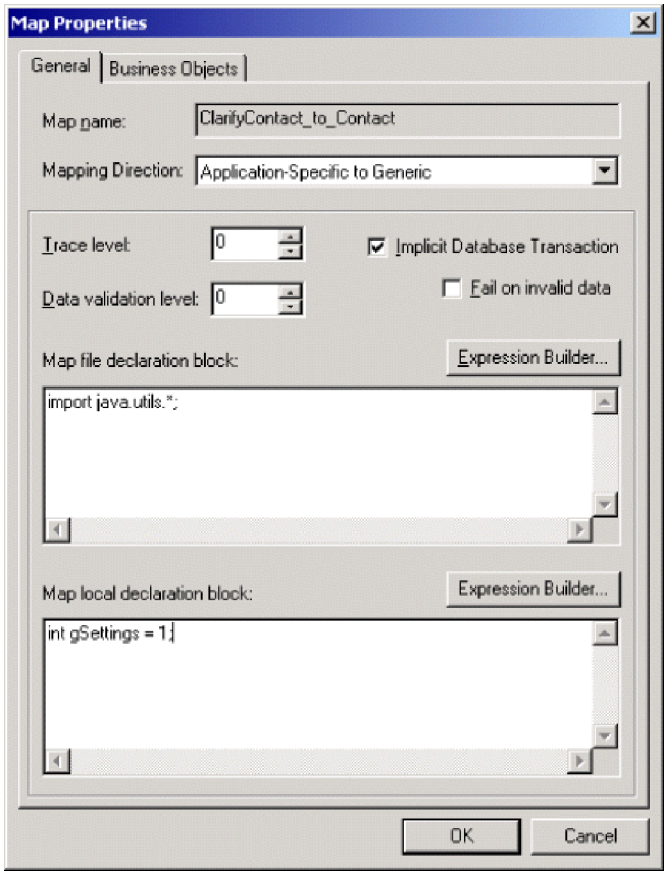


Figura 142. Livello di traccia per una mappa

Lo sviluppatore di mappe crea i livelli per cui è possibile richiedere la funzione di traccia delle mappe create, come descritto in "Assegnazione dei livelli di traccia" a pagina 525.

Nota: Se si modifica il livello di traccia per una mappa attivata, è necessario arrestare e riavviare la mappa prima che venga attivato il nuovo livello di traccia. Utilizzare il menu Componente di System Manager per arrestare e avviare una mappa.

Con l'impostazione del livello di traccia nella finestra Proprietà mappa di Map Designer Express, tale livello viene impostato per *tutte* le istanze mappe basate su questa definizione di mappa. Inoltre, è possibile impostare il livello di traccia per tutte le istanze mappe dalla finestra Proprietà mappa di System Manager.

Appendice. Proprietà dell'attributo

Tabella 152 elenca le proprietà degli attributi per definizioni di oggetti business.

