IBM WebSphere Business
Integration Server Express Plus

Guida allo sviluppo di collaborazioni

Versione 44

IBM WebSphere Business
Integration Server Express Plus

Guida allo sviluppo di collaborazioni

Versione 44

Nota!
Prima di utilizzare queste informazioni ed il prodotto supportato, consultare “Informazioni particolari” a pagina 491.

22 aprile 2005

Questa edizione della documentazione si riferisce a 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

Il prodotto IBM(R) WebSphere(R) Business Integration Server Express Plus comprende i seguenti componenti: Interchange Server Express, Toolset Express, CollaborationFoundation ed un insieme di adattatori di integrazione software. Gli strumenti di Toolset Express consentono di creare, modificare e gestire i processi business. E' possibile scegliere tra diversi adattatori preconfigurati per i processi business che sono condivisi da più applicazioni. La maschera dei processi standard, CollaborationFoundation, consente di creare rapidamente processi personalizzati.

In questo manuale sono descritte le modalità di utilizzo di Process Designer Express per la creazione di collaborazioni, che sono parte dell'infrastruttura InterChange Server Express. Le collaborazioni sono programmi che contengono la logica business per l'integrazione delle applicazioni.

Se non altrimenti specificato, le informazioni presenti in questo manuale si applicano a IBM WebSphere Business Integration Server Express Plus. Il termine "WebSphere Business Integration Server Express" fa riferimento a IBM WebSphere Business Integration Server Express Plus.

A chi è rivolto questo manuale

Questo documento si rivolge ai clienti, consulenti o rivenditori che creano o modificano collaborazioni. Prima di iniziare, è importante comprendere i concetti illustrati nel capitolo di panoramica di System Implementation Guide.

Per sviluppare una collaborazione, è necessario conoscere i principi e le procedure standard di programmazione. Inoltre, lo sviluppo di collaborazioni richiede una conoscenza di base del linguaggio di programmazione Java®. L'API di collaborazione si basa sul linguaggio di programmazione Java e gestisce le operazioni più comuni svolte dalla maggior parte delle collaborazioni, quali la gestione degli oggetti business. Se l'utente ha una conoscenza dei concetti di programmazione, gli esempi riportati in questo manuale gli consentono di scrivere semplici collaborazioni anche senza conoscere Java.

Se l'utente ha esperienza nell'analisi di business e delle applicazioni, ma non ha mai scritto un programma, oppure se la collaborazione richiede l'esecuzione di altre operazioni oltre a quelle comprese nell'ambito dell'API di collaborazione, si consiglia di lavorare insieme ad un programmatore Java.

Obiettivo di questo manuale

Il processo globale di sviluppo delle collaborazioni è costituito da numerose fasi e coinvolge più persone: esperti dell'applicazione, analisti business e programmatori. Dopo aver analizzato i problemi di integrazione di un'applicazione, il team di sviluppo della collaborazione realizza il processo business richiesto all'interno del sistema di integrazione WebSphere. Il team normalmente inizia con la creazione di un diagramma di flusso che viene poi trasformato in una collaborazione.

Questo manuale presuppone che si inizi a lavorare in base ad una specifica, un diagramma di flusso o un disegno su carta. Non sono illustrate le fasi di analisi dei processi business, lo sviluppo di connettori o la progettazione degli oggetti business.

Nota: In questo documento, le barre rovesciate (\) vengono utilizzare come convenzione per i percorsi di directory. Per le installazioni su sistemi UNIX, è necessario sostituire i caratteri barra retroversa con le barre (/). Tutti i nomi di percorso sono relativi alla directory di sistema in cui è installato il prodotto.

Come utilizzare questo manuale

Questo manuale ha la seguente struttura:

Parte I: Introduzione

Capitolo 1, "Introduzione allo sviluppo di collaborazioni", a pagina 3	Fornisce una panoramica delle collaborazioni e dell'ambiente di sviluppo di collaborazioni.
Capitolo 2, "Panoramica su Process Designer Express", a pagina 15	Fornisce informazioni dettagliate sull'interfaccia Process Designer Express.

Parte II: Creazione di una maschera di collaborazione

Capitolo 3, "Progettazione di una collaborazione", a pagina 29	Fornisce informazioni utili nella fase di progettazione dello sviluppo di collaborazioni.
Capitolo 4, "Creazione di una maschera di collaborazione", a pagina 83	Illustra come creare la definizione di una maschera di collaborazione.
Capitolo 5, "Utilizzo dei diagrammi di attività", a pagina 113	Descrive le modalità di utilizzo dei simboli e degli altri componenti per realizzare un diagramma di attività.
Capitolo 6, "Utilizzo di Activity Editor", a pagina 159	Descrive le modalità di utilizzo di Activity Editor per aggiungere in modo grafico le definizioni di una maschera di collaborazione.
Capitolo 7, "Gestione delle eccezioni", a pagina 179	Descrive le modalità di implementazione della gestione eccezioni in una maschera di collaborazione.
Capitolo 8, "Opzioni dell'area di lavoro e del layout", a pagina 197	Descrive alcune delle opzioni per l'area del diagramma e per la disposizione dei simboli in un diagramma di attività.
Capitolo 9, "Esempi e suggerimenti di codifica", a pagina 207	Contiene frammenti di codice e suggerimenti per illustrare le modalità di esecuzione delle più comuni operazioni.
Capitolo 10, "Creazione del file di messaggi", a pagina 247	Illustra le modalità di impostazione del file necessario alle collaborazioni per contenere i messaggi di log e di traccia.

Parte III: Blocchi funzione supportati

Capitolo 11, "Blocchi funzione di oggetto business", a pagina 255	Descrive i blocchi funzione utilizzati per gestire gli oggetti business.
Capitolo 12, "Blocchi funzione di vettore oggetto business", a pagina 281	Descrive i blocchi funzione utilizzati per gestire i vettori di oggetti business.

	Capitolo 14, “Blocchi funzione di connessione database”, a pagina 309	Descrive i blocchi funzione utilizzati per gestire le connessioni ai database.
	Capitolo 18, “Blocchi funzione di data”, a pagina 335	Descrive i blocchi funzione utilizzati per gestire le date.
	Capitolo 19, “Blocchi funzione di registro e traccia”, a pagina 341	Descrive i blocchi funzione utilizzati per gestire il log e la traccia.
	Capitolo 20, “Blocchi funzione di stringa”, a pagina 349	Descrive i blocchi funzione utilizzati per gestire gli oggetti String.
	Capitolo 21, “Blocchi funzione di utilità”, a pagina 357	Descrive i blocchi funzione utilizzati per gestire gli oggetti Vector e gli errori.
Parte IV: Riferimenti API di collaborazione		
	Capitolo 22, “Classe BaseCollaboration”, a pagina 367; Capitolo 23, “Classe BusObj”, a pagina 391; Capitolo 24, “Classe BusObjArray”, a pagina 411; Capitolo 25, “Classe CwDBConnection”, a pagina 425; Capitolo 26, “Classe CwDBStoredProcedureParam”, a pagina 437; Capitolo 27, “Classe CxExecutionContext”, a pagina 441 Capitolo 29, “Classe CollaborationException”, a pagina 449; Capitolo 31, “Classe Filter”, a pagina 461; Capitolo 32, “Classe Globals”, a pagina 467; Capitolo 33, “Classe SmartCollabService”, a pagina 471;	Contiene le pagine di riferimento per i metodi delle classi nell’API di collaborazione.
Glossario		Definisce i termini utilizzati nel manuale.

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 il nome di un comando, un nome file, informazioni immesse dall'utente o le informazioni visualizzate sullo schermo.
grassetto	Indica un termine nuovo la prima volta che viene visualizzato.
corsivo	Indica un nome variabile o un riferimento incrociato. Quando si visualizza un documento come file PDF, i riferimenti incrociati appaiono in blu e in corsivo. E' possibile fare clic su un riferimento incrociato per passare alla informazioni di destinazione.
`courier corsivo`	Indica un nome variabile all'interno di un testo letterale.
`courier con riquadro`	Separa un frammento di codice dal resto del testo.
testo blu	Un riquadro blu, visibile solo quando si visualizza il manuale in linea, indica un collegamento ipertestuale ad un riferimento incrociato. Fare clic all'interno del riquadro per passare all'oggetto del riferimento.
{ }	Su una riga della sintassi, le parentesi graffe racchiudono una serie di opzioni di cui ne va selezionata solo una.
`[ ]`	In una riga di sintassi, le parentesi quadre racchiudono un parametro facoltativo.
`...`	In una riga di sintassi, i puntini sospensivi indicano una ripetizione del parametro precedente. Ad esempio, `option[,...]` significa che è possibile inserire più opzioni separate da virgola.
ProductDir	Rappresenta la directory in cui viene installato il prodotto. Per IBM WebSphere Business Integration Server Express, la directory predefinita del prodotto è IBM\WebSphereICS. In ambiente IBM WebSphere Business Integration Adapters, la directory predefinita del prodotto è WebSphereAdapters.

Sommario degli aggiornamenti

Questa sezione fornisce un riepilogo delle funzioni nuove e modificate di IBM WebSphere Business Integration Server Express Plus e degli strumenti associati per lo sviluppo di collaborazioni. Queste funzioni e questi programmi sono illustrati in dettaglio in questo manuale.

Novità nel rilascio 4.4

Per il rilascio 4.4, sono state apportate le seguenti modifiche al presente manuale:

- Sono supportati gli script bidirezionali per la comunicazione con un ambiente esterno, utilizzando Access Interface. Consultare "Abilitazione delle collaborazioni per script bidirezionali tramite Access Interface" a pagina 74.
- Sono state aggiunte le regole business (Business Rule) per consentire di modificare in modo rapido e semplice il comportamento di runtime. Consultare "Regole di business" a pagina 126.
- L'analisi di oggetti business è stata utilizzata nelle regole business per definire il punto in cui una condizione business viene soddisfatta. Consultare "Analisi di oggetto business e regole di business" a pagina 126, "Utilizzo delle analisi di oggetti business" a pagina 124 e "Esempi di utilizzo delle regole di business" a pagina 127.
- E' stata migliorata la descrizione dei blocchi funzione. Consultare Parte 3, "Blocchi funzione supportati", a pagina 253.
- E' stato aggiunto il supporto degli script bidirezionali. Consultare Capitolo 3, "Progettazione di una collaborazione" e Capitolo 28, "Classe CxBiDiEngine", a pagina 445.
- E' stato migliorato l'Activity Editor. Consultare Capitolo 6, "Utilizzo di Activity Editor", a pagina 159.
- E' stata aggiunta la capacità di allineare automaticamente gli scenari in Process Designer Express. Consultare "Allineamento degli scenari" a pagina 204.
- E' stata aggiunta la capacità di configurare la sequenza o l'isolamento eventi. Consultare "Sequenza degli eventi" a pagina 58 e "Isolamento dell'evento" a pagina 58.
- E' stato migliorato il supporto per il ripristino degli eventi non riusciti. Consultare "Gestione degli eventi non riusciti" a pagina 78, "dropFailedEvent()" a pagina 368, "queryFailedEvents()" a pagina 380 e "resubmitFailedEvent()" a pagina 385.
- E' stato migliorato il supporto per i tentativi di riconnessione ad un database. Consultare "Configurazione della funzione di flessibilità della connessione al database" a pagina 75.
- Non è più supportata la classe StateManagement. I riferimenti a questa classe sono stati eliminati da questo manuale.

Novità nel rilascio 4.3.1

Questo 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. Introduzione

Capitolo 1. Introduzione allo sviluppo di collaborazioni

Benvenuti in Process Designer Express e nel processo di sviluppo di collaborazioni. Process Designer Express è un potente strumento per lo creazione di modelli e la generazione del codice, con il quale è possibile creare collaborazioni, programmi per la creazione di processi business aziendali che coinvolgono più applicazioni.

Questo capitolo è una introduzione al processo di sviluppo di collaborazioni e agli strumenti utilizzati per lo sviluppo di collaborazioni.

Definizione di collaborazione

Le collaborazioni sono moduli software che descrivono i processi business e che sono eseguiti all'interno di IBM WebSphere Business Integration Server Express. Questi processi business sono programmi che contengono la logica business per l'integrazione delle applicazioni. Una collaborazione può eseguire diversi tipi di operazioni Java. Tuttavia, spesso le collaborazioni eseguono operazioni su oggetti business, quali:

- Ottenimento e gestione di uno o più valori nell'evento di attivazione
- Invio di un oggetto business come richiesta ad un'applicazione, affinché l'applicazione crei, elimini o aggiorni un'entità specificata
- Invio di una richiesta ad un'applicazione per richiamare un'entità

Come mostrato nella Tabella 1, una collaborazione è un'entità costituita da due parti, una definizione di repository ed un oggetto runtime.

Tabella 1. Parti di una collaborazione

Entità repository	Oggetto runtime
Maschera di collaborazione	Oggetto di collaborazione

Quando si installa una collaborazione, si installa una *maschera di collaborazione*. Una maschera di collaborazione contiene tutta la logica di esecuzione della collaborazione, ma non è eseguibile. Per eseguire una collaborazione, è necessario prima creare un *oggetto di collaborazione* dalla maschera. L'oggetto di collaborazione diventa eseguibile dopo essere stato configurato collegandolo ai connettori o ad altri oggetti di collaborazione e specificando altre proprietà di configurazione.

Nota: Per un'introduzione alle modalità di funzionamento delle collaborazioni come componenti del sistema IBM WebSphere Business Integration Server Express, consultare *WebSphere Business Integration Server Express System Implementation Guide*. Questa sezione illustra in particolare la definizione di una collaborazione in base alle modalità di sviluppo.

In questo manuale, si utilizza il termine collaborazione per indicare sia le maschere di collaborazione che gli oggetti di collaborazione, a meno che non sia necessario distinguere tra una maschera e un oggetto.

Maschere di collaborazione

La definizione di una collaborazione inizia dalla maschera di collaborazione. Una maschera di collaborazione è una specifica della logica interna della collaborazione.

Una maschera di collaborazione viene definita con lo strumento Process Designer Express, che memorizza le informazioni opportune in System Manager. Lo sviluppo di una maschera di collaborazione comprende le seguenti fasi:

- Creazione della maschera di collaborazione
- Generazione delle parti della maschera di collaborazione
- Compilazione della maschera di collaborazione

Creazione di una maschera di collaborazione

Quando si sviluppa una collaborazione, si utilizza uno strumento denominato Process Designer Express per realizzare una maschera di collaborazione. Process Designer Express fornisce una interfaccia utente grafica (GUI) facile da utilizzare che evita la scrittura di gran parte del codice normalmente richiesto per lo sviluppo di un programma. Questa interfaccia rende più facile la dichiarazione delle variabili, la scrittura di frammenti di codice ed altre attività. IBM WebSphere Business Integration Server Express fornisce inoltre alcune maschere di collaborazione generiche per facilitare il processo di sviluppo. L'utilizzo di Process Designer Express rende più facile lo sviluppo di una maschera di collaborazione rispetto alla scrittura di un programma nei linguaggi di programmazione standard. Tuttavia, il risultato finale dello sviluppo di una collaborazione è un programma, sotto forma di una classe Java.

Process Designer Express salva le informazioni sulla maschera di collaborazione in System Manager fino alla distribuzione. Dopo la distribuzione di una collaborazione, tali informazioni sono disponibili in InterChange Server Express; le informazioni sono richiamate quando una collaborazione riceve un evento di attivazione. Per ulteriori informazioni su Process Designer Express, consultare Capitolo 2, "Panoramica su Process Designer Express", a pagina 15.

Generazione di una maschera di collaborazione

In Process Designer Express, la generazione di una maschera di collaborazione richiede un processo di sviluppo a due livelli:

- Diagrammi di attività
  - La creazione di diagrammi di attività, che sono descrizioni grafiche simboliche del processo business e del relativo flusso.
  - L'utilizzo di Activity Editor per implementare ulteriori dettagli del processo business.

  La compilazione della maschera converte i diagrammi e il codice associato in una classe Java eseguibile.
- Messaggi — La definizione di messaggi, che contengono il testo utilizzato nella registrazione di log e traccia e nella generazione di eccezioni.

  Quando si compila una maschera, il contenuto del messaggio viene scritto in un file di messaggi in System Manager. Dopo la distribuzione della collaborazione, il file di messaggi viene anche archiviato in InterChange Server Express, in modo da potervi accedere durante il runtime. Per ulteriori informazioni, consultare Capitolo 10, "Creazione del file di messaggi", a pagina 247.

Creazione dei diagrammi di attività: Una maschera di collaborazione è costituita da due componenti:

- *scenari*, che specificano l'insieme di azioni.

  Quando si sviluppa una maschera di collaborazione, si divide prima la logica business della collaborazione in uno o più scenari.

  Ogni maschera di collaborazione è costituita da una o più unità denominate scenari. Uno scenario specifica esattamente la risposta della collaborazione di

fronte ad una particolare attivazione del flusso. Lo scenario è simile ad un metodo in quanto descrive le azioni intraprese dalla collaborazione.

L'utente può scegliere se creare più scenari o inserire tutta la logica della collaborazione in un unico scenario.

- *diagrammi di attività*, che descrivono le azioni utilizzando frammenti di codice che rappresentano le singole azioni.

  Per ogni scenario, viene creato un diagramma di attività che descrive graficamente il processo dello scenario. Un diagramma di attività è un'implementazione grafica dello scenario, che comprende azioni, flusso di esecuzione e chiamate esterne. I diagrammi di attività si basano su UML (Unified Modeling Language), una notazione standard per la creazione di modelli dei processi business. L'utilizzo della programmazione visuale nei diagrammi rende più facile la creazione di uno scenario e riduce la quantità di codice effettivo.

  Le diverse fasi del diagramma di attività rappresentano le singole azioni o frammenti di codice.

  Ogni scenario contiene almeno un diagramma di livello principale, che costituisce il punto di ingresso dello scenario durante l'esecuzione e contiene il flusso logico complessivo dello scenario. I *diagrammi secondari* possono suddividere i dettagli della logica di scenario in più livelli nidificati.

  Un diagramma di attività è simile ad un diagramma di flusso. Tuttavia, a differenza di un diagramma di flusso, il diagramma di attività può creare il codice Java eseguibile rappresentato dal diagramma di attività.

La Figura 1 è un esempio di diagramma di attività.

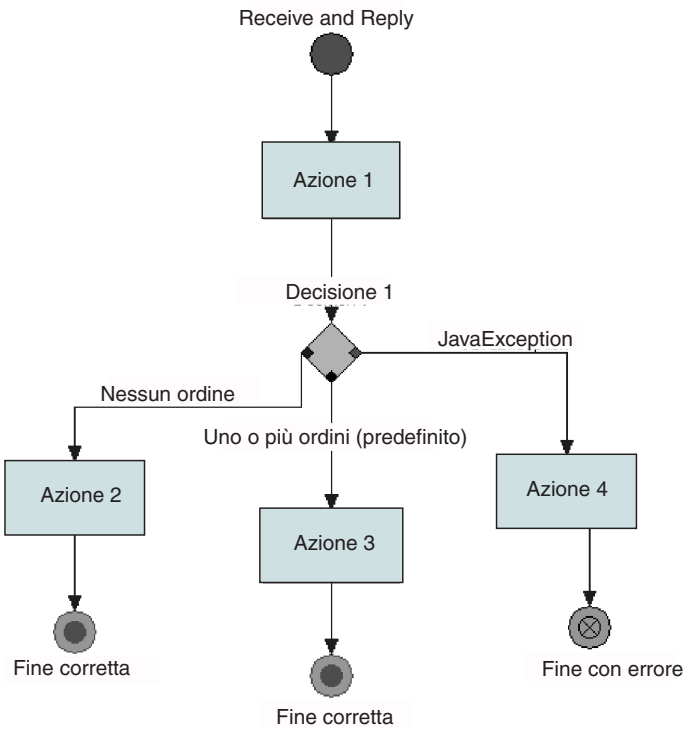


Figura 1. Diagramma di attività

L'unità di base in un diagramma di attività è una *azione*, rappresentata da un rettangolo. Un'azione specifica un'unità di lavoro nella collaborazione e viene utilizzata per creare e archiviare i frammenti di codice Java.

I diagrammi di attività rappresentano tutti i possibili comportamenti al momento dell'esecuzione. Il diagramma di attività nella Figura 1 presenta più percorsi di esecuzione. Un *percorso di esecuzione* è rappresentato da un insieme di simboli e collegamenti che passano dal simbolo di inizio in alto ad uno dei simboli di fine in basso. Un simbolo che presenta più collegamenti in uscita viene detto nodo decisione: è a questo punto che la collaborazione decide di seguire un percorso logico invece di un altro.

Implementazione di frammenti di codice Java: Ogni azione contiene un *frammento di codice* nel linguaggio di programmazione Java, al quale uno sviluppatore può aggiungere ulteriore codice. Process Designer Express incorpora il frammento di codice nel codice della maschera di collaborazione generato ed esegue il codice generato quando viene eseguito l'oggetto di collaborazione, come parte del flusso di collaborazione.

Se necessario, è possibile aggiungere un frammento di codice Java. L'utente può:

- Scrivere il proprio codice.

  Molto del codice in una maschera di collaborazione è costituito da chiamate all'*API di collaborazione* di InterChange Server Express. Tuttavia, il frammento di codice non è limitato alle sole chiamate all'API di collaborazione, può contenere qualsiasi tipo di codice Java. Per aggiungere una propria definizione di attività o per personalizzare una definizione di attività esistente, è possibile:
  - Utilizzare Activity Editor, un'interfaccia GUI che facilita le operazioni di aggiunta delle definizioni di attività. Consente di scrivere codice Java tradizionale o di creare modelli grafici della logica di programmazione con i blocchi funzione.
  - Immettere direttamente il frammento di codice Java nella finestra Frammento di codice nella finestra di dialogo Proprietà azione del nodo azione. (Notare che è necessario impostare le preferenze utente di Process Designer Express per abilitare questa funzione).
- Importare codice da un'altra classe Java.

  E' possibile importare pacchetti di classi Java esterni nella collaborazione ed utilizzare i relativi metodi all'interno delle azioni.

  Nota: La classe generata da Process Designer Express deve essere eseguita nel contesto di esecuzione di InterChange Server Express. Anche se è possibile importare o scrivere il proprio codice Java, il codice dovrebbe estendere un diagramma di attività. L'esecuzione di altre operazioni che possono alterare il flusso di esecuzione o favorire un uso eccessivo di risorse è sconsigliato.

Compilazione di una maschera di collaborazione

Una volta completata la definizione di una maschera di collaborazione, ovvero dopo aver definito gli scenari, generato i diagrammi di attività, personalizzato i frammenti e creato il file di messaggi, si effettua la compilazione dell'intera maschera. Il processo di compilazione della collaborazione crea tre tipi di file (.class, .java e .txt) che sono utilizzati dal runtime della collaborazione.

Quando si compila una collaborazione, questi file sono creati automaticamente nel progetto ICL (Integration Component Libraries) in System Manager. Quando si distribuisce l'oggetto di collaborazione al server, questi file sono spostati nella

directory *productDir*\collaborations. La Tabella 2 descrive i file e mostra dove sono collocati dopo la compilazione e la distribuzione.

Tabella 2. File di collaborazione

Tipo di file	Descrizione	Percorso
.class	Il file di classe eseguibile finale prodotto da Process Designer Express durante la compilazione	Dopo la compilazione: *ICLProject*\Templates\Classes Dopo la distribuzione: classes\UserCollaborations
.java	Il file del codice di origine prodotto da Process Designer Express durante la generazione del codice	Dopo la compilazione: *ICLProject*\Templates\Src Dopo la distribuzione: classes\UserCollaborations
.txt	Il file di messaggi che contiene il testo di tutti i messaggi aggiunti alla maschera durante lo sviluppo	Dopo la compilazione: *ICLProject*\Templates\messages Dopo la distribuzione: \messages

Importante

Effettuare tutte le modifiche ai messaggi tramite Process Designer Express; non effettuare mai direttamente modifiche al file contenente il testo dei messaggi. Dopo la distribuzione di una collaborazione, questo file viene utilizzato dall'ambiente runtime; la modifica diretta di tale file può provocare errori.

Dopo aver compilato una maschera di collaborazione, è possibile utilizzare System Manager per creare oggetti di collaborazione e distribuire questi oggetti e la maschera a InterChange Server Express.

Oggetti di collaborazione

Anche se una maschera di collaborazione contiene tutta la logica di esecuzione della collaborazione, è necessario effettuare le seguenti operazioni prima di poter eseguire la collaborazione:

1. Creare un oggetto di collaborazione.

   Un oggetto di collaborazione è un'istanza di una maschera di collaborazione. Per creare un oggetto di collaborazione si utilizza System Manager.

2. Configurare l'oggetto di collaborazione.

   L'oggetto di collaborazione diventa eseguibile dopo essere stato configurato. Per configurare l'oggetto di collaborazione, lo si collega ai connettori o ad altri oggetti di collaborazione e si specificano altre proprietà di configurazione.

   Il processo di specifica degli oggetti con i quali interagisce l'oggetto di collaborazione viene definito *binding*. Un oggetto di collaborazione può essere collegato ad uno dei seguenti oggetti:

   - Un connettore, altri oggetti di collaborazione o client di accesso con i quali l'oggetto di collaborazione interagisce. Quando si collega l'oggetto di collaborazione e si specificano i valori delle proprietà di configurazione, l'oggetto di collaborazione diventa eseguibile.

Per ulteriori informazioni sull'utilizzo di System Manager per la creazione e configurazione di oggetti di collaborazione, consultare *System Implementation Guide*.

La Figura 2 illustra la creazione di un oggetto di collaborazione denominato OrderStat dalla maschera OrderStatus.

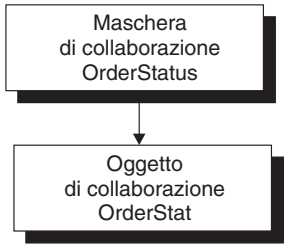


Figura 2. Creazione di un oggetto di collaborazione

La maschera di collaborazione OrderStatus è stata creata con due porte definite, attraverso le quali la collaborazione prevede di comunicare con gli oggetti di origine e di destinazione. Durante la configurazione dell'oggetto di collaborazione OrderStat, questo viene collegato a due oggetti esterni. La Figura 3 mostra l'oggetto di collaborazione OrderStat collegato al connettore SAP e al connettore Vantive.

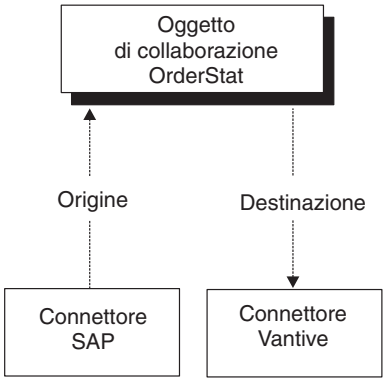


Figura 3. Oggetto di collaborazione collegato ai connettori

Dopo che un oggetto di collaborazione è collegato e configurato, è possibile utilizzare System Manager per il test e la distribuzione nell'ambiente runtime. Un oggetto di collaborazione può essere configurato per essere eseguito in un unico thread o in più thread, in cui ognuno gestisce un evento di attivazione. Per la gestione contemporanea di più eventi di attivazione, l'oggetto di collaborazione deve essere eseguito in modalità multi-thread.

Collaborazioni come LLBP (Long-Lived Business Process)

Gli oggetti di collaborazione possono essere distribuiti come LLBP (Long-Lived Business Process), consentendo la comunicazione asincrona tra processi business. Di conseguenza, i processi business possono durare a lungo. In un LLBP (Long-Lived Business Process), il contesto del flusso di eventi (che include una maschera globale o le variabili di porta e di oggetto business create in Process Designer Express, oltre a informazioni sul flusso di lavoro runtime) viene mantenuto durante una chiamata di servizio.

E' possibile specificare i valori di timeout per le chiamate di servizio asincrone in ingresso e sincrone, per definire ulteriormente i parametri di un LLBP.

Se si prevede di utilizzare un oggetto di collaborazione come LLBP, è necessario configurare la maschera di collaborazione in modo conforme. Prima di generare una maschera di collaborazione, consultare le informazioni presenti in "Progettazione di processi business di lunga durata" a pagina 55. Dopo aver progettato la maschera, consultare Capitolo 4, "Creazione di una maschera di collaborazione", a pagina 83 per informazioni sulle attività di configurazione specifiche richieste per il supporto di LLBP.

Collaborazioni e sistema WebSphere Business Integration Server Express

Il sistema WebSphere Business Integration Server Express utilizza un oggetto business per trasportare dati e richieste di azioni da un'applicazione ad un'altra. Una collaborazione inizia l'esecuzione quando uno scenario all'interno di un oggetto di collaborazione riceve un particolare oggetto business e un'azione (istruzione). Questa combinazione di oggetto business e di un istruzione, la cui ricezione da parte della collaborazione attiva l'esecuzione di uno scenario, è detta *attivazione flusso*. Durante la progettazione della maschera di collaborazione, lo sviluppatore della collaborazione specifica gli oggetti business e le istruzioni che agiscono come attivazioni di flusso per ciascuno scenario. Durante la configurazione dell'oggetto di collaborazione, la porta di ingresso della collaborazione viene collegata ad una particolare origine per l'attivazione flusso. Il tipo di origine che invia l'attivazione flusso alla porta in ingresso determina il tipo di attivazione flusso ricevuto dalla collaborazione.

Come mostrato nella Tabella 3, l'attivazione flusso può essere di diversi tipi, in base all'origine dell'oggetto business in ingresso:

Tabella 3. Tipo di attivazione flusso

Attivazione flusso	Origine dell'oggetto business in ingresso
Evento di attivazione	Connettore o altra collaborazione
Chiamata di accesso di attivazione	Client di accesso (attraverso Server Access Interface in InterChange Server Express)

Nota: Un client di accesso è un processo esterno che può richiedere l'esecuzione di collaborazioni mediante l'API Server Access Interface. Per ulteriori informazioni, consultare il manuale *Access Development Guide*.

Dal momento che i connettori sono la più comune origine delle attivazioni flusso, il termine "eventi di attivazione" viene spesso utilizzato per fare riferimento agli oggetti business in ingresso di una collaborazione. Ad esempio, la finestra Definizione maschera presenta una scheda Porte ed eventi di attivazione. Da questa scheda, è possibile definire le porte di collaborazione ed assegnare gli eventi di attivazione ai relativi scenari. Tuttavia, anche se i titoli della scheda e della tabella associata includono il termine "eventi di attivazione", questa scheda gestisce l'assegnazione di entrambi i tipi di attivazione flusso: eventi di attivazione o chiamate di accesso di attivazione. Se lo scenario riceve l'oggetto business da un connettore, la sua attivazione flusso è un evento di attivazione (come indicato dal nome della scheda). Se lo scenario riceve l'oggetto business da un client di accesso, la sua attivazione flusso è una chiamata di accesso di attivazione. In questo caso, si deve comunque utilizzare la tabella Porte ed eventi di attivazione per assegnare una chiamata di accesso di attivazione ad uno scenario.

Il tipo di attivazione flusso per la collaborazione non è effettivamente determinato fino a quando non viene configurata la porta dell'oggetto di collaborazione:

- Porta interna — quando la porta è collegata ad un connettore, riceve l'oggetto business come evento di attivazione.
- Porta esterna — quando la porta è collegata ad un client di accesso, riceve l'oggetto business come chiamata di accesso di attivazione.

Per ulteriori informazioni sulle modalità di configurazione di un oggetto di collaborazione, consultare *System Implementation Guide*. Per ulteriori informazioni sulla scheda Porte ed eventi di attivazione della finestra Definizione maschera, consultare "Definizione di porte ed eventi di attivazione (scheda Porte ed eventi di attivazione)" a pagina 97.

Strumenti per lo sviluppo di collaborazioni

La piattaforma per lo sviluppo di collaborazioni è Windows 2003. Le collaborazioni sono scritte in Java. Nella Tabella 4 sono riportati gli strumenti forniti da InterChange Server Express per lo sviluppo di collaborazioni.

Tabella 4. Strumenti per lo sviluppo di collaborazioni

Strumento	Descrizione	Per ulteriori informazioni
Process Designer Express	Strumento grafico di supporto per lo sviluppo della maschera di collaborazione.	"Process Designer Express"
API di collaborazione WebSphere Business Integration Server Express	Un insieme di classi Java con le quali è possibile personalizzare il codice di collaborazione generato. (Si può accedere ai metodi inclusi nell'API anche attraverso i blocchi funzione dell'Activity Editor).	"API di collaborazione" a pagina 10
System Manager	Strumento che fornisce le finestre grafiche per creare e configurare un oggetto di collaborazione.	"System Manager" a pagina 12
Integrated Test Environment (Test Connector)	Un gruppo di strumenti utilizzato per il test dei processi business. Utilizzare lo strumento Test Connector (disponibile in Integrated Test Environment e come strumento indipendente) per simulare un connettore generico, in modo da poter verificare la progettazione della collaborazione.	"Test Connector" a pagina 12

Process Designer Express

Process Designer Express viene utilizzato per creare, modificare, compilare ed eliminare maschere di collaborazione. Quando si modifica una maschera esistente, è possibile utilizzare Process Designer Express per modificare le proprietà della maschera, oltre ad aggiungere o modificare scenari e diagrammi di attività.

Per informazioni dettagliate sull'interfaccia Process Designer Express, consultare Capitolo 2, "Panoramica su Process Designer Express", a pagina 15.

API di collaborazione

L'API di collaborazione di WebSphere Business Integration Server Express fornisce numerose classi i cui metodi possono essere utilizzati in una maschera di collaborazione.

Nota: E' possibile accedere all'API mediante chiamate Java tradizionali o mediante i blocchi funzione supportati nell'Activity Editor. Consultare Capitolo 6, "Utilizzo di Activity Editor", a pagina 159 per ulteriori informazioni.

Interazione con un oggetto di collaborazione

La classe BaseCollaboration definisce in modo generico il comportamento e le funzioni di una collaborazione, quali l'ottenimento dei valori delle proprietà di configurazione, la scrittura su un file di log o la registrazione di una traccia.

Quando si crea una maschera di collaborazione, viene creata una classe Java come sottoclasse di BaseCollaboration. In questo modo, la collaborazione eredita tutti i metodi di BaseCollaboration. Questi metodi consentono ad una collaborazione di eseguire operazioni quali:

- Ottenimento del valore di una proprietà di configurazione
- Generazione di un'eccezione
- Scrittura di messaggi di errore, avvertenza e informativi nel file di log

Per ulteriori informazioni sui metodi della classe BaseCollaboration, consultare Capitolo 22, "Classe BaseCollaboration", a pagina 367.

Interazione con oggetti business

Una collaborazione generalmente interagisce e gestisce gli oggetti business. I metodi della classe BusObj consentono ad una collaborazione di eseguire operazioni quali:

- Ottenimento del nome di un oggetto business
- Ottenimento dei valori chiave di un oggetto business
- Ottenimento del numero di oggetti business secondari contenuti in un oggetto business gerarchico
- Verifica dell'eguaglianza dei valori di attributo di due oggetti business
- Copia dei valori di attributo da un oggetto business ad un altro

Per ulteriori informazioni sui metodi della classe BusObj, consultare Capitolo 23, "Classe BusObj", a pagina 391.

Interazione con vettori di oggetti business

Le collaborazioni spesso ottengono e impostano i valori degli attributi di oggetti business. Se un oggetto business è gerarchico, uno o più dei suoi attributi rappresenta un oggetto business secondario, o anche un vettore di oggetti business secondari. Un oggetto business secondario appare come un vettore per la collaborazione.

I metodi della classe BusObjArray consentono ad una collaborazione di interagire e gestire vettori di oggetti business. Questi metodi eseguono operazioni quali:

- Impostazione o ottenimento di elementi del vettore
- Copia di un vettore in un altro
- Aggiunta di un oggetto business al vettore
- Ottenimento del numero di elementi nel vettore

Per ulteriori informazioni sui metodi della classe `BusObjArray`, consultare Capitolo 24, "Classe BusObjArray", a pagina 411.

Interazione con eccezioni

Quando si verifica un errore in una collaborazione, la collaborazione o l'ambiente runtime della collaborazione generano un'eccezione. L'eccezione è contenuta in un oggetto della classe CollaborationException. Questa classe consente ad un oggetto di collaborazione di interagire con un oggetto eccezione e di eseguire le seguenti operazioni:

- Ottenimento del tipo e del tipo secondario di eccezione
- Ottenimento del messaggio di eccezione

Per ulteriori informazioni sui metodi della classe CollaborationException, consultare Capitolo 29, “Classe CollaborationException”, a pagina 449.

System Manager

System Manager è uno strumento grafico che fornisce un'interfaccia a ICS e al suo repository. Consente di effettuare le seguenti attività relative alla collaborazione:

- Creazione di un oggetto di collaborazione
- Collegamento di un oggetto di collaborazione
- Impostazione delle proprietà specifiche di collaborazione di un oggetto di collaborazione
- Verifica della progettazione di una collaborazione (mediante lo strumento Test Connector)
- Distribuzione di un oggetto di collaborazione nell'ambiente runtime

Per ulteriori informazioni sull'utilizzo di System Manager per la creazione, configurazione e distribuzione di oggetti di collaborazione, consultare *System Implementation Guide*.

Test Connector

Test Connector è uno strumento grafico per il test di collaborazioni e connettori. E' disponibile in Integrated Test Environment o come strumento indipendente.

Nota: Se si vuole effettuare il test di client di accesso, è necessario utilizzare Test Connector mediante Integrated Test Environment.

Lo strumento Test Connector simula un connettore reale, consentendo di verificare facilmente la progettazione della collaborazione inviando un evento di attivazione o inviando una richiesta di chiamata di servizio. Per ulteriori informazioni sull'utilizzo di Test Connector, consultare *System Implementation Guide*.

Panoramica sul processo di sviluppo

Questa sezione fornisce una panoramica del processo di sviluppo di collaborazioni, che include le seguenti fasi principali:

1. Installazione e configurazione del software IBM WebSphere Business Integration Server Express (inclusi Java Development Kit e tutti i prodotti di altri fornitori richiesti). Per istruzioni specifiche sull'installazione e la configurazione, consultare la guida all'installazione di WebSphere Business Integration Server Express per la piattaforma utilizzata.
2. Progettazione e implementazione della collaborazione.

Fasi di sviluppo della collaborazione

Le fasi di sviluppo della collaborazione sono le seguenti:

1. Progettazione del processo business che sarà implementato dalla collaborazione.
2. Creazione delle definizioni di oggetti business.
3. Creazione della maschera di collaborazione, incluse meta-informazioni e definizioni.

4. Creazione di tutti gli scenari e del relativo diagramma di attività.
5. Personalizzazione dei frammenti di codice richiesti.
6. Creazione del testo dei messaggi.
7. Compilazione della maschera.
8. Creazione di un oggetto di collaborazione dalla maschera di collaborazione.
9. Test e debug della collaborazione.
10. Distribuzione della collaborazione nell'ambiente runtime.

La Tabella 5 fornisce una panoramica del processo di sviluppo della collaborazione ed un riferimento rapido ai capitoli in cui è possibile trovare le informazioni sugli argomenti specifici.

Nota: Alcune delle attività di sviluppo della collaborazione sono al di fuori dell'ambito di sviluppo di una maschera di collaborazione, e non sono documentate in questo manuale. Per queste attività, la Tabella 5 fornisce il riferimento alla documentazione appropriata nella libreria di WebSphere Business Integration Server Express.

Notare che, se un team di persone è disponibile per lo sviluppo di una collaborazione, le attività principali dello sviluppo della collaborazione possono essere svolte in parallelo da diversi membri del team di sviluppo.

Tabella 5. Panoramica delle attività di sviluppo di una collaborazione

Attività:	**Fase:**	**Fare riferimento a:**
Progettazione dell'architettura della collaborazione	• Identificazione dei processi business • Individuazione dell'utilizzo delle maschere di collaborazione • Personalizzazione di flussi business particolari	Capitolo 3, "Progettazione di una collaborazione"
Progettazione e sviluppo di oggetti business	• Progettazione della struttura di oggetti business • Implementazione di oggetti business	Business Object Development Guide
Generazione della maschera di collaborazione	• Creazione della maschera di collaborazione e modifica delle sue proprietà	Capitolo 4, "Creazione di una maschera di collaborazione"
	• Creazione degli scenari e dei diagrammi di attività	Capitolo 5, "Utilizzo dei diagrammi di attività"
	• Personalizzazione dei frammenti di codice	Capitolo 6, "Utilizzo di Activity Editor"
	• Implementazione della gestione di errori e messaggi	Capitolo 7, "Gestione delle eccezioni"
Creazione dell'oggetto di collaborazione	• Creazione della definizione di collaborazione • Personalizzazione delle funzioni runtime, se necessario	*System Implementation Guide*
Test e debug	• Test della collaborazione nel sistema WebSphere Business Integration Server Express • Riscrittura del codice, se necessario	*System Implementation Guide*

Capitolo 2. Panoramica su Process Designer Express

Process Designer Express consente di eseguire le seguenti attività di sviluppo di collaborazioni:

- Creazione, modifica, compilazione o eliminazione di una definizione di maschera mediante la finestra Definizioni maschera.
- Definizione o modifica di un diagramma di attività per uno scenario della maschera di collaborazione mediante l'editor diagramma.

Questo capitolo contiene un'introduzione a Process Designer Express. Sono descritte l'interfaccia e le modalità di navigazione nelle finestre di Process Designer Express, i menu e le barre degli strumenti che consentono di eseguire le attività richieste per lo sviluppo di collaborazioni.

Avvio di Process Designer Express

Il metodo con il quale viene avviato Process Designer Express può essere diverso se si sta creando una nuova maschera di collaborazione o se si modifica una maschera di collaborazione esistente.

Importante
Prima di avviare Process Designer Express, è necessario assicurarsi che System Manager sia in esecuzione.

Le diverse modalità di avvio di Process Designer Express da System Manager, sono descritte nella Tabella 6.

Tabella 6. Avvio di Process Designer Express da System Manager

Metodo	Risultato
Fare clic con il tasto destro del mouse sulla cartella Maschere di collaborazione dalla vista di selezione oggetti, quindi fare clic su Crea nuovo modello di collaborazione dal menu di scelta rapida.	Process Designer Express si apre e visualizza la finestra di dialogo Maschera nuova.
Fare doppio clic su una maschera di collaborazione all'interno della cartella Maschere di collaborazione	Process Designer Express si apre e visualizza la definizione della maschera selezionata.
Creare ed utilizzare un collegamento dal progetto utente al componente nelle librerie ICL (Integration Component Libraries)	Process Designer Express si apre e visualizza la definizione della maschera associata al collegamento.

Process Designer Express può anche essere avviato dal menu Start. Fare clic su Start —> Programmi —> IBM WebSphere Business Integration Server Express —> IBM WebSphere Business Integration Toolset Express —> Programmi di sviluppo —> Process Designer Express.

Process Designer Express viene visualizzato in una propria finestra ancorabile. E' possibile avviare più di una istanza di Process Designer Express alla volta, per modificare più di una definizione di maschera di collaborazione.

Layout di Process Designer Express

Quando si avvia Process Designer Express, per impostazione predefinita la finestra principale viene visualizzata come mostrato nella Figura 4.

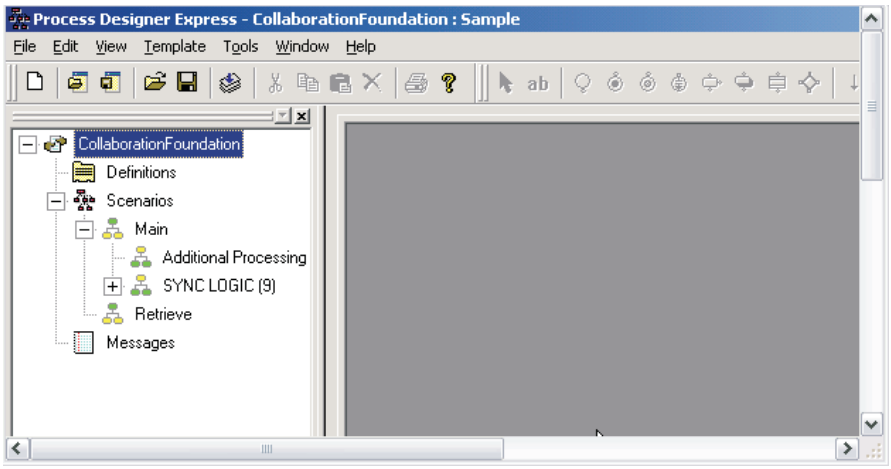


Figura 4. Finestra principale di Process Designer Express

Il layout della finestra Process Designer Express è costituito dalle seguenti aree:

- Struttura maschera (ancorabile)

  La vista Struttura maschera nel riquadro a sinistra utilizza un modello gerarchico per elencare le definizioni, gli scenari e i messaggi della maschera di collaborazione. Fare clic sul segno più (+) accanto ad un nodo di scenario esistente nella struttura per espandere i rami secondari e visualizzare gli scenari esistenti e i diagrammi secondari, se presenti.
- Area di lavoro della finestra principale, che può essere vuota o visualizzare le seguenti finestre:
  - Finestra Definizioni maschera

    Questa finestra viene utilizzata per fornire informazioni generali sulla maschera di collaborazione, le dichiarazioni di variabili o le informazioni sulle porte. Consultare “Finestra Definizioni maschera” a pagina 17.
  - Finestra Diagram editor

    Questa finestra viene utilizzata per visualizzare i nodi del diagramma di attività. Consultare “Finestra Diagram editor” a pagina 18.
  - Finestra Messaggi maschera

    Questa finestra viene utilizzata per scrivere o modificare il file di messaggi della maschera. Consultare “Finestra Messaggi maschera” a pagina 19.

  Le finestre Definizioni maschera, Messaggi maschera e Diagram Editor possono essere ridotte a icona, ingrandite o ridimensionate (aperte alla dimensione specificata dall’utente) all’interno dell’area di lavoro. Consultare “Visualizzazione delle finestre nell’area di lavoro” a pagina 26.
- Finestra di output della compilazione (ancorabile)

  La finestra di output della compilazione (spesso chiamata ″finestra Output″) visualizza i risultati della compilazione di una maschera di collaborazione. Process Designer Express visualizza automaticamente questa finestra quando si compila la maschera di collaborazione. Consultare “Compilazione di una maschera di collaborazione” a pagina 105.

Nota: Process Designer Express memorizza la configurazione della sua finestra principale al momento dell'uscita. Pertanto, le modifiche effettuate a questa configurazione sono visualizzate alla successiva apertura di Process Designer Express. (Consultare "Personalizzazione della finestra principale" a pagina 24.) La Figura 4 mostra la configurazione predefinita della finestra principale. Se in precedenti esecuzioni di Process Designer Express questa configurazione è stata modificata, la finestra principale può apparire diversa.

Per accedere alle funzionalità di Process Designer Express è possibile utilizzare:

- Menu a discesa nella parte superiore della finestra
- Icone nelle barre degli strumenti
- Menu sensibili al contesto (menu a comparsa a cui si accede facendo clic con il tasto destro del mouse)
- Tasti di accesso rapido

Finestre di Process Designer Express

Finestra Definizioni maschera

La finestra Definizioni maschera presenta quattro schede per la definizione delle proprietà della collaborazione.

- Scheda Generale — fornisce i campi in cui specificare le informazioni generali sulla maschera di collaborazione, quali il nome, la descrizione, il livello di transazione minimo e il pacchetto.
- Scheda Dichiarazioni — fornisce i campi in cui specificare le dichiarazioni delle variabili.
- Scheda Proprietà — fornisce i campi in cui specificare il nome, il tipo e il valore delle proprietà della maschera di collaborazione definite dall'utente.
- Scheda Porte ed eventi di attivazione — fornisce i campi in cui specificare i nomi di porta e i relativi oggetti business e le istruzioni associate.

Importante

IBM consiglia di non aggiungere, modificare o eliminare oggetti business dal repository utilizzando Business Object Designer Express o System Manager dopo aver collegato l'oggetto business ad una collaborazione in esecuzione in un ambiente di produzione. Per ulteriori informazioni, consultare "Definizione di porte ed eventi di attivazione (scheda Porte ed eventi di attivazione)" a pagina 97.

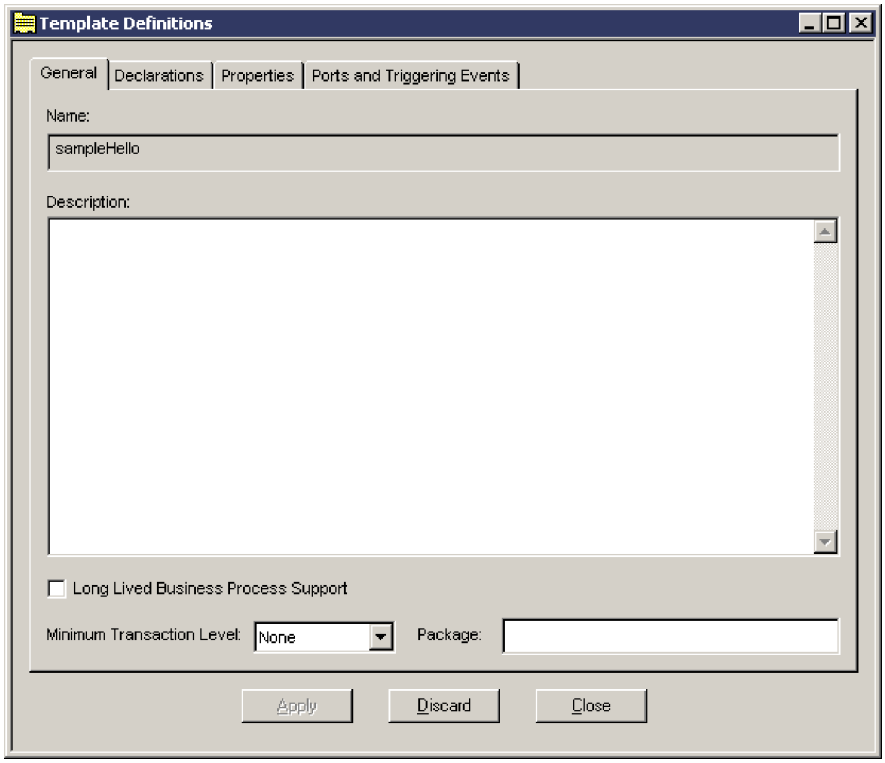


Figura 5. Finestra Definizioni maschera

Ci sono diversi metodi per aprire la finestra Definizioni maschera, che viene visualizzata nell'area di lavoro della finestra principale:

- Nella vista Struttura maschera, fare doppio clic su Definizioni.
- Nella vista Struttura maschera, selezionare Definizioni, fare clic con il tasto destro del mouse e scegliere Apri definizioni maschera.
- Dal menu a discesa Maschera, selezionare Apri definizioni maschera.
- Utilizzare la combinazione di tasti di accesso rapido `Ctrl+T`.

La finestra Definizioni maschera contiene i pulsanti Applica e Ignora; questi pulsanti sono visualizzati nella parte inferiore della finestra qualsiasi sia la scheda attualmente visualizzata. Il pulsante Applica convalida le modifiche alla maschera, ma non le salva (è necessario utilizzare il comando File —> Salva per salvare tutte le modifiche). Il pulsante Ignora consente di ripristinare la definizione precedentemente salvata, ignorando le modifiche non ancora salvate.

Nota: I pulsanti Applica e Ignora hanno effetto sui dati contenuti in tutte le schede, non solo su quella attualmente visualizzata.

Finestra Diagram editor

Il Diagram editor è uno strumento incluso in Process Designer Express che consente di creare e modificare i diagrammi di attività. Questa finestra viene visualizzata nell'area di lavoro della finestra principale quando si apre un diagramma di attività. Nel Diagram editor, è possibile aggiungere nodi, chiamate di servizio e collegamenti di transizione ad un diagramma di attività; modificare la posizione degli elementi; aggiungere e modificare etichette di testi e tipi di carattere; aggiungere e modificare proprietà di singoli componenti.

Il Diagram editor può essere aperto in diversi modi:

- Nella vista Struttura maschera, espandere il nodo degli scenari e fare doppio clic sul nome di un diagramma; in alternativa, fare clic con il tasto destro del mouse sul nome e scegliere Apri diagramma.
- Nella vista Struttura maschera, espandere il nodo degli scenari, selezionare il nome di un diagramma e scegliere Apri diagramma dal menu a discesa Maschera.
- Dal menu a discesa Maschera, selezionare Apri tutti i diagrammi.

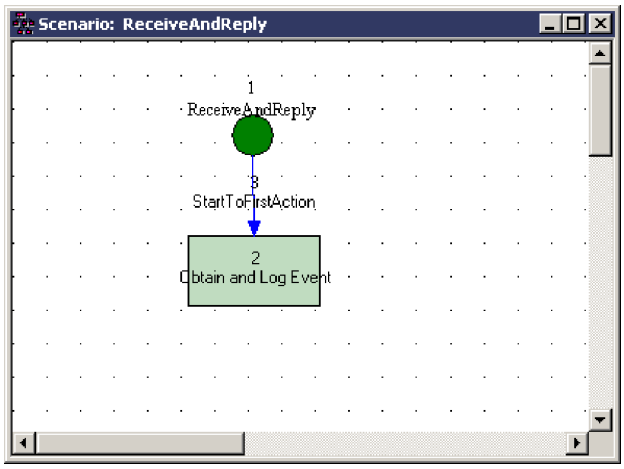


Figura 6. Finestra Diagram editor

Per ulteriori informazioni sull'utilizzo di Diagram editor, consultare "Creazione di un diagramma di attività" a pagina 104.

Finestra Messaggi maschera

La finestra Messaggi maschera fornisce un'area in cui è possibile scrivere o modificare il file di messaggi della maschera. Quando si compila la maschera, il testo del messaggio viene scritto nella directory Template\messages del progetto ICL (Integration Component Library) appropriato all'interno di System Manager.

La finestra Messaggi maschera può essere aperta in diversi modi:

- Nella vista Struttura maschera, fare doppio clic su Messaggi.
- Nella vista Struttura maschera, selezionare Messaggi, fare clic con il tasto destro del mouse e scegliere Apri messaggi.
- Dal menu a discesa Maschera, selezionare Apri messaggi maschera.
- Utilizzare la combinazione di tasti di accesso rapido `Ctrl+M`.

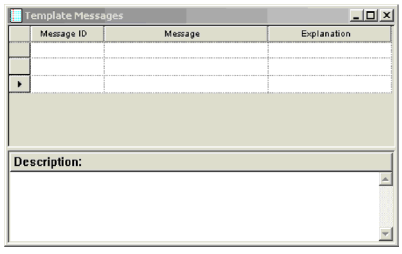


Figura 7. Finestra Messaggi maschera

Menu di Process Designer Express

In Process Designer Express, i menu e le opzioni di menu abilitate dipendono da cosa viene visualizzato nell'area di lavoro. Le seguenti sezioni illustrano i diversi menu di Process Designer Express quando l'area di lavoro è vuota o visualizza la finestra Definizioni maschera:

- "Funzioni del menu File"
- "Funzioni del menu Visualizza" a pagina 21
- "Funzioni del menu Maschera" a pagina 22
- "Funzioni del menu Finestra" a pagina 23

Nota: Quando l'area di lavoro visualizza il Diagram editor, Process Designer Express abilita le opzioni del menu Modifica ed abilita diverse opzioni in alcuni degli altri menu. La maggior parte delle funzioni sono relative alla gestione dei diagrammi di attività e sono illustrate in "Accesso alle funzioni dell'editor di diagrammi: i menu di Process Designer Express" a pagina 113.

Funzioni del menu File

Quando l'area di lavoro è vuota o visualizza la finestra Definizioni maschera o Messaggi maschera, il menu File presenta le seguenti opzioni:

- Nuovo — Crea una nuova maschera di collaborazione.
- Apri — Apre una definizione di maschera di collaborazione esistente. Contiene le seguenti due opzioni:
  - Da progetto — Apre una maschera di collaborazione da un progetto utente ICL (Integration Component Library).
  - Da file — Apre una maschera di collaborazione da un file .cwt presente nel file system.
- Chiudi — Chiude la maschera di collaborazione.
- Salva — Salva la maschera di collaborazione corrente. Contiene le seguenti due opzioni:
  - A progetto — Salva una maschera di collaborazione in un progetto utente ICL (Integration Component Library).
  - A file — Salva una maschera di collaborazione in un file .cwt che viene archiviato nel file system.
- Salva con nome — Salva la maschera di collaborazione corrente con un altro nome. Contiene le seguenti due opzioni:
  - A progetto — Salva una maschera di collaborazione in un progetto utente ICL (Integration Component Library).
  - A file — Salva una maschera di collaborazione in un file .cwt che viene archiviato nel file system.
- Elimina — Visualizza la finestra di dialogo Elimina maschera da progetto, dalla quale è possibile scegliere la maschera di collaborazione da eliminare.
- Compila — Compila la maschera di collaborazione. Per ulteriori informazioni, consultare "Compilazione di una maschera di collaborazione" a pagina 105.
- Compila tutto — Consente di compilare tutte le maschere di collaborazione presenti nel progetto, oppure di specificare un sottoinsieme per la compilazione. Per ulteriori informazioni, consultare "Compilazione di più maschere di collaborazione" a pagina 106.

- Importa — Importa i file nella definizione di maschera. Il programma di importazione di Process Designer Express può importare file BPEL e UML (in formato XMI); i file sono convertiti in file di maschera InterChange Server Express 4.2, se necessario.
- Esporta — Esporta i file. E' possibile esportare un file di maschera in UML (in formato XMI) o BPEL. Il programma di importazione di Process Designer Express esegue tutte le conversioni di formato richieste.
- Esci — Chiude Process Designer Express.

Quando nell'area di lavoro è visualizzato il Diagram editor, il menu File mostra le opzioni aggiuntive Imposta pagina, Anteprima di stampa e Stampa, che consentono di stampare un diagramma di attività. Quando nell'area di lavoro viene visualizzata la finestra Messaggi maschera, il menu File mostra un'opzione aggiuntiva che consente di stampare il file di messaggi.

Funzioni del menu Modifica

Le opzioni del menu Modifica sono disponibili solo quando il Diagram editor è attivo. Le opzioni includono i comandi di modifica standard Windows (ad esempio, Annulla, Ripeti, Copia e Incolla) e le seguenti opzioni speciali di Process Designer Express:

- Seleziona tutto — Seleziona tutti i nodi nel diagramma di attività corrente.
- Trova ID — Trova l'ID del diagramma di attività.
- Trova testo — Consente di ricercare il testo nel diagramma di attività corrente.
- Sostituisci testo — Consente di ricercare e sostituire il testo nel diagramma di attività corrente.
- Proprietà — Consente di modificare le proprietà di un simbolo selezionato. Questa opzione viene abilitata solo quando viene selezionato un simbolo nell'area di lavoro.
- Carattere — Consente di modificare il carattere e il colore delle etichette di testo dei simboli selezionati in un diagramma di attività. E' possibile modificare il tipo di carattere dei simboli e collegamenti attualmente selezionati oppure modificare il tipo di carattere per tutti i componenti utilizzando prima l'opzione Seleziona tutto, per selezionare tutti i componenti del diagramma, e poi applicando la modifica di carattere. Questa opzione viene attivata solo se è stato selezionato un simbolo nell'area di lavoro.

Funzioni del menu Visualizza

Le funzioni del menu Visualizza sono valide quando si apre Process Designer Express per la prima volta e quando nell'area di lavoro sono visualizzati componenti di aspetto grafico dei diagrammi di attività. Molte di queste funzioni possono essere attivate o disattivate:

- Preferenze — Apre la finestra di dialogo Preferenze, che consente di specificare le modalità di rappresentazione degli elementi in Process Designer Express.
- Struttura maschera — Quando questa opzione è attiva, Process Designer Express visualizza la vista di struttura maschera come riquadro a sinistra della finestra di Process Designer Express.
- Finestra output — Quando questa opzione è attiva, Process Designer Express visualizza i risultati della compilazione della maschera.
- Barre degli strumenti — Controlla la visualizzazione delle diverse barre degli strumenti di Process Designer Express. Le opzioni dei menu secondari sono:
  - Standard — Quando questa opzione è attiva, Process Designer Express visualizza i pulsanti dellabarra degli strumenti Standard.

  - Simboli — Quando questa opzione è attiva, Process Designer Express visualizza i pulsanti della barra degli strumenti Simboli.
  - Allineamento — Quando questa opzione è attiva, Process Designer Express visualizza i pulsanti della barra degli strumenti Allineamento.
  - Spostamento — Quando questa opzione è attiva, Process Designer Express visualizza i pulsanti della barra degli strumenti Spostamento.
  - Zoom — Quando questa opzione è attiva, Process Designer Express visualizza i pulsanti della barra degli strumenti Zoom/Panoramica.
  - Programmi — Quando questa opzione è attiva, Process Designer Express visualizza i pulsanti per accedere ad altri programmi di InterChange Server Express.
- Barra di stato — Quando questa opzione è attiva, Process Designer Express può visualizzare il messaggio di stato su una riga nella parte inferiore della finestra principale.

Inoltre, Process Designer Express consente le seguenti opzioni di diagramma quando è attivo il Diagram editor:

- Visualizza tipi — Se attivata, visualizza il tipo di un simbolo. Questa opzione risulta utile per imparare a riconoscere un nodo dalla forma.
- Visualizza UID — Se attivata, visualizza l'ID univoco (UID) di ciascun simbolo.
- Visualizza etichette — Se attivata, visualizza l'etichetta di simbolo fornita dall'utente.
- Blocca (sola lettura) — Se attivata, pone il diagramma di attività in modalità di sola lettura.
- Aggiorna — Aggiorna la visualizzazione del diagramma di attività.
- Griglia — Se attivata, visualizza le linee di griglia dell'area di lavoro. Se disattivata, le linee della griglia sono nascoste.
- Blocca sulla griglia — Se attivata, i simboli sono automaticamente allineati alle linee di griglia quando sono posizionati in un diagramma di attività.
- Proprietà griglia — Consente di impostare le proprietà della griglia. (Notare che il blocco sugli angoli non è applicabile ai diagrammi di attività, anche se può essere attivato/disattivato).
- Limiti di pagina — Mostra i limiti di pagina come linee tratteggiate.
- Comandi Zoom — Consentono di allargare il diagramma di attività o ingrandire una sezione. I comandi Zoom possono anche essere eseguiti dalla barra degli strumenti Zoom. Per ulteriori informazioni sulle funzioni di ingrandimento, consultare "Zoom o panoramica sui simboli" a pagina 200.
- Disposizione automatica — Allinea automaticamente lo scenario aperto nel Diagram editor.
- Disposizione automatica maschera — Allinea automaticamente tutti gli scenari all'interno di una maschera.

Funzioni del menu Maschera

Quando l'area di lavoro è vuota o visualizza la finestra Definizioni maschera o Messaggi maschera, il menu Maschera presenta le seguenti opzioni:

- Ogni volta che un oggetto, diverso da uno scenario, viene selezionato nella vista Struttura maschera:
  - Apri tutti i diagrammi — Apre tutti i diagrammi di attività definiti per la maschera di collaborazione.
  - Chiudi tutti i diagrammi — Chiude tutti i diagrammi di attività aperti.

- Nuovo scenario — Visualizza la finestra di dialogo Nuovo scenario.
- Apri definizioni maschera — Visualizza la finestra Definizioni maschera, da cui è possibile modificare le proprietà della maschera di collaborazione.
- Apri messaggi maschera — Visualizza la finestra Messaggi maschera, da cui è possibile definire o modificare il file di messaggi associato alla maschera di collaborazione.

• Ogni volta che viene selezionato uno scenario nella vista Struttura maschera, sono disponibili le seguenti voci di menu aggiuntive:
  - Apri diagramma — Apre il diagramma di attività per lo scenario corrente.
  - Rinomina scenario — Consente di rinominare lo scenario corrente.
  - Elimina scenario — Elimina lo lo scenario attualmente selezionato e i relativi diagrammi di attività.
  - Apri definizione scenario — Consente di modificare le variabili a livello di scenario.
• Quando viene aperto il Diagram editor, oltre alle voci di menu già descritte, sono disponibili le seguenti voci:
  - Ridimensiona diagramma — Ridimensiona il diagramma di attività come unità di pagina verticali ed orizzontali, ed è rilevante per la stampa del diagramma di attività. La finestra di dialogo Dimensione diagramma contiene le caselle di selezione per immettere il numero di pagine. Notare che la dimensione del diagramma è direttamente correlata all'orientamento di pagina (orizzontale o verticale) e alla selezione della carta.
  - Salva diagramma come file di testo — Salva il diagramma di attività corrente in un file in formato testo (.txt).

Funzioni del menu Strumenti

Il menu Strumenti consente di avviare altri strumenti di InterChange Server Express. Le opzioni sono le seguenti:

Map Designer Express	Apre Map Designer Express.
Business Object Designer Express	Apre Business Object Designer Express.
Relationship Designer Express	Apre Relationship Designer Express.

Funzioni del menu Finestra

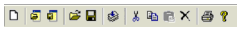
Le opzioni del menu a discesa Finestra comprendono le funzioni standard di visualizzazione della finestra MDI (Multiple Document Interface). Utilizzare queste opzioni per controllare le funzioni di visualizzazione, quali affiancamento, sovrapposizione e attivazione di finestre aperte.

Barre degli strumenti di Process Designer Express

Process Designer Express fornisce le barre degli strumenti per le attività comunemente eseguite. Queste barre degli strumenti sono ancorabili, cioè è possibile sganciarle dalla tavolozza della finestra principale e spostarle alla finestra principale o al desktop.

Nella Tabella 7 sono riportate le barre degli strumenti fornite da Process Designer Express.

Tabella 7. Barre degli strumenti di Process Designer Express

Nome barra degli strumenti	Aspetto barra degli strumenti	Per ulteriori informazioni
Standard		Nessuno
Simboli		"Introduzione ai simboli" a pagina 114
Allineamento		"Allineamento dei simboli" a pagina 197
Spostamento		"Spostamento di simboli" a pagina 199
Zoom/Panoramica		"Zoom o panoramica sui simboli" a pagina 200

Personalizzazione della finestra principale

Process Designer Express fornisce i seguenti modi per personalizzare la finestra principale:

- Scegliere quali finestre visualizzare
- Spostare una finestra ancorabile
- Scegliere le modalità di visualizzazione delle finestre nell'area di lavoro

Scelta delle finestre da visualizzare

Come mostrato nella Figura 4, quando si apre Process Designer Express per la prima volta, la vista Struttura maschera viene visualizzata nel riquadro a sinistra. L'area di lavoro viene visualizzata a destra ed è inizialmente vuota. La finestra Output non viene visualizzata. L'aspetto della finestra principale può essere personalizzato dal menu Visualizza.

La Tabella 8 descrive le opzioni del menu a discesa Visualizza ed il loro effetto sull'aspetto della finestra principale di Process Designer Express.

Tabella 8. Opzione del menu Visualizza per la personalizzazione della finestra principale

Opzione del menu Visualizza	Elemento visualizzato
Struttura maschera	Le definizioni di maschera, gli scenari e i messaggi come riquadro a sinistra.
Finestra Output	La finestra Output visualizzata come una piccola finestra sotto la vista Struttura maschera (se presente) e l'area di lavoro.

Tabella 8. Opzione del menu Visualizza per la personalizzazione della finestra principale (Continua)

Opzione del menu Visualizza	Elemento visualizzato	
Barre degli strumenti	Un menu che fornisce le opzioni per la visualizzazione delle barre degli strumenti Process Designer Express:	
	Standard	La barra degli strumenti principale nella tavolozza di Process Designer Express, che fornisce i pulsanti per connettersi o disconnettersi da InterChange Server Express, aprire una maschera dal server o da un file, salvare e compilare una maschera, tagliare. copiare, incollare e eliminare una maschera, stampare.
	Simboli	La barra degli strumenti Simboli diagramma fornisce i simboli da aggiungere ad un diagramma di attività.
	Allineamento	La barra degli strumenti Allineamento contiene le funzioni di allineamento per i simboli del diagramma di attività.
	Spostamento	La barra degli strumenti Spostamento contiene le funzioni per spostare leggermente i simboli selezionati in un diagramma di attività.
	Zoom	La barra degli strumenti Zoom/Panoramica contiene le funzioni per ingrandire o mostrare una panoramica dei simboli selezionati in un diagramma di attività.
Finestra Stato	Un riquadro costituito da una sola riga in cui Process Designer Express visualizza le informazioni di stato	

Quando una opzione di menu appare con un contrassegno alla sua sinistra, l'elemento associato viene visualizzato. Per disattivare la visualizzazione di un elemento, selezionare l'opzione di menu associata. Il contrassegno scompare per indicare che l'elemento non è attualmente visualizzato. Al contrario, è possibile attivare la visualizzazione di un elemento non visualizzato selezionando l'opzione di menu associata. In questo caso, il contrassegno appare affianco all'elemento da visualizzare.

Spostamento di una finestra ancorabile

Process Designer Express supporta le seguenti parti della finestra principale come finestre ancorabili:

- Vista Struttura maschera
- Finestra Output
- Barre degli strumenti

Per impostazione predefinita, una finestra ancorabile è normalmente collocata lungo i margini della finestra principale e viene spostata come parte della finestra principale. Quando si sposta una finestra ancorabile, questa viene staccata dalla finestra principale, consentendo alla finestra di comportarsi come finestra indipendente. Per spostare una finestra ancorabile, tenere premuto il tasto sinistro del mouse e trascinare il bordo della finestra sulla finestra principale o sul desktop.

Visualizzazione delle finestre nell'area di lavoro

L'area di lavoro della finestra principale di Process Designer Express consente di visualizzare le finestre al suo interno in uno dei seguenti modi:

- Ingrandite — Una finestra occupa tutta l'area di lavoro. E' possibile passare da una finestra ingrandita all'altra selezionando il nome della finestra desiderata dal menu a discesa Finestra. Se sono state mantenute le preferenze utente predefinite per le Finestre diagramma cartella di lavoro, è possibile anche passare da una finestra massimizzata all'altra selezionando la corrispondente scheda Cartella di lavoro sotto l'area di lavoro. Per ulteriori informazioni, consultare "Modifica della visualizzazione generale" a pagina 202.
- Ridimensionate — Ogni finestra dispone di una propria area separata all'interno dell'area di lavoro. E' possibile ridimensionare queste finestre e sovrapporle o spostarle nell'area di lavoro. Le finestre ridimensionabili sono utili quando si desidera visualizzare contemporaneamente più di un diagramma di attività oppure un diagramma di attività e la finestra Definizioni maschera o Messaggi maschera.
- Ridotte a icona — Ogni finestra viene rappresentata da un'icona nella parte inferiore dell'area di lavoro. E' possibile ripristinare una finestra ridotta a icona facendo doppio clic sulla sua rappresentazione minimizzata.

Parte 2. Creazione di una maschera di collaborazione

Capitolo 3. Progettazione di una collaborazione

Questo capitolo descrive le linee guida per la progettazione di collaborazioni riutilizzabili e ben strutturate.

In generale, è buona norma sviluppare una maschera di collaborazione standard per semplificare lo sviluppo di collaborazioni definite dall'utente. L'utilizzo di una simile maschera garantisce:

- La congruenza della progettazione della collaborazione

  Quando si basano su una maschera standard, tutte le collaborazioni possono:

  – Eseguire le stesse operazioni di istruzione sull'oggetto business nell'applicazione di destinazione che corrisponde all'attivazione del flusso della collaborazione
  – Gestire gli errori utilizzando lo stesso meccanismo di gestione errori, semplificando enormemente il supporto tecnico delle collaborazioni finali
  – Utilizzare le porte con nomi, tipi e comportamento previsto uguali
- Semplicità nel documentare la collaborazione

  Documentare il comportamento delle collaborazioni basate su una maschera standard può diventare molto più semplice perché può essere basato anch'esso su una maschera di informazioni di cui l'utente ha bisogno per comprendere il funzionamento della collaborazione.
- Inclusione delle "migliori metodologie"

  Le migliori metodologie che IBM raccomanda (consultare "Raccomandazioni relative alla codifica" a pagina 35) e che la propria sede aziendale sviluppa possono essere incorporate nella maschera standard ed essere automaticamente incluse nelle collaborazioni che si basano su quella maschera standard.

InterChange Server Express fornisce una maschera di collaborazione standard (CollaborationFoundation) ed una maschera di collaborazione standard wrapper (WrapperFoundation) che si può utilizzare così com'è o si può modificare secondo le proprie esigente. Per ulteriori informazioni consultare "Maschera CollaborationFoundation" a pagina 30 e "Maschera WrapperFoundation" a pagina 48.

Questo capitolo fornisce, inoltre, delle linee guida per le seguenti attività:

- Creazione di gruppi di oggetti collaborazione. Per ulteriori informazioni, consultare "Creazione di gruppi di collaborazioni" a pagina 53.
- Gestione di esecuzione parallela, incluso la sequenza degli eventi e l'isolamento di eventi. Per ulteriori informazioni, consultare "Progettazione di un'esecuzione parallela" a pagina 56.
- Creazione di una maschera di collaborazione internazionalizzata. Per ulteriori informazioni, consultare "Una collaborazione internazionalizzata" a pagina 65.
- Gestione degli eventi non riusciti. Per ulteriori informazioni, consultare "Gestione degli eventi non riusciti" a pagina 78.

Maschera CollaborationFoundation

La maschera di collaborazione CollaborationFoundation è uno strumento che semplifica lo sviluppo delle collaborazioni definite dell'utente che eseguono funzioni quali:

- Sincronizzazione

  Una collaborazione esegue una sincronizzazione quando replica i dati da un'applicazione ad un'altra. Una sincronizzazione di questo genere di solito elabora i dati da un'applicazione origine che non prevede un risultato.

  CollaborationFoundation contiene la logica dell'istruzione, il filtraggio dei dati e la gestione degli errori utilizzati dalla maggior parte delle collaborazioni che eseguono la sincronizzazione, vale a dire la logica di base di sincronizzazione. CollaborationFoundation ha cinque diagrammi secondari vuoti, ognuno denominato Ulteriore elaborazione, che consente all'utente di estendere il processo business di collaborazione senza dover modificare la logica di base di sincronizzazione.

  Questa sezione descrive principalmente la logica di base di sincronizzazione e fornisce le raccomandazioni per estendere la maschera CollaborationFoundation in modo da soddisfare i requisiti delle collaborazioni che si prevede di sviluppare.

- Acceso dati

  Un collaborazione può essere attivata da una richiesta Retrieve da un'applicazione origine che attende il risultato. Una collaborazione può anche essere attivata da un'applicazione origine che invia un evento di sincronizzazione ed attende che la collaborazione restituisca un oggetto business completo dall'applicazione di destinazione. Questi eventi richiedono un'applicazione origine che può inviare una richiesta sincrona o che utilizza Server Access Interface.

  Questo capitolo descrive la funzionalità di accesso nel Processo Retrieve e nel processo Additional_Retrieve.

- Piping di dati

  Una collaborazione esegue un piping di dati quando sposta i dati da un'applicazione ad un'altra senza eseguire alcun filtraggio o verifica dei dati. Per creare questo tipo di collaborazione da CollaborationFoundation, collegare la porta From all'applicazione origine e collegare entrambe le porte DestinationAppRetrieve e To all'applicazione di destinazione.

  Nota: E' anche possibile modificare una collaborazione distribuita con il prodotto per spostare semplicemente i dati da un origine ad una destinazione. Collegare la porta From all'applicazione origine, collegare entrambe le porte DestinationAppRetrieve e To all'applicazione di destinazione e collegare tutte le altre porte al connettore porte. E' importante non modificare il valore predefinito di alcuna proprietà d configurazione.

Le seguenti sezioni forniscono ulteriori informazioni sulla maschera CollaborationFoundation:

- "Funzioni di CollaborationFoundation" a pagina 31
- "Utilizzo della maschera CollaborationFoundation" a pagina 33
- "Porte CollaborationFoundation" a pagina 34
- "Estensione della CollaborationFoundation" a pagina 35

Funzioni di CollaborationFoundation

Un oggetto di collaborazione generato dalla maschera CollaborationFoundation può eseguire un'azione Crea, Recupera, Elimina o Aggiorna su un oggetto business nell'applicazione di destinazione che corrisponde all'oggetto business di attivazione della collaborazione.

Gestione dell'oggetto business di attivazione

Comportamento dell'istruzione di recupero (Retrieve): CollaborationFoundation è progettata per consentire all'oggetto business di attivazione di utilizzare l'istruzione Retrieve. Se la collaborazione è collegata ad un connettore origine che effettua una richiesta sincrona, i valori dell'oggetto business di attivazione vengono restituiti al connettore origine appena la collaborazione termina l'elaborazione. I valori vengono restituiti come se i valori dell'oggetto business fossero stati passati per riferimento.

Questa maschera fornisce le proprietà standard mostrate nella Tabella 9; queste proprietà influenzano il comportamento dei flussi di business.

Tabella 9. Proprietà standard di CollaborationFoundation per il flusso di business

Proprietà standard	Descrizione
CONVERT_CREATE	Configura un oggetto di collaborazione per convertire l'istruzione inviata alla destinazione da Create in Update se l'oggetto business di attivazione esiste nell'applicazione di destinazione.
CONVERT_UPDATE	Configura un oggetto di collaborazione per convertire l'istruzione Update in istruzione Create se l'oggetto business non esiste già nell'applicazione di destinazione.
USE_RETRIEVE	Configura un oggetto collaborazione per recuperare il relativo oggetto business di attivazione dall'applicazione di destinazione prima di sincronizzare i dati. Questa proprietà è utile quando si esegue l'elaborazione di compensazione e quando si imposta l'istruzione che si basa sull'esistenza o meno dell'oggetto business nell'applicazione di destinazione.
ADDITIONAL_RETRIEVE	Configura un oggetto di collaborazione per recuperare l'oggetto business dall'applicazione di destinazione dopo che ha sincronizzato i dati. Questa proprietà è utile quando l'applicazione origine richiede che un oggetto business con valori completi valutato venga restituito dall'applicazione di destinazione ma il connettore dell'applicazione di destinazione non restituisce un oggetto completo dopo la creazione o l'aggiornamento dei suoi dati.

La Figura 8 a pagina 32 mostra il flusso di business principale per una collaborazione quando la proprietà USE_RETRIEVE si risolve in `false`.

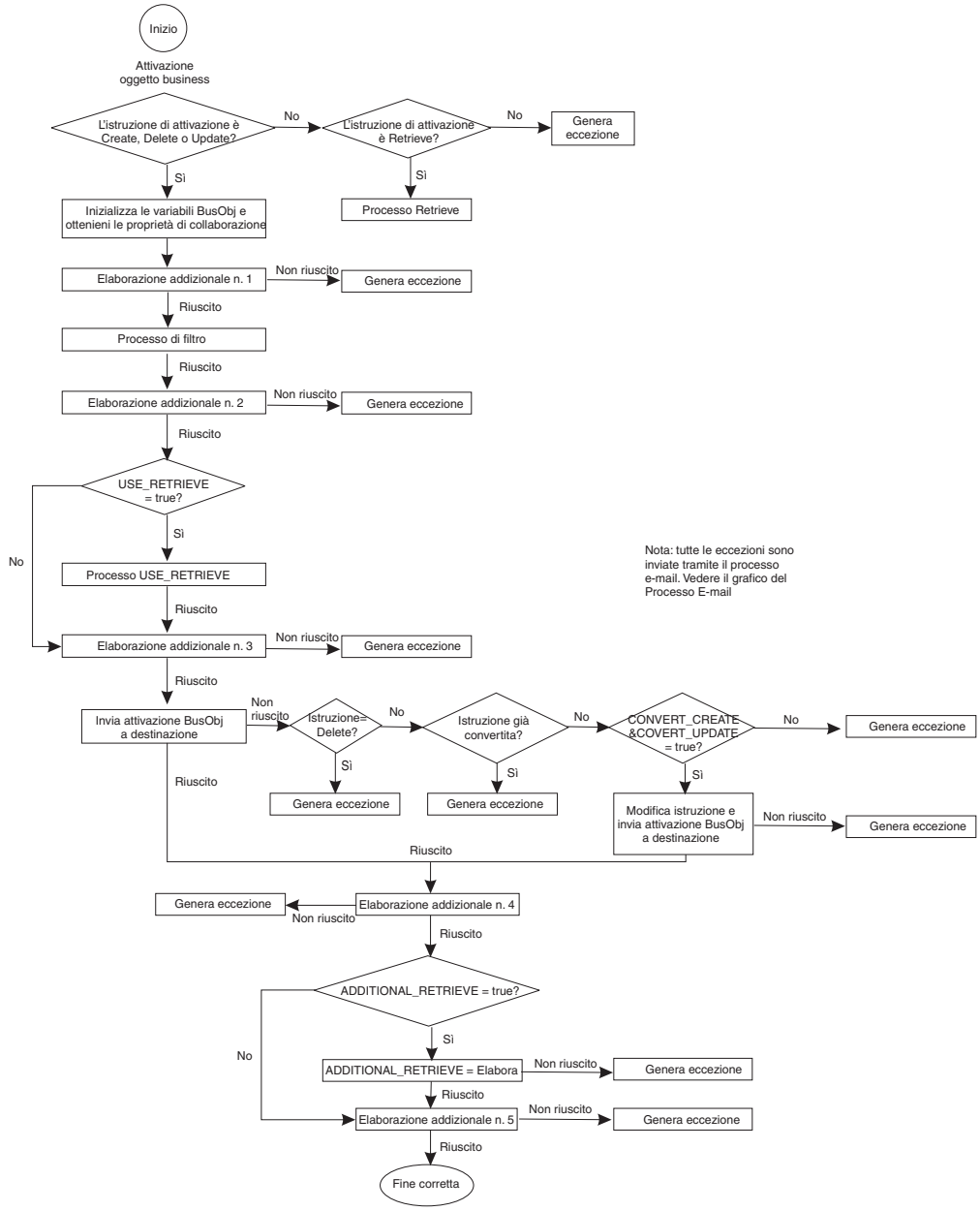


Figura 8. Flusso del processo di business principale di CollaborationFoundation

Filtraggio dei dati nell'oggetto business di destinazione: E' possibile configurare un oggetto di collaborazione per filtrare i dati in degli attributi specificati dell'oggetto business di attivazione. E' possibile utilizzare i risultati del filtraggio per stabilire se la collaborazione deve sincronizzare un oggetto business di attivazione con dei dati specifici. La maschera fornisce le seguenti quattro proprietà per il filtraggio dei dati:

- 1_FILTER_ATTRIBUTE
- 1_INCLUDE_VALUES
- 1_EXCLUDE_VALUES
- 1_FAIL_ON_INVALID_VALUE

Per informazioni sull'aggiunta di ulteriori proprietà che possono specificare più attributi di filtraggio, leggere la descrizione della proprietà 1_FILTER_ATTRIBUTE

in “Informazioni sulle collaborazioni standard”, a pagina 477. Inoltre, la Figura 93 a pagina 482 illustra il funzionamento del filtraggio della collaborazione.

Per filtrare su un secondo attributo, la maschera CollaborationFoundation fornisce la seguente serie di proprietà:

- 2_FILTER_ATTRIBUTE
- 2_INCLUDE_VALUES
- 2_EXCLUDE_VALUES
- 2_FAIL_ON_INVALID_VALUE

Per filtrare su più attributi, fornire una serie di proprietà di filtro per ciascuna proprietà. Fornire un numero univoco per ogni serie di proprietà. In altre parole, le proprietà del filtro per il terzo attributo possono utilizzare 3_ come prefisso. Anche se il prefisso deve essere diverso per ogni serie di attributi, i nomi stessi delle proprietà devono essere identici a quello mostrati prima.

Gestione degli errori

CollaborationFoundation fornisce delle proprietà standard per la gestione degli errori. La Tabella 10 descrive queste proprietà.

Tabella 10. Proprietà standard per la gestione degli errori

Proprietà della gestione errori	Descrizione
INFORMATIONAL_EXCEPTION	Specifica se un oggetto di collaborazione deve terminare con esito positivo ed inviare l’eccezione ad una traccia oppure terminare con esito negativo e generare un’eccezione da registrare.
SEND_EMAIL	Specifica se un oggetto di collaborazione invia email ad un indirizzo specificato quando si verifica un’eccezione, a prescindere dal valore di INFORMATIONAL_EXCEPTION.

La “Processo e-mail per la gestione degli errori” a pagina 483 illustra il processo email della collaborazione per la gestione degli errori.

Per entrambe le proprietà di gestione errori è possibile specificare il comportamento di tutte le eccezioni o di una serie più piccola di numeri di messaggi delimitati da virgole. I numeri di messaggi corrispondono a quelli contenuti nel file di messaggi della collaborazione (collaborations\messages\CollaborationFoundation.txt).

Utilizzo della maschera CollaborationFoundation

Per utilizzare la maschera CollaborationFoundation eseguire le seguenti attività:

1. Copiare l’intera maschera CollaborationFoundation nella propria area di sviluppo.
2. Ridenominare la maschera di collaborazione per riflettere il tipo di collaborazione che si sta creando.
3. All’interno di Process Designer Express, utilizzare la scheda Porte e eventi di attivazione della finestra Definizioni maschera per modificare il tipo di porte esistenti (DestinationAppRetrieve, To, From) per assegnar loro gli oggetti business appropriati. Per ulteriori informazioni, fare riferimento a “Porte CollaborationFoundation” a pagina 34.
4. Creare ulteriori serie di proprietà del filtro se si desidera che la collaborazione filtri più di un attributo di oggetto business.

InterChange Server Express fornisce una serie di proprietà di configurazione che consentono di includere o escludere gli oggetti business dalla sincronizzazione in base a dei valori specificati in un attributo specificato. Se si desidera filtrare la sincronizzazione in base a dei valori in più di un attributo è necessario creare un'ulteriore serie di proprietà di configurazione per ogni attributo che si desidera valutare. Per ulteriori informazioni, consultare "Proprietà standard per le maschere di collaborazione" a pagina 484.

5. Se necessario, creare ulteriori funzioni di collaborazione aggiungendo del codice nei diagrammi secondari Ulteriore elaborazione. Questi diagrammi non contengono funzioni nella maschera CollaborationFoundation. Esistono per consentire una facile personalizzazione della collaborazione.
6. Salvare la maschera di collaborazione e compilarla.

Porte CollaborationFoundation

La Figura 9 mostra le porte di CollaborationFoundation. Le tabelle riportate di seguito forniscono le informazioni su ciascuna porta.

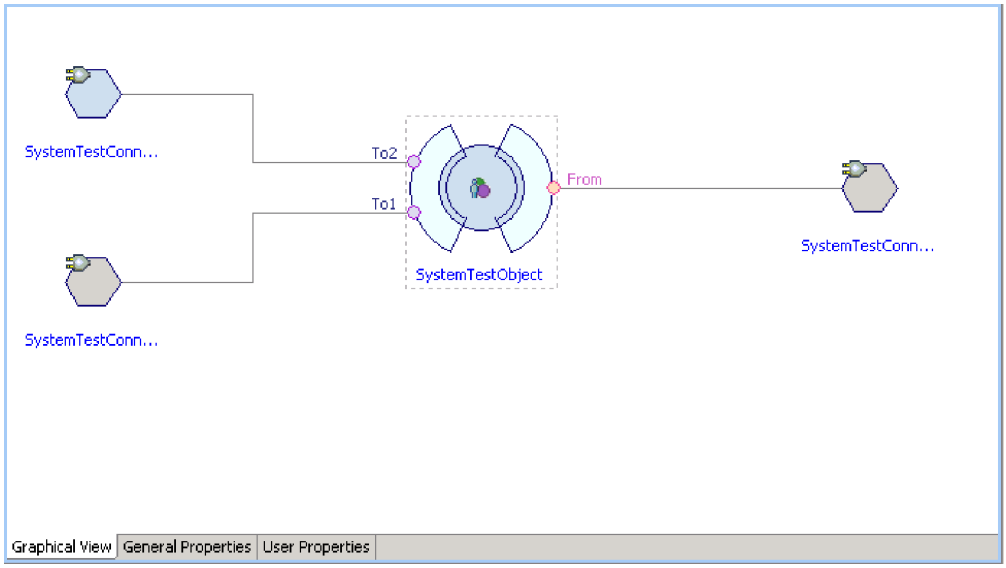


Figura 9. Porte della collaborazione CollaborationFoundation

La porta DestinationAppRetrieve

La Tabella 11 elenca le funzioni della porta DestinationAppRetrieve di CollaborationFoundation.

Tabella 11. Funzioni di porta (porta DestinationAppRetrieve)

Funzione della porta	Valore
Oggetto business	Un BusObj con valori di riferimento. Il valore predefinito è Controller.
Istruzioni utilizzate	Retrieve
Istruzioni di compensazione	Nessuna
Scopo	Richiama l'oggetto business con i valori completi dalla destinazione e lo utilizza per impostare l'istruzione.
Collegata a	Connettore dell'applicazione di destinazione

Tabella 11. Funzioni di porta (porta DestinationAppRetrieve) (Continua)

Funzione della porta	Valore
Proprietà di configurazione	USE_RETRIEVE

La porta To

La Tabella 12 elenca le funzioni della porta To di CollaborationFoundation.

Tabella 12. Funzioni di porta (porta To)

Funzione della porta	Valore
Oggetto business	Un BusObj con valori completi. Il valore predefinito è Controller.
Istruzioni utilizzate	Create, Update o Delete
Istruzioni di compensazione	Nessuna
Scopo	Invia un oggetto business al di fuori della collaborazione. Spesso la collaborazione invia una copia dell'oggetto business di attivazione attraverso questa porta alla destinazione.
Collegata a	Connettore dell'applicazione di destinazione o un'altra collaborazione (per ulteriori elaborazioni)
Proprietà di configurazione	Nessuna

La porta From

La Tabella 13 elenca le funzioni della porta From di CollaborationFoundation.

Tabella 13. Funzioni di porta (porta From)

Funzione della porta	Valore
Oggetto business	triggeringBusObj (il valore predefinito è Controller)
Istruzioni utilizzate	Create, Retrieve, Update o Delete
Istruzioni di compensazione	Nessuna
Scopo	Riceve l'oggetto business di attivazione
Collegata a	Applicazione origine o una collaborazione di chiamata
Proprietà di configurazione	Nessuna

Estensione della CollaborationFoundation

Questa sezione contiene le seguenti sezioni per semplificare la personalizzazione delle collaborazioni basate su CollaborationFoundation:

- "Raccomandazioni relative alla codifica"
- "Modifiche comuni" a pagina 39

Raccomandazioni relative alla codifica

Questa sezione descrive le metodologie di codifica per semplificare la standardizzazione del codice con quello delle collaborazioni fornite col prodotto.

- "Convenzioni di denominazione" a pagina 36
- "Elaborazione dell'attivazione del flusso" a pagina 36

Convenzioni di denominazione

E' buona norma stabilire le convenzioni di denominazione da utilizzare nelle maschere di collaborazione. L'elenco che segue fornisce alcune delle convenzioni di denominazione:

- Identificare ciascun tipo di variabile aggiungendo un prefisso al nome con una lettera significativa. Ad esempio, utilizzare come prefisso per un nome di variabile String la lettera "s"; come prefisso per un nome di variabile Boolean la lettera "b". Il seguente codice inizializza due variabili di questo genere:

  ```
  String sExceptionType
  Boolean bBranch
  ```

- Le proprietà di configurazione della collaborazione devono essere tutte a lettere maiuscole per distinguerle facilmente dalle variabili del programma. Il seguente codice ottiene il valore della proprietà SEND_EMAIL:

  ```
  bSendEmail = getConfigProperty("SEND_EMAIL");
  ```

Elaborazione dell'attivazione del flusso

Process Designer Express dichiara automaticamente una variabile di tipo BusObj denominata triggeringBusObj. Questa variabile conserva l'attivazione del flusso (di solito un evento di attivazione) che ha attivato l'esecuzione dello scenario.

Alcune situazioni possono richiedere di dover utilizzare l'attivazione del flusso anche dopo essere stata elaborata, ad esempio dovervi aggiungere dei dati dopo essere stata inviata all'esterno tramite le chiamate di servizio oppure dover modificare i valori degli attributi. Tali situazioni includono:

- Inviare l'attivazione del flusso all'esterno tramite chiamate di servizio per eseguire un rollback durante dei passi di compensazione di una collaborazione transazionale
- Confrontare i valori degli attributi dell'attivazione del flusso con quelli degli attributi di un oggetto business restituito da una chiamata di servizio o da una ricerca nel database

Per gestire queste situazioni, si raccomanda di creare una variabile BusObj intermedia che è una copia dell'attivazione del flusso, quindi modificare la variabile intermedia ed inviarla all'esterno tramite le chiamate di servizio, secondo le esigenze, piuttosto che modificare l'attivazione del flusso.

Nota: La creazione di copie degli oggetti business consuma risorse del sistema. Se il processo business non richiede una variabile intermedia (ad esempio, perché non ci sono requisiti transazionali e non si dovrà mai confrontare i valori degli attributi prima e dopo determinate situazioni), utilizzare l'attivazione del flusso, piuttosto che una copia, per conservare risorse.

Se la collaborazione è configurata per essere un processo business di lunga durata, tuttavia, il contenuto dell'oggetto business di attivazione flusso (triggeringBusObj) non viene conservato nelle chiamate di servizio. In questo caso, effettuare sempre una copia del flusso di attivazione.

Sono disponibili diverse API che consentono di copiare il contenuto di un oggetto business in un altro ed ognuna presenta dei vantaggi e degli svantaggi; le sezioni "Utilizzo del metodo copy()" e "Utilizzo del metodo duplicate()" a pagina 37 trattano ogni approccio.

Utilizzo del metodo copy(): Il metodo copy() può essere utilizzato per copiare il contenuto di una variabile di oggetto business in un'altra variabile di oggetto business dello stesso tipo. Si consiglia di tenere questo tipo di approccio perché lo

tengono le maschere di collaborazione fornite da InterChange Server Express, e la coerenza fra i componenti forniti e quelli creati su misura comporta una maggiore efficacia.

Per seguire questo approccio è necessario creare un'istanza di un nuovo oggetto BusObj dello stesso tipo dell'oggetto business di attivazione; si consiglia di eseguire la creazione dell'istanza nella definizione dello scenario e di denominare la variabile che memorizza la copia processingBusObj. Per soddisfare questi requisiti e raccomandazioni, aggiungere la seguente riga di codice alla definizione dello scenario:

```
BusObj processingBusObj;
processingBusObj = triggeringBusObj.duplicate();
```

Quindi eseguire il metodo copy() sulla variabile processingBusObj e passarle la variabile triggeringBusObj come argomento. E' buona norma effettuare queste operazioni nel primo nodo azione del diagramma di livello principale dello scenario, uno dedicato esclusivamente all'inizializzazione delle variabili. L'esempio di codice riportato qui sotto copia il contenuto della variabile triggeringBusObj nella variabile processingBusObj:

```
processingBusObj.copy(triggeringBusObj);
```

Utilizzo del metodo duplicate(): Per esempio, il frammento di codice riportato di seguito dichiara una variabile dello stesso tipo dell'attività di flusso ed imposta i suoi valori duplicando quelli contenuti nell'oggetto business dell'attivazione del flusso:

```
BusObj processingBusObj;
processingBusObj = triggeringBusObj.duplicate();
```

La collaborazione utilizza processingBusObj per manipolare i dati, come necessario. Quando è pronta per inviare i dati all'applicazione di destinazione, la collaborazione copia la variabile intermedia nella variabile ToBusObj. Utilizza ToBusObj nella sua chiamata di servizio per l'applicazione di destinazione. Il frammento di codice riportato di seguito mostra l'istruzione che copia i dati su ToBusObj:

```
ToBusObj.copy(processingBusObj);
```

Quando la chiamata di servizio torna con esito positivo alla collaborazione, la collaborazione copia i valori di ToBusObj su triggeringBusObj, come illustrato:

```
triggeringBusObj.copy(ToBusObj);
```

Le collaborazioni di InterChange Server Express di solito non modificano il valore originale di triggeringBusObj finché la collaborazione non riceve il ToBusObj restituito dalla porta To. L'utilizzo della variabile intermedia fa si che la collaborazione modifichi il valore di triggeringBusObj solo dopo aver ricevuto con esito positivo i valori dall'applicazione di destinazione.