Tabella 152. Proprietà dell'attributo

Proprietà	Descrizione
Nome	Un nome che descrive quale tipo di dati l'attributo contiene. Il nome deve essere inferiore o pari a 80 caratteri alfanumerici e caratteri di sottolineatura. Non può contenere spazi o determinati caratteri di punteggatura, come il punto, la parentesi sinistra ([) e destra (]), l'apostrofo o i doppi apostrofi.
Tipo	Il tipo dati dell'attributo. I tipi principali includono String, Boolean, Ddouble, Float, Integer e Date. Se l'attributo fornisce un oggetto business secondario, specificare il nome di una definizione di oggetto business secondario. Gli attributi che riportano oggetti business secondari sono definiti attributi composti.
IsKey	Un valore booleano, `true` o `false`, che specifica se si tratta un attributo chiave. Gli attributi chiave identificano in modo univoco un oggetto business creato dalla definizione. Ogni definizione di oggetto business dispone di almeno un attributo chiave.
IsForeignKey	Un valore booleano, `true` o `false`, che specifica se si tratta un attributo chiave esterna.
MaxLength	Un numero intero che rappresenta il numero massimo di byte che l'attributo può contenere. Per non specificare alcun limite, immettere zero (`0`).
AppSpecificInfo	Una stringa che fornisce informazioni sull'attributo per una particolare applicazione, come il nome di un campo in una tabella oppure il modulo che corrisponde all'attributo. I connettori utilizzano queste informazioni durante l'elaborazione dell'oggetto.
DefaultValue	Il valore da assegnare a questo attributo se non esistono valori di runtime.
IsRequired	Un valore booleano, `true` o `false`, che specifica se un valore è richiesto per questo attributo per creare un oggetto business.
ContainedObjectVersion	Il numero di versione della definizione dell'oggetto business secondario. IBM WebSphere System Manager visualizza tale valore sotto la dicitura Type Version.
Relazione	La relazione esistente tra l'oggetto business principale e l'oggetto business secondario. Nel release corrente, l'unica relazione valida è Containment.
cardinality	Il numero di oggetti business secondari che questo attributo indica. Se l'attributo indica un unico oggetto business secondario, il valore è 1. Se l'attributo può indicare molti oggetti business secondari, il valore è espresso con una lettera `n`.

Informazioni particolari

Queste informazioni sono state sviluppate per prodotti e servizi offerti negli Stati Uniti. È possibile che negli altri paesi l'IBM non offra i prodotti, le funzioni o i servizi illustrati in questo documento. Consultare il rappresentante locale IBM per informazioni sui prodotti e sui servizi disponibili attualmente nel proprio paese. Qualunque riferimento relativo a prodotti, programmi o servizi IBM non implica che solo quei prodotti, programmi o servizi IBM possano essere utilizzati. In sostituzione a quelli forniti dall'IBM possono essere utilizzati prodotti, programmi o servizi funzionalmente equivalenti che non comportino violazione dei diritti di proprietà intellettuale e di altri diritti dell'IBM. È comunque responsabilità dell'utente valutare e verificare la possibilità di utilizzare altri programmi e/o prodotti, fatta eccezione per quelli espressamente indicati dall'IBM. L'IBM può avere brevetti o domande di brevetto in corso relativi a quanto trattato nel presente documento. La fornitura di questa pubblicazione non implica la concessione di alcuna licenza su di essi. E' possibile inviare domande di licenza, per iscritto a:

IBM Director of Commercial Relations
IBM Europe
Schoenaicher Str. 220
D - 7030 Boeblingen
Deutschland

Per domande di autorizzazioni relative a informazioni DBCS, contattare IBM Intellectual Property Department nel proprio paese oppure inviare le domande a:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

Il seguente paragrafo non è valido per il Regno Unito o per tutti i paesi le cui leggi nazionali siano in contrasto con le disposizioni in esso contenute: L'INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE QUESTA PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA", SENZA ALCUNA GARANZIA, ESPLICITA O IMPLICITA, IVI INCLUSE EVENTUALI GARANZIE DI COMMERCIABILITA' ED IDONEITA' AD UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni; quindi la presente dichiarazione potrebbe essere non essere a voi applicabile. Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le informazioni incluse in questo documento vengono modificate su base periodica; tali modifiche verranno incorporate nelle nuove edizioni della pubblicazione. L'IBM si riserva il diritto di apportare miglioramenti e/o modifiche al prodotto o al programma descritto nel manuale in qualsiasi momento e senza preavviso. Tutti i riferimenti a siti Web non dell'IBM contenuti in questo documento sono forniti solo per consultazione e non rappresentano in alcun modo un'approvazione di tali siti Web. I materiali disponibile presso i siti Web non fanno parte di questo prodotto e l'utilizzo di questi è a discrezione dell'utente. Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente dall'IBM e diventeranno esclusiva della stessa. Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire: (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