Innalzamento delle eccezioni

Rilevare le eccezioni al livello in cui si verificano e innalzarle al processo principale della collaborazione. Rilevando l'eccezione è possibile specificare come gestire l'eccezione e controllare come appare all'utente; ad esempio, è possibile chiarire il contesto in cui si è verificata l'eccezione. Soprattutto, la creazione di nodi azione per la gestione dell'eccezione fornisce una documentazione visiva di ciascun punto nel codice in cui possono verificarsi le eccezioni.

In ogni collaborazione è necessario innalzare ciascuna eccezione catturata fino a che raggiunge l'ambiente di collaborazione runtime. Se si utilizza una chiamata di

servizio che attiva un'altra collaborazione, la collaborazione della chiamata deve verificare la presenza di eccezioni come risultato della chiamata di servizio.

Per innalzare il testo dell'eccezione ad un diagramma chiamante, dichiarare delle variabili string separate per memorizzare il testo del messaggio ed il tipo di eccezione. Ad esempio, il seguente codice dichiara due variabili string di quel tipo:

```
String sMessage
String sExceptionType
```

Utilizzare la funzione di diramazione per fornire un comportamento diverso quando la chiamata di servizio ha esito positivo o negativo. Nella diramazione che gestisce l'esito con l'errore, assegnare i valori in due variabili string. Ad esempio:

```
sMessage = currentException.getMessage();
sExceptionType = currentException.getType();
```

Prima di restituire il controllo al processo che ha effettuato la chiamata di servizio, innalzare l'eccezione. Ad esempio:

```
raiseException(ServiceCallException, 4000, SendRefBusObj.getType(),
   SendRefBusObj.getVerb(), SendRefBusObj.keysToString(),
   sExceptionType, sMessage);
```

Il codice citato specifica il messaggio di errore 4000, che è il messaggio di errore standard per l'esito negativo della collaborazione. Il file di messaggi include il seguente testo:

```
4000
Collaboration Failed: {1}.{2} with keys ({3}) synchronization failed
and the exception is {4}.{5}.
[EXPL]
The business object could not be synchronized in the destination.
```

Nel testo precedente, il metodo raiseException() sostituisce i valori mostrati nella Tabella 14.

Tabella 14. Valori sostituiti nella chiamata raiseException()

Variabile	Testo sostituito
{1}	Il valore che restituisce SendRefBusObj.getType()
{2}	Il valore che restituisce SendRefBusObj.getVerb()
{3}	Il valore che restituisce SendRefBusObj.keysToString()
{4}	Il valore nella variabile sExceptionType
{5}	Il valore nella variabile sMessage

Se il processo che effettua la chiamata di servizio *non* è il processo principale della collaborazione, deve innalzare l'eccezione al suo processo chiamante. Anche ogni processo al di sopra del processo chiamante deve innalzare l'eccezione in modo che il messaggio di errore possa essere registrato dal processo principale.

Diramazione

Il flusso di un diagramma di collaborazione spesso si basa sul valore di una proprietà di configurazione della collaborazione. La collaborazione può utilizzare il valore della proprietà per impostare una variabile boolean, che utilizza in seguito per determinare quale percorso prendere. Ad esempio, il seguente codice dichiara ed inizializza una variabile boolean denominata bBranch:

```
boolean bBranch = false;
```

InterChange Server Express imposta il valore della variabile di diramazione secondo le condizioni contenute nel codice. Queste condizioni possono essere basate sul valore di alcune variabili boolean. Ad esempio, si supponga che la collaborazione risolva la sua proprietà CONDITION_TWO solo se la proprietà CONDITION_ONE si risolve in `true`.

Il codice riportato di seguito basa una sezione sul valore di due variabili boolean:

- bCondition1, che contiene il valore configurato per la proprietà di collaborazione CONDITION_ONE
- bCondition2, che contiene il valore configurato per la proprietà di collaborazione CONDITION_TWO

Questo codice imposta il valore bBranch su `true` if CONDITION_ONE diviene `true` e CONDITION_TWO divienet `false`; imposta il valore di bBranch su `false` if CONDITION_ONE diviene `false` or CONDITION_TWO diviene `true`:

```
if (bCondition1 && !bCondition2)
   {
   bBranch = true;
   }
else
{
   bBranch = false;
}
```

Collaborazioni wrapper

Una *collaborazione wrapper* è una collaborazione che gestisce la verifica o la sincronizzazione di un oggetto business per un'altra collaborazione. La collaborazione chiamante invia alla collaborazione wrapper un oggetto business del livello principale, cui viene fatto riferimento nella sua stessa attivazione flusso.

Ad esempio, una collaborazione SalesOrderProcessing può sincronizzare l'oggetto business generico Order. Order generico contiene dei riferimenti ad un oggetto business generico Customer, che rappresenta il cliente che effettua l'ordine. Soprattutto, Order generico contiene un vettore di oggetti business generici OrderLineItem. Ogni OrderLineItem fa riferimento ad un oggetto business generico Item, che rappresenta gli articoli ordinati.

Per modularizzare la logica della collaborazione, è possibile fornire delle maschere di collaborazione separate per elaborare Order generico e gli oggetti business generici cui fa riferimento. Ad esempio, per elaborare un Order che fa riferimento agli oggetti business Customer e Item, è possibile fornire le seguenti maschere:

- SalesOrderProcessing — Elabora l'ordine.
- CustomerWrapper e CustomerSync — Elaborano il cliente cui si fa riferimento.
- ItemWrapper e ItemSync — Elabora gli articoli cui si fa riferimento.

La suddivisione dell'elaborazione degli oggetti business in diverse, specifiche collaborazioni non solo migliora il riutilizzo di ogni maschera di collaborazione ma impedisce anche che due collaborazioni modifichino gli stessi dati nello stesso momento. Per ulteriori informazioni, consultare "Problemi nell'elaborazione simultanea" a pagina 57.

Modifiche comuni

Questa sezione riguarda le seguenti comuni modifiche al codice di CollaborationFoundation:

- "Gestione dei dati dipendenti" a pagina 40

- "Delega della verifica dei dati dipendenti"
- "Esecuzione della verifica ripetitiva dei dati dipendenti" a pagina 45

Utilizzare le modifiche presentate in questa sezione per standardizzare le collaborazioni personalizzate con quelle fornite con il prodotto.

Gestione dei dati dipendenti

Si parla di oggetto business con dati dipendenti quando fa riferimento ad un altro oggetto business che influisce sulla propria elaborazione. Ad esempio, si supponga di voler sincronizzare in sede gli oggetti business Order, Customer e Item. Prima che l'oggetto collaborazione SalesOrderProcessing possa sincronizzare ciascun Order, è necessario verificare che il Customer cui si fa riferimento in Order esista già nell'applicazione di destinazione. Inoltre, si desidera che verificare che anche ogni Item menzionato da ciascun OrderLineItem esista nell'applicazione di destinazione e forse persino interrompere la sincronizzazione di Order finché non si è verificato che l'oggetto business cui si fa riferimento esiste. In un caso come questo, l'oggetto Order dipende dall'esistenza degli oggetti Customer e Item cui si fa riferimento.

Poiché InterChange Server Express fornisce diverse collaborazioni i cui oggetti business di attivazione possono fare riferimento ad un oggetto business Customer o Item, una singola installazione a volte esegue contemporaneamente più di un oggetto di collaborazione che sta tenta di sincronizzare gli stessi dati Customer o Item. Per mantenere la congruenza dei dati ed impedire che più collaborazioni modifichino contemporaneamente la stessa istanza dello stesso tipo di oggetto business, delegare la sincronizzazione degli oggetti business cui si fa riferimento a oggetti di collaborazione separati. Le collaborazioni richiamano le collaborazioni intermedie wrapper per gestire la verifica o la sincronizzazione degli oggetti business cui si fa riferimento nei relativi oggetti business di attivazione. Le collaborazioni wrapper facilitano l'isolamento dei dati in ambienti che eseguono più collaborazioni.

Utilizzare le collaborazioni wrapper per mantenere la congruenza dei dati in un ambiente che esegue più collaborazioni. Ad esempio, si supponga che due oggetti di collaborazione CustomerSync siano in esecuzione sulla stessa installazione. Uno è in esecuzione insieme a SalesOrderProcessing, mentre l'altro si esegue in maniera indipendente. In questo scenario, l'oggetto collaborazione indipendente CustomerSync sincronizza i dati di Customer fra le stesse due applicazioni dell'oggetto di collaborazione SalesOrderProcessing. Se si utilizza SalesOrderProcessing per delegare la sincronizzazione degli oggetti business Customer con valori di riferimento, con l'oggetto business intermedio CustomerWrapper, il software fornisce l'isolamento dell'evento. In altre parole, il sistema non consente all'oggetto di collaborazione indipendente CustomerSync di utilizzare lo stesso oggetto business contemporaneamente all'oggetto di collaborazione CustomerSync richiamato da SalesOrderProcessing.

Per ulteriori informazioni sulla sincronizzazione degli oggetti business menzionati da un oggetto business di attivazione, consultare "Problemi nell'elaborazione simultanea" a pagina 57.

Delega della verifica dei dati dipendenti

E' possibile estendere CollaborationFoundation per delegare una verifica di dipendenza ad un'altra collaborazione. Per fare ciò è necessario eseguire le seguenti attività:

- Aggiungere una porta che verrà collegata alla collaborazione chiamata.

- Aggiungere una proprietà di configurazione collaborazione che consente di specificare se la collaborazione verifica o sincronizza oggetti business cui si fa riferimento. Se si sceglie di sincronizzare gli oggetti business Item o Contact cui si fa riferimento, potrebbe essere necessario creare ulteriori proprietà di configurazione.
- Aggiungere i messaggi che forniscono le informazioni nel caso in cui la verifica o la sincronizzazione abbiano esito negativo.
- Aggiungere uno scenario per elaborare la verifica di dipendenza.

Aggiunta di una porta: Per delegare i dati dipendenti, creare una porta per ogni oggetto business dipendente. Mantenere la congruenza con le altre collaborazioni fornite con il prodotto denominando la nuova porta To *BOName*Wrapper, dove *BOName* si riferisce all'oggetto business dipendente, con valori di riferimento. Ad esempio, SalesOrderProcessing utilizza una porta denominata ToCustomerWrapper per inviare a CustomerWrapper gli oggetti business con valori di riferimento Customer.

Creare questo tipo di porta per ogni oggetto business con valori di riferimento da inviare ad una collaborazione wrapper. Ad esempio, anche SalesOrderProcessing utilizza le porte denominate SendContactRef e SendItemRef.

Aggiunta di una proprietà di configurazione: Per delegare la verifica di un oggetto business di riferimento ad un'altra collaborazione, creare una nuova proprietà di configurazione di collaborazione. Questa proprietà consente alla persona che configura la collaborazione di specificare se la collaborazione deve verificare o sincronizzare l'oggetto business di riferimento.

Per mantenere la congruenza con le collaborazioni fornite con il prodotto, denominare la nuova proprietà VERIFY_SYNC_*BOName*, dove *BOName* è l'oggetto business con valori di riferimento. Ad esempio, SalesOrderProcessing contiene una proprietà denominata VERIFY_SYNC_CUSTOMER, che utilizza per delegare la verifica di un oggetto business Customer. Stabilire i possibili valori di questa proprietà nel modo seguente:

- `neither` — impedisce la delega.
- `sync` — Fa sì che la collaborazione invii l'oggetto business con valori di riferimento con l'istruzione Sync alla collaborazione wrapper appropriata per la sincronizzazione.
- `verify` — fa si che la collaborazione invii l'oggetto business con valori di riferimento con l'istruzione Exists alla collaborazione wrapper appropriata per la verifica.

Per impostazione predefinita, il sistema imposta una proprietà VERIFY_SYNC_*BOName* su `neither`.

Proprietà per la gestione degli errori: Quando una collaborazione delega un vettore di oggetti business con valori di riferimento, deve essere preparato a gestire l'esito negativo della sincronizzazione o della verifica di qualsiasi elemento nel vettore. Le collaborazioni fornite con il prodotto dispongono di proprietà di configurazione che consentono di specificare in che modo gestire l'esito negativo di elementi individuali di un vettore durante la verifica o la sincronizzazione.

Ad esempio, l'oggetto business generico Order contiene un vettore di oggetti business secondai OrderLineItem;ognuno di questi oggetti OrderLineItem può fare riferimento ad un oggetto business Item. Pertanto la collaborazione che gestisce l'oggetto business Order può delegare un vettore di oggetti business Item.

Similarmente, l'oggetto business generico Order contiene un vettore di oggetti business secondari OrderContactRef e ciascun OrderContactRef può far riferimento ad un oggetto business Contact. Quindi, la collaborazione che gestisce l'oggetto business Order può delegare un vettore di oggetti business Contact.

La collaborazione SalesOrderProcessing, che gestisce gli oggetti business Order, fornisce le proprietà di configurazione che consentono di specificare cosa accade quando uno o più oggetti business con riferimenti, contenuti in un vettore, falliscono la sincronizzazione o la verifica.

Un opzione è di interrompere la sincronizzazione dell'oggetto business di attivazione, se un qualunque oggetto business di un vettore delegato di oggetti business con riferimenti fallisce la sincronizzazione o la verifica. Un'altra opzione è di rimuovere dal vettore l'oggetto business che avuto l'esito negativo e continuare la sincronizzazione dell'oggetto business di attivazione.

Una collaborazione valuta le opzioni per la gestione di un oggetto business con valori di riferimento con esito negativo solo se sta tentando di verificare o sincronizzare quell'oggetto business. Pertanto, la collaborazione deve valutare una proprietà che fornisce queste opzioni solo se la sua proprietà corrispondente VERIFY_SYNC_*BOName* si risolve in `verify` o `sync`.

Le collaborazioni fornite con InterChange Server Express hanno due proprietà di configurazione che forniscono le opzioni quando un oggetto business con riferimento, contenuto in un vettore, fallisce la verifica o la sincronizzazione. Una di queste proprietà gestisce gli oggetti business Contact con esito negativo; l'altra gestisce gli oggetti business Item con esito negativo.

L'esempio che segue dimostra come viene gestito un vettore di oggetti business Contact. In questo esempio, sia SalesOrderProcessing che InstalledProductSync delegano vettori di oggetti business con valori di riferimento Contact. Entrambe utilizzano la proprietà FAIL_ON_CONTACT_ERROR per determinare il comportamento della collaborazione quando la collaborazione richiamata non riesce a verificare o a sincronizzare uno degli oggetti business Contact contenuti nel vettore.

Se si modifica CollaborationFoundation per delegare un vettore di oggetti Contact, mantenere la congruenza con le altre collaborazioni codificando la collaborazione per valutare FAIL_ON_CONTACT_ERROR solo se VERIFY_SYNC_CONTACT si risolve in `verify` o `sync` e se ContactWrapper non riesce a verificare o sincronizzare uno degli oggetti Contact di riferimento. Le collaborazioni fornite con il prodotto eseguono le seguenti operazioni in base al valore di FAIL_ON_CONTACT_ERROR:

- True — La collaborazione interrompe l'elaborazione.
- False — La collaborazione continua l'elaborazione e rimuove dal vettore di oggetti business secondari ogni Contact non riuscito.

Per impostazione predefinita, la proprietà FAIL_ON_CONTACT_ERROR viene impostata su true.

I paragrafi che seguono dimostrano come vengono gestiti i vettori di oggetti business Item. Gli oggetti business InterChange Server Express possono contenere un vettore di riferimenti agli oggetti business Item e Contact. Quindi ogni Order generico può far riferimento ad un vettore di oggetti generici Item; è possibile configurare SalesOrderProcessing per delegare a ItemWrapper un vettore di oggetti business Item con riferimenti.

Per gestire gli errori di un singolo Item che fallisce la verifica o la sincronizzazione, SalesOrderProcessing utilizza la proprietà FIND_ALL_ITEM_ERRORS. Lae collaborazione valuta FIND_ALL_ITEM_ERRORS solo se VERIFY_SYNC_ITEM si risolve in `verify` o `sync` e se ItemWrapper non riesce a sincronizzare o verificare uno degli oggetti business Item con riferimenti.

La Figura 10 illustra il modo in cui CollaborationFoundation gestisce un vettore di oggetti business Item con riferimento quando VERIFY_SYNC_ITEM si risolve in `sync` e FIND_ALL_ITEM_ERRORS si risolve in `true`.

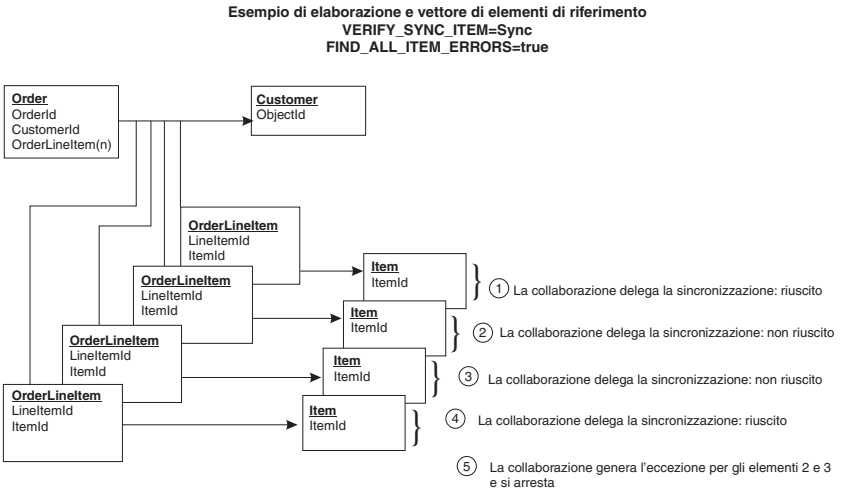


Figura 10. Esempio di elaborazione di un vettore di oggetti business Item con riferimenti

Le collaborazioni eseguono le seguenti operazioni in base al valore di FIND_ALL_ITEM_ERRORS:

- True — La collaborazione continua l'elaborazione e l'invio di oggetti business Item con riferimenti. Dopo aver gestito l'ultimo Item del vettore, la collaborazione innalza un'eccezione per ogni oggetto Item non riuscito ed interrompe l'elaborazione. Questa impostazione abilita un amministratore a ricevere tutti gli errori Item in una volta.
- False — La collaborazione interrompe l'elaborazione. L'impostazione richiede che un amministratore riavvii la collaborazione dopo ogni errore Item. L'amministratore deve trovare e correggere ogni errore.

Per impostazione predefinita, il valore di FIND_ALL_ITEM_ERRORS è `false`.

Elaborazione di più tipi di voci: InterChange Server Express fornisce gli oggetti business generici Item elencati nella Tabella 15.

Tabella 15. Oggetti business generici Item

Oggetto business generico Item	**Descrizione**
ItemBasic	Contiene gli attributi dei dati comuni a tutte le organizzazioni logiche che utilizzano i dati dell'elemento. Viene considerato un prerequisito per altri oggetti business generici Item di InterChange Server Express.
ItemOrder	Contiene gli attributi dei dati che catturano la serie di campi mantenuti per una specifica entità organizzativa di gestione ordine.
ItemPlanning	Contiene gli attributi di dati che riguardano la pianificazione dei requisiti futuri di un elemento e l'approvvigionamento.

Tabella 15. Oggetti business generici Item (Continua)

Oggetto business generico Item	Descrizione
Item	Un oggetto business semplice che rappresenta i dati dell'elemento. Alcune collaborazioni utilizzano questo oggetto business come segnaposto. L'Item generico consente alle collaborazioni di determinare al runtime il tipo effettivo di oggetto business Item da elaborare. Pertanto, l'Item generico *deve* esistere su un'installazione per poter collegare un connettore ad una collaborazione che elabora ItemBasic, ItemOrder e ItemPlanning generici.

Se un oggetto business di attivazione può fare riferimento ad un oggetto business Item, è importante consentire alla persona che configura l'oggetto collaborazione a specificare quale tipo di Item di riferimento verificare o sincronizzare. La proprietà ITEM_TYPE fornisce questa funzione.

Se la collaborazione creata da CollaborationFoundation ha un oggetto business di attivazione che fa riferimento a oggetti business Item, estendere la collaborazione all'utilizzo della proprietà ITEM_TYPE. Definire la sua serie di valori in modo da includere ogni tipo di oggetto business Item disponibile sul sistema.

Per impostazione predefinita, la proprietà ITEM_TYPE viene impostata su `Item`.

Aggiunta di messaggi: Per esaminare o modificare i messaggi di CollaborationFoundation, modificare il relativo file di messaggi (collaborations\messages\CollaborationFoundation.txt). Mantenere la congruenza con le altre collaborazioni di InterChange Server Express creando dei messaggi che forniscono le informazioni se un aspetto della verifica di dipendenza dovesse non riuscire.

Poiché le collaborazioni wrapper e di sincronizzazione di InterChange Server Express forniscono dei propri messaggi di errore, le collaborazioni sviluppate da CollaborationFoundation non necessitano di contenere dei messaggi di errore specifici per ogni sincronizzazione o verifica non riuscita degli oggetti business correlati. Tuttavia, queste collaborazioni personalizzate devono gestire esplicitamente i messaggi ricevuti dalle collaborazioni richiamate. Per ulteriori informazioni sulla gestione di questi messaggi, consultare "Innalzamento delle eccezioni" a pagina 37.

Aggiunta di scenari: Per delegare la verifica di un oggetto business di riferimento ad un'altra collaborazione, è necessario creare uno scenario di collaborazione che valuta se la collaborazione deve verificare o sincronizzare l'oggetto business di riferimento.

Lo scenario deve verificare il valore della proprietà VERIFY_SYNC_*BOName* della collaborazione per stabilire se la collaborazione è stata configurata per verificare o sincronizzare l'oggetto business di riferimento. E' anche possibile far eseguire allo scenario altre verifiche preliminari dell'oggetto business di riferimento, prima che richiami una collaborazione wrapper.

Ad esempio, il codice riportato di seguito migliora le prestazioni impedendo alla collaborazione di richiamare la collaborazione CustomerWrapper a meno che non sia veramente necessario:

```
// Get the value of the configuration property
String vsc = getConfigProperty("VERIFY_SYNC_CUSTOMER");

// By default, do not branch to Wrapper call.
bBranch= false;

// If VERIFY_SYNC_CUSTOMER evaluates to "neither" or
// if the source application's business object does not contain the
// Customer's primary key, do not change the value of bBranch.
if (vsc.equals("neither") || (processingBusObj.isNull("CustomerID")))"
     {}
else
     {
     // Get the Customer's primary key from the source
     // and destination application's business objects.
     // Note: The DestinationAppRetrieveBusObj variable
     // contains a value only if the USE_RETRIEVE property evaluates
     // to "true" and has already been processed.
     String sc=processingBusObj.getString("CustomerID");
     String dc=DestinationAppRetrieveBusObj.getString("CustomerID");

     // If the Customer's primary key is the same in both the
     // source and destination application (therefore, no need to
     // synchronize), the verb is Update, and USE_RETRIEVE evaluates
     // to "true", do not change the value of bBranch.
     if (sVerb.equals("update") && bUseRetrieve && sc.equals.(DC))
          {}
     else
          {
          // If the verb is Create, or if the Customer's primary
          // keys are not identical and the verb is Update, call
          // the Wrapper collaboration.
          bBranch = true;
          }
}
```

Per un esempio di maschera di collaborazione che include il codice precedentemente menzionato, vedere SalesOrderProcessing. Il relativo diagramma secondario Additional processing 3 include del codice che controlla i valori delle proprietà VERIFY_SYNC_CUSTOMER, VERIFY_SYNC_ITEM e VERIFY_SYNC_CONTACT per stabilire quale percorso seguire fra i diversi presenti.

Poiché SalesOrderProcessing elabora il diagramma USE_RETRIEVE prima di elaborare il diagramma secondario Additional processing 3, la variabile DestinationAppRetrieveBusObj può già contenere l'oggetto business dell'applicazione di destinazione.

Esecuzione della verifica ripetitiva dei dati dipendenti

La maggior parte degli oggetti business fanno riferimento ad altri oggetti business i cui tipi sono diversi dai propri. Ad esempio, Order fa riferimento a Customer, Contact e Item. Order non fa riferimento ad un altro oggetto business Order. Poiché Order non è l'unico oggetto business generico che può fare riferimento agli oggetti Customer, Contact e Item objects, la collaborazione che modifica Order deve delegare l'elaborazione degli oggetti business di riferimento alle collaborazioni wrapper appropriate. Per mantenere la congruenza dei dati negli ambienti che eseguono più collaborazioni, non vi sono due collaborazioni fornite col prodotto che modificano lo stesso tipo di oggetto business.

La discussione sulla modifica degli oggetti business diventa più complicata, tuttavia, quando si analizzano le collaborazioni che modificano gli oggetti business InstalledProduct e Item.

Gli oggetti business InstalledProduct ed i relativi elementi principali InstalledProduct: Come per Order, un oggetto business InstalledProduct può far riferimento a Customer, Contact e Item. Diversamente da qualsiasi altro oggetto generico business, tuttavia, InstalledProduct può anche far riferimento ad un altro InstalledProduct ed al suo elemento principale. La Figura 11 illustra la relazione fra più oggetti business InstalledProduct.

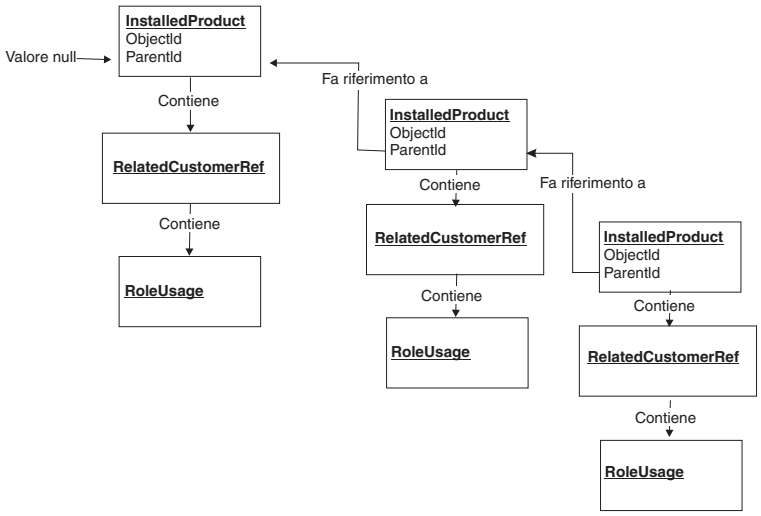


Figura 11. Relazione fra più oggetti business InstalledProduct

La Figura 11 mostra che l'elemento principale InstalledProduct non contiene il suo elemento secondario InstalledProduct. Invece, l'elemento secondario contiene un riferimento al suo elemento principale nell'attributo ParentId, il cui tipo è String. Qualsiasi InstalledProduct con un valore null nell'attributo ParentId è in cima alla gerarchia del prodotto.

Oggetti business Item e relativi oggetti Item prerequisiti: Anche gli oggetti business Item introducono un nuovo livello di complessità alla modifica dell'oggetto business. Come mostrato nella Tabella 15 a pagina 43, InterChange Server Express fornisce una famiglia di oggetti business Item. Il sistema di integrazione business gestisce gli oggetti Item in maniera flessibile; le collaborazioni utilizzano il valore della loro proprietà ITEM_TYPE per stabilire quale tipo di Item sono stati configurati a gestire.

Anche se un Item non fa riferimento ad un altro come elemento principale, un oggetto business Item può fare riferimento ad un altro oggetto business della famiglia Item come suo prerequisito. Ad esempio, ItemOrder e ItemPlanning possono entrambi fare riferimento a ItemBasic come prerequisito. Per impostazione predefinita, le proprietà PREQ_ITEMORDER e PREQ_ITEMPLANNING della collaborazione ItemSync specificano ItemBasic come prerequisito per entrambi gli oggetti business Item.

Elaborazione ripetitiva degli oggetti business dello stesso tipo o o famiglia del tipo: Quando un oggetto business fa riferimento ad un altro oggetto business dello stesso tipo o famiglia del tipo, la stessa collaborazione che modifica l'oggetto business di attivazione può modificare l'oggetto business di riferimento. In altre parole, la collaborazione InstalledProductSync deve delegare la gestione dei suoi oggetti di riferimento Customer, Contact e Item dell'oggetto business di attivazione.

Tuttavia, InstalledProductSync può sincronizzare non solo il suo InstalledProduct di attivazione ma anche l'elemento principale InstalledProduct dell'oggetto business di attivazione, e l'elemento principale dell'elemento principale. Soprattutto, ItemSync può sincronizzare non solo il suo ItemOrder di attivazione ma anche ItemBasic prerequisito dell'oggetto business di attivazione. Se il prerequisito dell'oggetto business di attivazione ha un suo proprio prerequisito, ItemSync li può sincronizzare tutti.

InterChange Server Express fornisce due collaborazioni che elaborano in maniera ripetitiva uno stack di oggetti business:

- InstalledProductSync — Crea uno stack di oggetti business InstalledProduct principali iniziando dal basso con l'oggetto business di attivazione e risalendo verso l'oggetto principale. La collaborazione quindi sincronizza gli oggetti business InstalledProduct, spostandosi nello stack dall'alto verso il basso. Ad ogni nodo nella gerarchia, la collaborazione delega la gestione di tutti gli oggetti business di riferimento che non sono di tipo InstalledProduct *prima* di sincronizzare realmente l'oggetto business InstalledProduct. In altre parole, InstalledProductSync può, in maniera ripetitiva, sincronizzare più oggetti business utilizzando il metodo LIFO (Last-In-First-Out).

  Ad esempio, si supponga che un'azienda che costruisce macchine gestisce i componenti come prodotti installati separati all'interno di una gerarchia di prodotti. In questo caso, l'iniettore di carburante viene gestito come un prodotto installato, l'elemento principale dell'iniettore di carburante (il motore) viene gestito da un prodotto installato e l'elemento principale del motore (l'automobile) viene gestito da un prodotto installato. Per sincronizzare l'iniettore di carburante, InstalledProductSync costruisce uno stack; l'iniettore di carburante si trova in fondo allo stack. La collaborazione colloca la macchia in cima allo stack. Quando si esegue la sincronizzaizone, InstalledProductSync sincronizza l'automobile prima e l'iniettore di carburante per ultimo.

  InstalledProductSync si basa sulla supopsizione che l'elemento principale di un oggetto InstalledProduct deve esistere *prima* che venga creato InstalledProduct, come deve esistere un capo prima che venga assunto un impiegato.
- ItemSync — si muove lungo una gerarchia di oggetti business Item prerequisiti, rimuovendo un oggetto business dalla gerarchia se esiste già il suo oggetto business prerequisito nell'applicazione di destinazione. In altre parole, ItemSync esegue una prima ricerca approfondita quando esegue la sincronizzazione in maniera ripetitiva degli oggetti business prerequisiti nella gerarchia.

  Il processo di ItemSync differisce dal processo InstalledProductSync nel seguente modo:

  – ItemSync può gestire qualsiasi numero di dipendenze. La collaborazione è progettata per semplificare l'elaborazione di oggetti business generici sviluppati nell'installazione di un utente. Con modifiche minime, ItemSync può accettare gli oggetti business definiti dall'utente come oggetti business di attivazione o come prerequisiti.
  – Man mano che si sposta verso la parte bassa dell'elenco di prerequisiti, ItemSync valuta ogni prerequisito non riuscito per stabilire se il prerequisito ha un altro prerequisito sopra di lui nella gerarchia, che è stato recuperato con esito positivo. Se è così, ItemSync mantiene nell'elenco solo il prerequisito non riuscito.

  Se si prevede di modificare CollaborationFoundation per eseguire l'elaborazione ripetitiva, esaminare il codice delle collaborazioni InstalledProductSync e ItemSync. E' possibile copiare gli scenari di quelle collaborazioni nella collaborazione che si crea da CollaborationFoundation. E' possibile quindi

modificare gli scenari per soddisfare le esigenze specifiche. Per ulteriori informazioni su questa collaborazioni, consultare le pagine di riferimento di InstalledProductSync e ItemSync.

Maschera WrapperFoundation

La maschera di collaborazione WrapperFoundation è uno strumento che semplifica lo sviluppo delle collaborazioni wrapper definite dall'utente che aderiscono agli standard InterChange Server Express. Una collaborazione wrapper è una collaborazione che gestisce la verifica o la sincronizzazione di un oggetto business per un'altra collaborazione. La collaborazione chiamante invia alla collaborazione wrapper un oggetto business del livello principale, cui viene fatto riferimento nel suo stesso oggetto business di attivazione.

Ad esempio, la collaborazione SalesOrderProcessing sincronizza l'oggetto business generico Order. Order generico contiene dei riferimenti ad un oggetto business generico Customer, che rappresenta il cliente che effettua l'ordine. Soprattutto, Order generico contiene un vettore di oggetti business generici OrderLineItem. Ogni OrderLineItem fa riferimento ad un oggetto business generico Item, che rappresenta gli articoli ordinati.

Per modularizzare la logica della collaborazione in questa situazione, InterChange Server Express fornisce delle maschere di collaborazione separate per elaborare l'Order generico e gli oggetti business generici cui fa riferimento. Ad esempio, InterChange Server Express fornisce le seguenti maschere per elaborare un oggetto business Order che fa riferimento agli oggetti business Customer e Item:

- SalesOrderProcessing — Elabora l'ordine.
- CustomerWrapper e CustomerSync — Elaborano il cliente di riferimento.
- ItemWrapper e ItemSync — Elaborano gli articoli di riferimento.

La suddivisione dell'elaborazione degli oggetti business in diverse, specifiche collaborazioni non solo migliora il riutilizzo di ogni maschera di collaborazione ma impedisce anche che due collaborazioni modifichino gli stessi dati nello stesso momento. Per ulteriori informazioni, consultare "Problemi nell'elaborazione simultanea" a pagina 57.

Per mantenere la congruenza in tutta la sincronizzazione ed accedere alle collaborazioni, InterChange Server Express crea le maschere per queste collaborazioni dalla maschera CollaborationFoundation. Per mantenere la congruenza in tutte le collaborazioni wrapper, tutte le maschere di collaborazione wrapper sono create dalla maschera WrapperFoundation. E' possibile utilizzare WrapperFoundation per sviluppare le proprie collaborazioni wrapper.

Funzioni di WrapperFoundation

Un oggetto di collaborazione wrapper generato dalla maschera WrapperFoundation può verificare l'esistenza del suo oggetto business di attivazione nell'applicazione di destinazione o semplificare la sua sincronizzazione. Questa sezione contiene i seguenti argomenti:

- "Elaborazione della verifica"
- "Elaborazione della sincronizzazione" a pagina 49

Elaborazione della verifica

Quando una collaborazione wrapper riceve il suo oggetto business di attivazione con valori di riferimento, con l'istruzione Exists, verifica l'esistenza del suo oggetto

business di attivazione recuperandolo dall'applicazione di destinazione. La collaborazione wrapper utilizza l'istruzione Retrieve per questa verifica.

Se la verifica ha esito positivo, la collaborazione wrapper restituisce alla collaborazione chiamante uno stato di esito positivo.

Se la verifica non riesce, il comportamento della collaborazione wrapper dipende dall'impostazione della proprietà di configurazione CONTINUE_WITH_WARNING.

Per ulteriori informazioni e una illustrazione del processo di verifica, consultare la guida per l'utente della specifica collaborazione wrapper che si sta utilizzando.

Elaborazione della sincronizzazione

Quando una collaborazione wrapper riceve il suo oggetto business di attivazione con valori di riferimento, con l'istruzione Sync, semplifica la sincronizzazione dell'oggetto business. Lo fa recuperando tutti i valori dall'applicazione origine ed inviando l'oggetto business completo con i valori con l'istruzione Create all'appropriata collaborazione di sincronizzazione.

Se la sincronizzazione ha esito positivo nella creazione dell'oggetto business nell'applicazione di destinazione, la collaborazione wrapper restituisce alla sua collaborazione chiamante uno stato di esito positivo.

Se la collaborazione di sincronizzazione non riesce a creare l'oggetto business nell'applicazione di destinazione, il comportamento della collaborazione wrapper dipende dall'impostazione della proprietà CONTINUE_WITH_WARNING.

Per impedire che la collaborazione di sincronizzazione richiamata tenti di creare un oggetto esistente e restituisca alla collaborazione wrapper uno stato di esito negativo, utilizzare le seguenti impostazioni per la collaborazione di sincronizzazione:

- Impostare la proprietà di configurazione USE_RETRIEVE su `true`.
- Impostare la proprietà di configurazione INFORMATIONAL_EXCEPTIONS su `3010`.

Per ulteriori informazioni e una illustrazione del processo di sincronizzazione, consultare la guida per l'utente della specifica maschera di collaborazione wrapper che si sta utilizzando.

Utilizzo della maschera WrapperFoundation

Eseguire le seguenti attività quando si desidera utilizzare la maschera WrapperFoundation:

1. Copiare l'intera maschera WrapperFoundation nella proria area di sviluppo.
2. Ridenominare la maschera per riflettere il tipo di collaborazione che si sta creando.
3. All'interno di Process Designer Express, utilizzare la scheda Porte e eventi di attivazione della finestra Definizioni maschera per modificare il tipo di porte esistenti (DestinationAppRetrieve, To, SourceApp e From) per assegnar loro gli oggetti business appropriati. Ad esempio, per creare unamaschera InvoiceWrapper è necessario modificare l'oggetto business di ogni porta dal valore predefinito (Controller) all'oggetto business Invoice. Per ulteriori informazioni, consultare "Porte WrapperFoundation" a pagina 50.

Porte WrapperFoundation

La Figura 12 illustra le porta di WrapperFoundation. Le tabelle che seguono la figura forniscono le informazioni su ciascuna porta.

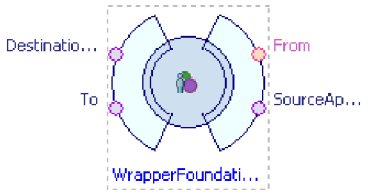


Figura 12. Porte della collaborazione WrapperFoundation

Porta DestinationAppRetrieve di WrapperFoundation

La Tabella 16 elenca le funzioni della porta DestinationAppRetrieve della maschera WrapperFoundation.

Tabella 16. Funzioni di porta per la porta DestinationAppRetrieve di WrapperFoundation

Funzione della porta	**Valore**
Oggetto business	L'oggetto business per il quale viene denominata la collaborazione wrapper. Ad esempio, CustomerWrapper invia un oggetto business con valori di riferimento Customer. Questo oggetto business viene impostato sul valore predefinito Controller.
Istruzioni utilizzate	Retrieve
Scopo	Recupera l'oggetto business dall'applicazione di destinazione
Collegata a	Connettore dell'applicazione di destinazione

Porta From di WrapperFoundation

La Tabella 17 elenca le funzioni della porta From della maschera WrapperFoundation.

Tabella 17. Funzioni di porta della porta From di WrapperFoundation

Funzione della porta	**Valore**
Oggetto business	L'oggetto business per il quale viene denominata la collaborazione wrapper. Ad esempio, CustomerWrapper viene avviato da un oggetto business Customer. Il valore predefinito è Controller.
Istruzioni utilizzate	Sync, Exists
Scopo	Riceve l'oggetto business di attivazione dalla collaborazione chiamante
Collegata a	La porta To*BusObj*Wrapper della collaborazione chiamante

Porta SourceApp di WrapperFoundation

La Tabella 18 a pagina 51 elenca le funzioni della porta SourceApp della maschera WrapperFoundation.

Tabella 18. Funzioni di porta della porta SourceApp di WrapperFoundation

Funzione della porta	Valore
Oggetto business	L'oggetto business per il quale viene denominata la collaborazione wrapper. Ad esempio, CustomerWrapper recupera un oggetto business Customer dall'applicazione origine. L'oggetto business viene impostato sul valore predefinito Controller.
Istruzioni utilizzate	Retrieve
Scopo	Recupera l'oggetto business di attivazione dall'applicazione origine
Collegata a	Connettore dell'applicazione origine

Porta To di WrapperFoundation

La Tabella 19 elenca le funzioni della porta To della maschera WrapperFoundation.

Tabella 19. Funzioni di porta per la porta To di WrapperFoundation

Funzione della porta	Valore
Oggetto business	L'oggetto business per il quale viene denominata la collaborazione wrapper. Ad esempio, CustomerWrapper invia un oggetto business con valori di riferimento Customer. Questo oggetto business viene impostato sul valore predefinito Controller.
Istruzioni utilizzate	Create
Scopo	Invia un oggetto business al di fuori della collaborazione. Di solito utilizzata per inviare un oggetto business completo con valori alla collaborazione di sincronizzazione di pertinenza
Collegata a	La porta From della collaborazione che si sincronizza o o si sottoscrive all'oggetto business

Estensione di WrapperFoundation

CustomerPartnerWrapper e ItemWrapper forniscono degli esempi di collaborazioni InterChange Server Express che sono state create da WrapperFoundation e sono state modificate per soddisfare delle esigenze specifiche. Questa sezione descrive quelle modifiche sotto forma di esempi come si può modificare la maschera WrapperFoundation.

CustomerPartnerWrapper

Poiché diverse applicazioni utilizzano i dati di CustomerPartner in modi significativamente differenti, CustomerPartnerWrapper gestisce le chiavi dell'oggetto business in un modo univoco. Alcune applicazioni richiedono che CustomerPartner contenga l'identificativo del suo oggetto Customer associato, oltre al proprio identificativo. Altre applicazioni non richiedono l'identificativo dell'oggetto Customer e non forniscono quell'identificativo quando inviano l'oggetto business al sistema di integrazione business InterChange Server Express.

Se una collaborazione riceve un oggetto business CustomerPartner da un sistema che non fornisce l'identificativo Customer ma deve sincronizzare l'oggetto ad un sistema che richiede l'identificativo, la collaborazione deve ottenere dei dati aggiuntivi e fornirli all'applicazione di destinazione. Per questo motivo CustomerPartnerWrapper *non* utilizza il comando fornito da WrapperFoundation per impostare le chiavi dell'oggetto business:

```
SourceAppBusObj.setKeys(triggeringBusObj);
```

Invece di usare il comando appena riportato, che WrapperFoundation fornisce nel nodo Recupera da origine, dello scenario Sync, CustomerPartnerWrapper ottiene i valori della chiave tramite le seguenti due istruzioni separate:

```
SourceAppBusObj.set("ObjectId",
     triggeringBusObj.getString("ObjectID"));
SourceAppBusObj.set("AdditionalKey",
     triggeringBusObj.getString("AdditionalKey"));
```

Ubicate anche loro nel nodo Recupera da origine nello scenario Sync, queste istruzioni utilizzano entrambi gli attributi ObjectId e AdditionalKey come potenziale chiave.

Nel nodo Prepara oggetto per collaborazione Sync, dello scenario Sync, WrapperFoundation contiene il seguente codice:

```
ToBusObj =(SourceAppBusObj).duplicate();
ToBusObj.setKeys(triggeringBusObj);
```

nel nodo equivalente, CustomerPartnerWrapper utilizza due istruzioni separate per impostare la chiave per ToBusObj:

```
ToBusObj =(SourceAppBusObj).duplicate();
ToBusObj.set("ObjectId", triggeringBusObj.getString("ObjectId"));
ToBusObj.set("AdditionalKey", triggeringBusObj.getString("AdditionalKey"));
```

ItemWrapper

La Tabella 15 a pagina 43 elenca i quattro diversi tipi di oggetti business Item generici distribuiti con InterChange Server Express. Si presuppone che l'utente prevede di definire dei propri tipi di oggetti business Item generici per soddisfare delle necessità specifiche del proprio ambiente. Quindi, la collaborazione ItemSync è stata progettata per essere facilmente estesa per gestire degli oggetti business item di attivazione e prerequisiti definiti dall'utente. Per ulteriori informazioni, fare riferimento alla guida per l'utente della collaborazione ItemSync.

Poiché ItemWrapper può essere attivato da qualsiasi tipo di oggetto business Item generico, la collaborazione deve prima stabilire il tipo di oggetto business che l'ha attivata. Quindi le Definizioni scenario di ItemWrapper di entrambi gli scenari Verify e Sync contengono la seguente istruzione, che non è presente nelle altre maschere di collaborazione wrapper:

```
BusObj getItemFlvrBusObj = new BusObj(triggeringBusObj.getType());
```

L'istruzione precedente crea un oggetto business univoco per ItemWrapper (getItemFlvrBusObj). Questa istruzione inoltre ottiene il tipo di oggetto business di attivazione e lo utilizza per impostare il tipo oggetto business utilizzato negli scenari di verifica e di sincronizzazione di ItemWrapper.

Tutte le collaborazioni wrapper creano un oggetto business DestinationAppBusObj e un SourceAppBusObj nella sezione Dichiarazioni della Definizione maschera della collaborazione. Il tipo di questi oggetti business è lo stesso dell'oggetto business di attivazione. Mentre le altre collaborazioni wrapper utilizzano DestinationAppBusObj e SourceAppBusObj per recuperare i dati dalle applicazioni di destinazione e di origine, ItemWrapper usa getItemFlvrBusObj. Prima di recuperare l'elemento dall'applicazione di destinazione nello scenario Verify o dall'applicazione origine nello scenario Sync, ItemWrapper utilizza i valori chiave di triggeringBusObj per impostare le chiavi di getItemFlvrBusObj, nel seguente modo:

```
getItemFlvrBusObj.setKeys(triggeringBusObj);
```

Nello scenario Sync, ItemWrapper utilizza i valori in getItemFlvrBusObj per creare l'oggetto business che invia alla destinazione, nel modo seguente:

```
toItemFlvrBusObj = getItemFlvrBusObj.duplicate();
//toItemFlvrRef.copy(getItemFlvrRef);
toItemFlvrBusObj.setKeys(triggeringBusObj);
```

Una collaborazione wrapper standard, come ContactWrapper, crea un ToBusObj, che istanzia da SourceAppBusObj. L'esempio che segue illustra questa creazione e creazione di istanza:

```
ToBusObj =(SourceAppBusObj).duplicate();
ToBusObj.setKeys(triggeringBusObj);
```

Poiché ItemWrapper non utilizza FromBusObj standard, il codice della collaborazione commenta il codice nel nodo Inizializza variabili di entrambi gli scenari Verify e Sync.

Altre estensioni alla maschera

E' possibile estendere WrapperFoundation in modo da contenere ulteriori porte, proprietà, logica di elaborazione e messaggi. Per ulteriori informazioni, consultare "Estensione della CollaborationFoundation" a pagina 35.

Creazione di gruppi di collaborazioni

Un *gruppo di collaborazioni* è una serie di oggetti di collaborazione che rappresenta un processo business combinato. Un gruppo di collaborazioni consente di combinare delle unità distinte di logica. Gli oggetti di collaborazione sono collegati fra loro tramite gli stessi tipi di porte attraverso le quali possono collegarsi anche ai connettori.

I gruppi di collaborazioni forniscono i seguenti vantaggi:

- Consentono di modularizzare la logica. E' possibile sviluppare e testare una unità di logica solo una volta e poi distribuirla più volte.
- Consente di espandere le collaborazioni esistenti. E' possibile creare le maschere di collaborazione che richiamano o (vengono richiamate da) collaborazioni esistenti.

I gruppi di collaborazioni sono composti da due o più collaborazioni. All'interno di un gruppo, le collaborazioni sono collegate ad altre collaborazioni e c'è sempre la consapevolezza di una collaborazione chiamante ed una collaborazione chiamata . In ogni coppia di collaborazioni collegate fra loro, una è quella chiamante e l'altra la collaborazione chiamata. Una collaborazione chiamante è collegata in modo tale che una delle sue chiamate di servizio invia un oggetto business che attiva l'esecuzione di un'altra collaborazione. La collaborazione chiamata riceve l'oggetto business, che è il suo evento di attivazione. La collaborazione chiamata restituisce il risultato al chiamante dopo l'esecuzione. Consultare Figura 13.

Figura 13. Collaborazioni chiamanti e chiamate

In un gruppo di collaborazioni, una collaborazione che non supporta i processi business di lunga durata non può collegarsi ad una collaborazione distribuita come processo business di lunga durata.

Esempio di gruppo di collaborazioni: Customer Manager

un esempio di gruppo di collaborazioni è il Customer Manager del prodotto WebSphere Business Integration Server Express, che è composto dalle seguenti collaborazioni:

- CustomerSync
- CustomerWrapper
- CustomerPartnerSync
- CustomerPartnerWrapper

Quando si installa la collaborazione Customer Manager, si ricevono tutte le maschere di collaborazione. E' possibile configurarle e collegarle (stabilire le comunicazioni fra le collaborazioni utilizzando le porte) in vari modi per formare un processo unificato.

La collaborazione CustomerSync sincronizza un cliente SoldTo; vale a dire, la collaborazione CustomerSync unisce eventi e dati con un cliente SoldTo. E' anche possibile scegliere di sincronizzare i dati e gli eventi sulle informazioni correlate del cliente. In quel caso, è possibile collegare CustomerSync a CustomerPartnerWrapper, che esegue una pre-elaborazione, e quindi collegare CustomerPartnerWrapper a CustomerPartnerSync. La Figura 14 illustra questa serie di collegamenti.

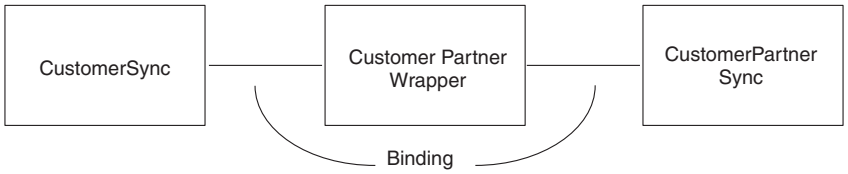


Figura 14. Gruppo di collaborazioni collegato

Creazione di un gruppo di collaborazioni

Passi generali per la creazione di un gruppo di collaborazioni:

- Nella collaborazione chiamante:
  - Creare una porta per il tipo di oggetto business da passare alla collaborazione chiamata.
  - Configurare una chiamata di servizio che passa l'oggetto business insieme all'istruzione che si desidera gestisca la collaborazione chiamata.
  - Gestire i risultati della chiamata di servizio normalmente.

    Nota: Se una delle collaborazioni del gruppo di collaborazioni è configurata per la persistenza della chiamata di servizio in transito, tutte le collaborazioni di quel gruppo verranno automaticamente configurate dal sistema WebSphere Business Integration Server Express a mantenere il comportamento del ripristino congruente. Per ulteriori informazioni, consultare "Chiamate di servizio e richieste exactly-once (esattamente una volta)" a pagina 192.
- Nella collaborazione chiamata:

– Creare una porta per il tipo di oggetto business che deve ricevere dalla collaborazione chiamante.
– Creare uno scenario ed assegnare l'evento di attivazione dello scenario.

Inclusione dei servizi Web

IBM WebSphere Business Integration Server Express supporta l'utilizzo dei servizi Web nelle collaborazioni. Un servizio Web è un'applicazione modulare le cui interfacce e i cui collegamenti pubblici sono definiti con XML, ed è accessibile tramite i protocolli aperti, come HTTP e SOAP. Un servizio Web può essere incluso in una definizione di attività della maschera di collaborazione. Quando viene eseguito l'oggetto collaborazione corrispondente, il servizio Web viene richiamato automaticamente. Non sono necessarie modifiche di InterChange Server Express.

Utilizzare System Manager per individuare e registrare i servizi Web. I servizi Web registrati divengono parte del progetto ICL (Integration Component Library). Inoltre, qualsiasi oggetto business richiesto per i servizi Web viene generato automaticamente e collocato nel progetto ICL. Per ulteriori informazioni sull'utilizzo di System Manager per registrare e gestire i servizi Web, consultare il manuale *System Implementation Guide*.

Per includere un servizio Web in una maschera di collaborazione è necessario esportarlo dal progetto ICL in System Manager. Ogni metodo viene esportato come blocco funzione in Activity Editor, dove può essere messo in una definizione di attività. Per ulteriori informazioni sull'esportazione di un servizio Web e la sua aggiunta ad una definizione di attività, consultare "Blocchi funzione dei servizi web" a pagina 166.

Progettazione di processi business di lunga durata

Se si prevede di distribuire la collaborazione coma processo business di lunga durata, tenere presente quanto riportato di seguito, durante la progettazione e la creazione della maschera di collaborazione:

- Utilizzare le variabili di maschera o di porta globali per tutti i dati che si desidera persistano nel processo business.
- I riferimenti a tutti gli oggetti CwDBConnectionvengono rilasciati prima di una chiamata di servizio in un processo business di lunga durata, e viene implicitamente effettuato il commit di tutte le transazioni attive del database. Se necessario, progettare la maschera a riacquisire gli oggetti CwDBConnection al termine della chiamata di servizio. Inoltre, reinizializzare il contesto della transazione database dopo la chiamata di servizio, se si sta utilizzando il bracketing esplicito della transazione database.
- Se la collaborazione verrà collegata ad un adattatore, verificare che l'adattatore sia configurato per utilizzare JMS come meccanismo di trasporto. I processi business di lunga durata non possono usare un adattatore con nessun altro tipo di trasporto.
- Le collaborazioni processo business di lunga durata non possono essere collegate a client di accesso esterni.
- In un gruppo di collaborazioni, le collaborazioni che non supportano i processi business di lunga durata non possono collegarsi ad una collaborazione processo business di lunga durata.

Chiamate di servizio dinamiche

Esistono due approcci che una collaborazione può utilizzare per richiedere i servizi in InterChange Server Express. Il primo metodo è tramite chiamata statica; si definisce la collaborazione di destinazione o il connettore per la chiamata di servizio quando si progetta la collaborazione e la si collega ad una porta specifica. Questo rende la chiamata di servizio statica, nel senso che può solo comunicare con la destinazione collegata esplicitamente.

Il secondo metodo è effettuando una chiamata di servizio dinamica alla collaborazione o connettore di destinazione. Questo metodo consente di distribuire gli oggetti business a più destinazioni non note al momento della progettazione della collaborazione. Una chiamata di servizio dinamica si implementa utilizzando dynamicSend().

I vantaggi di utilizzare le chiamate di servizio dinamiche sono:

- Non è necessario collegare una porta al momento della progettazione. Le informazioni necessarie per elaborare il progetto business fanno parte della chiamata di servizio.
- La collaborazione può interagire con molti connettori fornendo una maggiore flessibilità in un ambiente di elaborazione dinamico.
- Le collaborazioni gestiscono le richieste contemporaneamente.

Quando si deve decidere se utilizzare le chiamate dinamiche, considerare quanto segue:

- E' necessario comprendere la sequenza degli eventi quando si utilizzano le chiamate di servizio dinamiche. Poiché non vi sono porte configurate esplicitamente in una chiamata di servizio dinamica, la chiamata di servizio dinamica potrebbe modificare la coda istanze della sequenza eventi.
- Le chiamate di servizio dinamiche non supportano la compensazione. Non è disponibile un metodo rollback quando si utilizza una chiamata di servizio dinamica.
- InterChange Server non considera una chiamata di servizio dinamica ad un'altra collaborazione, quando stabilisce un gruppo di collaborazioni. Questo perché non esiste un collegamento esplicito ad una porta fra le due collaborazioni.
- Un processo business di lunga durata non può essere un consumer di una chiamata di servizio dinamica.
- Un processo business di luna durata può effettuare una chiamata di servizio dinamica ad un connettore ma il connettore deve utilizzare JMS come tipo di trasporto.

Progettazione di un'esecuzione parallela

InterChange Server Express fornisce un ambiente per esecuzioni parallele: può eseguire più collaborazionicontemporaneamente in thread separati e può anche eseguire più thread della stessa collaborazione (noto come multithreading).

Attenzione: Il pool di thread per le collaborazioni viene utilizzato *solo* per flussi attivati da eventi e non per flussi attivati da chiamate. Tuttavia, anche i flussi attivati da chiamate sono a più thread nell'esecuzione, nel senso che utilizzano il pool di thread di ORB (Object Request Broker) IBM Java.

Funzioni multithreading

Ogni server ha un numero massimo di thread che possono essere prodotte contemporaneamente per elaborare le sottoscrizioni degli oggetti business. E' possibile impostare il proprio numero massimo di da generare, in base alla situazione individuale e a cosa si ritiene sia ottimale per le prestazioni. Naturalmente il numero impostato non deve superare il numero di thread consentito dal server.

Per impostare il numero massimo di thread che possono potenzialmente essere prodotti per un singolo oggetto di collaborazione, specificare il numero massimo nel campo **Numero massimo di eventi simultanei**, che System Manager visualizza nella scheda Proprietà generali della collaborazione dell'oggetto collaborazione. E' possibile accedere alla scheda dall'opzione Proprietà del menu di scelta rapida di un oggetto di collaborazione, dall'interno del progetto o dalla vista Server.

Nota: Se il connettore di destinazione è configurato per l'elaborazione parallela, codificare la maschera di collaborazione per verificare che la richiesta sia stata inviata con esito positivo all'applicazione. Aggiungere questo codice al nodo presente subito dopo il collegamento di transizione dell'eccezione per la chiamata di servizio. Per ulteriori informazioni, consultare getSubType() in Capitolo 29, "Classe CollaborationException", a pagina 449.

Importante: Se il connettore di destinazione è a thread singolo, è necessario configurarlo per l'elaborazione parallela per usufruire dei vantaggi della collaborazione a più thread. Per ulteriori informazioni, consultare il manuale *System Implementation Guide*.

Problemi nell'elaborazione simultanea

In qualsiasi ambiente di elaborazione simultanea, c'è sempre il pericolo di dell'incongruenza dei dati. L'incongruenza dei dati può verificarsi se l'elaborazione simultanea è mediante più processi o più thread. Se due programmi o due thread accedono agli stessi dati contemporaneamente, c'è sempre la possibilità che uno possa modificare i dati e influenzare negativamente le operazioni dell'altro programma o thread in modi non previsti. Gli ambienti ad elaborazione simultanea gestiscono questo problema sincronizzando l'accesso ai dati condivisi; un thread o un processo blocca la parte di dati in modo che un altro thread o processo non possa accedervi contemporaneamente.

Per un esempio semplice del problema così come potrebbe verificarsi nell'ambiente di integrazione business, considerare la situazione seguente:

- Un utente di applicazione in un'applicazione origine InterChange Server Express deve aggiungere 10.000 euro allo stipendio di 40.000 euro di un impiegato.
- L'utente dell'applicazione per sbaglio inserisce un aumento di stipendio di 100.000 euro. L'utente realizza che l'immissione non era corretta ed aggiorna di nuovo lo stipendio, stavolta togliendo i 100.000 euro.
- Entrambe le operazioni comportano l'invio di un evento Employee.Update per lo stesso ID impiegato al sistema InterChange Server Express.
- Gli eventi vengono elaborati immediatamente ed inviati ad un'altra applicazione per la sincronizzazione.
- Nell'applicazione di destinazione, la prima operazione di aggiornamento tenta di impostare lo stipendio dell'impiegato su -60.000 euro — tenta di sottrarre 100.000 euro dallo stipendio dell'impiegato che è di 40.000. Questo provoca un errore semantico e dei risultati non previsti perché gli stipendi non possono andare sotto lo zero. Anche il secondo aggiornamento rileva un errore.

InterChange Server Express dispone delle seguenti funzioni per assicurare la congruenza dei dati ed occuparsi di questo problema:

- "Sequenza degli eventi"
- "Isolamento dell'evento"

Sequenza degli eventi

La *sequenza degli eventi* fa si che due thread della stessa collaborazione non utilizzino gli stessi dati contemporaneamente. Se più eventi hanno lo stesso tipo di oggetto business e gli stessi valori chiave, il server li mette in coda e li distribuisce in ordine di arrivo. Il thread della collaborazione che riceve il primo evento deve essere terminato prima che la collaborazione possa ricevere l'evento successivo. La sequenza eventi quindi mantiene l'ordine di esecuzione, anche in presenza di esecuzioni multithread, anche se i diversi thread potrebbero andare in esecuzione a velocità differenti.

Non è necessario progettare una collaborazione in un modo particolare per usufruire dei vantaggi della sequenza eventi; viene effettuata automaticamente.

Nota: E' possibile disattivare la sequenza eventi modificando il valore della proprietà **ControllerEventSequencing** del connettore in false e deselezionando la casella **Isolamento evento** nella configurazione della collaborazione. Consultare *"Configuring connectors"* nel manuale *System Implementation Guide*.

Isolamento dell'evento

L'*Isolamento evento* fa si che due collaborazioni non utilizzino gli stessi dati contemporaneamente. Alle volte più collaborazioni gestiscono gli stessi tipi di oggetti business. Un evento arriva ed attiva una particolare collaborazione. Questa collaborazione avvia l'esecuzione e, mentre è in esecuzione, dispone dell'accesso esclusivo a quell'istanza di oggetto business in InterChange Server Express. Se arriva un altro evento relativo agli stessi dati, InterChange Server Express mette in coda l'evento appena arrivato finché la collaborazione in esecuzione non termina l'elaborazione del primo evento. Esistono delle limitazioni che riguardano questa funzione; sono descritte nelle sezioni che seguono.

InterChange Server Express non esegue l'isolamento dell'evento automaticamente. Gli sviluppatori di collaborazioni devono progettare delle maschere in un determinato modo, per usufruire dei vantaggi dell'isolamento evento. Questa sezione descrive le regole e fornisce qualche esempio di decisioni di progettazione che possono essere utili allo scopo.

Nota: Queste linee guida riguardano solo le collaborazioni che eseguono operazioni che modificano i dati ed operano in un ambiente in cui vengono utilizzate più collaborazioni. Se si sta sviluppando una collaborazione che esegue solo operazioni di recupero e sarà sempre l'unica collaborazione che utilizza quel tipo di oggetto business su quel server, è possibile ignorare queste linee guida.

Quando viene applicato l'isolamento evento

InterChange Server Express stabilisce l'applicazione dell'isolamento evento al of evruntime, in base all'analisi degli eventi che arrivano e alle porte delle collaborazioni attive. I criteri per l'analisi degli eventi sono gli stessi della sequenza di eventi: gli eventi sono gli stessi, quando il tipo di oggetto business ed i valori chiave sono uguali.

InterChange Server Express utilizza anche il valore configurato della proprietà **Isolamento evento** della collaborazione per stabilire se applicare l'isolamento dell'evento. Consultare *"Configuring collaboration objects"* nel manuale *System Implementation Guide*.

Importante: Se si deseleziona la proprietà **Isolamento evento** di una collaborazione che necessita l'isolamento, esiste la possibilità di aggiornamenti errati. Ad esempio, se le collaborazioni A, B e C fanno tutte riferimento allo stesso oggetto business ed hanno la stessa destinazione, dovrebbero tutte avere l'isolamento dell'evento abilitato. Se una delle collaborazioni ha questa proprietà disattivata, quella collaborazione potrebbe accedere ad un oggetto business contemporaneamente ad un'altra collaborazione che sta aggiornando i valori nell'oggetto.

L'analisi delle collaborazioni attive considera la serie di porte di ciascuna collaborazione collegate ai connettori. Durante l'*corrispondenza delle porte*, InterChange Server Express verifica se:

- In tutte le altre collaborazioni, le porte siano collegate alla stessa serie di connettori
- Nelle porte collegate alla stessa serie connettori, le porte collegate allo stesso connettore abbiano lo stesso tipo di oggetto business

Ad esempio, due collaborazioni hanno le porte corrispondenti se entrambe hanno questi collegamenti di porta:

Connector1/Tipo oggetto business A

Connector2/Tipo oggetto business B

Non è importante se una porta viene utilizzata per eventi in ingresso o richieste e risposte in uscita; sono rilevanti solo il collegamento connettore ed il tipo di oggetto business.

Le porte collegate ad altre collaborazioni non vengono considerate durante la determinazione delle collaborazioni per le quali è valido l'isolamento evento.

Esempio di corrispondenza delle porte: porte che corrispondono: La Figura 15 illustra due collaborazioni, X e Y, per le quali sarebbe valido l'isolamento evento. I piccoli rettangoli neri sui margini delle collaborazioni indicano le porte.

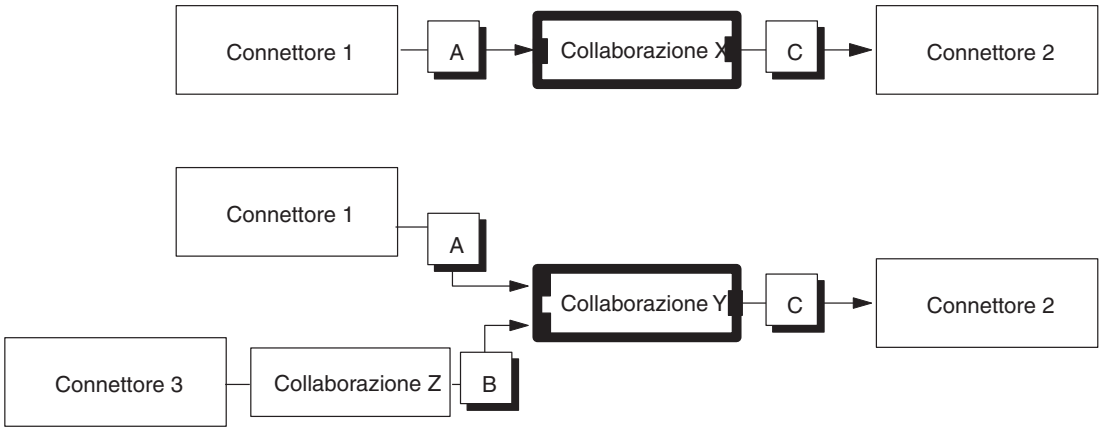


Figura 15. Corrispondenza delle porte

Nella Figura 15, X ha due porte e Y ha tre porte. Tuttavia, la corrispondenza delle porte considera solo le due porte di Y che sono collegate ai connettori; ignora la porta collegata alla collaborazione Z. Entrambe le collaborazioni hanno le seguenti porte collegate a connettori:

- Una porta definita per il tipo di oggetto business A e collegata al connettore 1.
- Una porta definita per il tipo di oggetto business C e collegata al connettore 2.

Questo esempio soddisfa i criteri per l'isolamento dell'evento ed il server isola l'evento in ingresso o di attivazione. Pertanto, le istanze dell'evento A sarebbero soggette ad isolamento in queste due collaborazioni.

Esempio di corrispondenza delle porte: porte che non corrispondono: Ricordarsi che il server considera tutte le porte, quando confronta le collaborazioni; non limita l'analisi della corrispondenza delle porte alle porte che ricevono eventi di attivazione. Se due collaborazioni ricevono lo stesso tipo di evento dallo stesso tipo di connettore ma inviano un oggetto business in uscita a due diversi connettori, loro eventi non vengono isolati.

La Figura 16 illustra due collaborazioni le cui porte in uscita sono collegate a connettori diversi. Le loro istanze evento non vengono isolate.

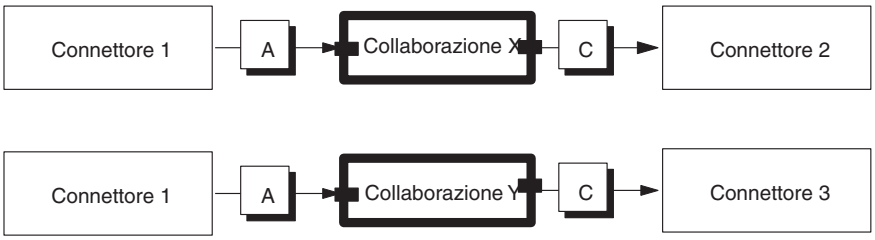


Figura 16. Porte non corrispondenti

Regole di progettazione

E' necessario progettare le collaborazioni in un determinato modo se si desidera beneficiare dell'isolamento dell'istanza evento. Questa sezione descrive come:

- Utilizzare la delega per formare i gruppi di collaborazioni
- Gestire gli oggetti business secondari come oggetti con valori di riferimento

Utilizzo della delega: Ogni maschera di collaborazione che modifica un oggetto business dovrebbe essere dedicata a modificare solo quel tipo di oggetto business. Se la collaborazione deve modificare un altro tipo di oggetto business, ad esempio un oggetto business secondario, è necessario creare una collaborazione separata il cui scopo è modificare l'altro tipo di oggetto business. Quindi, far si che la prima collaborazione *deleghi* (passi) l'altro oggetto business alla seconda collaborazione, per la modifica.

La regola di dedicare una singola collaborazione a modificare solo un tipo di oggetto business aiuta a mantenere la congruenza dei dati. Impedisce al più collaborazioni di modificare contemporaneamente la stessa istanza dello stesso tipo di oggetto business. La delega assicura il mantenimento della congruenza dei dati di un oggetto business secondario nel rispetto delle istanze dello stesso oggetto business elaborate da a ltre collaborazioni. Ricordarsi che è possibile utilizzare un oggetto business in un contesto come elemento secondario e in un altro contesto come elemento indipendente.

Si supponga di dove scrivere un processo business che ha a che fare con un l'oggetto business A ed una serie di informazioni B, ad esso associate come oggetto business secondario. La struttura dell'oggetto business potrebbe essere simile a quella illustrata nella Figura 17, dove l'oggetto business B è un elemento secondari dell'oggetto business A.

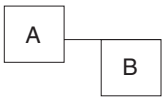


Figura 17. Esempio di un oggetto business gerarchico

Se esiste già una collaborazione che elabora gli oggetti business B, delegare a quella collaborazione il lavoro relativo all'oggetto business secondario B. In alternativa, potrebbe essere necessario creare un'altra collaborazione.

Quando si desidera modificare i dati associati all'oggetto business A, è necessario lavorare sia sull'oggetto business A che sulla sua serie di dati B, o oggetto business secondario. Si creerebbero quindi due diverse maschere di collaborazione — una collaborazione modifica l'oggetto business A e l'altra modifica l'oggetto business secondario B — ed è possibile combinare le due maschere in un gruppo di collaborazioni. Ogni maschera di collaborazione gestisce le operazioni su un oggetto business.

La Figura 18 illustra un gruppo di collaborazioni A/B, che contiene una collaborazione Processore A ed una collaborazione Processore B. La collaborazione Processore A elabora l'oggetto business A. Quando la collaborazione Processore A deve modificare l'oggetto business secondario B, utilizza una chiamata di servizio per inviare l'oggetto business B alla collaborazione Processore B. Nella Figura 18, una riga tratteggiata mostra la delega.

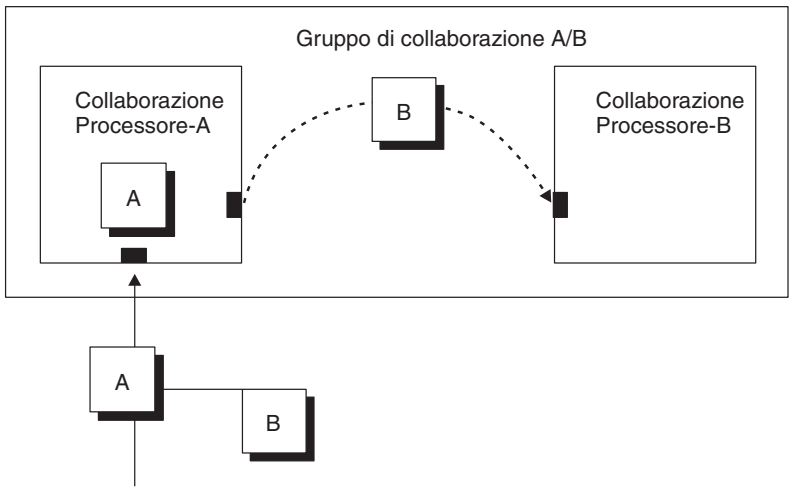


Figura 18. Delega di un oggetto business secondario

Gestione degli oggetti business secondari come oggetti con valori di riferimento: Quando una collaborazione riceve un oggetto business secondario delegato (come nel caso della collaborazione Processore B nella Figura 18), deve considerare l'oggetto business come un oggetto con valori di riferimento. Un *oggetto business con valori di riferimento* contiene solo i valori per gli attributi definiti come chiavi primarie per l'oggetto business. Un *oggetto business con valori completi*, al contrario, contiene i valori degli altri attributi.

Nelle figure di questo capitolo, gli oggetti business con valori di riferimento sono contrassegnati (r) e gli oggetti con valori completi sono contrassegnati (f). La Figura 19 è un esempio di oggetto business secondario con con valori di riferimento.

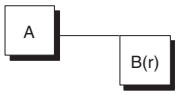


Figura 19. Oggetto business gerarchico secondario con valori di riferimento

A seconda del connettore di origine, gli eventi possono essere inviati con oggetti business secondari con valori di riferimento o con valori completi. Una collaborazione che riceve un oggetto business secondario delegato potrebbe quindi ricevere tutti i suoi valori di attributo o solo con i suoi valori di chiavi primarie. Tuttavia, la collaborazione deve sempre considerare l'oggetto business secondario delegato ricevuto come oggetto con valori di riferimento. Deve presupporre solo che i valori delle chiavi primarie siano corretti; deve ignorare i valori degli attributi che non sono chiavi primarie.

Se la collaborazione deve eseguire delle operazioni sugli attributi non chiavi dell'oggetto business secondario deve risolvere il riferimento recuperando la versione con valori completi dell'oggetto business dall'applicazione origine. Se l'oggetto business secondario e di tipo con valori di riferimento, l'operazione di recupero ottiene i valori attributo aggiuntivi. Se l'oggetto business secondario e di tipo con valori completi, l'operazione di recupero assicura che i dati associati siano attuali e validi.

La Figura 18 illustra la delega di B come oggetto business con valori di riferimento e la risoluzione della collaborazione del riferimento, recuperando i valori dalla collaborazione origine.

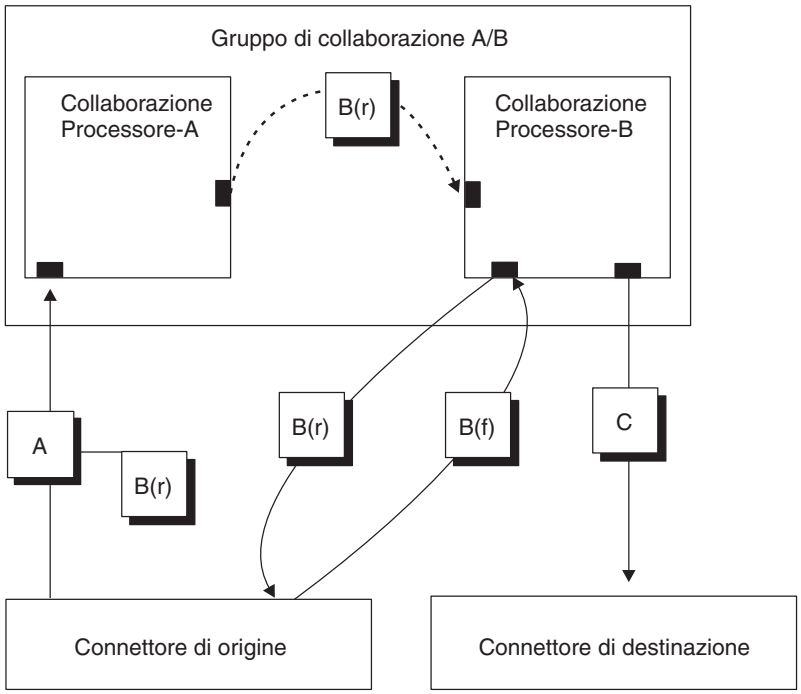


Figura 20. Risoluzione del riferimento

Esempi

La Figura 21 illustra un ambiente in cui l'isolamento evento è attivo fra due diverse collaborazioni, collaborazione Prcessore B e collaborazione da-B-a-C.

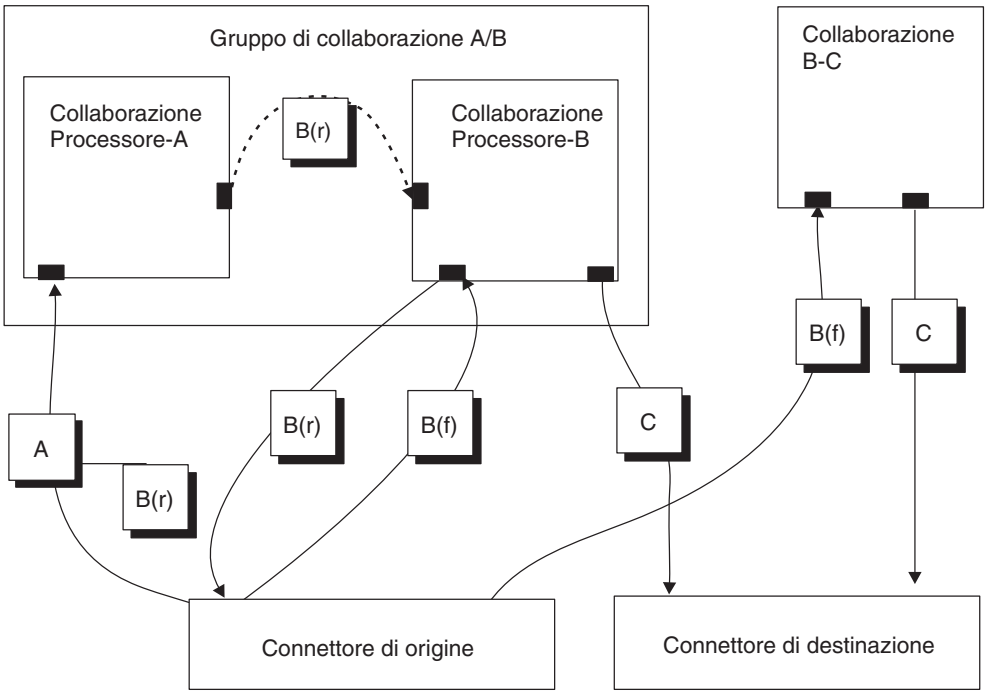


Figura 21. Due collaborazioni soggette a isolamento evento

Entrambe le collaborazioni Processore B e da-B-a-C:

- ricevono gli eventi dell'oggetto business di tipo B
- producono oggetti business di tipo C
- sono collegati alla stessa serie di connettori

Pertanto, la corrispondenza delle risulterebbe nell'isolamento evento per queste collaborazioni.

Il prossimo esempio (mostrato nella Figura 22) illustra come è possibile utilizzare gli oggetti di collaborazione creati dalla stessa maschera in due modi diversi nello stesso ambiente. Questa modalità consente di riutilizzare ed estendere una maschera di collaborazione esistente, come quando si desidera aggiungere delle funzioni alla collaborazione.

Si supponga che esista la maschera di collaborazione Processore Y e che l'oggetto collaborazione istanziato dalla maschera Processore Y, Processore Y Collaboration1, sia in uso. Si desidera creare delle nuove funzioni della collaborazione che includono ed estendono le funzioni della maschera di collaborazione Processore Y.

Un modo per farlo è di riutilizzare la maschera della collaborazione Processore Y e creare un nuovo oggetto collaborazione Processore Y che si utilizza in un gruppo di collaborazioni. Vale a dire, si crea un'istanza di un secondo oggetto collaborazione Processore Y, Processore T Collaboration2, dalla maschera di Processore Y e posizionarlo in un gruppo di collaborazioni. Ora esistono due collaborazioni Processore Y, entrambe che necessitano dell'isolamento evento. Una collaborazione intermedia — Collaborazione Z nell'esempio — può fornire ulteriori funzioni ed assicurare l'isolamento evento, senza richiedere modifiche a Processore Y.

Nella Figura 22, il server applica l'isolamento evento agli oggetti business Y ricevuti dalle collaborazioni con i margini evidenziati in scuro, Collaborazione Z e Processore Y Collaboration1. I numeri indicano la sequenza dell'elaborazione.

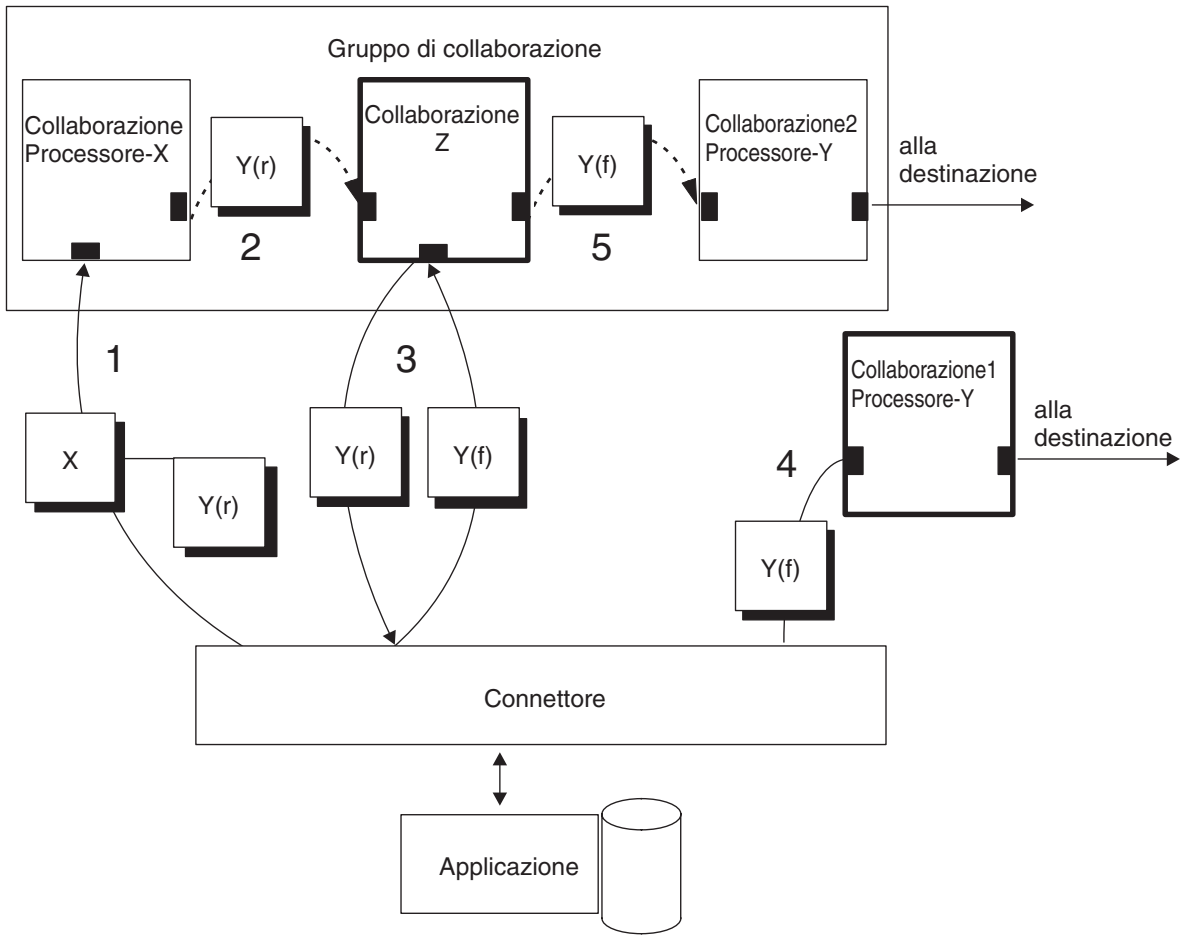


Figura 22. Recupero di un oggetto business con valori completi

Collaborazione Z e Processore Y Collaboration2 lavorano come un team, in termini di isolamento evento. Le linee guida per gli oggetti business delegati vengono seguite da Collaborazione Z per conto di Processore Y Collaboration2.

Una collaborazione internazionalizzata

Una *collaborazione internazionalizzata* è una collaborazione che è stata scritta in modo tale da poter essere personalizzata per una determinata locale. Una *locale* è la parte di un ambiente utente raccoglie le informazioni su come gestire i dati specifici per la nazione, la lingua o il territorio specifici dell'utente finale. Di solito la locale viene installata come parte del sistema operativo. La creazione di una collaborazione che gestisce i dati sensibili alla locale viene denominata l'*internazionalizzazione* (I18N) della collaborazione. La preparazione di una collaborazione internazionalizzata per una particolare locale viene denominata la *localizzazione* (L10N) della collaborazione.

Questa sezione fornisce le seguenti informazioni su una collaborazione internazionalizzata:

- "Descrizione di locale" a pagina 66

- "Considerazioni sulla progettazione di una collaborazione internazionalizzata"

Descrizione di locale

Una *locale* fornisce le seguenti informazioni per l'ambiente dell'utente:

- Convenzioni culturali secondo la lingua e la nazione (o il territorio):
  - Formati di dati:
    - Date: definisce i nomi completi ed abbreviati di giorni della settimana e dei mesi, oltre alla struttura della data (incluso il separatore della data).
    - Numeri: definisce i simboli per il separatore delle migliaia ed il punto decimale, oltre che dove vengono posti questi simboli all'interno del numero.
    - Orari: definisce gli indicatori per la suddivisione del tempo in 12 ore (come gli indicatori AM e PM) oltre alla struttura dell'indicazione dell'ora.
    - Valori monetari: definisce i simboli numerici e di valuta, oltre che dove vengono collocati all'interno del valore monetario.
  - Ordine alfabetico: come ordinare i dati in base ad un particolare codeset di caratteri ed alla lingua.
  - La gestione delle stringhe include attività quali confronto "minuscolo e maiuscolo" (lettere maiuscole e lettere minuscole), stringhe secondarie e concatenazione.
- Una *codifica carattere* — la mappatura da un carattere (una lettera dell'alfabeto) ad un valore numerico in un codeset di caratteri. Ad esempio, il codeset di caratteri ASCII codifica la lettera "A" come 65, mentre il codeset di caratteri EBCIDIC codifica questa lettera come 43. Il *codeset di caratteri* contiene le codifiche di tutti i caratteri in uno o più alfabeti della lingua.

Un nome locale dispone del seguente formato:

ll_TT.codeset

dove *ll* è un codice di lingua a due caratteri (di solito in minuscolo), *TT* è un codice di territorio e paese a due lettere(di solito in maiuscolo) e *codeset* è il nome della serie di codici di caratteri associati. La parte *codeset* del nome è spesso facoltativa. La locale viene di solito installata come parte dell'installazione del sistema operativo.

Considerazioni sulla progettazione di una collaborazione internazionalizzata

Questa sezione fornisce le seguenti categorie di considerazioni sulla progettazione di una collaborazione internazionalizzata:

- "Principi di progettazione sensibili alla locale"
- "Principi di progettazione per la codifica di caratteri" a pagina 72

Principi di progettazione sensibili alla locale

Per essere internazionalizzata una collaborazione deve essere codificata per essere sensibile alla locale, vale a dire, il suo comportamento deve prendere in considerazione l'impostazione locale ed eseguire l'attività appropriata a quella locale. Ad esempio, per locale che utilizzano l'inglese, la collaborazione deve ottenere i messaggi di errore da un file di messaggi in lingua inglese.

IBM WebSphere Business Integration Server Express fornisce l'utente di una versione internazionalizzata di InterChange Server Express e dell'ambiente di collaborazione runtime. Fornisce inoltre CollaborationFoundation, una serie di

maschere di collaborazione che semplificano lo sviluppo degli standard definiti dall'utente che aderiscono agli standard InterChange Server Express. (Per ulteriori informazioni su CollaborationFoundation, consultare "Maschera CollaborationFoundation" a pagina 30.) Il codice di CollaborationFoundation è internazionalizzato. Pertanto, qualsiasi parte della maschera di collaborazione che utilizza il codice di CollaborationFoundation sarà internazionalizzata. Tuttavia, è necessario assicurarsi che qualsiasi codice personalizzato creato dall'utente sia internazionalizzato.

Nota: Il codice della collaborazione che crea Process Designer Express *non* è internazionalizzato. Quando Process Designer Express genera del codice di collaborazione, è necessario eseguire i passi evidenziati in questa sezione per internazionalizzare la propria maschera di collaborazione.

La Tabella 20 elenca i principi di progettazione sensibili alla locale che una collaborazione internazionalizzata deve seguire.

Tabella 20. Principi di progettazione sensibili alla locale, per le collaborazioni

Principio progettazione	Per ulteriori informazioni
il testo di tutti i messaggi di errore e di stato deve essere isolato dalla maschera di collaborazione in un file di messaggi e tradotto nella lingua della locale.	"Stringhe di testo"
La locale dell'oggetto business deve essere conservata durante l'esecuzione della collaborazione.	"Locale dell'oggetto business" a pagina 69
Le proprietà della configurazione della collaborazione devono essere gestite in modo da includere possibili inserimenti di caratteri multibyte.	"Proprietà di configurazione della collaborazione" a pagina 71
Devono essere considerate altre attività specifiche per la locale.	"Altre attività sensibili alla locale" a pagina 71

Stringhe di testo: E' una buona norma di programmazione progettare una collaborazione internazionalizzata in modo che faccia riferimento ad un file di messaggi esterno quando deve ottenere le stringhe di testo piuttosto vincolare le stringhe di testo nel codice della collaborazione. Quando una collaborazione deve generare un messaggio di testo, recupera il messaggio appropriato tramite il suo numero di messaggio dal file di messaggi. Una volta raccolti tutti i messaggi in un unico file di messaggi, il file può essere localizzato traducendone il testo nella lingua o lingue appropriate. Per ulteriori informazioni sui file di messaggi globalizzati, consultare Capitolo 10, "Creazione del file di messaggi", a pagina 247.

Per globalizzare le operazioni di registrazione, eccezione e email, verificare che tutte queste operazioni utilizzino dei file di messaggi per generare i messaggi di testo. Collocando le stringhe di messaggi in un file di messaggi, si assegna un identificativo univoco ad ogni messaggio. La Tabella 21 elenca i tipi di operazioni che utilizzano un file di messaggi ed i metodi API di collaborazione associati contenuti nella classe`BaseCollaboration` che la maschera di collaborazione utilizza per richiamare i relativi messaggi da un file di messaggi.

Tabella 21. Metodi per richiamare i messaggi da un file di messaggi

Operazione file di messaggi	Meotdo BaseCollaboration
Creazione log	`logInfo()`, `logError()`, `logWarning()`

Tabella 21. Metodi per richiamare i messaggi da un file di messaggi (Continua)

Operazione file di messaggi	Meotdo BaseCollaboration
Gestione eccezioni	`raiseException()`
Notifica Email	`sendMail()`

Nota: Gli standard InterChange Server Express raccomandano che i messaggi di traccia *non* siano inclusi del file di messaggi di una collaborazione. I messaggi di traccia non hanno bisogno di essere visualizzati nella lingua della locale dell'utente perché servono per il processo di debug del prodotto.

Gestione della registrazione e dei messaggi di eccezione: Per essere sicuri che la creazione di log e la gestione delle eccezioni siano sempre ottenute dal file di messaggi della collaborazione, *non* utilizzare i formati dei metodi contenuti nella Tabella 21 che consentono di specificare la stringa di messaggi direttamente nella chiamata. Ad esempio, per registrare un errore nella destinazione log, *non* utilizzare la seguente chiamata a logError():

```
logError("Log this message to the log destination");
```

Creare, invece, un identificativo univoco per il messaggio e mettere il testo all'interno del file di messaggi della collaborazione. Se al messaggio fosse assegnato l'identificativo univoco 712, la sua immissione del file di messaggi avrebbe questo aspetto:

```
712
Log this message to the log destination.
```

Facoltativamente è possibile aggiungere dei parametri messaggio a questo messaggio, se necessario.

In una collaborazione internazionalizzata, la chiamata precedente a logError() deve essere sostituita con la seguente chiamata, che ottiene il messaggio log dal file di messaggi della collaborazione:

```
logError(712);
```

in modo simile, si devono ottenere tutti i messaggi di eccezione dal file di messaggi della collaborazione evitando di utilizzare il seguente formato di raiseException():

```
void raiseException(String exceptionType, String message)
```

Utilizzare, invece, uno dei formati raiseException() che include un numero di messaggio.

Gestione dei messaggi email: Il metodo sendEmail() consente di inviare un messaggi a dei destinatari email specifici. In una collaborazione internazionalizzata, i messaggi email devono andare nel file di messaggi della collaborazione. Tuttavia, il metodo sendEmail() *non* fornisce un formato che consente di specificare un identificativo univoco di un messaggio. Pertanto, per inviare un messaggio email è necessario prima estrarre il messaggio dal file di messaggi e poi utilizzare sendEmail() per inviare la stringa messaggio recuperata. La Tabella 22 mostra il metodo che la collaborazione può utilizzare per recuperare un messaggio da un file di messaggi.

Tabella 22. Metodo per recuperare un messaggio dal file di messaggi

Classe libreria collaborazione	Metodo BaseCollaboration
`BaseCollaboration`	`getMessage()`

Il seguente frammento di codice richiama il messaggio 100 dal file di messaggi della collaborazione ed include questo messaggio come parte di un messaggio email:

```
String retrievedMsg = getMessage(100);
sendEmail(retrievedMsg, subjectLine, recipientList);
```

Gestione delle stringhe miste: Oltre a gestire le operazioni del file di messaggi in Tabella 21, una maschera di collaborazione internazionalizzata *non* deve contenere stringhe codificate in maniera mista. Isolare anche queste stringhe nel file di messaggi.

Per globalizzare le stringhe codificate, eseguire i seguenti passi:

- Generare per la stringa codificata un messaggio numerato in maniera univoca nel file di messaggi della collaborazione.

  Nota: Nel file di messaggi è anche possibile includere una spiegazione facoltativa alla stringa isolata. In questa spiegazione è possibile includere il nome dello scenario ed il numero del nodo azione in cui viene utilizzata la stringa. Queste informazioni possono facilmente tracciare la posizione dell'origine ed apportare delle modifiche quando necessario.
- Nella maschera di collaborazione, utilizzare il metodo `getMessage()` per specificare la stringa isolata tramite il numero di messaggio.

Ad esempio, la maschera di collaborazione contiene la seguente riga di codice con una stringa codificata:

```
String imsg100 = "********Before entering order-to-ATP map********";
```

Per isolare questa stringa codificata dal codice della collaborazione creare un messaggio nel file di messaggi ed assegnare un numero di messaggio univoco (100):

```
100
********Before entering order-to-ATP map********
[EXPL]
ATP Transaction: 162
```

Nella maschera di collaborazione, sostituire il codice che contiene la stringa codificata che recupera la stringa isolata (messaggio 100) dal file di messaggi:

```
String imsg100 = getMessage(100);
//retrieve the message numbered '    100'
String imsg100 = getMessage(100);
//display the retrieved message
```

Per ulteriori informazioni sull'utilizzo dei file di messaggi, consultare Capitolo 10, "Creazione del file di messaggi", a pagina 247.