Tali informazioni possono essere disponibili, in base ad appropriate clausole e condizioni, includendo in alcuni casi, il pagamento di una tassa. Il programma su licenza descritto in questo manuale e tutto il materiale su licenza ad esso relativo sono forniti dall'IBM nel rispetto delle condizioni previste dalla licenza d'uso. Tutti i dati contenuti in questa pubblicazione sono stati determinati in ambiente controllato. Pertanto, i risultati ottenuti in ambienti operativi diversi possono variare in modo considerevole. Alcune misurazioni potrebbero essere state fatte su sistemi di livello di sviluppo per cui non si garantisce che queste saranno uguali su tutti i sistemi disponibili. Inoltre, alcune misurazioni possono essere state stimate tramite estrapolazione. I risultati attuali possono quindi variare. Gli utenti del presente documento dovranno verificare i dati applicabili per i propri ambienti specifici. Le informazioni relative a prodotti non-IBM sono state ottenute dai fornitori di tali prodotti. L'IBM non ha verificato tali prodotti e, pertanto, non può garantirne l'accuratezza delle prestazioni. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti. Tutti le dichiarazioni riguardanti la direzione o le decisioni future di IBM sono soggette a variazione o ritiro senza preavviso e costituiscono solo degli obiettivi. Questa pubblicazione contiene esempi di dati e prospetti utilizzati quotidianamente nelle operazioni aziendali. Pertanto, può contenere nomi di persone, società, marchi e prodotti. Tutti i nomi contenuti nella pubblicazione sono fittizi e ogni riferimento a nomi e indirizzi reali è puramente casuale. LICENZA DI COPYRIGHT: Queste informazioni contengono programmi applicativi di esempio in lingua originale, che illustrano le tecniche di programmazione su diverse piatteforme operative. E' possibile copiare, modificare e distribuire queste esempi di programmi sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in modo conforme alle API (Application Programming Interface) a seconda della piattaforma operativa per cui gli esempi dei programmi sono stati scritti. Questi esempi non sono stati testati approfonditamente tenendo conto di tutte le condizioni possibili. La IBM, quindi, non può garantire o assicurare l'affidabilità, la praticità o il funzionamento di questi programmi. Se questa pubblicazione viene visualizzata in formato elettronico, è possibile che le fotografie e le illustrazioni a colori non vengano visualizzate.

Informazioni sull'interfaccia di programmazione

Le informazioni sull'interfaccia di programmazione, se fornite, sono designate per creare il software applicativo utilizzando questo programma. Le interfacce di programmazione di utilizzo generale consentono di scrivere il software applicativo che ottiene i servizi degli strumenti di questo programma. Tuttavia, queste informazioni possono contenere anche le informazioni sull'ottimizzazione, modifica e diagnosi. Le informazioni sulle diagnosi, le modifiche e sull'ottimizzazione vengono fornite come supporto per l'esecuzione del debug del software applicativo.

Avvertenza: Non utilizzare queste informazioni sulle diagnosi, le modifiche e sull'ottimizzazione come un'interfaccia di programmazione poiché sono soggette a cambiamenti.

Marchi e marchi di servizio

I seguenti termini sono marchi della International Business Machines Corporation negli Stati Uniti e/o in altri paesi: i5/OS
IBM
il logo IBM
AIX
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS
Microsoft, Windows, Windows NT e il logo Windows sono marchi di Microsoft Corporation negli Stati Uniti e/o in altri paesi. MMX, Pentium e ProShare sono marchi di Intel Corporation negli Stati Uniti e/o in altri paesi. Java e tutti i marchi basati su Java sono marchi di Sun Microsystems, Inc. negli Stati Uniti e/o in altri paesi. Linux è un marchio di Linus Torvalds negli Stati Uniti e/o in altri paesi. Altri nomi di società, di servizi e prodotti possono essere marchi di altre società.

WebSphere Business Integration Server Express e Express Plus includono il software sviluppato da Eclipse Project (http://www.eclipse.org/).

WebSphere Business Integration Server Express, Versione 4.4 e WebSphere Business Integration Server Express Plus, Versione 4.4

Indice analitico

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

Printed in Denmark by IBM Danmark A/S