Locale dell'oggetto business: Durante l'esecuzione di un oggetto collaborazione, esistono due diverse impostazioni di locale:

- Una collaborazione eredita la sua locale, denominata *locale della collaborazione*, dall'istanza InterChange Server Express in cui la collaborazione è in esecuzione.

La locale della collaborazione determina la locale dei messaggi di testo che la collaborazione utilizza per la creazione di log, per la traccia, per le eccezione e per l'email.

- Una collaborazione utilizza una *locale del flusso* per gli oggetti business di attivazione. La locale del flusso determina le impostazioni di locale degli oggetti business coinvolti durante l'esecuzione della collaborazione.

Quando viene creato un oggetto business, ha sempre una locale associata ai dati. Per impostazione predefinita ogni oggetti business creato in una collaborazione utilizza la locale della collaborazione. Tuttavia, un oggetto business spesso necessita della locale dell'oggetto business di attivazione (la locale del flusso). Poiché la locale della collaborazione potrebbe essere diversa da quella del flusso, potrebbe essere necessario assegnare la locale del flusso agli oggetti business. La Tabella 23 mostra il metodo che la collaborazione può utilizzare per recuperare la locale associata al flusso.

Tabella 23. Metodo per recuperare la locale del flusso della collaborazione

Classe libreria collaborazione	Metodo
`BaseCollaboration`	`getLocale()`

La maschera di collaborazione deve assicurarsi che le locale degli oggetti business siano ben conservate ed utilizzate in modo appropriato durante il flusso di qualsiasi scenario della collaborazione. La collaborazione può accedere a questa locale con i metodi mostrati nella Tabella 24.

Tabella 24. Metodi per accedere alla locale dell'oggetto business

Classe libreria collaborazione	Metodo
`BusObj`	`getLocale()`, `setLocale()`

Quando Process Designer Express crea una nuova porta per una maschera di collaborazione, crea un nuovo oggetto `BusObj` per la porta con nome *`nomePorta`* `BusObj`, dove *`nomePorta`* è il nome della porta. Ad esempio, se si crea una porta denominata `To`, Process Designer Express crea un oggetto `BusObj` denominato `ToBusObj` con del codice simile a questo:

```
BusObj ToBusObj = new BusObj("Item");
```

Il costruttore della classe `BusObj` crea un oggetto `BusObj` con la locale impostata su quella della collaborazione. Se l'oggetto business deve associare i suoi dati alla locale del flusso, la maschera di collaborazione deve modificare la locale dell'oggetto.

Ad esempio, Process Designer Express genera del codice nella Figura 23 per creare degli oggetti `BusObj` per due porte, `To` e `From`.

```
BusObj ToBusObj = new BusObj(triggeringBusObj.getType());
BusObj FromBusObj = new BusObj(triggeringBusObj.getType());
```

Figura 23. Codice generato per creare degli oggetti business per le porte

il seguente frammento di codice internazionalizza il codice generato nella Figura 23, facendo sì che la locale del flusso sia impostata in questi nuovi oggetti `BusObj`:

```
BusObj ToBusObj = new BusObj(triggeringBusObj.getType());
BusObj FromBusObj = new BusObj(triggeringBusObj.getType());

// get flow locale from BaseCollaboration
triggerLocale = getLocale();

// set newly created BusObj objects' locale to flow locale
ToBusObj.setLocale(triggerLocale);
FromBusObj.setLocale(triggerLocale);
```

Il costruttore `BusObj()` accetta anche un nome locale come argomento. Quindi, un modo alternativo di riscrivere il codice generato nella Figura 23 è di passare la locale del flusso direttamente alla chiamata costruttore come indicato di seguito:

```
// get flow locale from BaseCollaboration
triggerLocale = getLocale();

BusObj ToBusObj = new BusObj(triggeringBusObj.getType(), triggerLocale);
BusObj FromBusObj = new BusObj(triggeringBusObj.getType(), triggerLocale);
```

Nota: I metodi copy() e duplicate() della classe `BusObj` gestiscono automaticamente l'assegnazione della locale dell'oggetto business. Pertanto, se l'oggetto business origine ha la locale corretta, l'oggetto business di destinazione avrà la stessa locale.

Proprietà di configurazione della collaborazione: Come trattato in "Definizione delle proprietà di configurazione della collaborazione (scheda Proprietà)" a pagina 94, una maschera di collaborazione può utilizzare due tipi di proprietà di configurazione per personalizzare la sua esecuzione:

- Le proprietà di configurazione standard sono disponibili per tutte le collaborazioni.
- Le proprietà di configurazione specifiche per la collaborazione sono univoche per la particolare maschera di collaborazione in cui sono definite.

I nomi di tutte le proprietà di configurazione della collaborazione devono utilizzare *solo* i caratteri definiti nel codeset associato alla locale U.S English (`en_US`). Tuttavia, i valori di queste proprietà di configurazione possono contenere i caratteri provenienti dal codeset associato alla locale della collaborazione.

La maschera di collaborazione ottiene i valori delle proprietà di configurazione con il metodo `getConfigProperty()` o `getConfigPropertyArray()` della classe `BaseCollaboration`. Questi metodi gestiscono correttamente i caratteri provenienti da codest multibyte. Tuttavia, per essere sicuri che la maschera di collaborazione sia internazionalizzata, il suo codice deve gestire correttamente i valori delle proprietà di configurazione, una volta che li recupera. La maschera di collaborazione *non* deve dare per scontato che i valori delle proprietà di configurazione contengano solo caratteri single-byte.

Altre attività sensibili alla locale: Una collaborazione internazionalizzata deve anche gestire le seguenti attività sensibili alla locale:

- Ordinamento e confronto dei dati: la collaborazione deve utilizzare un ordine di confronto appropriato per la lingua e la nazione della locale.
- Elaborazione della stringa (ad esempio confronto, stringhe secondarie e maiuscolo/minuscolo): la collaborazione deve garantire che qualunque elaborazione eseguita sia appropriata per i caratteri nella lingua della locale.
- Formati di date, numeri e ora: la collaborazione deve garantire che qualsiasi formattazione eseguita sia appropriata per la locale.

Principi di progettazione per la codifica di caratteri

Se vengono trasferiti dei dati da una ubicazione che utilizza un codeset tad una ubicazione che utilizza un codeset diverso, devono essere eseguite alcune forme di conversione carattere sui dati, per conservarne il significato. L'ambiente runtime Java in JVM (Java Virtual Machine) rappresenta i dati in Unicode. Il set di caratteri Unicode è un set di caratteri universale che contiene le codifiche di caratteri dei più comuni codeset di caratteri (sia single-byte che multibyte). Esistono diversi formati di codifica di Unicode. Le seguenti codifiche vengono utilizzate più frequentemente nel sistema di integrazione business:

- Set di caratteri codificato universale a più ottetti: UCS-2

  La codifica UCS-2 è il set di caratteri Unicode codificato in 2 byte (ottetti).
- UCS Transformation Format, formato a 8 bit: UTF-8

  La codifica UTF-8 è progettata per considerare l'utilizzo dei dati in carattere Unicode in ambienti UNIX. Supporta tutti i valori ASCII (0...127) in modo da non essere mai interpretati con altro che un vero codice ASCII. Ogni valore del codice viene di solito rappresentato come valore a 1, 2 o 3 byte.

La maggior parte dei componenti nel sistema WebSphere Business Integration Server Express, incluso InterChange Server Express e il suo ambiente runtime di collaborazione sono scritti in Java. Pertanto, quando i dati vengono trasferiti fra una collaborazione ed altri componenti interni a InterChange Server Express, viene codificato nel codeset Unicode e non è necessaria la conversione dei caratteri.

Progettazione delle collaborazioni per gli script bidirezionali

IBM WebSphere Business Integration Server Express supporta gli script bidirezionali. Questo supporto è in un formato di tipo standard Windows (logica da sinistra a destra). (Esempi di lingue che implicano la progettazione bidirezionale sono l'arabo e l'ebraico). Grazie a questo supporto, tutte le collaborazioni supportano anche gli script bidirezionali. Tuttavia, i dati che avviano una collaborazione possono venire da:

- Un adattatore che supporta script bidirezionali. Per determinare se un adattatore supporta script bidirezionali, consultare la guida per l'utente del proprio adattatore.
- Un componente che non supporta gli script bidirezionali, un adattatore che non li supporta o i dati importati da un origine esterna dove il supporto bidirezionale è sconosciuto.

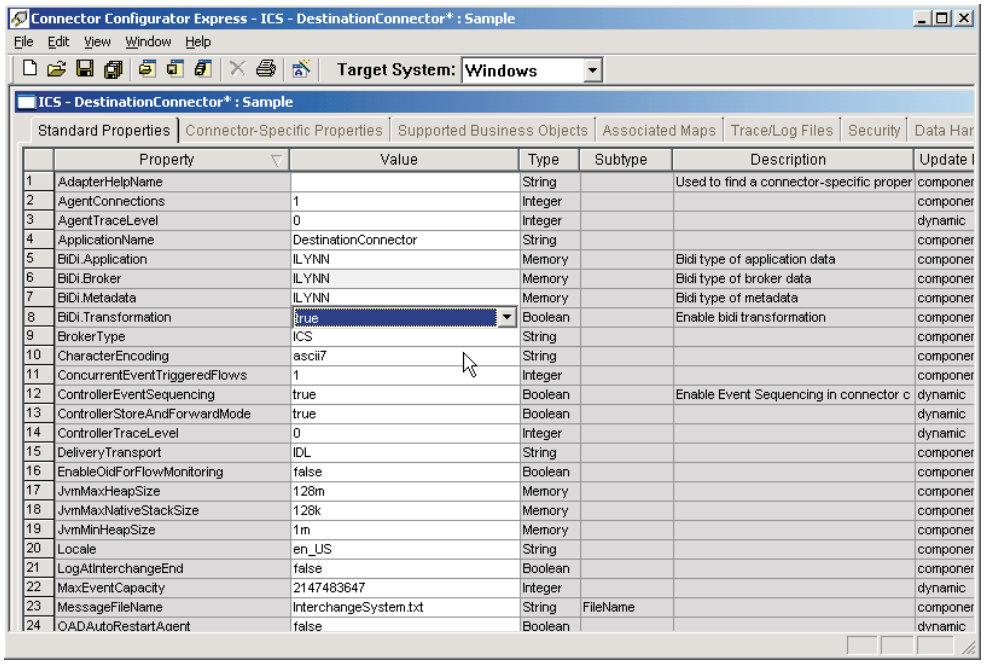
Le incongruenze del formato bidirezionale causano confronti all'interno di una collaborazione e restituiscono risultati non corretti. Questi tipi di errori possono essere evitati seguendo le seguenti istruzioni:

- Accettare input solo da fonti che applicano lo stesso formato bidirezionale di IBM WebSphere Business Integration Server Express come gli adattatori già abilitati con questo supporto.
- Abilitare i connettori che utilizzano questa collaborazione per applicare il formato bidirezionale correttot. (Consultare "Abilitazione dei connettori per script bidirezionali".)
- Utilizzare le API della classe CxBidiEngine per trasformare tutti i dati in un formato bidirezionale congruente (consultare Capitolo 28, "Classe CxBiDiEngine", a pagina 445).

Abilitazione dei connettori per script bidirezionali

Per abilitare un connettore a supportare uno script bidirezionale:

1. Impostare la proprietà BiDiTransformation su `true` nella scheda **Standard** di Connector Configurator Express. (Consultare Figura 24). Impostando il valore su `true` si consente la visualizzazione di altri parametri che supportano gli script bidirezionali per il connettore.



	Property	Value	Type	Subtype	Description	Update I
1	AdapterHelpName		String		Used to find a connector-specific proper	componer
2	AgentConnections	1	Integer			componer
3	AgentTraceLevel	0	Integer			dynamic
4	ApplicationName	DestinationConnector	String			componer
5	BiDi.Application	ILYNN	Memory		Bidi type of application data	componer
6	BiDi.Broker	ILYNN	Memory		Bidi type of broker data	componer
7	BiDi.Metadata	ILYNN	Memory		Bidi type of metadata	componer
8	BiDi.Transformation	true	Boolean		Enable bidi transformation	componer
9	BrokerType	ICS	String			componer
10	CharacterEncoding	ascii7	String			componer
11	ConcurrentEventTriggeredFlows	1	Integer			componer
12	ControllerEventSequencing	true	Boolean		Enable Event Sequencing in connector c	dynamic
13	ControllerStoreAndForwardMode	true	Boolean			dynamic
14	ControllerTraceLevel	0	Integer			dynamic
15	DeliveryTransport	IDL	String			componer
16	EnableOidForFlowMonitoring	false	Boolean			componer
17	JvmMaxHeapSize	128m	Memory			componer
18	JvmMaxNativeStackSize	128k	Memory			componer
19	JvmMinHeapSize	1m	Memory			componer
20	Locale	en_US	String			componer
21	LogAtInterchangeEnd	false	Boolean			componer
22	MaxEventCapacity	2147483647	Integer			dynamic
23	MessageFileName	InterchangeSystem.txt	String	FileName		componer
24	OADAutoRestartAgent	false	Boolean			dynamic

Figura 24. La proprietà BiDiTransformation nel programma di configurazione del connettore

2. Specificare le opzioni bidirezionali per:
   - BiDi.Application
   - BiDi.Broker
   - BiDi.MetaData

   Ciascuna proprietà visualizza una finestra di dialogo (consultare Figura 25) in cui si scelgono i parametri bidirezionali da supportare. (Consultare Tabella 25 per una descrizione dei parametri).

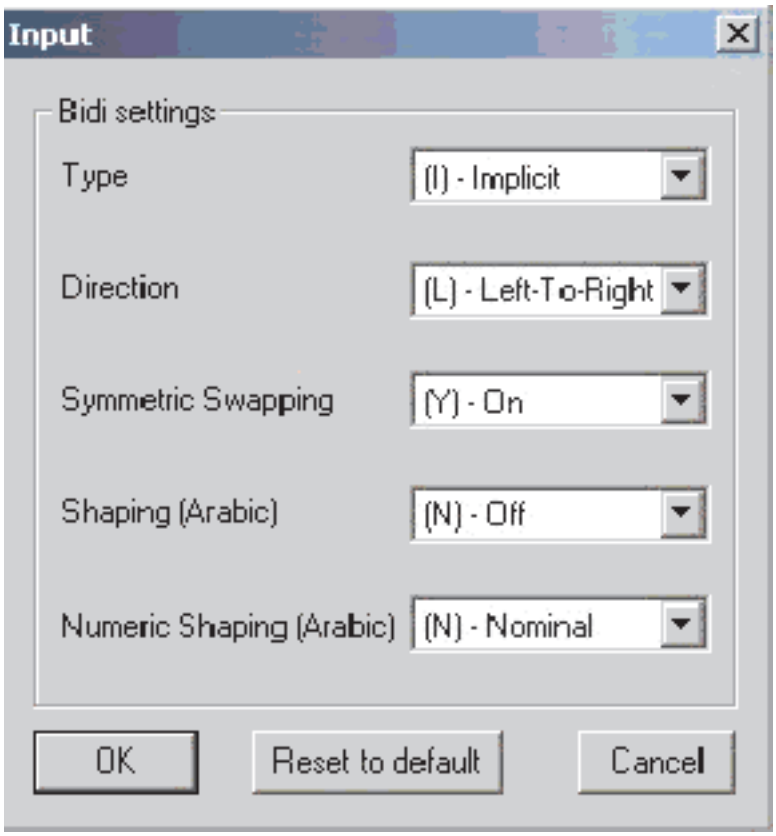


Figura 25. Finestra di dialogo di input di parametri di script bidirezionali

Tabella 25. Valori per stringhe di formato bidirezionale

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	

3. Distribuzione del connettore. Per ulteriori informazioni sulla distribuzione dei connettori consultare il manuale *System Implementation Guide*.

Abilitazione delle collaborazioni per script bidirezionali tramite Access Interface

WebSphere Business Integration Server Express supporta gli script bidirezionali per i collegamenti di comunicazione da un ambiente esterno. I sistemi esterni utilizzano Access Interface per collegarsi a WebSphere Business Integration Server

Express. I dati degli script bidirezionali provenienti da un'origine vengono convertiti nel formato bidirezionale standard di tipo Windows per l'elaborazione. I dati vengono quindi convertiti di nuovo nel formato originale per le comunicazioni con l'origine.

Poiché la configurazione, e non lo sviluppo, è necessaria per la conversione bidirezionale, l'amministratore di sistema, piuttosto che lo sviluppatore, è responsabile del supporto di questa funzione.

E' possibile definire le proprietà per la maschera di collaborazione in Process Designer Express e fornire i valori delle proprietà alla distribuzione in System Manager.

Per ulteriori informazioni sull'impostazione delle proprietà dell'oggetto collaborazione consultare il manuale *System Implementation Guide*.

Configurazione della maschera di collaborazione per dati bidirezionali con Access Interface: Per definire un formato bidirezionale per tutti i dati dell'oggetti business passati attraverso tutte le porte della collaborazione, aggiungere la seguente proprietà:

`BiDi_AI_Application` -- Il nome non è sensibile a maiuscolo e minuscolo. Valori validi sono stringhe da 5 lettere che specificano il formato bidirezionale dei dati dell'oggetto business passati fra l'applicazione esterna e la collaborazione. Per un elenco dei valori validi per questa proprietà, consultare Tabella 25 a pagina 74.

Per definire dei formati bidirezionali diversi per dati di oggetti business passati alla collaborazione attraverso porta diverse, aggiungere la seguente proprietà a ogni porta:

`BiDi_<port name>_AI_Application` -- Il nome non è sensibile a maiuscolo e minuscolo. Valori validi sono stringhe da 5 lettere che specificano il formato bidirezionale dei dati dell'oggetto business passati fra l'applicazione esterna e la collaborazione. Per un elenco dei valori validi per questa proprietà, consultare Tabella 25 a pagina 74.

Nota: : Se l'utente non definisce una specifica proprietà per il formato bidirezionale per una porta di collaborazione, al runtime viene utilizzata quella generale `BiDi_AI_Application`. Se non ne viene definita nessuna, non avviene la trasformazione.

Configurazione della funzione di flessibilità della connessione al database

Molte volte una collaborazione è critica al successo di un'impresa. In questi casi, la disponibilità dei database implicati nella collaborazione è cruciale. Tuttavia vi sono occasioni in cui i database potrebbero non essere disponibili per dei periodi di tempo (ad esempio, mancanza di connessione al database o manutenzione di routine). La funzione di flessibilità della connessione al database fornisce una protezione da questi temporanei periodi di non disponibilità di un database. Le sezioni che seguono descrivono cosa implica assicurarsi che i database critici siano disponibili per le collaborazioni. La flessibilità della connessione al database non influenza direttamente la progettazione della collaborazione.

Personalizzazione della funzione di flessibilità della connessione al database

Per personalizzare la funzione di flessibilità della connessione al database, nel file Server.cfg di InterChange, è possibile impostare le proprietà DB_CONNECT_RETRIES e DB_CONNECT_INTERVAL per ogni database utilizzato da InterChange Server Express per la gestione eventi, le transazioni, il repository, il controllo del flusso, le relazioni ed il registro utente:

- DB_CONNECT_RETRIES - specifica il numero massimo di volte che InterChange Server Express ritenta la connessione al database.
- DB_CONNECT_INTERVAL - specifica l'importo del tempo che InterChange Server Express attende fra i tentativi.

Le proprietà predefinite globali per tutti i database che utilizzano queste proprietà sono definite nella sezione DB_CONNECTIVITY del file.

E' possibile impostare i valori predefiniti per DB_CONNECT_RETRIES e DB_CONNECT_INTERVAL per dei singoli database nelle sezioni del file InterchangeServer.cfg mostrato nella Tabella 26. La configurazione dei valori predefiniti per i database fornisce separatamente la flessibilità quando il sistema tenta di ricollegarsi ai database.

Tabella 26. Sezioni di InterchangeSystem.cfg per impostare i valori predefiniti della flessibilità database

Sezione	Database
EVENT_MANAGEMENT	Gestione eventi
FLOW_MONITORING	Monitoraggio del flusso
REPOSITORY	Repository
TRANSACTIONS	Transazioni
USER_REGISTRY	Registro utente

Nota: E' possibile configurare i valori predefiniti per i database di relazione solo nella sezione DB_CONNECTIVITY del file InterchangeSystem.cfg.

Impostare le proprietà DB_CONNECT_RETRIES e DB_CONNECT_INTERVAL nella configurazione del server in System Manager. Queste proprietà vengono impostate sui valori predefiniti anche se non sono definiti nel file InterchangeSystem.cfg. Per personalizzare le proprietà:

1. Collegarsi a InterChange Server Express utilizzando System Manager.
2. Aprire la configurazione server in System Manager.
3. Impostare le proprietà DB_CONNECT_RETRIES e DB_CONNECT_INTERVAL.
4. Salvare le modifiche.
5. Riavviare InterChange Server Express.

Impostare la proprietà DB_CONNECT_INTERVAL su un valore né troppo grande né troppo piccolo. Se il valore è troppo grande, ad esempio 10 minuti, dopo un tentativo non riuscito di ripristinare le connessioni al database, il successivo tentativo inizia dopo 10 minuti anche se il database potrebbe essere di nuovo disponibile un minuto dopo. Se il valore è troppo piccolo, ad esempio 2 secondi, in quello stesso periodo di 1 minuto InterChange Server Express tenta di recuperare la connessione al database 30 volte e tutti i tentativi falliscono.

Se una mancata connessione al database pianificata causa un riavvio a freddo del database, impostare DB_CONNECT_RETRIES e DB_CONNECT_INTERVAL sui valori che possono includere la mancanza di collegamento, se è nota. L'impostazione delle proprietà che calcolano i tempi morti impedisce inutili tentativi di recupero.

Altre due proprietà sono correlate alla flessibilità della connessione al database per tutti i database utente: maxConnectRetries e maxConnectRetryInterval. Queste proprietà sono definite in ogni connessione a database definiti dall'utente. Funzionano allo stesso modo di DB_CONNECT_RETRIES e DB_CONNECT_INTERVAL e fanno sì che InterChange Server Express tenti di ricollegarsi ai database utente come per i database di sistema. Per configurare queste proprietà:

1. Aprire System Manager.
2. Aprire o creare una connessione database.
3. Impostare le proprietà maxConnectRetries e maxConnectRetryInterval.
4. Salvare la connessione database utente.
5. Distribuire la connessione database utente a InterChange Server Express.

Impostazioni per la flessibilità della connessione al database

Dopo l'installazione di InterChange Server Express il valore predefinito di DB_CONNECT_RETRIES è 3 ed il valore per DB_CONNECT_INTERVAL è 60 secondi. E' possibile cambiare le impostazioni predefinite modificando i valori nella sezione DB_CONNECTIVITY del file InterchangeSystem.cfg. E' anche possibile impostare i valori nella finestra di dialogo Connettività del database di System Manager (consultare Figura 26).

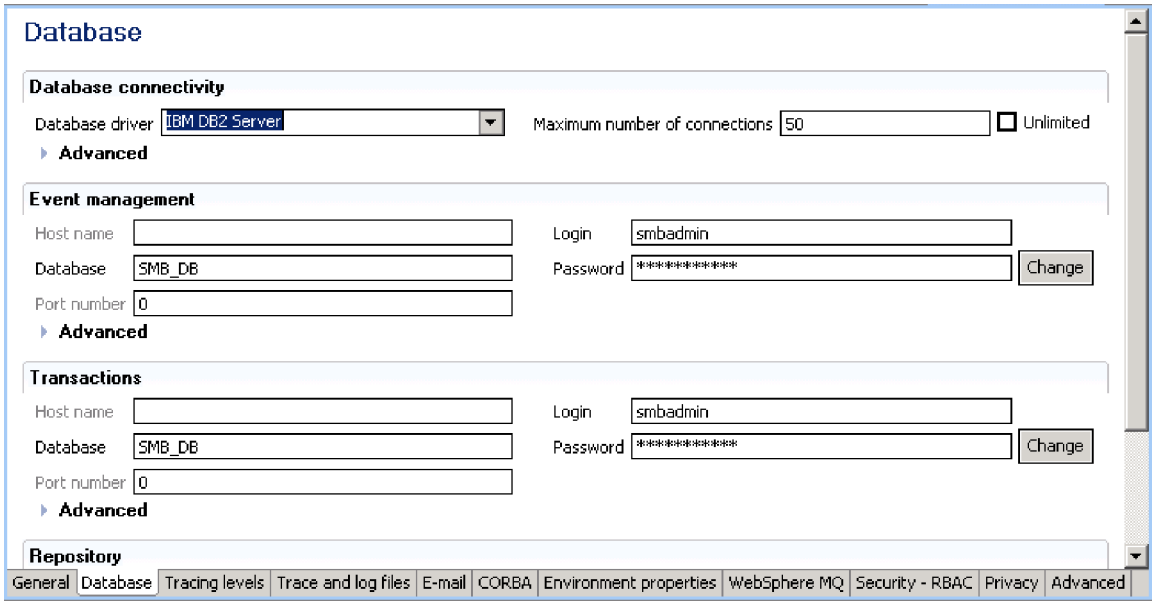


Figura 26. System Manager con i valori predefiniti per la flessibilità del database

Gestione degli eventi non riusciti

In alcuni casi potrebbe essere necessario gestire degli eventi non riusciti nella collaborazione invece di utilizzare Flow Manager. Ad esempio, si potrebbe voler salvare solo i flussi non riusciti in un processo batch per un'ulteriore gestione. Esistono delle API che consentono ad una collaborazione in esecuzione di gestire automaticamente gli eventi non riusciti durante l'elaborazione della collaborazione. Queste API consentono di interrogare, salvare, cancellare o inviare di nuovo degli eventi non riusciti nella collaborazione. Le API forniscono una maggiore flessibilità dell'utilizzo di Flow Manager perché è possibile creare l'elaborazione all'interno del flusso business per gestire automaticamente gli eventi non riusciti. Questa funzione consente un processo batch per poter continuare e allo stesso tempo fornisce la possibilità di reinoltrare o eliminare gli eventi non riusciti in maniera asincrona.

Utilizzare i metodi elencati per fornire alla collaborazione la gestione degli eventi non riusciti:

- "queryFailedEvents()" a pagina 380 – recupera i record degli eventi non riusciti dal relativo database. Ogni evento può avere uno dei seguenti stati:
  - PENDING
  - USER_GENERATED
  - DEFERRED_RECOVERY
  - SVC_CALL_IN_TRANSIT
  - DELIVERY_POSSIBLE_DUPLICATE
  - SVC_CALL_WAITING (per processi business di lunga durata)

  Nota: Flow Manager visualizza gli eventi non riusciti di tipo USER_GENERATED come USER_SAVED.
- "saveFailedEvent()" a pagina 386 – salva gli eventi non riusciti per una successiva elaborazione.
- "dropFailedEvent()" a pagina 368 – elimina gli eventi non riusciti dal database.
- "resubmitFailedEvent()" a pagina 385 – inoltra di nuovo gli eventi non riusciti per una nuova elaborazione.

Salvataggio di un evento non riuscito

Durante la gestione degli errori verificatisi nell'esecuzione della collaborazione, si potrebbe voler decidere che determinate condizioni richiedono un intervento esterno prima di elaborare l'evento. In questi casi, la logica della collaborazione potrebbe salvare l'evento per una successiva analisi ed un nuovo inoltro. Questa funzione consente alla collaborazione di completare l'elaborazione degli oggetti che vengono corretti per evitare ritardi nel processo globale. Ad esempio, un sistema di spedizione degli ordini continua l'elaborazione delle voci corrette in modo che il magazzino possa prendere gli elementi buoni evitando che quelli non corretti tengano in sospeso l'intero orine.

La Figura 27 a pagina 79 mostra la logica della collaborazione di una consegna di un ordine, progettata per elaborare gli ordini anche in presenza di elementi con errore. Notare che invece di salvare le voci immediatamente, la collaborazione aggiunge un oggetto business non riuscito all'oggetto business principale e quindi restituisce l'esito positivo. Quando il controllo torna al diagramma principale, vengono salvati tutti gli eventuali elementi secondari dell'oggetto business principale, utilizzando il metodo saveFailedEvent() a pagina 386.

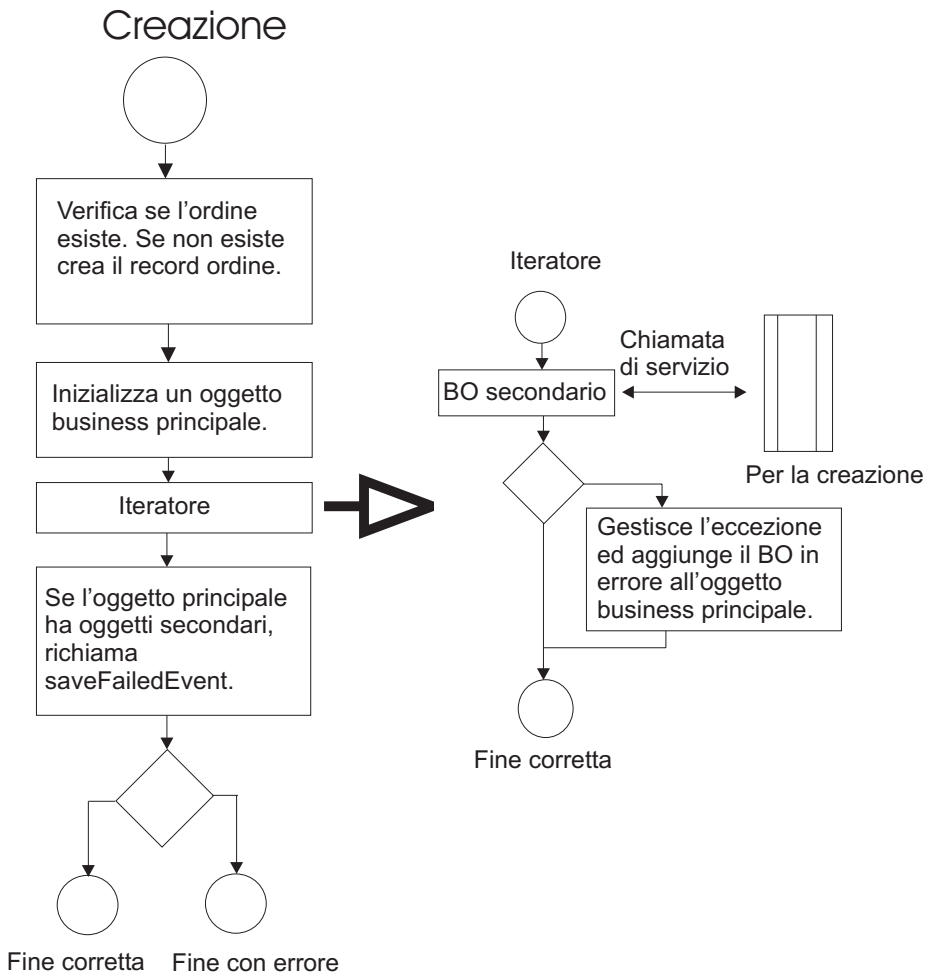


Figura 27. Collaborazione di una consegna di un ordine con la possibilità di salvare gli elementi errati

E' anche possibile creare un processo parallelo per vedere gli eventi non riusciti, correggere gli errori ed inoltrarli di nuovo per l'elaborazione. Per ulteriori informazioni consultare:

- "queryFailedEvents()" a pagina 380
- "dropFailedEvent()" a pagina 368
- "resubmitFailedEvent()" a pagina 385

Codifica per il salvataggio di un evento non riuscito

L'esempio che segue implementa la logica mostrata nella Figura 27.

Dichiarazione:

```
// Define an empty parent BO:
BusObj iterOrderItem = new BusObj("OrderItem');
BusObj parentBo = new BusObj("SalesOrder");
```

Nel passo di azione per l'inizializzazione dell'oggetto business principale:

```
// Set values for parent BO:
// Set the values which are the same as triggeringBusObj
parentBo.set("SalesOrderId",triggeringBusObj.getString("SalesOrderId"));
parentBo.set("CustomerId",triggeringBusObj.getString("CustomerId"));
```

```
parentBo.set("OrderDate",triggeringBusObj.getString("OrderDate"));
//Set Verb to the parent bo
parentBo.setVerb(triggeringBusObj.getVerb());
```

Nel passo di iterazione per salvare l'oggetto business secondario:

```
//Add the failed business object to parent bo
parentBo.set("OrderItems",iterOrderItem);
```

Nel passo di azione per controllare e salvare l'oggetto business principale:

```
// Check to see if the parent bo contains at least one child bo
BusObjArray boArray = parentBo.getBusOvjArray("OrderItems");
int nChildBo = 0;
if (boArray != null)
              nChildBo = boArray.size();

if(nChildBo>0) {
    // Call collaboration API to save the failed event
    saveFailedEvent(parentBO);

}
```

Abilitazione di una collaborazione al salvataggio di eventi non riusciti

Per abilitare una collaborazione al salvataggio di eventi non riusciti aggiornare la maschera della collaborazione con una definizione di attività per individuare e salvare l'evento non riuscito. E' possibile effettuarlo direttamente nel blocco azione in Activity Editor nella vista Java.

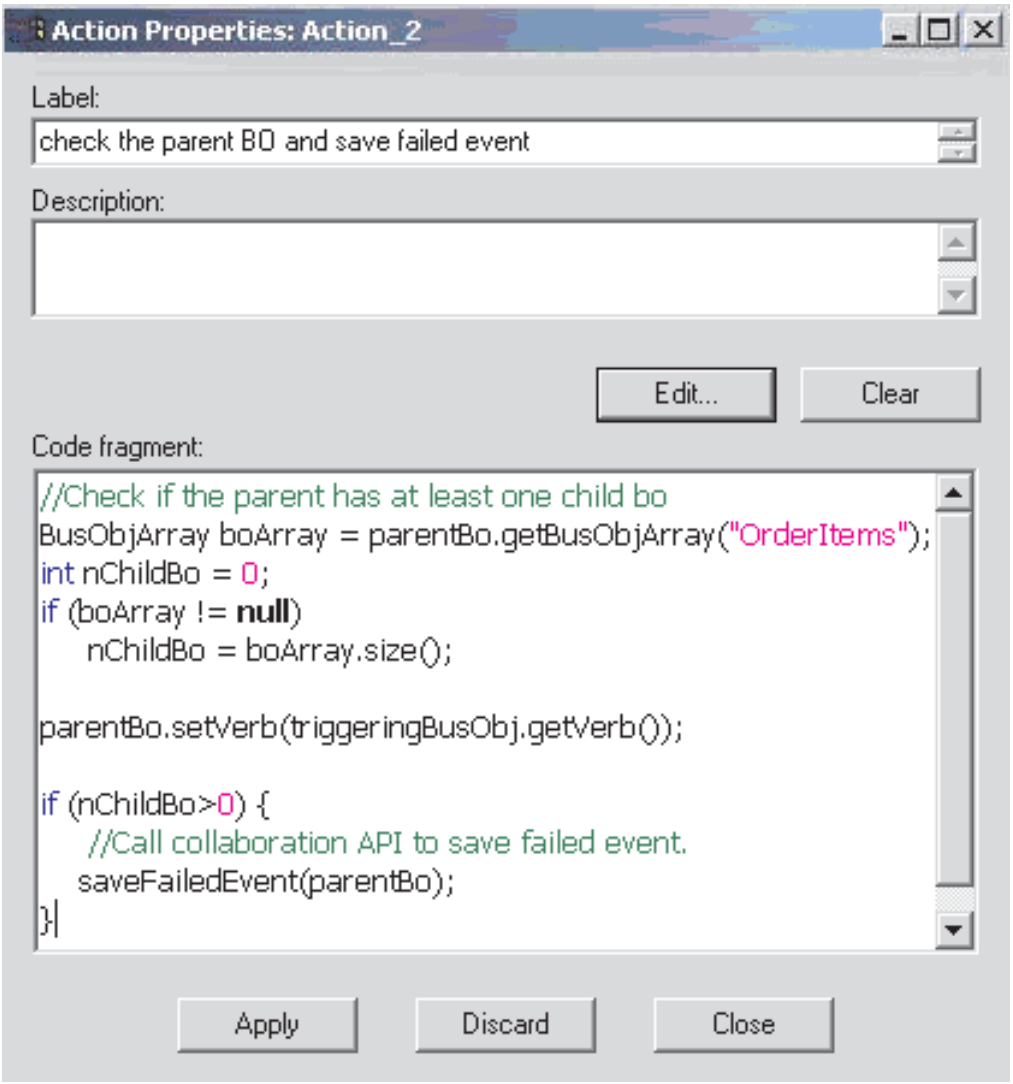


Figura 28. Salvataggio di un evento non riuscito

Nota: Non esistono blocchi funzione in Activity Editor per queryFailedEvents(), dropFailedEvent(), resubmitFailedEvent() o saveFailedEvent().

Strutture dati richieste

Per implementare con esito positivo la gestione di eventi non riusciti nelle collaborazioni è necessario importare le seguenti strutture di dati:

- CxCommon.EventManagement.FailedEventInfo;
- CxCommon.EventManagementEventQueryDef;
- CxCommonEventManagement.EventKeyAttrDef;

Queste strutture vengono importate automaticamente nella collaborazione durante la compilazione delle maschere di collaborazione. Per ulteriori informazione sulle strutture, consultare Capitolo 30, “Pacchetto EventManagement”, a pagina 455.

Reinoltro di eventi non riusciti

Dopo aver salvato gli eventi non riusciti, la collaborazione può inoltrarli di nuovo per l'elaborazione. InterChange Server Express utilizza lo stato USER_GENERATED per differenziare questi eventi non riusciti dagli altri eventi

non riusciti che non sono stati salvati dalla collaborazione. Quando una collaborazione inoltra di nuovo un evento USER_GENERATED, ICS modifica lo stato dell'evento. La collaborazione elabora l'evento come se fosse nuovo. Se l'evento ha di nuovo esito negativo, lo stato viene modificato in USER_GENERATED e non FAILED, in modo che la collaborazione può stabilire come procedere nel ripristino dal secondo errore.

Nota: E' possibile anche utilizzare Flow Manager per individuare manualmente e inoltrare di nuovo questi eventi.

Capitolo 4. Creazione di una maschera di collaborazione

Questo capitolo descrive come creare e modificare un definizione di maschera di collaborazione. E' necessario eseguire le seguenti attività:

1. Creare la definizione della maschera. Consultare "Creazione di una maschera di collaborazione" per ulteriori informazioni.
2. Fornire le informazioni relative alle proprietà della maschera. Consultare "Informazioni sulle proprietà della maschera" a pagina 85 per ulteriori informazioni.
3. Definire uno scenario. Consultare "Definizione di scenari" a pagina 100 per ulteriori informazioni.
4. Creare un diagramma delle attività relativo allo scenario definito. Per informazioni generali, consultare "Creazione di un diagramma di attività" a pagina 104 e per istruzioni dettagliate fare riferimento a Capitolo 5, "Utilizzo dei diagrammi di attività", a pagina 113.
5. Definire ulteriori scenari, se necessario, e creare i diagrammi associati delle attività.
6. Creare un file di messaggi per la maschera. Consultare "Creazione del file di messaggi" a pagina 105 per ulteriori informazioni.
7. Compilare la maschera della collaborazione. Consultare "Compilazione di una maschera di collaborazione" a pagina 105 per ulteriori informazioni.
8. Se lo si desidera, testare la collaborazione. Per verificare se opera nelle modalità previste, è possibile utilizzare Test Connector nell'ambiente di test integrato (Integrated Test Environment). Consultare "Test di una collaborazione" a pagina 112 per ulteriori informazioni.

Se si dispone di una maschera creata con una versione precedente di Process Designer Express, è possibile convertire quella maschera nel formato richiesti dalla versione 4.2. Per istruzioni sulla conversione, consultare "Conversione delle maschere" a pagina 106.

Creazione di una maschera di collaborazione

Utilizzare Process Designer Express per creare, modificare e compilare una maschera di collaborazione. Le sezioni che seguono descrivono come definire una nuova maschera e fornire le informazioni di base richieste.

E' necessario fornire determinate informazioni prima di costruire la maschera; altri tipi di informazioni possono essere fornite in qualsiasi momento durante lo sviluppo. Per la creazione della maschera sono richieste le seguenti informazioni:

Nome maschera	Consultare, "Creazione della definizione di una maschera" a pagina 84
Porte	Consultare, "Definizione di porte ed eventi di attivazione (scheda Porte ed eventi di attivazione)" a pagina 97
Definizioni di scenari	Consultare, "Definizione di scenari" a pagina 100

Le seguenti informazioni sono facoltative ed è possibile fornirle in qualsiasi momento durante il processo di sviluppo:

Supporto per processi business di lunga durata	Consultare, “Aggiunta del supporto per processi business di lunga durata” a pagina 86
Nome pacchetto	Consultare, “Specifica di un pacchetto di collaborazioni” a pagina 88
Livello di transazione minimo	Consultare, “Specifica del livello di transazione minimo” a pagina 86
proprietà configurazione	Consultare, “Definizione delle proprietà di configurazione della collaborazione (scheda Proprietà)” a pagina 94
Variabili maschera	Consultare, “Dichiarazione e modifica delle variabili di una maschera (scheda Dichiarazioni)” a pagina 88
Istruzioni di importazione	Consultare, “Importazione di pacchetti Java” a pagina 89

Creazione della definizione di una maschera

Per creare una nuova maschera di collaborazione, effettuare le seguenti operazioni da System Manager:

1. Fare clic con il tasto destro del mouse sulla cartella Maschere di collaborazione del progetto e quindi selezionare l’opzione Crea nuova maschera di collaborazione. Process Designer Express apre e visualizza la finestra Nuova maschera, come illustrato nella Figura 29.

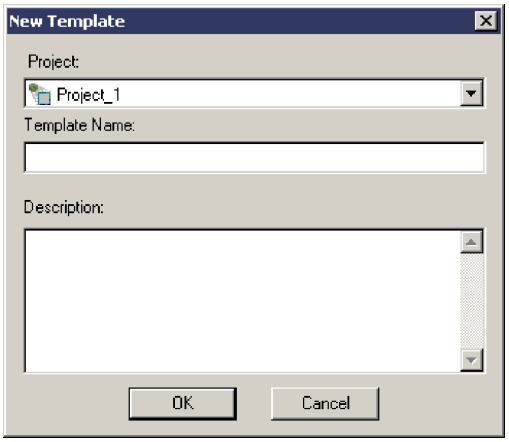


Figura 29. Finestra Nuova maschera

2. nel campo Progetto, utilizzare l’elenco a discesa per selezionare il nome del progetto utente ICL (Integration Component Library) a cui appartiene la maschera.
3. Immettere il nome della maschera nel campo Nome maschera. Un nome di maschera può includere i caratteri alfabetici, i numeri ed i caratteri di sottolineatura. Poiché Process Designer Express crea un file origine (.java) ed un file di classe Java (.class) in base al nome della maschera, si consiglia di seguire le convenzioni di denominazione della classe Java:
   - Iniziare il nome di una maschera di collaborazione con una lettera maiuscola.
   - Se il nome contiene più parole, iniziare ogni parola con una lettera maiuscola (ad esempio, CustomerSync).
   - Non includere degli spazi nel nome.
4. Se lo si desidera, includere una breve descrizione della maschera nel campo Descrizione.

Nota: Nella descrizione della maschera non includere un ritorno a capo.

5. Fare clic su OK. Process Designer Express apre e visualizza la nuova maschera e la struttura ad albero al livello principale nel riquadro Struttura maschera.

Informazioni sulle proprietà della maschera

Una finestra di definizioni della maschera di collaborazione fornisce le quattro schede elencate nella Tabella 27 per la definizione delle proprietà di una maschera di collaborazione.

Tabella 27. Schede della finestra Definizioni

Schede di Definizioni maschera	Descrizione	Per ulteriori informazioni
Generale	Consente la definizione delle seguenti informazioni per una maschera di collaborazione: • Descrizione della maschera • Supporto LLBP (Long-lived Business Processes) • Livello di transazione minimo • Informazioni sul pacchetto	"Definizione delle informazioni generali sulle proprietà (scheda Generale)" a pagina 85
Dichiarazioni	Consente di definire le variabili della maschera e di visualizzare le variabili di maschera generate dal sistema.	"Dichiarazione e modifica delle variabili di una maschera (scheda Dichiarazioni)" a pagina 88
Proprietà	Consente di specificare il nome, il tipo ed il valore delle proprietà della maschera di collaborazione definite dall'utente.	"Definizione delle proprietà di configurazione della collaborazione (scheda Proprietà)" a pagina 94
Porte ed eventi di attivazione	Consente di definire le porte e gli eventi di attivazione per la maschera di collaborazione.	"Definizione di porte ed eventi di attivazione (scheda Porte ed eventi di attivazione)" a pagina 97

Definizione delle informazioni generali sulle proprietà (scheda Generale)

La scheda Generale della finestra Definizioni maschera (vedere la Figura 30) visualizza le informazioni generali sulle proprietà della maschera di collaborazione, incluso quelle elencate nella Tabella 28.

Tabella 28. Informazioni generali della definizione di una maschera

Proprietà maschera generale	Descrizione	Per ulteriori informazioni
Descrizione della maschera di collaborazione	Questo campo della maschera di collaborazione è facoltativo. Vi è possibile inserire del testo che sarà disponibile per tutti gli utenti della maschera di collaborazione.	Nessuno
Supporto per processi business di lunga durata	Specifica se la maschera supporta i processi business di lunga durata (supporto LLBP).	"Aggiunta del supporto per processi business di lunga durata" a pagina 86
Livello di transazione (solo per le collaborazioni di transazioni)	Imposta un livello minimo di transazione per tutte le operazioni della collaborazione.	"Specifica del livello di transazione minimo" a pagina 86
Pacchetto di collaborazioni	Il pacchetto Java in cui risiede la collaborazione	"Specifica di un pacchetto di collaborazioni" a pagina 88

La Figura 30 mostra la scheda Generale contenuta nella finestra Definizioni.

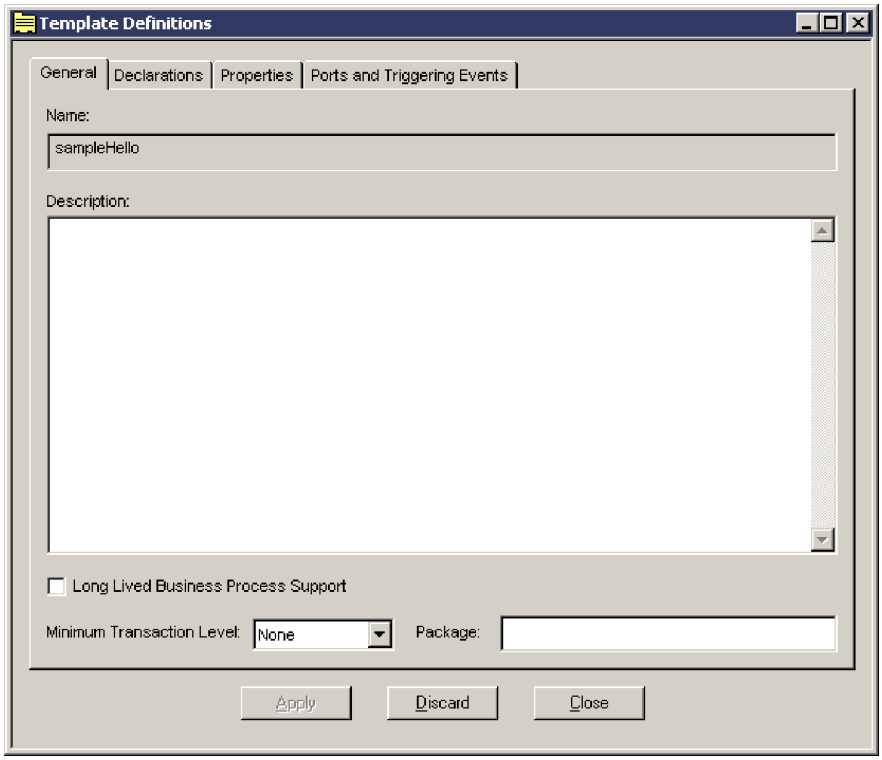


Figura 30. Proprietà generali della collaborazione

Aggiunta del supporto per processi business di lunga durata

Il supporto LLBP per processi business di lunga durata consente di distribuire una collaborazione come processo business di lunga durata e di specificare un valore timeout per le chiamate di servizio in questo ambiente. Per poter utilizzare questa funzione, è necessario effettuare le seguenti operazioni:

- Selezionare l'opzione Supporto processo business di lunga durata sulla scheda Generale
- Facoltativamente, creare una proprietà della maschera di collaborazione definita dall'utente che rappresenti il valore timeout per la chiamata di servizio utilizzata nell'elaborazione business di lunga durata. Consultare "Definizione delle proprietà di configurazione della collaborazione (scheda Proprietà)" a pagina 94 per ulteriori informazioni.

Specifica del livello di transazione minimo

Quando una collaborazione è transazionale, InterChange Server Express esegue il rollback della collaborazione se si verificano degli errori nella transazione. Il rollback esegue delle compensazioni definite nella maschera per regredire le modifiche apportate ai dati della collaborazione. Per una spiegazione sulle collaborazioni transazionali, consultare "Utilizzo delle funzioni transazionali" a pagina 154.

Il livello di transazione determina il meccanismo secondo il quale vengono eseguiti gli scenari della collaborazione. Gli oggetti della collaborazione vengono eseguiti in uno dei livelli di transazione descritti nella Tabella 29.

Tabella 29. Livelli di transazione

Livello di transazione	Effetto	Comportamento del sistema
Nessuno	La collaborazione non è transazionale.	Se si verifica un errore durante l'esecuzione della collaborazione, il sistema lo invia semplicemente al log e termina l'esecuzione.
Minimo sforzo	La collaborazione è transazionale; ha definite delle compensazioni per le transazioni secondarie del suo scenario.	Se si verifica un errore durante l'esecuzione di quello scenario, InterChange Server Express effettua il rollback dello scenario, eseguendo la compensazione su ciascun passo delle transazioni secondarie.
Massimo sforzo	Ha lo stesso comportamento del Minimo sforzo; in aggiunta alle compensazioni, viene utilizzato l'isolamento dei dati per garantirne la correttezza.	InterChange Server Express verifica che i dati vengano isolati virtualmente per la durata del loro utilizzo nella collaborazione transazionale controllando che il valore dei dati non sia cambiato dal precedente utilizzo. La verifica di isolamento di Massimo sforzo lascia un breve lasso di tempo durante il controllo in cui i dati sono vulnerabili alle modifiche da parte di altre transazioni dell'applicazione.
Rigoroso	Ha lo stesso comportamento del Massimo sforzo ma elimina la vulnerabilità della finestra di isolamento dei dati.	L'applicazione blocca i dati quando viene verificato l'isolamento. Supportato da applicazioni le cui API supportano un'operazione "testa e imposta" atomica.

Uno sviluppatore di maschere di collaborazione imposta il livello minimo di transazione per gli oggetti della collaborazione creati dalla maschera. Ad esempio, se una collaborazione ha a che fare con dati critici e si desidera essere sicuri che venga eseguito sempre il rollback quando ci sono errori, è possibile impostare il livello di transazione minimo su Minimo sforzo. Se si progetta una collaborazione per l'esecuzione transazionale ma che può essere utilizzata con esito positivo senza le funzioni transazionali, è possibile impostare il livello di transazione minimo su Nessuno.

Un amministratore può aumentare il livello di transazione dell'oggetto di una collaborazione se i relativi connettori supportano il livello di transazione superiore. Il livello di transazione di un oggetto collaborazione, tuttavia, non può essere inferiore al minimo specificato nella maschera.

Suggerimenti

E' possibile creare la compensazione per consentire l'operazione transazionale anche se si imposta il livello di transazione minimo delle maschere di collaborazione su Nessuno. Se è necessario un maggiore rigore ed i connettori utilizzati possono supportare un livello di transazione superiore, un amministratore può aumentare il livello di transazione dell'oggetto di una collaborazione durante il collegamento. Per ulteriori informazioni sulla compensazione, consultare "Definizione della compensazione" a pagina 140.

Per assegnare un livello di transazione minimo alla maschera di collaborazione, eseguire le seguenti operazioni:

1. Verificare che la finestra Definizioni maschera sia aperta e che la scheda Generale sia visualizzata.

2. Utilizzare il menu a discesa Livello transazione per selezionare il livello di transazione minimo che si desidera utilizzare. Se si sta modificando una maschera di collaborazione che non è transazionale, mantenere il valore predefinito Nessuno.
3. Fare clic su Applica per salvare le modifiche.

Specifica di un pacchetto di collaborazioni

Un pacchetto è un gruppo di collaborazioni che hanno delle funzioni correlate. Tutte le collaborazioni alle quali accede Process Designer Express appartengono al pacchetto UserCollaborations o ad un pacchetto secondario di UserCollaborations. Pertanto il pacchetto UserCollaborations include:

- Le collaborazioni fornite con il software InterChange Server Express
- Le collaborazioni create da altri sviluppatori di collaborazioni

E' possibile creare dei pacchetti secondari in UserCollaborations per raggruppare delle maschere di collaborazione personalizzate. Ad esempio, se si creano diverse maschere di collaborazione che hanno a che fare con la gestione delle forniture per l'ufficio, è possibile creare un pacchetto secondario denominato OfficeSupplyMgmt. In esso è possibile collocarvi le collaborazioni PaperClipMgmt e PencilInventory.

Se si specifica che una maschera di collaborazione fa parte di un pacchetto, Process Designer Express utilizza il nome del pacchetto per creare una directory secondaria nella directory Template\Classes del progetto ICL (Integration Component Library). (Durante la distribuzione viene creata la directory *ProductDir*\collaborations\classes\UserCollaborations per memorizzare le informazioni relative al pacchetto).

L'installazione del prodotto imposta la variabile di ambiente CLASSPATH per includere tutte le collaborazioni di UserCollaborations nel percorso classe. Process Designer Express colloca nella directory secondaria un file .class ed un file .java della maschera di collaborazione.

Per specificare un pacchetto in cui memorizzare la maschera di collaborazione:

1. Verificare che la finestra Definizioni maschera sia aperta e che la scheda Generale sia visualizzata.
2. Nel campo Pacchetto, immettere il nome del pacchetto in cui memorizzare la maschera di collaborazione.

   Quando si specifica il nome di un pacchetto esistente, Process Designer Express aggiunge la maschera di collaborazione al pacchetto. Se si specifica il nome di un pacchetto che non esiste ancora, Process Designer Express lo crea.
3. Fare clic su Applica per salvare le modifiche.

E' possibile rivedere la definizione di una maschera di collaborazione esistente in qualsiasi momento, per aggiungervi un nome di pacchetto o modificare quello esistente.

Dichiarazione e modifica delle variabili di una maschera (scheda Dichiarazioni)

La scheda Dichiarazioni della finestra Definizioni maschera visualizza le informazioni relative alle variabili di maschera della maschera di collaborazione. *Le variabili di maschera* sono delle variabili di collaborazione il cui ambito sono tutti gli scenari contenuti in una collaborazione, vale a dire una variabile di maschera è globale per tutti gli scenari di una collaborazione. (Sono paragonabili alle variabili

di classe nel linguaggio di programmazione Java). Ad esempio, una collaborazione che implica delle transazioni con clienti può avere una variabile di maschera customerID che identifica il cliente in tutti gli scenari. E' possibile creare o modificare le variabili di maschera in qualunque momento durante lo sviluppo.

La Figura 31 mostra la scheda Dichiarazioni nella finestra Definizioni maschera.

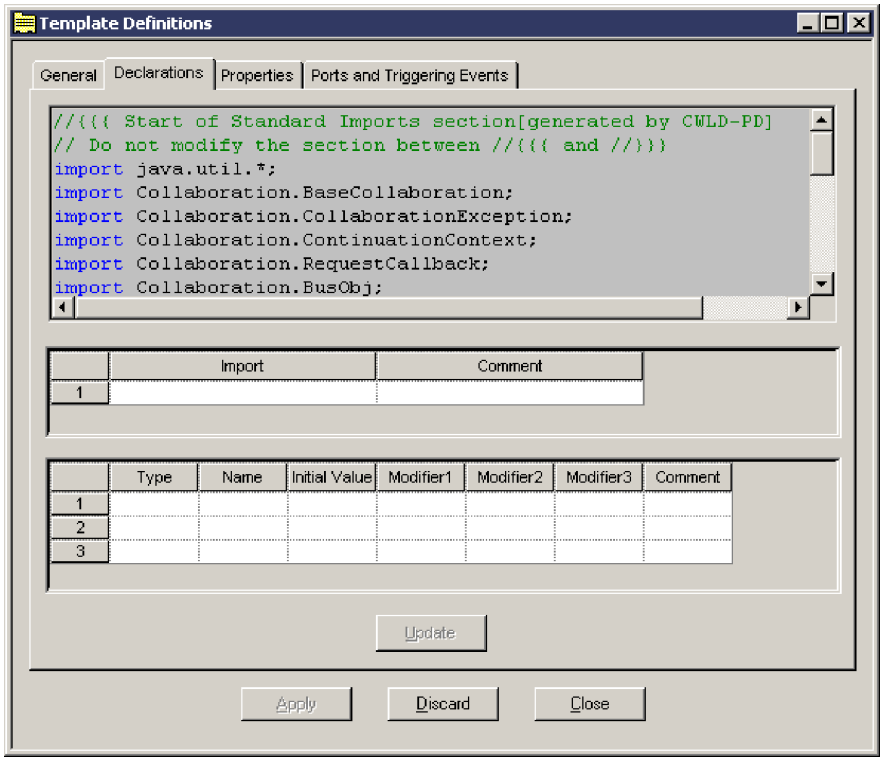


Figura 31. Scheda Dichiarazioni della finestra Definizioni maschera

Nella scheda Dichiarazioni sono possibili le seguenti attività:

- Specificare il codice di qualunque istruzione import. Consultare "Importazione di pacchetti Java" per ulteriori informazioni.
- Inserire il testo della dichiarazione per le variabili di maschera definite dall'utente. Consultare "Dichiarazione delle variabili di maschera" a pagina 92 per ulteriori informazioni.
- Visualizzare le variabili di maschera generate dal sistema (non sono modificabili). Consultare "Variabili generate dal sistema" a pagina 93 per ulteriori informazioni.

Importazione di pacchetti Java

E' possibile utilizzare la scheda Dichiarazioni per importare nella collaborazione delle specifiche classi Java. Una classe Java importa un pacchetto di altre classi per ottenere l'accesso alle loro funzioni. Ad esempio, una classe importa i pacchetti java.math, java.security e java.text per utilizzare rispettivamente le loro funzioni di aritmetica, sicurezza ed internazionalizzazione. Poiché una maschera di collaborazione è una classe, può utilizzare le classi o gruppi di classi (denominati pacchetti) forniti da JDK (Java Development Kit) o da prodotti forniti da terzi.

Tutte le classi Java per impostazione predefinita importano implicitamente le classi nel pacchetto java.lang. Inoltre, Process Designer Express implicitamente importa le classi nel pacchetto java.util da utilizzare con tutte le maschere di collaborazione.

La seguente dichiarazione di importazione importa le classi java.math da JDK. (l'asterisco indica di importare tutte le classi nel pacchetto specificato):

```
java.math.*;
```

In alternativa, la seguente istruzione importa solo la classe BigDecimal del pacchetto:

```
java.math.BigDecimal;
```

E' possibile aggiungere delle dichiarazioni di importazione in qualunque momento al codice durante lo sviluppo di una collaborazione.

Per importare le classi Java:

1. Verificare che la finestra Definizioni maschera sia aperta e che la scheda Dichiarazioni sia visualizzata.
2. Posizionare il cursore nella cella dell'intestazione della tabella di importazione. Fare clic con il tasto destro del mouse e selezionare Aggiungi, come illustrato nella Figura 32. Viene aggiunta una nuova riga alla tabella.

   Nota: E' possibile aggiungere una nuova riga anche facendo clic sull'ultima riga presente al momento sulla tabella.

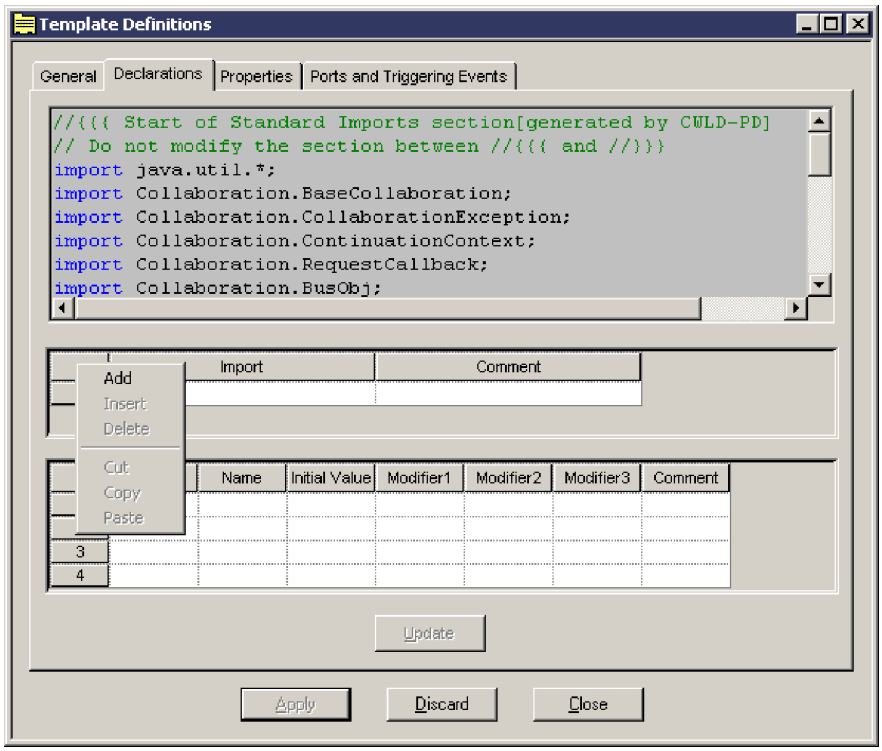


Figura 32. Aggiunta di una dichiarazione di importazione

3. Immettere la dichiarazione di importazione nella colonna Importazione. Ad esempio:

   ```
   java.math.*
   ```

4. Facoltativamente, inserire una breve descrizione della dichiarazione di importazione nella colonna Commento.
5. Fare clic su Applica per salvare le modifiche.

6. Ripetere dal passo 2 a pagina 90 al passo 5 a pagina 90 per aggiungere altre dichiarazioni di importazione. E' possibile inserire un numero illimitato di istruzioni import in una maschera di collaborazione.

Se le classi importate sono contenute in un pacchetto fornito da terzi, piuttosto che nel JDK, è necessario modificare la sezione *ProductDir*\bin\cwtools.cfg `[codeGeneration]` per riflettere il percorso del pacchetto prima di compilare la maschera.

Prima di distribuire una collaborazione che utilizza le classi importate da un pacchetto fornito da terzi, è necessario aggiornare la variabile JCLASSES nel sistema su cui viene distribuita la collaborazione. Se le classi importate sono contenute in un pacchetto fornita da terzi piuttosto che nel JDK, è necessario aggiungerle al percorso delle classi importate nella variabile JCLASSES. IBM consiglia di utilizzare qualche tipo di meccanismo per differenziare in JCLASSES le classi standard da quelle personalizzate. Ad esempio è possibile creare una nuova variabile per contenere solo le classi personalizzate ed inserirla in JCLASSES nel seguente modo:

1. Collocare il file CwMacroUtils.jar in una directory propria. Ad esempio, creare una directory \dependencies nella directory del prodotto e mettervi il file .jar.
2. Modificare il file utilizzato per avviare InterChange Server Express (per impostazione predefinita, *ProductDir*\bin\start_server.bat o *ProductDir*/bin/CWSharedEnv.sh), in modo da includere il nuovo percorso per il file CWMacroUtils.jar. Aggiungere la seguente voce al file:

   ```
   set DEPENDENCIES=ProductDir/dependencies/CwMacroUtils.jar
   ```

   dove *ProductDir* è la posizione in cui è installato InterChange Server Express.
3. Aggiungere la nuova variabile DEPENDENCIES alla voce JCLASSES nel modo indicato di seguito, a seconda del sistema operativo.

   Su un sistema Unix, utilizzare la seguente sintassi, dove *ExistingJarFiles* rappresenta i file .jar già inclusi in JCLASSES:

   ```
   set JCLASSES = $JCLASSES: ExistingJarFiles:$DEPENDENCIES
   ```

   Su un sistema Windows utilizzare la seguente sintassi, dove *ExistingJarFiles* rappresenta i file .jar già inclusi in JCLASSES:

   ```
   set JCLASSES = ExistingJarFiles;%DEPENDENCIES%
   ```
4. In ogni collaborazione che utilizza le classi, includere il *PackageName.ClassName* specificato nel file CwMacroUtils.jar.
5. Riavviare InterChange Server Express per rendere i metodi disponibili alle collaborazioni.

Quando si importa una classe personalizzata, si potrebbe ricevere un messaggio di errore che indica che il software InterChange Server Express non è stato in grado di trovare 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. Accertarsi che il codice della classe personalizzata includa un'istruzione di pacchetto corretta e che sia presente all'inizio del file origine, prima di qualsiasi dichiarazione della classe o dell'interfaccia.
- Verificare che l'istruzione di importazione sia corretta nella maschera di collaborazione. L'istruzione di importazione deve fare riferimento al nome pacchetto corretto; può inoltre specificare il nome della classe personalizzata o

fare riferimento a tutte le classi contenute nel pacchetto. Ad esempio, se il nome del pacchetto è COM.acme.graphics e la classe personalizzata è Rectangle, è possibile importare l'intero pacchetto:

```
COM.acme.graphics.*;
```

Oppure, è possibile importare solo la classe personalizzata Rectangle:

```
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 o quello per la classe personalizzata stessa, se non è contenuta in un pacchetto.

Ad esempio, quando si importa una classe personalizzata, è possibile creare una cartella denominata *ProductDir*\lib\com\crossworlds*pacchetto*, dove *ProductDir* è la posizione in cui InterChange Server Express è stato installato e *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.

Dichiarazione delle variabili di maschera

E' possibile utilizzare la scheda Dichiarazioni anche per dichiarare le proprie variabili di maschera utilizzate dalla collaborazione.

Per utilizzare una variabile è necessario prima dichiararla specificandone il tipo ed il nome. Una variabile di una maschera di collaborazione può essere uno dei seguenti tipi di dati:

- Un tipo di dati base, o primitivo, come un byte, short, int, long, float, double, char o boolean
- Una classe Java, come String o Integer
- Una classe definita da InterChange Server Express, come BusObj, BusObjArray o CollaborationException
- Una classe definita dall'utente, se si tratta di un utente avanzato

Nota: LongText e Date sono delle designazioni specifiche del prodotto per delle stringhe con scopi speciali negli attributi dell'oggetto business. Utilizzare nel codice il tipo di dati String per rappresentare una variabile per un attributo di oggetto business il cui tipo è LongText o Date.

Per dichiarare le variabili di maschera, effettuare le seguenti operazioni:

1. Verificare che la finestra Definizioni maschera sia aperta e che la scheda Dichiarazioni sia visualizzata.
2. Posizionare il cursore nella cella dell'intestazione della tabella di variabili. Fare clic con il tasto destro del mouse e selezionare Aggiungi. Viene aggiunta una nuova riga alla tabella.

   Nota: E' possibile aggiungere una nuova riga anche facendo clic sull'ultima riga presente al momento sulla tabella.
3. Utilizzare il menu a discesa nella colonna Tipo per specificare il tipo di variabile che si desidera dichiarare.
4. Specificare il nome della variabile nella colonna Nome.
5. Specificare il valore iniziale della variabile nella colonna Valore iniziale.

   Nota: I valori di stringa devono essere immessi fra virgolette (ad esempio, la stringa Sì deve essere inserita come "Sì."

6. Specificare gli eventuali modificatori che si desidera applicare alla variabile (ad esempio, public, private, protected) nelle colonne Modifier1, Modifier2, e Modifier3. Non è necessario specificare un modificatore in tutte e tre le colonne.

   Nota: Non utilizzare il modificatore `Statico` quando si definiscono le variabili di maschera.
7. Fare clic su Aggiorna per aggiungere la nuova variabile all'elenco di dichiarazioni nella parte superiore della tabella, e fare clic su Applica per salvare le modifiche.

E' possibile dichiarare le variabili i cui valori restano costanti nei molteplici richiami di una collaborazione. Si supponga di voler tenere un conteggio di un'azione in una collaborazione e si desidera che questo conteggio venga incrementato ad ogni esecuzione separata della collaborazione. Utilizzare la tabella delle variabili nella scheda Dichiarazioni per creare una variabile numero intero (integer) denominata ctr, pubblica.

Successivamente, all'interno dello stesso codice della collaborazione, aumentare il conteggio:

```
ctr = ctr+1;
```

La variabile ctr aumenta ad ogni esecuzione della collaborazione.

Considerazioni particolari per le variabili utilizzate con i processi business di lunga durata: Se una collaborazione deve essere distribuita come processo business di lunga durata, verificare che tutte le variabili che si desidera restino costanti siano definite come variabili di maschera globali o variabili di porta globali.

Inoltre, verificare che quelle variabili appartengano ad uno dei seguenti tipi:

- Tipi di dati Java serializzabili, incluso byte, short, int, long, float, double, char, boolean, string, Integer o qualsiasi tipo di dati definito dall'utente che implementi l'interfaccia Java Serializable o Externalizable
- Tipo di dati BusObj di InterChange Server Express
- Tipo di dati BusObjArray InterChange Server Express

Le variabili di altro tipo non persistono in un processo business di lunga durata.

Variabili generate dal sistema

Process Designer Express dichiara automaticamente le seguenti variabili di collaborazione:

- Due variabili di collaborazione che sono disponibili in tutte le collaborazioni: triggeringBusObj e currentException.
- Una variabile per ogni porta.

La Tabella 30 elenca e descrive le variabili generate dal sistema.

Tabella 30. Variabili generate dal sistema

Variabile	Descrizione
triggeringBusObj	La variabile triggeringBusObj contiene l'attivazione del flusso (evento di attivazione o chiamata di accesso di attivazione) di uno scenario. L'attivazione del flusso è composta da un oggetto business ed un'istruzione (verb). Un evento di attivazione rappresenta un evento dell'applicazione ed i relativi dati. L'arrivo dell'attivazione del flusso avvia l'esecuzione di uno scenario. Questa è una variabile di maschera, vale a dire che il suo ambito è l'intera collaborazione.
currentException	La variabile currentException contiene un oggetto eccezione generato dall'azione, azione secondaria o iteratore immediatamente precedenti. Process Designer Express dichiara implicitamente currentException, il cui ambito è l'azione che segue immediatamente la generazione di un'eccezione. Uno scenario deve verificare il valore di currentException nel collegamento della transazione o nel frammento di codice che segue immediatamente l'attività che ha prodotto l'eccezione.
Variabili di porta	Process Designer Express dichiara una variabile di maschera per l'oggetto business associato ad ogni porta nella collaborazione. Queste dichiarazioni generate sono visibili nella scheda Dichiarazioni della finestra Definizioni maschera. Il nome di ogni variabile di porta è il nome della porta con `BusObj` attaccato. Ad esempio, se il nome porta è SourceInvoice, il nome variabile è SourceInvoiceBusObj. La dichiarazione inoltre crea un'istanza BusObj dello stesso tipo per il quale è definita la porta. Inizialmente imposta gli attributi dell'oggetto business su null. E' possibile utilizzare queste variabili di porta per gestire l'evento di attivazione. Per ulteriori informazioni fare riferimento a "Copia dell'evento di attivazione" a pagina 220.

Definizione delle proprietà di configurazione della collaborazione (scheda Proprietà)

le maschere di collaborazione dispongono di due tipi di proprietà di configurazione:

- *Proprietà standard* fornisce le informazioni necessarie a tutte le collaborazioni, come il livello di traccia e l'indirizzo email per le notifiche dei messaggi. Tutte le collaborazioni hanno le stesse proprietà di configurazione standard, che sono definite da InterChange Server Express. Consultare "Proprietà standard per le maschere di collaborazione" a pagina 484 per ulteriori informazioni.
- *Proprietà specifiche per la collaborazione*, che sono facoltative; vengono definite da uno sviluppatore di collaborazioni. La collaborazione utilizza il valore della proprietà per determinare un aspetto del suo comportamento. Le proprietà possono essere di uno dei seguenti tipi:
  - Data (Date)
  - Doppio (Double)
  - Mobile (Float)
  - Intero (Integer)
  - Booleano (Boolean)
  - Stringa (String)
  - Ora (Time)

– URL

Un amministratore utilizza entrambi i tipi di proprietà quando configura una collaborazione.

Come sviluppatore di collaborazioni, l'utente decide se una collaborazione necessita di proprietà di configurazione specifiche per la collaborazione. In tal caso, ne definisce i nomi ed i valori predefiniti. Queste proprietà di configurazione abilitano un utente di collaborazione a specificare i dati che ne influenzano il comportamento.

La Tabella 31 fornisce alcuni esempi di tipi di proprietà che è possibile creare.

Tabella 31. Esempi di proprietà di configurazione specifici per la collaborazione

Tipo di proprietà	**Esempio**
Un valore che la collaborazione utilizza per impostare il valore di un attributo.	Una collaborazione potrebbe richiedere che un'applicazione crei le fatture dei clienti. La collaborazione può impostare il valore di un attributo Rate in un oggetto business Invoice. Se la collaborazione ha una proprietà denominata BILLING_RATE, un amministratore può aumentare o diminuire la tariffa in base alla prassi corrente del business.
Valore true o false, che determina se la collaborazione deve prendere un particolare percorso di esecuzione.	Le collaborazioni di InterChange Server Express che sincronizzano le modifiche con le entità delle applicazioni, in genere hanno una proprietà denominata CONVERT_CREATE. Quando la collaborazione riceve un evento di aggiornamento, cerca nell'applicazione di destinazione l'entità da aggiornare. Se l'entità non esiste, la collaborazione verifica il valore della proprietà CONVERT_CREATE. Se la proprietà è impostata su true, la collaborazione converte la richiesta Aggiorna in una richiesta Crea.

L'utilizzo delle proprietà di configurazione specifiche per collaborazione è facoltativo e in una maschera è possibile utilizzarne un numero illimitato. E' possibile aggiungere queste proprietà in qualunque momento durante lo sviluppo. Se fin dall'inizio è al corrente di quali proprietà necessita la collaborazione, è possibile crearle prima di produrre i modelli degli scenari. Quando, inoltre, ci si trova nel mezzo della creazione di un modello di scenario, è possibile definire ulteriori proprietà per supportare la logica della collaborazione.

Per creare una proprietà di configurazione specifica per la maschera di collaborazione:

1. Verificare che la finestra Definizioni maschera sia aperta e che la scheda Proprietà sia visualizzata.

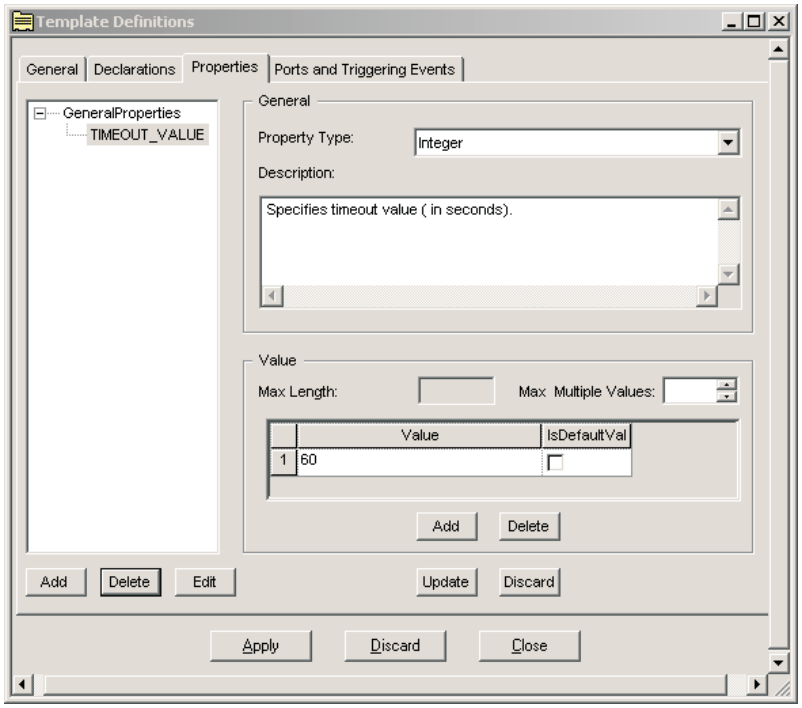


Figura 33. Aggiunta di proprietà specifiche per collaborazione

2. Fare clic sul pulsante Aggiungi per creare una proprietà di configurazione. Viene aperta la finestra di dialogo Nome.
3. Inserire il nome della proprietà nel campo Nome e fare clic su OK.

   Nota: Per convenzione, i nomi delle proprietà di configurazione sono a lettere maiuscole e utilizzano i caratteri di sottolineatura per separare le parole. IBM raccomanda che i nomi delle proprietà di configurazione indichino chiaramente lo scopo o la funzione della proprietà, poiché gli amministratori devono leggere e comprendere ogni proprietà.
4. Utilizzare il menu a discesa Tipo proprietà per selezionare il tipo di proprietà.
5. Se lo si desidera, fornire una descrizione della nuova proprietà nel campo Descrizione.
6. Se il tipo di proprietà è una stringa, specificare un valore nel campo Lunghezza massima.
7. Facoltativamente, utilizzare il campo Numero massimo valori multipli per specificare il numero massimo di valori multipli accettato per la proprietà. Il numero specificato in questo campo limita anche il numero di valori predefiniti di cui può disporre la proprietà. Ad esempio, se si imposta Numero massimo valori multipli su 2, è possibile avere solo due valori predefiniti per la proprietà, a prescindere da quanti valori possibili sono associati ala proprietà. Se non si specifica un valore in questo campo, il valore predefinito è 1.

   Nota: L'attributo Numero massimo valori multipli di una proprietà specifica per collaborazione non viene usato spesso. La maggior parte delle proprietà specifiche per collaborazione accetta solo un unico valore.
8. Fare clic su Aggiungi nel riquadro Valore. Viene aggiunta una nuova riga alla tabella e viene visualizzata la finestra di dialogo Valore proprietà.

9. Inserire un valore nel campo Valore oppure specificare un intervallo di valori nel campo Intervallo da, quindi fare clic su OK. La finestra di dialogo viene chiusa e la colonna Valore è compilata con l'informazione.
10. Se si tratta di un valore predefinito, contrassegnare la casella di controllo nella colonna IsDefaultValue.
11. Ripetere dal passo 8 a pagina 96 al passo 10 per ogni valore che si desidera aggiungere alla definizione della proprietà.
12. Fare clic su Aggiorna.
13. Ripetere dal passo 2 al passo 12 per aggiungere tutte le proprietà di configurazione necessarie.
14. Quando si è terminato di aggiungere le proprietà di configurazione, fare clic su Applica per salvare le modifiche.

Per eliminare una proprietà di configurazione specifica per collaborazione, selezionare il nome della proprietà dall'elenco nel riquadro a sinistra della scheda e fare clic su Elimina.

Aggiunta delle proprietà per supportare i processi business di lunga durata

Se si desidera supportare i processi business di lunga durata (LLBP) con i valori di timeout dinamici della chiamata di servizio, è possibile utilizzare una variabile Java oppure una proprietà specifica per collaborazione. Se si desidera utilizzare una proprietà specifica per collaborazione, è necessario crearla durante la definizione della maschera di collaborazione. L'utilizzo di proprietà specifiche per collaborazione abilita l'impostazione del valore timeout durante il runtime, piuttosto che utilizzare un valore statico fornito durante la creazione iniziale della chiamata di servizio. Utilizzare il tipo di dati integer (numero intero), quando si creano le proprietà per i valori timeout dinamici.

Ad esempio, se si prevede di avere una chiamata di servizio dalla porta A che invia un oggetto business con una richiesta di creazione, è possibile definire una proprietà di collaborazione denominata CreateTimeout. Quando si definisce la chiamata di servizio, utilizzare la proprietà CreateTimeout per specificare il punto in cui quella chiamata di servizio supera il tempo limite. Per i dettagli sulla creazione di chiamate di servizio consultare "Chiamate di servizio" a pagina 134.

E' possibile utilizzare anche un valore timeout fisso specificato durante la creazione e la definizione di una chiamata di servizio; in questo caso non è necessaria la proprietà di collaborazione. Consultare "Definizione del tipo di chiamata di servizio" a pagina 138.

Definizione di porte ed eventi di attivazione (scheda Porte ed eventi di attivazione)

La scheda Porte ed eventi di attivazione della finestra Definizioni maschera visualizza le informazioni relative a quanto segue:

- Le porte della collaborazione

  In una maschera di collaborazione, una *porta* è una variabile che rappresenta un oggetto business che l'oggetto collaborazione riceve o produce al runtime.
- L'evento di attivazione della collaborazione

  Un oggetto business rappresenta l'evento di attivazione o l'azione. Quando una collaborazione riceve un oggetto business da un connettore, di solito risponde con un'azione. Questi oggetti business sono noti come attivazioni o eventi di attivazione.

Nota: Il termine generale di un oggetto business che una collaborazione riceve è un'attivazione flusso. Quando una collaborazione riceve un oggetto business da un connettore, questa attivazione flusso è un evento di attivazione. Quando una collaborazione riceve un oggetto business da un client di accesso, questo oggetto business viene denominato chiamata di accesso di attivazione. Una porta il cui oggetto business è associato ad una chiamata di accesso di attivazione viene definita nello stesso modo di quella associata ad un evento di attivazione.

Per informazioni dettagliate sull'utilizzo della scheda Porte ed eventi di attivazione per la definizione di un evento di attivazione, consultare "Assegnazione di eventi di attivazione agli scenari" a pagina 101.

Quando una collaborazione porta a termine un'operazione, di solito invia un oggetto business al connettore che ha avviato l'azione. Pertanto, InterChange Server Express spesso fa riferimento alle porte in termini di eventi di attivazione o di invio.

Nota: Se si aggiunge, modifica o elimina un oggetto business nel repository utilizzando Business Object Designer Express o System Manager, l'elenco delle definizioni di oggetti business visualizzato in Process Designer Express viene aggiornato automaticamente. *Non* è necessario riavviare System Manager o Process Designer Express per visualizzare i risultati delle modifiche dinamiche nel campo oggetto business della tabella Porte ed eventi di attivazione nella finestra Definizioni maschera.

Importante

IBM raccomanda l'utilizzo di questa funzione di aggiornamento dinamico solo in un ambiente di sviluppo. Possono risultare possibili complicazioni dall'aggiornamento di un oggetto business. L'aggiornamento dinamico può influenzare le altre funzioni nel sistema, incluso come elaborare qualsiasi evento che utilizza la vecchia definizione dell'oggetto business e come inoltrare di nuovo i flussi non risolti inviati originariamente sulla vecchia definizione dell'oggetto business. Questi ed altri scenari possono provocare una discrepanza fra le definizioni dell'oggetto business in fase di elaborazione e quelle presenti in memoria. Quindi, nel sistema di produzione, IBM raccomanda di eseguire gli aggiornamenti alle definizioni dell'oggetto business solo quando non vengono elaborati eventi sul sistema.

Per ulteriori argomentazioni sulle porte della collaborazione, consultare il capitolo relativo alla collaborazione nel manuale *System Implementation Guide*.

Creazione di una porta

Per creare una porta, effettuare le seguenti operazioni:

1. Verificare che la finestra Definizioni maschera sia aperta e che la scheda Porte ed eventi di attivazione sia visualizzata.

   Nella parte superiore della finestra è presente una tabella che contiene i nomi porta, i tipi di oggetti business e le istruzioni. La tabella è vuota se non è stata ancora creata una porta per questa maschera di collaborazione.
2. Per aggiungere una nuova porta alla tabella Porte fare clic su Aggiungi.
3. Immettere un nome porta nella colonna Porta della tabella.

   Seguire queste linee guida per la definizione di un nome per una porta:

   - Iniziare il nome con un carattere alfabetico e nel nome utilizzare solo caratteri alfanumerici ed il simbolo di sottolineatura.

- Anche se i nomi porta non sono sensibili a maiuscolo e minuscolo, è necessario fare riferimento al nome porta sempre utilizzando il tipo di carattere in cui è stato definito.
- In generale, è utile assegnare dei nomi porta che aiutino a ricordare lo scopo della porta. I nomi porta vengono utilizzati in tutto il processo di sviluppo ed i nomi porta auto esplicativi semplificano l'impegno per lo sviluppo.
- Gli sviluppatori di collaborazioni spesso creano un nome porta combinando il tipo di oggetto business e la sua designazione del ruolo, come "In" e "Out" o "Origine" e "Destinazione". Per esempio, si può chiamare una porta OrigineStudio per indicare che si tratta della porta per l'applicazione origine e che è configurata per oggetti business di studio.

4. Selezionare il tipo di porta dall'elenco a discesa nella colonna Tipo oggetto business. Questo è il tipo di definizione di oggetto business che questa porta supporta.
5. Fare clic su Applica per salvare le modifiche.

Nota: In alcune situazioni, non tutte le porte di un oggetto collaborazione sono necessarie; in tal caso è necessario configurare la logica di collaborazione per impedire l'esecuzione di chiamate di servizio per porte non utilizzate.

Poiché InterChange Server Express richiede che tutte le porte di collaborazione siano collegate, è necessario collegare ad un connettore Porta anche le porte non utilizzate. Un *connettore Porta* è una definizione generica di connettore che viene utilizzata per chiudere una porta non utilizzata. Il connettore Porta deve essere utilizzato unitamente alla corretta logica di collaborazione; qualsiasi chiamata di servizio inviata ad una porta collegata ad un connettore Porta blocca il thread della collaborazione.

Modifica di un nome porta

Per modificare il nome di una porta è necessario eliminare e ricreare la porta utilizzando il nuovo nome; non è possibile modificarne semplicemente il nome. Per ridenominare una porta effettuare le seguenti operazioni:

1. Selezionare la porta nella tabella sulla scheda Porte ed eventi di attivazione.
2. Fare clic su Elimina porta.
3. Seguire le istruzioni contenute in "Creazione di una porta" a pagina 98 per creare una porta con il nuovo nome.

La Tabella 32 riassume cosa accade quando una porta viene eliminata e ricreata.

Tabella 32. Risultato della modifica di un nome porta

Cosa fa Process Designer Express	Cosa deve fare l'utente
La variabile di maschera generata dal sistema che utilizza il nome porta cambia.	Se si dispone di codice che utilizza la variabile dichiarata con il vecchio nome di porta, modificare il nome variabile nel codice. Trovare tutti i nodi azione e le chiamate di servizio in cui viene visualizzata la variabile dichiarata con il vecchio nome porta. Il compilatore rileva gli eventuali nomi non corretti rimasti.
L'assegnazione di attivazioni di flusso (eventi di attivazione o chiamate di accesso di attivazione) viene eliminata.	Riassegnare le attivazioni di flusso. Consultare "Compilazione di una maschera di collaborazione" a pagina 105.

Definizione di scenari

Uno *scenario* è il codice della maschera di collaborazione che gestisce un determinato oggetto business o serie di oggetti business in ingresso. Questo oggetto business può rappresentare un evento (proveniente da un connettore) o una chiamata di accesso (proveniente da un client di accesso). Si può pensare ad uno scenario come ad un metodo di gestione eventi della classe di maschere di collaborazione. I diagrammi di attività contengono il codice che specifica come gestire l'evento.

Informazioni sugli scenari

Gli scenari vengono utilizzati per effettuare una suddivisione del problema business che una collaborazione risolve. E' possibile raggruppare tutta la logica della collaborazione in un unico scenario oppure si possono creare diversi scenari, ognuno peculiare per un aspetto del problema. Raggruppare tutta la logica di collaborazione in un unico scenario è simile ad un programma che contiene tutta la sua logica in una funzione main(), mentre l'utilizzo di più scenari è paragonabile ad un programma strutturato in diverse funzioni separate.

In genere gli scenari vengono denominati in base alla funzione che eseguono. Quando una collaborazione contiene più scenari, ognuno dei quali gestisce un tipo di oggetto business, prendere in considerazione di denominare ciascuno scenario in base all'oggetto business che gestisce. Ad esempio, se la collaborazione gestisce un tipo di oggetto business con diverse possibili istruzioni, è possibile sviluppare degli scenari Crea, Aggiorna ed Elimina. Se la collaborazione gestisce diversi tipi di oggetti business, è possibile sviluppare uno scenario per ogni definizione di oggetto business.

Uno scenario gestisce solo un flusso di attivazione (evento di attivazione o chiamata di accesso di attivazione) per ogni esecuzione. Lo stesso scenario, tuttavia, può potenzialmente gestire una serie di possibili flussi di attivazione. Ad esempio, lo stesso scenario può gestire un flusso Crea, Aggiorna o Elimina.

In generale, quando una logica identica gestisce diversi tipi di oggetto business, è più efficace utilizzare un unico scenario per quegli oggetti business. In questo modo si elimina la necessità di testare ed eseguire il debug di più parti di codice.

Nota: Uno scenario non può passare il controllo ad un altro scenario della stessa collaborazione. Se i piani preliminari per una suddivisione della logica di collaborazione indicano che uno scenario deve chiamarne un altro, collocare tutta la logica della collaborazione nello stesso scenario. All'interno dello scenario, la progettazione è molto flessibile. In alternativa, è possibile creare un gruppo di collaborazioni, suddividendo la logica fra le collaborazioni contenute nel gruppo.

Creazione di uno scenario

Per creare un nuovo scenario seguire i passi riportati di seguito:

1. Da Process Designer Express, fare clic su Maschera —> Nuovo scenario. Viene visualizzata ala finestra di dialogo Crea scenario.
2. Immettere il nome dello scenario nel campo Nome scenario.

   Il nome è una stringa che può contenere caratteri alfanumerici ed i simboli di sottolineatura. Se lo scenario gestisce gli eventi con una particolare istruzione, può essere utile includere l'istruzione nel nome dello scenario.
3. Facoltativamente, immettere una descrizione nel campo Descrizione.

4. Fare clic su OK. Nella vista ad albero della maschera, il nome del nuovo scenario viene visualizzato nella struttura ad albero dello scenario. Inoltre, l'editor di diagrammi viene aperto nella finestra principale.
5. E' necessario assegnare allo scenario almeno un'attivazione flusso. La mancata assegnazione dell'attivazione flusso provoca un errore di runtime. Per istruzioni sull'assegnazione degli eventi di attivazione alla definizione del nuovo scenario, consultare "Assegnazione di eventi di attivazione agli scenari".

Assegnazione di eventi di attivazione agli scenari

L'assegnazione di un evento di attivazione ad uno scenario viene effettuata nella tabella Porte ed eventi di attivazione nell'omonima scheda. Ad ogni scenario creato è necessario assegnare il relativo evento di attivazione. L'evento di attivazione è rappresentato da un oggetto business ed un'istruzione.

Nota: Il termine generale utilizzato per l'oggetto business in ingresso e l'istruzione che uno scenario riceve è "attivazione flusso". Se l'attivazione flusso ha origine da un connettore, viene denominata evento di attivazione. Se ha origine da un client di accesso è un metodo di accesso di attivazione. La scheda Porte ed eventi di attivazione consente di assegnare un'attivazione flusso ad uno scenario, a prescindere che si tratti di evento di attivazione o metodo di accesso di attivazione. Questa sezione utilizza i termini "evento di attivazione" ed "evento" perché le attivazioni di flusso ricevute da connettori sono di gran lunga le più comuni.

Le definizioni di una porta di collaborazione specificano i tipi di oggetti business che la collaborazione può inviare e ricevere. Dopo aver definito le porte e gli scenari della collaborazione, è necessario specificare:

- La porta o le porte attraverso le quali gli eventi di attivazione entrano ed escono

  Nella scheda Porte ed eventi di attivazione, scegliere la riga nella tabella che corrisponde al nome della porta attraverso la quale entra l'evento di attivazione ed il nome oggetto business che rappresenta l'evento.
- L'oggetto che attiva l'esecuzione della collaborazione

  L'attivazione flusso è rappresentata dall'oggetto business della porta e da un'istruzione (combinazioni *oggetto-business.istruzione*). Nella riga della porta e dell'oggetto business per i quali si stanno definendo le attivazioni di flusso, specificare l'attivazione flusso scegliendo la relativa istruzione.
- Lo scenario che gestisce ogni attivazione flusso

La Figura 34 mostra queste associazioni in una maschera di collaborazione la cui porta, Da, supporta il tipo di oggetto business Widget. Lo scenario Crea gestisce l'evento di attivazione Widget.Create e lo scenario Elimina Widget.Delete.

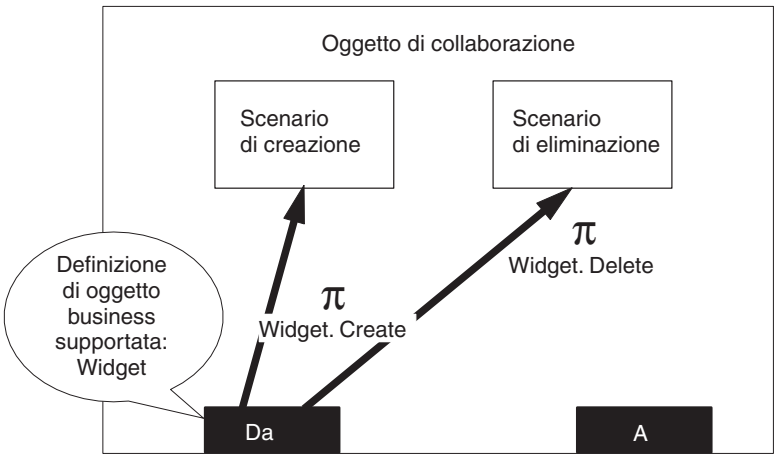


Figura 34. Relazione fra porta, evento di attivazione e scenario

Specificare lo scenario di ogni attivazione flusso effettuando le seguenti operazioni:

1. Verificare che la finestra Definizioni maschera sia aperta e che la scheda Porte ed eventi di attivazione sia visualizzata.
2. Nella tabella Porte ed evento di attivazione individuare la riga che rappresenta la porta dalla quale arriva l'attivazione flusso e l'oggetto business che rappresenta l'attivazione flusso.
3. In quella riga, fare clic sull'elenco a discesa nella colonna Crea. L'elenco contiene tutti gli scenari definiti per la maschera; selezionare lo scenario desiderato.
4. Ripetere il passo 2 ed il passo 3 per ogni porta, oggetto business ed istruzione a cui si desidera assegnare l'attivazione flusso.
5. Una volta terminata l'assegnazione degli eventi di attivazione, fare clic su Applica per salvare le assegnazioni.

Definizione delle variabili di scenario

Dopo aver creato lo scenario, è possibile aggiungere le variabili specifiche per lo scenario nella finestra Definizioni scenario (vedere la Figura 35 a pagina 103).

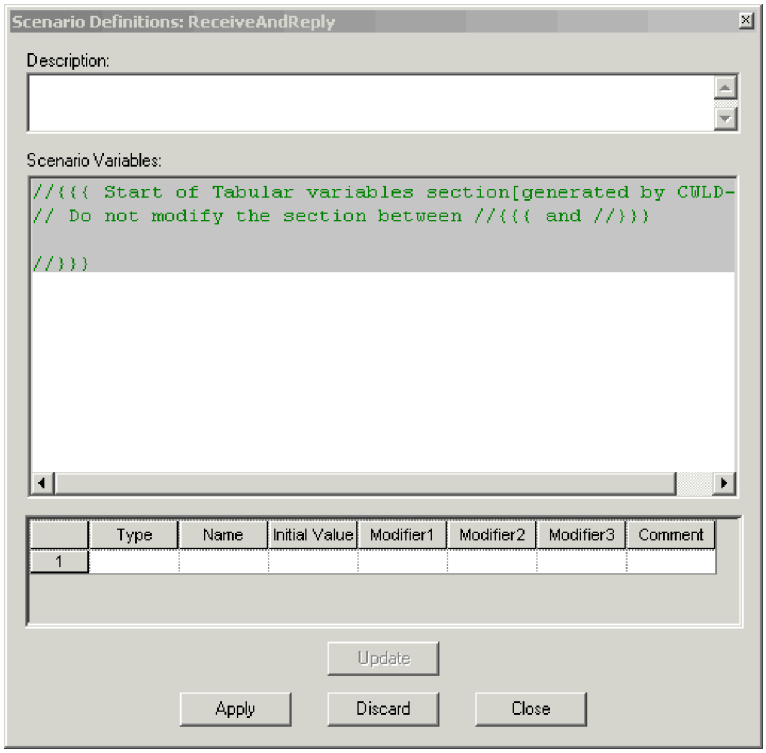


Figura 35. Finestra Definizioni scenario

Le *variabili di scenario* sono delle variabili di collaborazione il cui ambito sono tutte le azioni e tutti i collegamenti contenuti in un unico scenario. (Sono paragonabili alle variabili di classe nel linguaggio di programmazione Java). E' possibile impostare le variabili di scenario in qualunque momento durante il processo di sviluppo della maschera di collaborazione.

Per aggiungere le variabili alla definizione di scenario, procedere come segue:

1. Aprire la finestra Definizioni scenario effettuando una delle seguenti operazioni:
   - Selezionare uno scenario nella vista ad albero della maschera e fare clic su Maschera —> Apri definizione scenario.
   - Selezionare uno scenario nella vista ad albero della maschera e fare clic con il tasto destro del mouse per richiamare il menu di scelta rapida. Da quel menu, fare clic su Apri definizione scenario.
   - Da un diagramma di attività nell'editor di diagrammi, fare clic con il tasto destro del mouse per richiamare il menu d scelta rapida. Da quel menu, fare clic su Apri definizione scenario.
2. Fare clic con il tasto destro del mouse sulla cella di intestazione a sinistra nella tabella delle variabili e selezionare Aggiungi dal menu di scelta rapida. Nella tabella viene visualizzata una nuova riga.

   Nota: E' possibile aggiungere una nuova riga anche facendo clic sull'ultima riga presente al momento sulla tabella.
3. Utilizzare l'elenco a discesa nella colonna Tipo per specificare il tipo di variabile che si desidera dichiarare.
4. Specificare il nome della variabile nella colonna Nome.
5. Specificare il valore iniziale della variabile nella colonna Valore iniziale.

Nota: I valori di stringa devono essere immessi fra virgolette (ad esempio, la stringa Sì deve essere inserita come "Sì."

6. Specificare gli eventuali modificatori che si desidera applicare alla variabile (ad esempio, transient, private, protected) nelle colonne Modifier1, Modifier2, e Modifier3. Non è necessario specificare un modificatore in tutte e tre le colonne.

   Nota: *Non* includere le parole chiave `public` e `static` nella dichiarazione di una variabile di scenario.

7. Fare clic su Aggiorna per aggiungere la nuova variabile all'elenco di dichiarazioni nella parte superiore della tabella e fare clic su Applica per salvare le modifiche.

Considerazioni particolari per le variabili di scenario in processi business di lunga durata

Le variabili di scenario non persistono automaticamente come parte del contesto del flusso di eventi di un processo business di lunga durata. Se si desidera utilizzare le variabili di scenario in una collaborazione di processi business di lunga durata è necessario impostare manualmente la variabile su null prima della chiamata di servizio e quindi reinizializzare la variabile al termine della chiamata di servizio. Queste attività vengono eseguite nel nodo azione che effettua la chiamata di servizio.

Nel seguente esempio, una variabile di scenario denominata poolName viene impostata su null nel nodo azione prima che abbia luogo la chiamata di servizio:

```
String poolName;
poolName = null;
```

Al termine della chiamata di servizio, poolName viene reinizializzato nel nodo azione nel seguente modo:

```
poolName = getConfigProperty("Pool_A");
```

Eliminazione di uno scenario

E' possibile utilizzare Process Designer Express per eliminare gli scenari. Una volta effettuata, l'eliminazione di scenari non può essere annullata.

Per eliminare una definizione di scenario, effettuare le seguenti operazioni:

1. Dalla vista ad albero della maschera selezionare lo scenario che si desidera eliminare.
2. Fare clic su Maschera —> Elimina scenario. Viene visualizzata una finestra di conferma dell'eliminazione.
3. Fare clic su Sì per eliminare lo scenario.

Creazione di un diagramma di attività

Ogni scenario deve avere un diagramma di attività. Un *diagramma di attività* utilizza UML (Unified Modified Language) per costruire il processo business della collaborazione. UML rappresenta i passi e le decisioni del processo business. Un diagramma di attività viene creato nell'editor di diagrammi di Process Designer Express.

Per istruzioni dettagliate sulla creazione di un diagramma di attività, consultare Capitolo 5, "Utilizzo dei diagrammi di attività", a pagina 113.

Creazione del file di messaggi

Parte del processo di creazione di una maschera di collaborazione è la definizione dei relativi messaggi. L'ambiente di collaborazione runtime utilizza il contenuto del file di messaggi come testo per la registrazione, la traccia ed i messaggi di eccezione.

Process Designer Express fornisce una vista Messaggi maschera per semplificare la creazione dei messaggi. Il testo del messaggio specificato viene memorizzato come parte della maschera di collaborazione. Quando si compila e si distribuisce la maschera, Process Designer Express estrae il contenuto del messaggio e crea o aggiorna il file di messaggi per l'utilizzo nel runtime.

Per istruzioni dettagliate sulla creazione di un file di messaggi, consultare Capitolo 10, "Creazione del file di messaggi", a pagina 247.

Compilazione di una maschera di collaborazione

L'attività finale richiesta per la creazione di una maschera di collaborazione è la compilazione della maschera. Dopo aver definito le proprietà della maschera, gli scenari, i diagrammi di attività ed il file di messaggi, è necessario compilare la maschera di collaborazione. Durante la compilazione vengono creati i seguenti file:

- File origine Java (*CollaborationName*.java)
- File eseguibile (C*ollaborationName*.class)
- File di testo dei messaggi (*CollaborationName*.txt)

Dopo la compilazione della maschera in Process Designer Express, questi file vengono creati nel progetto utente Integration Component Library in System Manager. (Per l'esatta posizione, consultare "Compilazione di una maschera di collaborazione" a pagina 6.)

Process Designer Express fornisce due metodi per compilare le maschere di collaborazione:

- "Compilazione di una maschera singola"
- "Compilazione di più maschere di collaborazione" a pagina 106

Compilazione di una maschera singola

Esistono diversi metodi per avviare la compilazione di una singola maschera di collaborazione:

- Da Process Designer Express, fare clic su File —> Compila.
- Selezionare un nome di maschera nella vista ad albero della maschera e fare clic con il tasto destro del mouse per richiamare il menu di scelta rapida, e fare clic su Compila maschera.
- Utilizzare il tasto di accesso rapido `Ctrl+F7`.

Se la finestra Output compilazione non è già aperta Process Designer Express la apre in fondo alla finestra principale per visualizzare i messaggi di compilazione.

Se durante la compilazione si verificano degli errori, eseguire le seguenti operazioni:

1. Tracciare l'errore facendo doppio clic sul messaggio di errore nella finestra di output.

Viene visualizzato il diagramma di attività il cui codice ha generato l'errore di compilazione, con il nodo che ha l'errore selezionato.

2. Correggere il problema e ricompilare. Ripetere il processo finché non si riceve il messaggio:

   ```
   Generatore di codice: generazione del codice riuscita.
   ```

Compilazione di più maschere di collaborazione

Il menu File di Process Designer Express include un'opzione di menu Compila tutto che abilita la compilazione di tutte le maschere di collaborazione (o un gruppo secondario) contenute nel progetto utente Integration Component Library. Per compilare più maschere eseguire i seguenti passi:

1. Se in Process Designer Express è aperta una maschera, chiuderla ora.
2. Fare clic su File —> Compila tutto. Viene aperta la finestra di dialogo Compila tutte le maschere. Visualizza una griglia di tutte le maschere del progetto utente. Per impostazione predefinita, tutte le maschere sono selezionate per la compilazione.
3. Annullare la selezione delle caselle di controllo accanto alle maschere che non si desidera compilare.
4. Fare clic su Continua.
5. Alla richiesta di conferma della compilazione, fare clic su Sì.

Conversione delle maschere

Process Designer Express fornisce la seguente funzione di conversione:

- Importa — Process Designer Express può importare i file BPEL (Business Process Execution Language) e UML (Unified Modeling Language) in XMI 1.1 da utilizzare come maschere di collaborazione. Consultare "Importazione di file".
- Esporta — Process Designer Express può esportare la maschera di collaborazione in formato BPEL o UML (in XMI 1.1). Consultare "Esportazione di una maschera di collaborazione" a pagina 107.
- Aggiorna — Process Designer Express può convertire vecchie maschere di collaborazione da utilizzare con la versione corrente di InterChange Server Express. Consultare "Aggiornamento di maschere di rilasci precedenti" a pagina 107.

Importazione di file

Process Designer Express può importare file BPEL e UML (in XMI 1.1) da utilizzare in una maschera di collaborazione. Utilizzare le informazioni contenute in quei file per creare una nuova definizione di maschera.

Eseguire le seguenti attività per creare una nuova maschera di collaborazione basata su file BPEL o UML (in XMI 1.1) esistenti:

1. Verificare che Process Designer Express sia aperto.
2. Fare clic su File —> Importa. Si apre il programma di importazione di Process Designer Express.
3. Selezionare il tipo di file che si desidera importare e fare clic su Avanti.
4. Selezionare la posizione del(i) file origine BPEL o UML.
5. Selezionare il(i) file che si desidera importare.

Nota: Se si prevedere di utilizzare i file BPEL è necessario importare tutti e tre i file .bpel, .wsdl e .bpelGUI.xml. Utilizzare il tasto Ctrl per selezionare tutti e tre i file per l'importazione.

6. Fare clic su Avanti per iniziare il processo di importazione. Al termine dell'importazione, viene visualizzata la finestra di dialogo Nuova maschera.
7. Selezionare nel campo Progetto il nome del progetto utente al quale appartiene la maschera.
8. Immettere nel campo Nome maschera il nome della maschera che si sta creando. Un nome di maschera può includere i caratteri alfabetici, i numeri ed i caratteri di sottolineatura.
9. Fare clic su OK. Process Designer Express crea la nuova maschera di collaborazione e la compila con tutte le informazioni contenute nei file BPEL o UML di origine.

Esportazione di una maschera di collaborazione

E' possibile esportare la maschera di collaborazione in formato BPEL o UML (in XMI 1.1) da utilizzare nelle altre applicazioni. Quando una maschera di collaborazione di InterChange Server Express viene esportata in formato BPEL, vengono creati i seguenti file:

- *.bpel — Questo file contiene le principali informazioni della maschera.
- *.wsdl — Questo file contiene le informazioni sull'interfaccia esterna.
- *.bpelGUI.xml — Questo file contiene le informazioni sulla rappresentazione grafica dei diagrammi di attività. Viene utilizzato in situazioni in cui i file BPEL vengono importati di nuovo in InterChange Server Express.

Quando una maschera di collaborazione viene esportata in UML (in XMI 1.1), viene creato un file *.xmi.

Per esportare una maschera di collaborazione di InterChange Server Express, eseguire i passi riportati di seguito:

1. Verificare che Process Designer Express sia aperto e che la maschera di collaborazione sia stata salvata e che sia stata compilata senza errori.
2. Fare clic su File —> Esporta. Si apre il programma di esportazione di Process Designer Express.
3. Selezionare il formato in cui si desidera esportare la maschera e fare clic su Avanti.
4. Selezionare la posizione in cui si desidera salvare il file o i file della maschera esportata.
5. Nel campo Nome file, specificare il nome del file della maschera esportata. Se si sta esportando in BPEL, *non* specificare un'estensione file nel campo Nome file.
6. Fare clic su Avanti per iniziare il processo di esportazione. La finestra di dialogo del programma di esportazione di Process Designer Express mostra l'avanzamento della conversione.
7. Al termine del processo di esportazione fare clic su Chiudi.

Aggiornamento di maschere di rilasci precedenti

Se si dispone di una maschera di collaborazione di un rilascio precedente, è possibile utilizzarla con l'ultima versione di InterChange Server Express. Il

processo di importazione e di aggiornamento di queste maschere varia a seconda che la vecchia maschera sia memorizzata in InterChange Server Express o in un file system locale.

Se la maschera esistente è memorizzata in InterChange Server Express, utilizzare System Manager per importarla in un progetto nel seguente modo:

1. Verificare che System Manager sia aperto e collegato a InterChange Server Express, dove è memorizzata la vecchia maschera.
2. Individuare in Integration Component Library la cartella progetto appropriata e fare clic con il tasto destro del mouse.
3. Dal menu di scelta rapida della cartella del progetto, fare clic su Importa componenti dal server —> Maschere di collaborazione. Viene visualizzata la finestra Importa componenti.
4. Selezionare la maschera di collaborazione che si desidera importare e fare clic su Fine.

La maschera viene automaticamente aggiornata per essere utilizzata con l'ultima versione di Integration Server Express. Dopo l'importazione e l'aggiornamento della maschera, è possibile utilizzare Process Designer Express per modificarla, se necessario.

Se si desidera utilizzare una maschera di collaborazione esistente che non è stata memorizzata in InterChange Server Express, è necessario aprirla in Process Designer Express con il comando File —> Apri —> Da file. Se la maschera non è compatibile con l'ultima versione di InterChange Server Express, Process Designer Express chiede di aggiornarla. Confermare l'aggiornamento; Process Designer Express converte la maschera esistente in un formato utilizzabile.

Al termine della conversione, è buona norma esaminare visivamente il nuovo diagramma di attività per verificarne la correttezza. Una volta terminate le modifiche alla maschera, salvarla per conservare le modifiche.

Considerazioni sulla modifica delle maschere

Le collaborazioni possono richiedere delle modifiche nel tempo, in base alla crescita e ai cambiamenti del business. Alcune delle modifiche saranno per il codice Java e altre per la definizione della maschera stessa.

E' possibile modificare il codice Java che supporta una collaborazione senza influenzare negativamente le collaborazioni esistenti; esse utilizzeranno il codice modificato alla successiva creazione di istanza della collaborazione.

Se si modifica la definizione di una maschera, è **necessario** creare di nuovo gli eventuali oggetti della collaborazione basati sulla maschera. Le modifiche che richiedono l'oggetto collaborazione includono (ma non sono limitate a):

- Definizione di nuovi attributi
- Rimozione di attributi esistenti
- Rimozione di porte
- Aggiunta di porte
- Modifica di oggetti di attivazione
- Rimozione di oggetti di attivazione
- Aggiunta di oggetti di attivazione
- Aggiunta di oggetti business
- Rimozione di oggetti business

- Modifica di oggetti business
- Aggiunta di nuovi scenari
- Rimozione di scenari

Quando si crea di nuovo l'oggetto collaborazione, compilarlo e testarlo come descritto in "Test di una collaborazione" a pagina 112.

Considerazioni particolari per le collaborazioni che utilizzano le proprietà SEND_EMAIL e INFORMATIONAL_EXCEPTIONS

Molte maschere di collaborazione basate sulla maschera CollaborationFoundation o BaseCollaboration contengono le proprietà di configurazione SEND_EMAIL e INFORMATIONAL_EXCEPTIONS (di solito nei nodi azione nello scenario principale). Queste proprietà determinano come la collaborazione gestisce le eccezioni; i valori possono includere dei numeri specifici di messaggi di eccezione.

Nei rilasci precedenti, il codice riportato di seguito era stato utilizzato per ottenere i messaggi di eccezione ed analizzarli per il numero di messaggio:

```
//Get exception message
sMessage=currentException.getMessage();

//Get exception type
sExceptionType=currentException.getType();

//Determine whether to send email. If SEND_EMAIL is set to none,
//then never send email. If SEND_EMAIL is set to all, then always
//send email. Else, make sure the message number is included in
//the comma-delimited list of SEND_EMAIL values.

StringTokenizer emailtoken=new StringTokenizer(sSendEmail, ",");
bBranch=false;

if (sSendEmail.trim().toLowerCase().equals("all"))
   {
     bBranch=true;
   }
else while (emailtoken.hasMoreTokens())
   {
     sToken=emailtoken.nextToken();
     sToken=sToken.trim().toLowerCase();
     sToken="Error"+sToken;
     if (sMessage.lastIndexOf(sToken)>-1)
       {
         bBranch=true;
         break;
       }
   }
```

Il metodo getMsgNumber() è stato aggiunto alla classe CollaborationExceptions; restituisce i numeri dei messaggi di eccezione che a loro volta vengono utilizzati dalle proprietà SEND_EMAIL e INFORMATIONAL_EXCEPTIONS nello stabilire come viene gestita l'eccezione. Come risultato, può essere necessario aggiornare manualmente il codice in una maschera di collaborazione esistente, per poter beneficiare del metodo getMsgNumber().

Stabilire se la maschera che si sta convertendo contiene le proprietà di configurazione SEND_EMAIL o INFORMATIONAL_EXCEPTIONS. Se la maschera contiene una o entrambe le proprietà, effettuare le seguenti operazioni:

1. Valutare ogni scenario che contiene SEND_EMAIL o INFORMATIONAL_EXCEPTIONS per identificare eventuale codice personalizzato (codice diverso da quello fornito nelle maschere di

collaborazione incluse nel prodotto) nei nodi azione. Se lo scenario *non* contiene codice personalizzato, continuare con il passo 2.

Se lo scenario contiene del codice personalizzato, per valutare una stringa di messaggio di eccezione di un numero di messaggio, rimuovere quel codice e continuare con il passo 2.

2. Aggiungere il seguente codice ad ogni nodo azione che restituisce i messaggi di eccezione ed esegue un'analisi per i numeri di messaggi di eccezione:

```
//Get Exception Number
iMessageNum=currentException.getMsgNumber();
sMessageNum=String.valueOf(iMessageNum);
```

3. Nello stesso nodo azione, sostituire il loop while esistente che restituisce il numero messaggio cercando la stringa `Error` con il nuovo codice, nel seguente modo:

Codice esistente da sostituire

```
else while (emailtoken.hasMoreTokens())
   {
     sToken=emailtoken.nextToken();
     sToken=sToken.trim().toLowerCase();
     sToken="Error"+sToken;
     if (sMessage.lastIndexOf(sToken)>-1)
       {
         bBranch=true;
         break;
       }
   }
```

Nuovo codice

```
else while (emailtoken.hasMoreTokens())
   {
     sToken=emailtoken.nextToken();
     sToken=sToken.trim().toLowerCase();

     if (sToken.equals(sMessageNum))
       {
         bBranch=true;
         break;
       }
   }
```

4. Se il nodo azione utilizza SEND_EMAIL per determinare la gestione dell'eccezione di collaborazione, aggiungere il codice riportato di seguito per supportare la possibilità di specificare un intervallo di numeri messaggi di eccezione. Questo codice è nidificato all'interno di un loop while che è stato modificato nel Passo 3; l'esempio riportato di seguito mostra l'intero loop while in modo da poter individuare il punto di inserimento del nuovo codice.

```
else while (emailtoken.hasMoreTokens())
   {
     sToken = emailtoken.nextToken();
     sToken = sToken.trim().toLowerCase();

     trace (3,"***** Current sToken value is: "+sToken);

     String sSeparator="-";

     if (sToken.lastIndexOf(sSeparator)>-1)
       {
         int pos = sToken.indexOf(sSeparator);

         sLowerRange = sToken.subtring(0,pos);
         sLowerRange = sLowerRange.trim();
         iTemp = new Integer(sLowerRange);
         iLowerRange = (((Integer)iTemp).intValue());
```

```
            sUpperRange = sToken.substring(pos+1,sToken.length());
            sUpperRange = sUpperRange.trim();
            iTemp = new Integer(sUpperRange);
            iUpperRange = (((Integer)iTemp).intValue());

            trace(3," ***** iLowerRange value is -" +iLowerRange
                  +" *****iUpperRange value is -"
                  +iUpperRange +" *****");

            if ((iMessageNum>=iLowerRange) && (iMessageNum<=iUpperRange))
              {
                bBranch = true;
                trace(3," ***** The current exception falls within
                      this range *****");
                break;
              }
        }
        else
          {
           if (sToken.equals(sMessageNum))
             {
               bBranch=true;
               break;
             }
          }
    }
```

5. Per ogni scenario che contiene le proprietà SEND_EMAIL o INFORMATION_EXCEPTIONS, aggiungere le seguenti variabili di scenario nella finestra di dialogo Definizioni scenario. Per ulteriori informazioni sull'aggiunta di variabili di scenario, consultare "Definizione delle variabili di scenario" a pagina 102.

Tabella 33. Variabili di scenario da aggiungere durante la conversione della maschera

Tipo di variabile	Nome variabile	Valore iniziale
String	sMessageNum	null
String	sLowerRange	null
String	sUpperRange	null
int	iMessageNum	0
int	iLowerRange	0
int	iUpperRange	0

6. Ricompilare la maschera e testarla per assicurarsi che le modifiche siano corrette. Per ulteriori informazioni su queste attività consultare "Compilazione di una maschera di collaborazione" a pagina 105 e "Test di una collaborazione" a pagina 112.

Eliminazione di una maschera di collaborazione

Importante

Non cancellare una maschera di collaborazione che contiene degli oggetti collaborazione associati, a meno che non si prevede di eliminare anche quelli. L'eliminazione di una maschera di collaborazione rende inutilizzabili tutti gli oggetti creati da quella maschera. Per istruzioni sull'eliminazione di oggetti collaborazione e quindi sull'eliminazione della maschera di collaborazione in System Manager, consultare il manuale *Implementation Guide for InterChange Server Express*.

Utilizzare Process Designer Express per eliminare una maschera di collaborazione che non ha oggetti collaborazione creati da essa. Per eliminare una maschera, effettuare le seguenti operazioni:

1. Aprire Process Designer Express e verificare che System Manager sia in esecuzione.
2. Fare clic su File —> Elimina. Process Designer Express visualizza la finestra Elimina maschera dal progetto '*NomeProgetto*'.
3. Dall'elenco a discesa Progetto, selezionare il nome del progetto che contiene la maschera che si desidera eliminare.
4. Dall'elenco di maschere di collaborazione selezionare il nome della maschera che si desidera eliminare e fare clic su OK.
5. Il programma chiede di confermare la cancellazione. Fare clic su Sì.

Test di una collaborazione

Dopo aver creato e compilato con esito positivo una maschera di collaborazione, è possibile testarne la progettazione. Per verificare che la collaborazione funzioni come previsto è necessario creare un oggetto collaborazione ed utilizzare lo strumento Test Connector per testarne la funzionalità.

Test Connector, che fa parte dell'ambiente di test di InterChange Server, simula un connettore reale. Utilizzare Test Connector per inviare degli eventi e delle risposte alle collaborazioni. Consente di configurare gli oggetti business e gli eventi di attivazione che testano la funzionalità di una collaborazione.

Se la collaborazione che si sta testando dispone di una porta per un connettore, aprire un'istanza di Test Connector. Se la collaborazione utilizza una porta in ingresso da un connettore ed un'altra porta verso un connettore diverso, aprire due istanze Test Connector, una per ogni connettore.

Dai menu di Test Connector, designare il file di configurazione e la definizione del connettore da emulare. Configurare i valori per gli oggetti business selezionati, quindi inviare e ricevere l'oggetto business.

Per informazioni dettagliate sull'utilizzo di Integrated Testing Environment e di Test Connector, consultare il manuale *System Implementation Guide*.

Debug di una collaborazione

Lo strumento Collaboration Debugger consente di esaminare tutta la logica business in una maschera di collaborazione per verificare che funzioni come previsto. Collaboration Debugger fornisce la possibilità di impostare più punti di interruzione in uno scenario; il processo di debug può interrompersi in qualsiasi punto di interruzione per consentire la raccolta di dati rilevanti sull'evento di attivazione ed il suo flusso attraverso lo scenario.

Collaboration Debugger è disponibile in System Manager (come prospettiva) e tramite Integrated Test Environment. Per ulteriori informazioni sull'utilizzo di Collaboration Debugger, consultare il manuale *System Implementation Guide*.

Capitolo 5. Utilizzo dei diagrammi di attività

Questo capitolo descrive come utilizzare Process Designer Express per modificare un diagramma di attività. Un *diagramma di attività* definisce il flusso di controllo di una parte particolare di una collaborazione; viene creato automaticamente quando si crea uno scenario. Il diagramma è una serie di passi che vengono eseguiti secondo uno specifico ordine. Un diagramma di attività contiene i simboli che specificano questi passi, il loro ordine e la logica che ne determina l'esecuzione.

Per informazioni sul layout e la visualizzazione dell'area di lavoro, consultare Capitolo 8, "Opzioni dell'area di lavoro e del layout", a pagina 197.

Per modificare il diagramma di attività di uno scenario, eseguire le seguenti attività:

1. Visualizzare l'editor di diagrammi nell'Area di lavoro.

   E' possibile richiamare l'editor di diagrammi in uno dei seguenti modi:

   - Selezionare un nome scenario o un nome diagramma nella vista ad albero della maschera e scegliere l'opzione Apri diagramma dal menu a discesa Maschera.
   - Selezionare un nome scenario o un nome diagramma nella vista ad albero della maschera e fare clic con il tasto destro del mouse per richiamare il menu di scelta rapida. Dal menu di scelta rapida scegliere Apri diagramma.
   - Fare doppio clic sul nome scenario.

   Process Designer Express visualizza l'editor di diagrammi con il diagramma di attività relativo allo scenario selezionato.
2. Modificare il diagramma attività nell'editor di attività, utilizzando i simboli forniti nella barra strumenti Simboli.
3. Salvare il diagramma di attività con l'opzione Salva del menu File oppure utilizzare la combinazione di tasti di accesso rapido `Ctrl+S`.

Utilizzo delle funzioni dell'editor di diagrammi

E' possibile richiamare le funzioni dell'editor di diagrammi in uno dei seguenti modi:

- tramite le barre degli strumenti
- con spostamenti del mouse sui simboli contenuti nel diagramma di attività

Accesso alle funzioni dell'editor di diagrammi: i menu di Process Designer Express

Process Designer Express dispone di menu a discesa dai quali vengono avviate molte delle operazioni relative al diagramma. Per alcune di queste funzioni sono disponibili dei tasti di accesso rapido e dei menu di scelta rapida. Per informazioni dettagliate sui menu, consultare "Menu di Process Designer Express" a pagina 20.

Accesso alle funzioni dell'editor di diagrammi: spostamenti del mouse

L'editor di diagrammi riconosce i seguenti spostamenti del mouse:

- Fare clic con il tasto sinistro del mouse per selezionare un componente o un simbolo di un diagramma di attività.

  L'editor di diagrammi circonda il simbolo con dei riquadrati grigi di ancoraggio, denominati *margine di selezione*, per indicare che il simbolo è selezionato. Per deselezionare il simbolo, fare clic altrove nell'area di lavoro.
- Fare clic con il tasto destro del mouse per selezionare e richiamare il menu di scelta rapida relativo al simbolo.

  Il menu di scelta rapida contiene le seguenti opzioni:
  - Proprietà — Visualizza l'appropriata finestra di dialogo Proprietà del simbolo relativa al simbolo selezionato. E' uguale all'opzione Proprietà del menu Modifica. Per ulteriori informazioni, vedere la descrizione dell'opzione Proprietà in "Funzioni del menu Modifica" a pagina 21.
  - Carattere — Controlla il carattere in cui viene visualizzato il testo del simbolo. E' uguale all'opzione Carattere del menu modifica. Per ulteriori informazioni, vedere la descrizione dell'opzione Carattere in "Funzioni del menu Modifica" a pagina 21.
- Fare clic con il tasto destro del mouse in qualunque punto nel diagramma di attività (ma non su un simbolo specifico) per andare al diagramma principale del diagramma di attività.

Simboli del diagramma di attività

Un diagramma di attività utilizza dei simboli per rappresentare le fasi di esecuzione. Questa sezione fornisce le seguenti informazioni sui simboli di un diagramma di attività:

- Tipi di simboli esistenti
- Confronto dei simboli con quelli contenuti in un diagramma di flusso
- Proprietà di ciascun simbolo

Introduzione ai simboli

La Figura 36 a pagina 115 mostra i simboli di un diagramma di attività ed i relativi pulsanti associati contenuti nella barra degli strumenti Simboli. Questa barra degli strumenti diviene attiva quando l'editor di diagrammi viene visualizzato nell'area di lavoro.

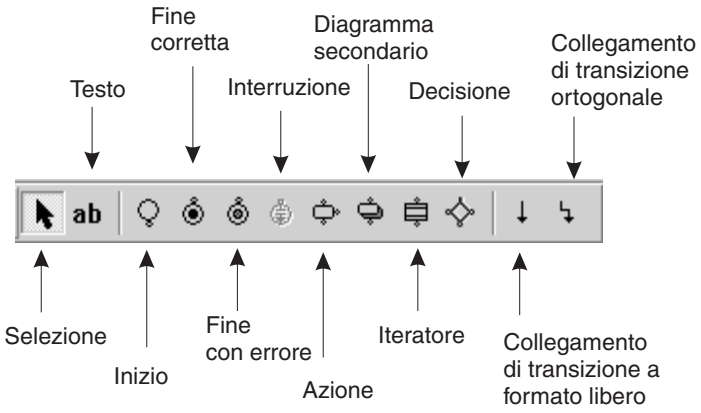


Figura 36. Barra degli strumenti Simboli

I diagrammi di attività contengono tre tipi principali di simboli: nodi, collegamenti di transizioni e chiamate di servizio. Contengono, inoltre, i simboli di inizio e di fine.

Simboli di inizio e di fine

Quando viene creato un diagramma di attività, un simbolo di inizio viene automaticamente viene collocato nel diagramma. Questo simbolo rappresenta l'inizio del flusso; ogni diagramma di attività deve contenere un simbolo di inizio.

Il simbolo Inizio può essere utilizzato per inizializzare un attributo di correlazione. Per ulteriori informazioni, consultare "Utilizzo di un attributo di correlazione" a pagina 141.

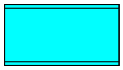
Process Designer Express fornisce due simboli di fine per i diagrammi di attività: Fine corretta e Fine con errore. Ogni percorso di esecuzione in un diagramma di attività deve terminare con uno di questi simboli (ad eccezione di un diagramma di attività iteratore, che termina con un simbolo di interruzione). Per ulteriori informazioni sull'utilizzo dei simboli di fine, consultare "Termine del percorso di esecuzione" a pagina 154.

Simboli Nodo

Un *nodo* è un simbolo che rappresenta una fase in una collaborazione. Esistono quattro tipi di nodi: *azioni*, *decisioni*, *diagrammi secondari*, e *iteratori*. Ciascun nodo viene rappresentato da un unico simbolo nella barra degli strumenti Simboli (vedere la Figura 36).

La Tabella 34 a pagina 116 mostra il simbolo posizionato nel diagramma di attività per ciascun tipo di nodo.

Tabella 34. Simboli dei nodi

Tipo di nodo	Simbolo nel diagramma di attività	Per ulteriori informazioni
Azione		"Nodi azione" a pagina 117
Decisione		"Nodi decisione" a pagina 129
Diagramma secondario		"Diagrammi secondari" a pagina 144
Iteratore		"Iteratori" a pagina 149

Simboli Collegamento di transizione

Un *collegamento di transizione* rappresenta il flusso di controllo fra i nodi. Poiché il flusso di un diagramma va dall'alto verso il basso, un collegamento di transizione è orientato sempre verticalmente. Se da un nodo sono disponibili più percorsi, il collegamento di transizione deve essere utilizzato insieme ad un nodo decisione. La logica del nodo decisione determina qual percorso viene preso.

L'editor di diagrammi può rappresentare un collegamento di transizione in uno di due modi: collegamenti in formato libero e collegamenti ortogonali. La Tabella 35 mostra il simbolo del diagramma di attività che rappresenta ciascun tipo di collegamento di transizione.

Tabella 35. Simboli Collegamento di transizione

Tipo di collegamento di transizione	Simbolo nel diagramma di attività	Per ulteriori informazioni
Collegamento di transizione in formato libero		"Collegamenti di transizione" a pagina 121
Collegamento di transizione ortogonale		"Collegamenti di transizione" a pagina 121

Simbolo Chiamata di servizio

Una *chiamata di servizio* rappresenta una richiesta o una risposta nei confronti di un'entità esterna, attraverso una porta. E' sempre orientata orizzontalmente. Una chiamata di servizio è collegata ad un nodo azione. Per impostazione predefinita, l'etichetta di una chiamata di servizio ne descrive il tipo. La chiamata di servizio può essere una delle seguenti:

- Chiamata di servizio sincrona
- Chiamata di servizio in uscita asincrona
- Chiamata di servizio in ingresso asincrona

La barra degli strumenti Simboli non contiene un simbolo per le chiamate di servizio. La funzionalità della chiamata di servizio è disponibile tramite il menu di scelta rapida che viene visualizzato quando si fa clic con il tasto destro del mouse su un nodo azione.

Per ulteriori informazioni sui tipi di chiamate di servizio e su come includerle in un diagramma di attività, consultare "Chiamate di servizio" a pagina 134.

Proprietà dei simboli del diagramma

Un simbolo di un diagramma di attività può avere le proprietà visualizzate nella Tabella 36.

Tabella 36. Proprietà di un simbolo

Proprietà del simbolo	Descrizione
Un identificativo univoco (UID)	Ogni simbolo di un diagramma di attività ha un identificativo univoco (UID). E' possibile scegliere se visualizzare o meno l'UID nel diagramma. Anche se è possibile assegnare delle proprie etichette ai simboli, l'etichetta non sostituisce l'UID. L'UID identifica il simbolo nei messaggi di compilazione e di traccia. E' possibile scegliere se visualizzare i UID con l'opzione Visualizza UID del menu a discesa Visualizza.
Un'etichetta facoltativa	L'etichetta serve da nome descrittivo che rende il diagramma di attività più leggibile (quando le etichette sono visualizzate). E' possibile scegliere se visualizzare le etichette con l'opzione Visualizza etichette del menu a discesa Visualizza.
Una descrizione facoltativa	La descrizione è un commento.
Proprietà specifiche per tipo	Alcuni simboli, come i nodi azione, hanno un frammento di codice associato.

E' possibile modificare le proprietà della maggior parte dei simboli. Richiamare la finestra di dialogo Proprietà del simbolo in uno dei seguenti modi:

- Fare clic con il tasto destro del mouse su un simbolo di un diagramma di attività per richiamare il menu di scelta rapida, dal quale selezionare Proprietà.
- Selezionare un simbolo di un diagramma di attività, quindi selezionare Proprietà dal menu a discesa Modifica.
- fare doppio clic su un simbolo di un diagramma di attività per aprire la relativa finestra Proprietà del simbolo.
- Utilizzare la combinazione di tasti di accesso rapido `Ctrl + Invio`.

Nodi azione

Un nodo azione (spesso denominato semplicemente *azione*) rappresenta una fase in una collaborazione. E' il principale fondamento di costruzione della logica della collaborazione. La scomposizione della logica di collaborazione in nodi azione dipende dall'utente. E' possibile scrivere molte righe di codice complesso in una singola azione o suddividere la logica in numerose azioni individuali. La suddivisione di una logica di collaborazione in nodi azione è simile allo sviluppo del codice di programma. E' possibile scrivere un programma con una breve routine principale che richiama una serie di routine secondarie o chiamate di metodi per portare a compimento la funzione del programma. Oppure è possibile scrivere una routine principale più lunga che include in linea tutta la logica del programma.

Aggiunta di un'azione ad un diagramma

Per aggiungere un nodo azione ad un diagramma di attività:

1. Nella barra degli strumenti Simboli, fare clic sul pulsante Azione.
2. Fare clic nell'area di lavoro per posizionare il simbolo Azione.

Nota: Un nodo azione può effettuare una chiamata di servizio se si collega all'azione il simbolo Chiamata di servizio. Per informazioni sulle chiamate di servizio, consultare "Chiamate di servizio" a pagina 134.

Definizione delle proprietà del nodo azione

Dopo che il nodo azione viene visualizzato nel diagramma di attività, utilizzare la finestra Proprietà azione per definire una qualsiasi delle seguenti proprietà per il nodo:

- Etichetta — Fornisce un'etichetta per il nodo azione. L'utilizzo di testo descrittivo invece dell'UID predefinito rende il diagramma più facile da leggere e da utilizzare. Questa proprietà è facoltativa.
- Descrizione — Fornisce una descrizione dello scopo del nodo azione. Questa proprietà è facoltativa.
- Frammento di codice — Definisce come agisce il nodo azione. Per ulteriori informazioni, consultare "Aggiunta di definizioni di attività ad un nodo azione" a pagina 119.

Aprire la finestra di dialogo Proprietà dell'azione effettuando una delle seguenti operazioni:

- Fare doppio clic sul nodo azione selezionato.
- Fare clic con il tasto destro del mouse sul nodo azione per richiamare il menu di scelta rapida e scegliere Proprietà.
- Selezionare il nodo azione e fare clic su Proprietà dal menu a discesa Modifica.
- Selezionare il nodo azione ed utilizzare i tasti di accesso rapido `Ctrl + Invio`.

Viene visualizzata la finestra di dialogo Proprietà dell'azione con il nome del nodo azione nella parte superiore della finestra. Il nome ha il seguente formato:

`Action_`*`UID`*

dove *UID* specifica l'identificativo univoco del nodo azione. La Figura 37 mostra la finestra di dialogo Proprietà dell'azione.

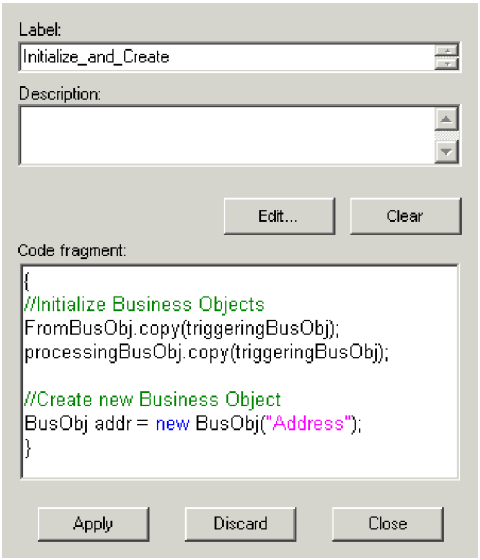


Figura 37. Finestra di dialogo Proprietà dell'azione

Aggiunta di definizioni di attività ad un nodo azione

I nodi azione contengono delle definizioni di attività. Le definizioni di attività (dette anche *frammenti di codice*) consistono in chiamate all'API della collaborazione o ad altro codice Java, e possono contenere operazioni quali:

- Richiamo ed impostazione di valori attributo negli oggetti business
- Verifica dell'istruzione di un evento in ingresso
- Confronto dei valori attributo con le costanti i con altri valori attributo
- Configurazione delle variabili dell'oggetto business da utilizzare nelle chiamate di servizio
- Registrazione dei messaggi

Le definizioni di attività possono contenere qualsiasi costrutto del linguaggio di programmazione Java.

Le definizioni di attività possono essere aggiunte ai nodi azione in due modi:

- Inserendo il codice Java direttamente nella finestra Frammento di codice della finestra di dialogo Proprietà dell'azione. Consultare "Aggiunta di una definizione di azione nella finestra Proprietà dell'azione".
- Utilizzando Activity Editor per aggiungere dei blocchi funzione grafici oppure del codice Java tradizionale. Consultare "Aggiunta di una definizione di attività in Activity Editor".

Aggiunta di una definizione di azione nella finestra Proprietà dell'azione

Per aggiungere del codice Java direttamente nella finestra Frammento di codice nella finestra di dialogo Proprietà dell'azione, effettuare le seguenti operazioni:

1. Abilitare la modifica diretta della finestra Frammento di codice come segue:
   a. In Process Designer Express, fare clic su Visualizza > Preferenze. Viene visualizzata la finestra Preferenze utente.
   b. Nella scheda Diagramma selezionare le opzioni Abilita modifica in locale per nuovi nodi azione e Abilita modifica in locale per nodi azione esistenti.
   c. Fare clic su Applica e su OK.
2. Fare clic con il tasto destro del mouse sul nodo azione per visualizzare il relativo menu di scelta rapida.
3. Dal menu di scelta rapida, fare clic su Proprietà. Viene visualizzata la finestra di dialogo Proprietà dell'azione.
4. Immettere il codice Java nella finestra Frammento di codice.
5. Fare clic su Applica per salvare le modifiche.

Aggiunta di una definizione di attività in Activity Editor

Per aggiungere una definizione di attività utilizzando Activity Editor, effettuare le seguenti operazioni:

1. Fare clic con il tasto destro del mouse sul nodo azione per visualizzare il relativo menu di scelta rapida.
2. Dal menu di scelta rapida, fare clic su Proprietà. Viene visualizzata la finestra di dialogo Proprietà dell'azione.
3. Fare clic su Modifica per aprire Activity Editor. Per impostazione predefinita, Activity Editor presenta la vista Grafica.

Nota: Se per questo nodo è già stata creata una definizione di attività e se la definizione contiene del codice Java personalizzato che non viene riconosciuto dalla vista Grafica, Activity Editor visualizza la vista Java.

4. Aggiungere la definizione di attività. Consultare "Definizioni delle attività" a pagina 165 per ulteriori informazioni.
5. Chiudere Activity Editor. La finestra Proprietà dell'azione è ancora aperta ed ora visualizza il frammento di codice associato alla definizione di attività.
6. Fare clic su Applica per salvare le modifiche.

Utilizzo degli operatori Java relazionali delle definizioni di attività

E' possibile utilizzare gli operatori Java nelle definizioni di attività, specialmente gli operatori relazionali e condizionali, oltre agli operatori aritmetici.

Tutte le classi Java ereditano dalla classe di base `Object`. In questo modo ogni classe dispone di un metodo `equals()`. La sua firma è la seguente:

```
public boolean equals(Object obj);
```

Il metodo `equals()` si occupa dell'uguaglianza del valore. Confronta l'oggetto richiamando il metodo e l'oggetto di riferimento `obj` per stabilirne l'uguaglianza, restituendo true se hanno lo stesso valore e false nel caso contrario. Notare che è diverso dagli operatori di uguaglianza `==` e `!=`. Questi ultimi stabiliscono se due riferimenti fanno riferimento allo stesso oggetto, a prescindere dal valore dell'oggetto. Questa è una distinzione importante.

La Tabella 37 riassume gli operatori di uguaglianza disponibili. Ogni operatore di uguaglianza o relazionale genera un risultato con valore booleano.

Tabella 37. Operatori Java relazionali

Operatore relazionale	Significato
>	maggiore di
>=	maggiore di o uguale a
<	minore di
<=	minore di o uguale a
==	uguale a
!=	non uguale a
!	operatore unario, che inverte un valore booleano.

I risultati delle espressioni booleane possono essere uniti ai simboli condizionali `AND` (`&&`) e `OR` (`||`).

La Tabella 38 riassume gli operatori aritmetico, inclusi gli operatori di incremento e decremento.

Tabella 38. Operatori Java aritmetici e di assegnazione

Operatore aritmetico	Significato
+	addizione
-	sottrazione
*	moltiplicazione
/	divisione
%	resto
-	unario, per la negazione o l'inversione del segno di un numero
++	operatore d'incremento

Tabella 38. Operatori Java aritmetici e di assegnazione (Continua)

Operatore aritmetico	Significato
--	operatore di decremento
=	operatore di assegnazione

Utilizzo dei servizi Web nelle definizioni di attività

Se si desidera includere un servizio Web in una definizione di attività di una maschera di collaborazione è necessario prima esportare il servizio Web dal progetto ICL (Integration Component Library) in System Manager. Ogni metodo contenuto nel servizio Web viene esportato come blocco funzione in Activity Editor, dove può essere inserito nella definizione di attività. Per ulteriori informazioni, consultare "Blocchi funzione dei servizi web" a pagina 166.

Ulteriori informazioni sulle definizioni di attività

Per ulteriori informazioni sulla logica di codifica di un nodo azione, fare riferimento alla seguente tabella:

Tabella 39. Reperimento di ulteriori informazioni sui frammenti di codice

Ulteriori risorse	Ubicazione
Descrizione dettagliata di Activity Editor, inclusi gli esempi sull'utilizzo dei blocchi funzione	Capitolo 6, "Utilizzo di Activity Editor", a pagina 159
Esempi di frammenti di codice	Capitolo 9, "Esempi e suggerimenti di codifica", a pagina 207.
Pagine di riferimento per i singoli blocchi funzione	Dal Capitolo 11 al Capitolo 21
Pagine di riferimento per i singoli metodi nell'API della collaborazione	Dal Capitolo 22 al Capitolo 29.

Collegamenti di transizione

I collegamenti di transizione rappresentano il flusso di controllo del diagramma di attività. Essi collegano i nodi in cui si verificano le attività, come azioni, decisioni, diagrammi secondari e iteratori, e collegano questi nodi a dei simboli di inizio e di fine. I collegamenti di transizione possono contenere delle analisi degli oggetti business che controllano i calori delle istanze oggetto business.

Nota: Nei diagrammi di attività i collegamenti di transizione non rappresentano il flusso di dati. I dati passano di nodo in nodo quando un nodo imposta una variabile ed un altro accede alla variabile. Il meccanismo del flusso di dati del diagramma di attività è paragonabile a quello di una classe o di un programma, in cui il codice imposta una variabile che un altro codice utilizza. Questo modello si differisce da quello utilizzato dagli strumenti di creazione di modelli di passaggio eventi, che mostra i dati che si spostano lungo i collegamenti.

Process Designer Express fornisce sia i collegamenti ortogonali che a formato libero. Quando possibile, utilizzare i collegamenti ortogonali. Utilizzare invece i collegamenti in formato libero quando non si riesce ad ottenere la forma desiderata dai collegamenti ortogonali. Fare clic con il tasto destro del mouse per visualizzare il menu di scelta rapida che mostra l'ortogonalità del collegamento. Utilizzare questo menu per passare dal collegamento ortogonale e quello in formato libero e viceversa.

Quanti collegamenti possono coesistere?

La Tabella 40 visualizza il numero di collegamenti in ingresso ed in uscita che i diversi tipi di nodi possono avere.

Tabella 40. Collegamenti in ingresso e in uscita consentiti per tipo di nodo

Tipo di nodo	Collegamenti in ingresso	Collegamenti in uscita
Azione	Nessun limite	Uno
Decisione	Uno	Sette
Diagramma secondario o iteratore	Nessun limite	Uno

Creazione di un collegamento di transizione

Per creare un collegamento di transizione, i due simboli che si desidera collegare devono essere disponibili sull'area di lavoro.

Per aggiungere un collegamento di transizione ad un diagramma di attività:

1. Nella barra degli strumenti Simboli, fare clic sul pulsante Collegamento transizione.
2. Nell'area di lavoro, fare clic sul margine inferiore del simbolo dove si desidera che inizi il collegamento di transizione.
3. Fare clic sul margine superiore del simbolo in cui si desidera che termini il collegamento di transizione.

Process Designer Express consente di posizionare delle connessioni valide fra i simboli. Non consente di collegare due simboli per i quali un collegamento non è valido. Piuttosto che visualizzare un messaggio di errore, Process Designer Express non consente la creazione del collegamento di transizione non valido.

Process Designer Express indica se il tentativo di posizionare un collegamento di transizione su un simbolo è valido. Quando si posiziona il puntatore del mouse (con il collegamento) sul margine di un simbolo, il puntatore diviene un segno più incluso in un cerchio, se la connessione è valide. Sarà quindi possibile fare clic e posizionare la connessione sul simbolo. Se la connessione non è valida, il puntatore del mouse non diviene un segno più e non è possibile posizionare la connessione.

Annullamento di un collegamento

Interrompere o annullare un collegamento di transizione premendo il tasto Escape (ESC). Ogni volta che si preme il tasto ESC viene annullato l'ultimo segmento della linea di connessione. L'utilizzo del tasto ESC è l'unico modo per annullare un tentativo di connessione per il quale non esiste un simbolo valido nel diagramma di attività.

Ad esempio, si supponga di aver posizionato nel diagramma due simboli, uno di inizio e uno di fine. Selezionare un collegamento di transizione e fare clic sul simbolo di inizio. Viene visualizzato il segmento della linea del collegamento di transizione, collegato al simbolo di inizio. Tuttavia non vi sono simboli o porte validi a cui è possibile collegare una linea di collegamento di transizione. A questo punto, premere il tasto ESC è l'unico modo per annullare l'attività di connessione e continuare a modificare la sessione.

Definizione delle proprietà del collegamento di transizione

Una volta visualizzato il collegamento di transizione nel diagramma di attività, è possibile definirne le proprietà nella finestra di dialogo Proprietà del collegamento (vedere la Figura 38).

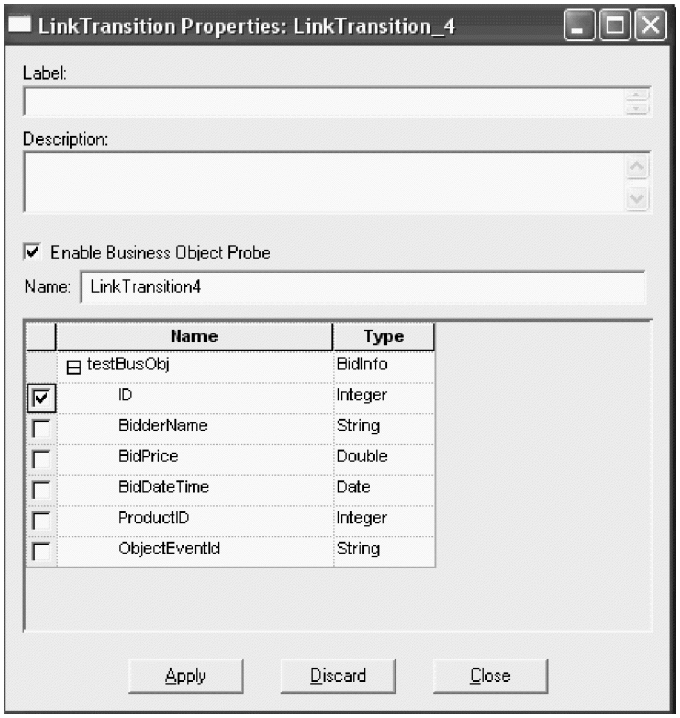


Figura 38. Finestra di dialogo Proprietà del collegamento di transizione

La finestra di dialogo Proprietà del collegamento identifica il collegamento di transizione con un nome nella parte superiore della finestra nel seguente formato:

`LinkTransition_`*`UID`*

dove *UID* è l'identificativo univoco per il collegamento.

Le proprietà di un collegamento di transizione includono l'etichetta, la definizione e se contiene un'analisi di oggetto business.

Per definire le proprietà di un collegamento di transizione:

1. Visualizzare la finestra di dialogo Proprietà del collegamento.

   E' possibile visualizzare questa finestra di dialogo in uno qualunque dei seguenti modi:

   - Fare doppio clic sul collegamento di transizione selezionato.
   - Fare clic con il tasto destro del mouse sul collegamento di transizione per richiamare il menu di scelta rapida e scegliere Proprietà.
   - Selezionare il collegamento di transizione e scegliere Proprietà dal menu a discesa Modifica.
   - Selezionare il collegamento di transizione ed utilizzare il tasto di accesso rapido `Ctrl + Invio`.

   Viene visualizzata la finestra di dialogo Proprietà del collegamento. La Figura 38 a pagina 123 mostra un esempio di finestra Proprietà del collegamento.

2. Facoltativamente, specificare l'etichetta e la descrizione per questo collegamento di transizione.

   Per ulteriori informazioni sulle etichette, consultare "Etichettatura di un collegamento" a pagina 124.
3. Fare clic su Applica per salvare le proprietà del collegamento.

Etichettatura di un collegamento

Le etichette del collegamento possono essere utili per garantire la leggibilità di un diagramma di attività. Tentare di cogliere la logica di decisione nello stesso modo che etichettando un nodo decisione in un diagramma del flusso. Le etichette del collegamento, col nome assegnato in modo logico, spiegano il flusso dello scenario. Ad esempio:

- Se due collegamenti di transizione si diramano in base al valore di una proprietà di configurazione denominata `CONVERT_VERB`, le etichette potrebbero essere `DoConvert` e `DoNotConvert`.
- Se un collegamento di transizione gestisce una chiamata di servizio riuscita ed un altro collegamento gestisce un'eccezione di chiamata di servizio, le etichette potrebbero essere `Success` e `ServiceCallException`.

Per etichettare un collegamento di transizione, inserire il testo dell'etichetta nella casella Etichetta della finestra Proprietà del collegamento. L'etichetta viene visualizzata nel diagramma di attività esattamente come appare nella casella di testo. Utilizzare il tasto di andata a capo nella casella di testo per suddividere l'etichetta su più righe in modo da non ricoprire gli altri collegamenti .

Utilizzo delle analisi di oggetti business

Un'analisi oggetto business controlla i valori dell'istanza oggetto business durante il runtime. L'analisi viene posizionata su un collegamento di transizione durante la creazione di un diagramma di attività e viene attivata o disattivata durante il runtime tramite la finestra di dialogo Proprietà della collaborazione di System Manager.

Per impostazione predefinita, un'analisi di oggetto business viene visualizzata sotto forma di un quadratino rosso sul collegamento di transizione in un diagramma di attività. Nella Figura 39 il collegamento Default branch contiene un'analisi dell'oggetto business.

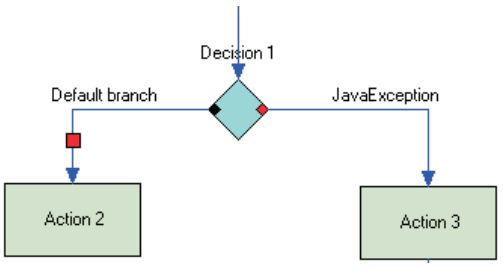


Figura 39. Collegamento di transizione con analisi oggetto business

E' possibile utilizzare un'analisi di oggetto business per controllare qualsiasi oggetto business specificato nella scheda Porte ed eventi di attivazione della finestra Definizioni maschera, con le seguenti eccezioni:

- Le analisi di oggetti business non possono essere utilizzate sui collegamenti di transizione in ingresso per un nodo decisione.

- Le analisi di oggetti business non possono essere utilizzate sui collegamenti di chiamata di servizio.

Per ogni oggetto business è possibile scegliere gli attributi specifici da controllare. Tutti i valori istanza di questi attributi vengono presentati in un prospetto fornito da System Monitor.

Aggiunta di un'analisi di oggetto business

Per aggiungere un'analisi di oggetto business eseguire le seguenti attività:

1. Verificare che Process Designer Express sia aperto e che il diagramma di attività sia visualizzato.
2. Fare clic con il tasto destro del mouse sul collegamento di transizione al quale si desidera aggiungere l'analisi oggetto business.
3. Nel menu di scelta rapida fare clic su **Proprietà**. Viene visualizzata la finestra di dialogo Proprietà del collegamento di transizione.
4. Fare clic sulla casella di controllo **Abilita analisi oggetto business**. Il campo Nome diviene attivo.
5. Nel campo Nome inserire un nome per l'analisi dell'oggetto business. Il nome deve essere univoco nella maschera di collaborazione corrente e non deve superare i 100 caratteri. Utilizzare solo lettere, numeri o caratteri di sottolineatura. Gli altri caratteri non sono consentiti.
6. Fare clic sul segno più (+) accanto all'oggetto business che si desidera controllare per visualizzarne l'elenco attributi.
7. Selezionare gli attributi specifici che si desidera controllare.
8. Fare clic su **Applica** per salvare le modifiche.
9. Durante il runtime, utilizzare System Manager per abilitare o disabilitare l'analisi oggetto business in base alle esigenze.

Modifica di un'analisi di oggetto business

Per modificare le proprietà dell'analisi dell'oggetto business effettuare i seguenti passi:

1. Verificare che Process Designer Express sia aperto e che il diagramma di attività sia visualizzato.
2. Fare clic con il tasto destro del mouse sul collegamento di transizione dove è inserita l'analisi oggetto business.
3. Nel menu di scelta rapida, fare clic su **Proprietà**. Viene aperta la finestra di dialogo Proprietà del collegamento di transizione.
4. Modificare le proprietà appropriate. Se si sta modificando il nome, deve essere univoco nella maschera di collaborazione corrente e non deve superare i 100 caratteri. Utilizzare solo lettere, numeri o caratteri di sottolineatura. Gli altri caratteri non sono consentiti.
5. Fare clic su **Applica** per salvare le modifiche.

Per ulteriori informazioni sulla gestione dell'analisi dell'oggetto business consultare il manuale *WebSphere Business Integration Server Express System Administration Guide*.

Modifica di un collegamento di transizione

E' possibile scollegare e ricollegare i collegamenti di transizione e modificare l'aspetto dei segmenti della linea di transizione.

- Per scollegare e ricollegare, selezionare un un punto di collegamento — la fine di una linea — con il mouse e trascinarlo nella direzione desiderata. Lo scollegamento e ricollegamento consente di spostare un collegamento di transizione.
- Per creare un nuovo segmento della linea di transizione in un collegamento di transizione esistente, fare clic sulla sezione centrale di un segmento della linea con il tasto Ctrl premuto. L'attaccatura fra i due segmenti della linea viene contrassegnata da un quadratino.
- Per rimuovere un segmento di linea di transizione, fare clic sul quadratino del segmento della linea con il tasto Ctrl premuto.

Nota: La modifica dei segmenti della linea non è valida per i collegamenti ortogonali.

Per verificare che i segmenti della linea di transizione abbiano solo angoli destri, tenere premuto il tasto Maiusc quando si crea un nuovo segmento di linea.

Regole di business

Per semplificare i cambiamenti nelle necessita del business esistente, WebSphere Business Integration Server Express fornisce una funzione "regola di business". Le regole di business consentono di alterare la flessibilità di comportamento del runtime senza dover ridefinire e ridistribuire la maschera di collaborazione. E' possibile aggiungere, eliminare o modificare le regole di business senza influire sull'esecuzione degli altri componenti.

Ad esempio, un requisito business potrebbe essere di applicare uno sconto promozionale del 20% stagionale o di portare ad un livello superiore il tesseramento di un cliente per acquisti superiori a 300 euro in un periodo promozionale. In questi casi, la modifica della maschera di collaborazione sarebbe piuttosto laboriosa. E' possibile, tuttavia, utilizzare le regole di business per modificare il processo business in modo includere facilmente i nuovi requisiti di business.

Le sezioni che seguono spiegano l'utilizzo delle regole di business per soddisfare i propri requisiti di business:

- "Analisi di oggetto business e regole di business"
- "Esempi di utilizzo delle regole di business" a pagina 127

Per ulteriori informazioni sull'utilizzo della procedura guidata per le nuove regole di business consultare il manuale *System Implementation Guide*. Per ulteriori informazioni sulla visualizzazione e la gestione delle regole di business utilizzando System Monitor, consultare il manuale *System Administration Guide*.

Analisi di oggetto business e regole di business

Con la funzionalità delle regole di business, le analisi di oggetti business abilitano il motore delle regole di business a controllare i dati dell'oggetto business per la rilevazione di eventuali regole. Il motore delle regole di business elabora l'evento, se stabilisce che vi sono delle regole associate all'analisi dell'oggetto business. Per informazioni sulle altre funzioni dell'analisi oggetto business consultare "Utilizzo delle analisi di oggetti business" a pagina 124.

Ad esempio, lo Scenario 2 descrive una situazione in cui un utente desidera rilevare l'importo accumulato degli acquisti di un cliente. Poiché questa

informazione è memorizzata come attributo nell'oggetto business Customer, il sistema controlla l'attributo ed esegue la regola quando la condizione business viene soddisfatta. L'analisi dell'oggetto business fornisce la soluzione ideale per la definizione del punto specifico in cui i dati business vengono controllati sulla base di una regola di business. Consultare "Scenario 2: Applicazione delle regole di business utilizzando le analisi di oggetti business" a pagina 128.

Panoramica dei passi per l'utilizzo delle analisi di oggetti business con le regole di business

Per definire le analisi di oggetti business da utilizzare con le regole di business, eseguire i seguenti passi:

1. Decidere che determinati attributi di oggetti business verranno controllati dalle analisi business per rilevare la regola di business.
2. Determinare dove verranno controllati i dati business nel diagramma di attività (ad esempio, dopo un determinato nodo azione in una maschera di collaborazione).
3. In Process Designer Express, definire le analisi di business sui collegamenti di transizione nella maschera di collaborazione.
4. In System Manager, creare una collaborazione basata sulla maschera. Per informazioni sull'utilizzo di System Manager, consultare il manuale *System Administration Guide*.
5. nella procedura guidata Nuova regola di business, seguire i passi per definire le informazioni specifiche per la regola utilizzando le analisi di business definite prima. Per informazioni sull'utilizzo della procedura guidata Nuova regola di business, consultare il manuale *System Implementation Guide*.
6. Distribuire la nuova regola di business in WebSphere Business Integration Server Express.

Esempi di utilizzo delle regole di business

I seguenti esempi illustrano la regola dell'utente nell'aggiunta di determinata logica di business a WebSphere Business Integration Server Express. Questi esempi illustrano i passi per definire la logica business ed implementare le regole di business.

Scenario 1: Applicare una regola di business in un periodo stabilito

In questo scenario, il punto vendita ha un nuovo requisito di business: applicare uno sconto promozionale del 20% all'inizio della stagione commerciale natalizia. Nel progetto WebSphere Business Integration Server Express esistono le seguenti condizioni:

- Una maschera di collaborazione, PurchaseOrder Processing, che contiene la proprietà "Discount."
- Una collaborazione, OrderProcessing, basata sulla maschera PurchaseOrderProcessing.

Per applicare lo sconto per il periodo specificato eseguire i seguenti passi:

1. Aprire System Manager, dove è memorizzata la collaborazione OrderProcessing. Per informazioni sull'utilizzo di System Manager, consultare il manuale *System Administration Guide*.
2. Richiamare la procedura guidata Nuova regola di business per iniziare la creazione della regola di business. Per informazioni sull'utilizzo della procedura guidata Nuova regola di business, consultare il manuale *System Implementation Guide*.

3. Seguire i passi della procedura guidata Nuova regola di business per definire le informazioni specifiche per la regola (ad esempio, new discount rate = 20%, rule apply time = 1 dicembre 2004, 24:00).
4. Salvare la nuova regola di business nel progetto WebSphere Business Integration Server Express.
5. Distribuire la nuova regola di business in WebSphere Business Integration Server Express.

Risultato: il server aggiorna il motore delle regole con la nuova definizione di regola e memorizza la regola nel repository del server.

Al momento stabilito nella regola di business la collaborazione esegue OrderProcessing. Il cliente riceve uno sconto del 20%.

Scenario 2: Applicazione delle regole di business utilizzando le analisi di oggetti business

In questo scenario, l'azienda offre un vantaggio promozionale: promuove al livello Oro il tesseramento dei clienti i cui ordini accumulati in una stagione superano i 300 euro. Nel progetto WebSphere Business Integration Server Express esistono le seguenti condizioni:

- Un oggetto business, Order, che contiene il totale degli acquisti accumulati per il cliente.
- Una maschera di collaborazione, PurchaseOrderProcessing.
- Una collaborazione, OrderProcessing, basata sulla maschera PurchaseOrderProcessing.
- Una collaborazione, UpgradeMember, che contiene la logica dell'aggiornamento del livello di tesseramento di un cliente.

Per aggiornare il livello di tesseramento dei clienti che superano i 300 euro di acquisti per la stagione, eseguire i seguenti passi:

1. Definire un'analisi di business nella maschera di collaborazione PurchaseOrderProcessing in modo che ogni volta che il sistema elabora un ordine di acquisto, l'analisi di business controlla l'importo di acquisto del cliente.
2. Salvare la maschera di collaborazione.
3. Aprire System Manager. Per informazioni sull'utilizzo di System Manager, consultare il manuale *System Administration Guide*.
4. Richiamare la procedura guidata Nuova regola di business per iniziare la creazione di una nuova regola di business. Per informazioni sull'utilizzo della procedura guidata Nuova regola di business, consultare il manuale *System Implementation Guide*.
5. Seguire i passi della procedura guidata Nuova regola di business per definire le informazioni specifiche per la regola: la condizione che l'importo totale degli acquisti sia maggiore di 300 euro, l'analisi di business per il controllo e la l'azione della regola per aggiornare al livello superiore il tesseramento.
6. Salvare la nuova regola di business nel progetto WebSphere Business Integration Server Express.
7. Distribuire la nuova regola di business in WebSphere Business Integration Server Express.

Risultato: il server aggiorna il motore delle regole con la nuova definizione di regola e memorizza la regola nel repository del server.

La collaborazione OrderProcessing elabora gli ordini del cliente come al solito. Quando viene soddisfatta la condizione della regola di business che gli acquisti superano i 300 euro, la collaborazione UpgradeMember promuove il cliente al livello Oro.

Nodi decisione

Se si desidera che da un'azione si passi alla successiva a prescindere dalle condizioni, un collegamento di transizione è tutto ciò che serve. Se, invece, si desidera effettuare una diramazione del flusso verso più di un'azione in base ad una serie di condizioni, è necessario includere un nodo decisione. Nel suo utilizzo più comune, un nodo di decisione collega un'azione a tutte i suoi possibili risultati, incluso altre azioni, diagrammi secondari e simboli di fine. I nodi decisione possono essere utilizzati con i nodi azione, diagramma secondario e iteratore. Non posizionare un nodo decisione subito dopo un simboli di inizio.

Un nodo decisione di solito ha almeno due diramazioni; il numero massimo di diramazioni è sette. Ogni diramazione ha associata una condizione che determina se quella diramazione viene presa o meno.

Importante
Quando si implementa un nodo decisione, definire le condizioni in modo tale che ve ne sia sempre una che si risolve in true. Se nessuna delle condizioni contenute nel nodo decisione si risolve in true, si verifica un errore runtime.

Esistono tre tipi di diramazione in un nodo decisione:

- Normale — Una diramazione normale ha associata una condizione; se quella condizione viene soddisfatta, viene presa quella diramazione. E' possibile avere più diramazioni normali. Per impostazione predefinita, le diramazioni normali sono rappresentate da un quadratino blu.
- Eccezione — Una diramazione eccezione ha associato un tipo specifico di eccezione. La condizione di una diramazione eccezione testa se la variabile di sistema currentException è uguale al tipo di eccezione in cui si imposta la diramazione. E' possibile avere più diramazioni eccezione. Per impostazione predefinita le diramazioni di eccezione sono rappresentate da un quadratino rosso.
- Predefinita — La diramazione predefinita viene presa quando nessuna delle condizioni delle altre diramazioni è true. Ogni nodo decisione può avere una (e solo una) diramazione predefinita. Questa diramazione è facoltativa. Per impostazione predefinita, viene rappresentata da un quadratino nero.

Queste diramazioni vengono definite nella finestra di dialogo Proprietà della decisione, dove vengono anche impostate le condizioni come mostrato nella Figura 40 a pagina 130.

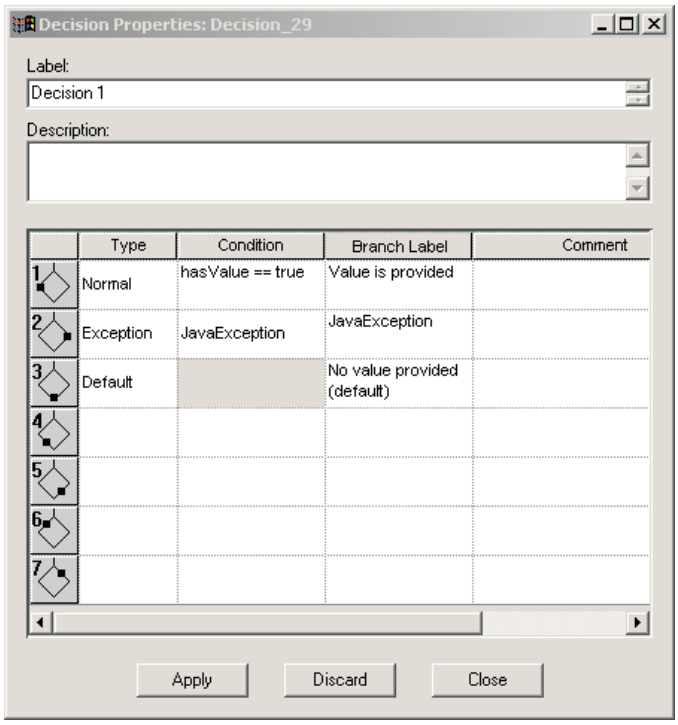


Figura 40. Finestra di dialogo Proprietà della decisione

Ogni diramazione definita di un nodo decisione deve avere un collegamento di transizione che la collega al risultato associato (ad esempio un nodo azione o un simbolo di fine).

La Figura 41 a pagina 131 illustra un esempio di diagramma di attività con un nodo decisione. In questo esempio, il nodo decisione ha tre diramazioni. La diramazione normale passa il flusso ad Azione 2 se la condizione si risolve in true. La diramazione eccezione passa il flusso a Fine con errore se viene generata un'eccezione JavaException. La diramazione predefinita passa il flusso ad Azione 3 se la condizione della diramazione normale si risolve in false e non viene rilevata l'eccezione JavaException.

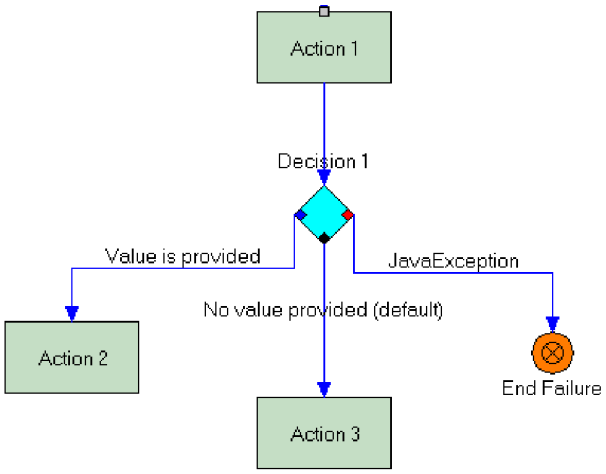


Figura 41. Diagramma di attività con un nodo decisione

Per aggiungere un nodo decisione al diagramma di attività eseguire i seguenti passi:

1. Verificare che l'editor di diagrammi sia aperto e che sia stato già posizionato il simbolo che passerà al nodo decisione. I nodi decisione possono essere utilizzati da qualsiasi azione, diagramma secondario o nodo iteratore.
2. Fare clic sul pulsante Nodo decisione nella barra degli strumenti Simboli.
3. Nel diagramma, posizionare il cursore sotto il simbolo che utilizzerà il nodo decisione e fare clic per collocare il nodo nel diagramma.
4. Creare un collegamento di transizione fra il nodo decisione ed il simbolo che lo richiama. Per ulteriori informazioni sulla creazione di collegamenti di transizione consultare "Creazione di un collegamento di transizione" a pagina 122.

Definizione di una diramazione normale

Ogni diramazione normale richiede una condizione. Queste condizioni vengono create con le variabili che vengono definite nella maschera di collaborazione o nello scenario. Prima di poter creare una diramazione normale si deve definire la variabile necessaria per la condizione. Per ulteriori informazioni consultare "Dichiarazione e modifica delle variabili di una maschera (scheda Dichiarazioni)" a pagina 88 e "Definizione delle variabili di scenario" a pagina 102.

Per definire una diramazione normale in un nodo decisione, eseguire i seguenti passi:

1. Nell'editor di diagrammi, fare doppio clic sul simbolo del nodo decisione. Viene aperta la finestra di dialogo Proprietà della decisione.
2. Nella linea della diramazione che si sta creando, fare clic sulla cella della tabella nella colonna Tipo e selezionare Normale dall'elenco a discesa dei tipi di diramazioni.
3. Fare clic con il tasto destro del mouse sulla cella della tabella nella colonna Condizione e selezionare Generatore di condizioni dal menu di scelta rapida.

   Nota: E' anche possibile immettere la condizione direttamente nella cella di tabella Condizione, invece di utilizzare l'editor di condizioni.

   Viene visualizzata la finestra di dialogo Editor di condizioni.

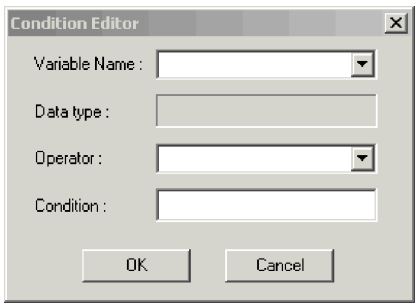


Figura 42. Editor delle condizioni

4. Nel campo Nome variabile, utilizzare l'elenco a discesa per selezionare la variabile che si desidera venga risolta per la condizione. Questo elenco contiene tutte le variabili della collaborazione definite per lo scenario.

   Quando si seleziona una variabile, il campo Tipo di dati viene automaticamente aggiornato in modo da includere il tipo di variabile (ad esempio, Boolean o String).
5. Nel campo Operatore utilizzare l'elenco a discesa per selezionare l'operatore appropriato da usare per risolvere la variabile. L'elenco contiene solo gli operatori supportati dal tipo di variabile che si sta utilizzando.
6. Nel campo Condizione, immettere il valore che si desidera utilizzare per la condizione. (Ad esempio, se si ha una variabile Boolean denominata hasValue, è possibile impostare la condizione su true o false).
7. Fare clic su Ok per chiudere l'editor delle condizioni e tornare alla finestra di dialogo Proprietà della decisione.
8. Facoltativamente, inserire un'etichetta per la diramazione nella cella di tabella Etichetta ramo. L'etichettatura delle diramazioni può migliorare la leggibilità del diagramma di attività.
9. Facoltativamente, inserire una descrizione della diramazione nella cella di tabella Commento.
10. Fare clic su Applica per aggiungere la diramazione al nodo decisione. Nel diagramma attività il nodo decisione ora contiene un quadratino blu per indicare la diramazione normale appena creata.

Dopo aver aggiunto una diramazione normale, è necessario collegarla al suo risultato associato con un collegamento di transizione.

Definizione di una diramazione eccezione

Per definire una diramazione eccezione in un nodo decisione, eseguire i seguenti passi:

1. Nell'editor di diagrammi, fare doppio clic sul simbolo del nodo decisione. Viene aperta la finestra di dialogo Proprietà della decisione.
2. Nella linea della diramazione che si sta creando, fare clic sulla cella della tabella nella colonna Tipo e selezionare Eccezione dall'elenco a discesa dei tipi di diramazione.
3. Fare clic sulla cella della tabella nella colonna Condizione e selezionare il tipo di eccezione dall'elenco a discesa dei tipi di eccezioni.
4. Facoltativamente, inserire un'etichetta per la diramazione nella cella di tabella Etichetta ramo. L'etichettatura delle diramazioni può migliorare la leggibilità del diagramma di attività.

5. Facoltativamente, inserire una descrizione della diramazione nella cella di tabella Commento.
6. Fare clic su Applica per aggiungere la diramazione al nodo decisione. Nel diagramma attività il nodo decisione ora contiene un quadratino rosso per indicare la diramazione eccezione appena creata.

Dopo aver aggiunto una diramazione eccezione, è necessario collegarla al suo risultato associato con un collegamento di transizione.

Definizione di una diramazione predefinita

Ogni nodo decisione può avere solo una diramazione predefinita. L'aggiunta di una diramazione predefinita è facoltativa.

Per aggiungere una diramazione predefinita al nodo decisione eseguire i seguenti passi:

1. Nell'editor di diagrammi, fare doppio clic sul simbolo del nodo decisione. Viene aperta la finestra di dialogo Proprietà della decisione.
2. Nella linea della diramazione che si sta creando, fare clic sulla cella della tabella nella colonna Tipo e selezionare Predefinita dall'elenco a discesa dei tipi di diramazioni.
3. Facoltativamente, inserire un'etichetta per la diramazione nella cella di tabella Etichetta ramo. L'etichettatura delle diramazioni può migliorare la leggibilità del diagramma di attività.
4. Facoltativamente, inserire una descrizione della diramazione nella cella di tabella Commento.
5. Fare clic su Applica per aggiungere la diramazione al nodo decisione. Nel diagramma attività il nodo decisione ora contiene un quadratino nero per indicare la diramazione predefinita appena creata.

Non è possibile specificare una condizione per la diramazione predefinita. La condizione è implicita; si risolve in true quando tutte le condizioni associate alle altre diramazioni si risolvono in false.

Dopo aver aggiunto una diramazione predefinita, è necessario collegarla al suo risultato associato con un collegamento di transizione.

Combinazione di un'eccezione e di una condizione della logica di diramazione

Una diramazione può essere normale o un'eccezione ma non entrambe. Vi sono situazioni, tuttavia, in cui si potrebbe voler specificare il percorso di esecuzione da prendere in risposta a sue condizioni simultanee: si è verificata un'eccezione ed un'altra condizione è true. Questa combinazione equivale all'utilizzo di un operatore `AND` in un'espressione condizionale.

Ad esempio, si supponga di voler creare queste due condizioni:

```
Eccezione == JavaException && hasValue == false
Eccezione == JavaException && hasValue == true
```

Per modellare un simile costrutto, creare due livello di nodi decisione, posizionando un nodo azione fra loro, come mostrato di seguito:

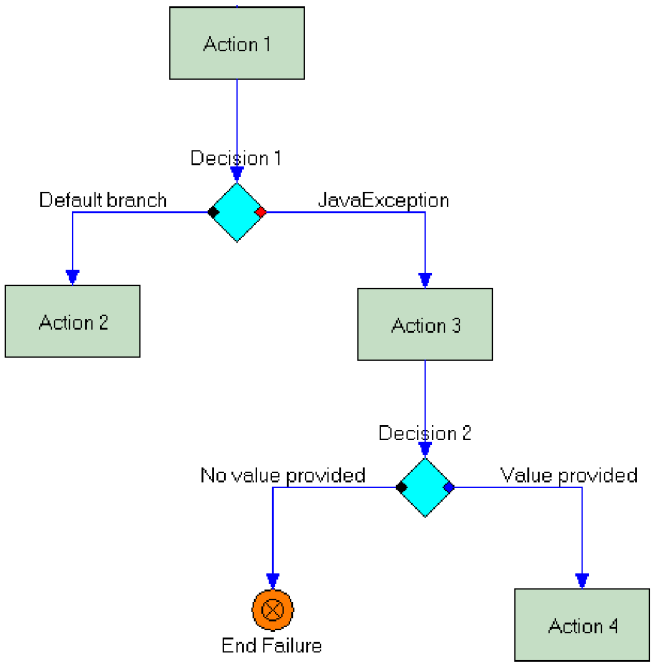


Figura 43. Combinazione di un'eccezione e di una condizione

Scambio di diramazioni di nodi decisione

E' possibile scambiare le diramazioni in un nodo decisione essenzialmente trascinando le diramazioni nel diagramma di attività. Process Designer Express aggiorna automaticamente la finestra di dialogo Proprietà della decisione per riflettere la nuova posizione di ciascuna diramazione.

Per scambiare la posizione di due diramazioni in un nodo decisione, eseguire i seguenti passi:

1. Nell'editor dei diagrammi, individuare il nodo decisione di cui si desidera scambiare le diramazioni.
2. Fare clic sul collegamento di transizione di una delle diramazioni che si desidera scambiare.
3. Tenendo premuto il tasto Alt, trascinare l'inizio del collegamento di transizione dalla sua posizione originaria alla nuova diramazione.
4. Rilasciare il tasto Alt. Process Designer Express scambia i due collegamenti di transizione e le relative condizioni associate ed aggiorna l'elenco di diramazioni nella finestra di dialogo Proprietà della decisione.

Chiamate di servizio

Un nodo azione non può, da solo, inviare una richiesta ad un connettore o ad un'altra collaborazione. E' necessario, invece, collegare il nodo azione ad una chiamata di servizio. InterChange Server Express supporta sia le chiamate sincrone che quelle asincrone. Quello che segue è un esempio di chiamata di servizio sincrona che utilizza una richiesta Retrieve:

```
status = retrieve(business-object, port);
```

Stato della chiamata completata

Tipo di richiesta

Dati, inviati e restituiti nella stessa variabile

Una chiamata di servizio si collega sempre ad un'azione. L'azione genera la chiamata di servizio e ne gestisce i risultati sui relativi collegamenti di transizione in uscita. Un'azione ed una chiamata di servizio funzionano in coppia; la chiamata di servizio esegue la funzione input/output remota dell'azione.

La Figura 44 illustra la relazione fra un nodo azione e la chiamata di servizio. La figura mostra l'ordine in cui l'ambiente di collaborazione runtime elabora l'azione e la relativa chiamata di servizio.

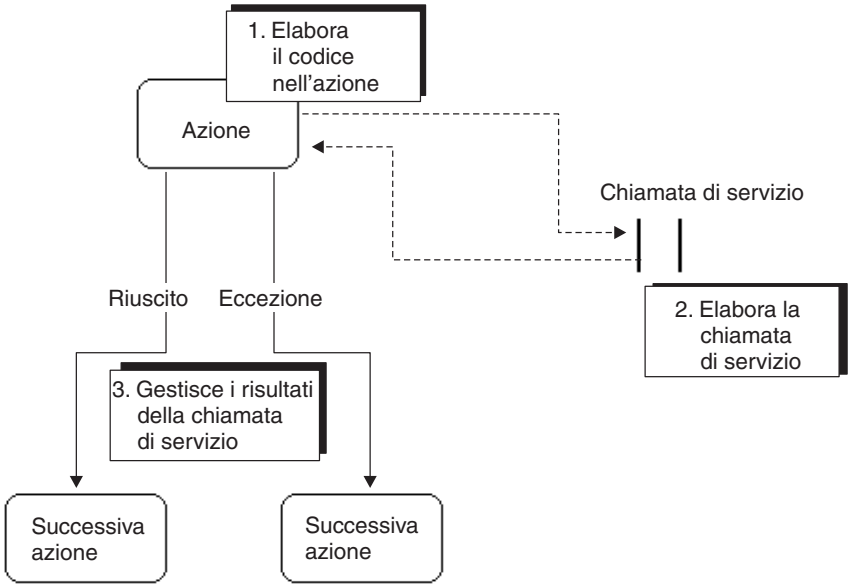


Figura 44. Relazione fra nodo azione e chiamata di servizio

La Figura 44 mostra che l'ambiente di collaborazione runtime prima elabora il codice nell'azione e poi esegue la chiamata di servizio. Quando termina la chiamata di servizio, i collegamenti di transizione del nodo azione gestiscono il risultato. Quando un nodo azione genera una chiamata di servizio, è buona norma includere un collegamento eccezione in uscita che rileva eventuali ServiceCallException.

Tipi di chiamate di servizio

InterChange Server Express supporta tre tipi di chiamate di servizio: sincrona, in uscita asincrona e in ingresso asincrona. Le sezioni che seguono descrivo ciascun tipo di chiamata.

Chiamata di servizio sincrona

Questo tipo di chiamata di servizio utilizza un meccanismo di richiesta/risposta sincrona. La chiamata di servizio invia la richiesta ma non termina finché non arriva la risposta e non viene elaborata.

Le chiamate di servizio sincrone supportano la compensazione. Supportano, inoltre, un valore timeout per i processi business di lunga durata.

Per impostazione predefinita, tutte le chiamate di servizio aggiunte ad un diagramma di attività sono sincrone. E' possibile modificare il tipo, se richiesto dallo scenario.

Chiamata di servizio in uscita asincrona

Una chiamata di servizio in uscita asincrona invia una richiesta ma non prevede o aspetta una risposta per continuare la sua elaborazione. Una maschera di collaborazione deve supportare i processi business di lunga durata, per poter utilizzare le chiamate di servizio in uscita asincrone.

Se un una chiamata di servizio in uscita asincrona ha una porta collegata ad una collaborazione invece che ad un connettore, la chiamata di servizio diviene automaticamente sincrona.

Le chiamate di servizio in uscita asincrone supportano la compensazione ma non un valore timeout per i processi business di lunga durata.

Chiamata di servizio in ingresso asincrona

Una chiamata di servizio in ingresso asincrona attende di ricevere un evento in ingresso in base ad un attributo di correlazione o una serie di attributi di correlazione che identificano l'evento. Viene utilizzata in congiunzione con i processi business di lunga durata. Per ulteriori informazioni consultare "Utilizzo di un attributo di correlazione" a pagina 141.

Quando viene creata una chiamata di servizio in ingresso asincrona, le viene assegnato un valore timeout; se la chiamata di servizio non riceve un evento in ingresso prima che scada il timeout, viene generata l'eccezione TimeoutException.

Le chiamate di servizio asincrone sono disponibili solo se la maschera di collaborazione supporta i processi business di lunga durata. E' possibile abilitare questo supporto in qualunque momento durante i processi di sviluppo della maschera facendo clic sull'opzione Supporto processi business di lunga durata nella scheda Generale della finestra Definizioni maschera.

Le chiamate di servizio in ingresso asincrone non supportano la compensazione.

Creazione di una chiamata di servizio

Eseguire i seguenti passi per aggiungere una chiamata di servizio al diagramma di attività:

1. Verificare che l'editor dei diagrammi sia aperto.
2. Nell'area di lavoro, fare clic con il tasto destro del mouse sul simbolo del nodo azione al quale si desidera associare una chiamata di servizio.
3. Dal menu di scelta rapida, fare clic su Aggiungi nodo azione. La chiamata di servizio viene aggiunta al diagramma di attività; una linea di puntini collega la chiamata di servizio al nodo azione. Per impostazione predefinita la chiamata di servizio è sincrona.

Nota: Un'azione può supportare solo una chiamata di servizio (statica o dinamica), che deve essere sola nel nodo azione.

Definizione di una chiamata di servizio

Dopo aver creato la chiamata di servizio è necessario definirla. Utilizzare la finestra di dialogo Proprietà della chiamata di servizio per specificare le proprietà richieste:

- La porta a cui viene inviata la chiamata di servizio
- La variabile che contiene l'oggetto business da inviare
- L'istruzione dell'oggetto business
- La serie di correlazioni (richiesta per le chiamate di servizio in ingresso asincrone)

Facoltativamente, è anche possibile specificare:

- Il supporto per la compensazione se si sta utilizzando una collaborazione transazionale. Per ulteriori informazioni, consultare "Definizione della compensazione" a pagina 140
- Il tipo di chiamata di servizio (per impostazione predefinita, tutte le chiamate di servizio sono sincrone ma possono essere modificate in chiamate asincrone in uscita o asincrone in ingresso). Per ulteriori informazioni, consultare "Definizione del tipo di chiamata di servizio" a pagina 138.
- Il valore timeout da utilizzare con le chiamate di servizio sincrone e asincrone in ingresso. Per ulteriori informazioni, consultare "Definizione del tipo di chiamata di servizio" a pagina 138.
- La serie di correlazioni utilizzato per la corrispondenza attributi (per le chiamate di servizio sincrone e asincrone in uscita).

Suggerimenti

Alla restituzione di una chiamata di servizio, la variabile dell'oggetto business contiene il risultato della chiamata. I dati contenuti nell'oggetto business originale andranno persi se la chiamata di servizio richiama dei nuovi dati per l'oggetto business. Pertanto, se si prevede di aver bisogno dei valori dell'oggetto originale, è utile copiare l'oggetto business originale in una variabile temporanea in un'azione che richiama la chiamata di servizio.

Per definire una normale chiamata di servizio (una che non utilizza la compensazione o delle serie di correlazioni) eseguire i seguenti passi):

1. Nel diagramma di attività, fare doppio clic su simbolo della chiamata di servizio già creato. Viene visualizzata la finestra di dialogo Proprietà della chiamata di servizio, come illustrato nella Figura 45 a pagina 138. Non è possibile immettere direttamente un valore nel campo Etichetta per una chiamata di servizio. Process Designer Express assegna l'etichetta al termine della definizione della chiamata di servizio.

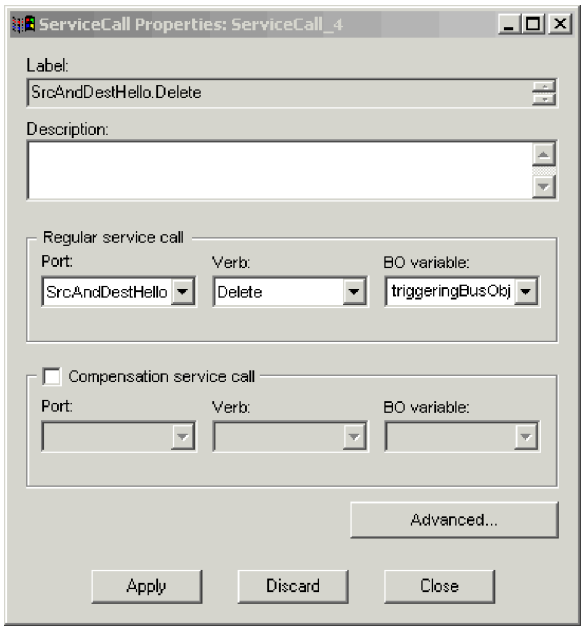


Figura 45. Finestra di dialogo Proprietà della chiamata di servizio

2. Facoltativamente, specificare la descrizione della chiamata di servizio.
3. Utilizzare l'elenco a discesa Porta per selezionare la porta che la chiamata di servizio utilizzerà per inviare o ricevere le richieste.
4. Utilizzare l'elenco a discesa Istruzione per specificare quale istruzione verrà utilizzata nella richiesta. Ad esempio, per aggiornare un'applicazione con in dati contenuti nell'oggetto business che invia la chiamata di servizio, utilizzare l'istruzione Aggiorna (update).
5. Usare l'elenco a discesa Variabile oggetto business per selezionare la variabile che contiene l'oggetto business che invia la chiamata di servizio. Se si prevede di supportare i processi business di lunga durata, deve essere una variabile globale di maschera o di porta; le variabili di scenario non possono essere utilizzate in questa funzione dei processi di business di lunga durata.
6. Fare clic su Applica per salvare la definizione.

Definizione del tipo di chiamata di servizio

Per modificare il tipo di chiamata di servizio, effettuare le seguenti operazioni:

1. Se la finestra di dialogo Proprietà della chiamata di servizio non è aperta, visualizzarla facendo doppio clic sul simbolo della chiamata di servizio nel diagramma di attività.
2. Accertarsi di aver fornito il nome porta richiesto, l'istruzione (verb) ed il nome variabile dell'oggetto business per la chiamata di servizio.
3. Fare clic sul pulsante Avanzate nella finestra di dialogo Proprietà della chiamata di servizio. Viene visualizzata la finestra di dialogo Proprietà avanzate della chiamata di servizio.

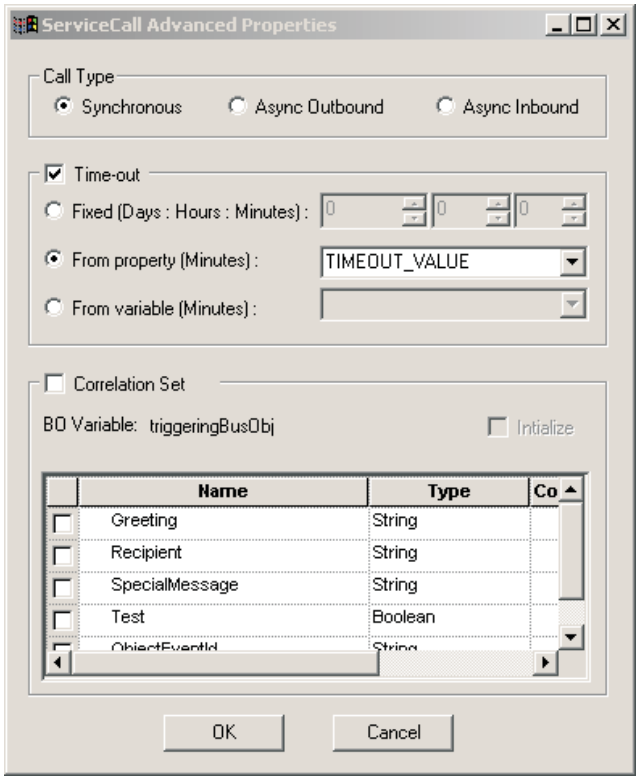


Figura 46. Finestra di dialogo Proprietà avanzate della chiamata di servizio

4. Nella casella Tipo di chiamata, fare clic sul pulsante di opzione accanto al tipo di chiamata di servizio che si desidera utilizzare.

   Nota: L'opzione Asincrona in entrata è disponibile solo se è abilitato il supporto per i processi business di lunga durata nella definizione della maschera.

5. Se si sta utilizzando una chiamata di servizio sincrona o una chiamata di servizio asincrona in ingresso e si desidera specificare un valore timeout da usare con in processi business di lunga durata, fare clic sulla casella di controllo Timeout ed impostare il valore su uno dei seguenti:
   - Fisso — Questa opzione richiede di specificare il valore timeout in giorni, ore e minuti. Selezionare questa opzione se si desidera utilizzare sempre lo stesso valore timeout per la chiamata di servizio. Questo valore non può essere modificato durante la configurazione della collaborazione.
   - Da proprietà — Questa opzione consente l'utilizzo di una proprietà specifica per collaborazione per specificare dinamicamente il valore timeout (in secondi) durante la configurazione della collaborazione. Nell'elenco a discesa Da proprietà, selezionare la proprietà creata per rappresentare il valore timeout.
   - Da variabile — Questa opzione consente di utilizzare una variabile globale di tipo oggetto Java per impostare il valore timeout (misurato in secondi) durante il runtime. Selezionare il nome variabile appropriato dall'elenco a discesa Da variabile.

   Nota: Se la porta di una chiamata di servizio sincrona è collegata ad una collaborazione invece che ad un connettore, qualsiasi valore timeout specificato verrà ignorato.

6. Fare clic su Ok per chiudere la finestra di dialogo Proprietà avanzate della chiamata di servizio.
7. Fare clic su Applica nella finestra Proprietà della chiamata di servizio per salvare le modifiche.

Definizione della compensazione

I passi di transazione secondaria di una collaborazione definiscono il comportamento transazionale di una collaborazione transazionale. Una *transazione secondaria* è un'operazione in cui la collaborazione invia una richiesta che provoca una modifica dei dati transazionali nella memoria dati di un'applicazione. Le chiamate di servizio implementano i passi di transazione secondaria. Una chiamata di servizio che ha una delle richieste Crea, Elimina o richiesta Aggiorna è un passo di transazione secondaria; una Recupera non lo è (perché non modifica i dati). Altre istruzioni potrebbero essere o non essere transazionali, a seconda se modificano i dati nella memoria dati di un'applicazione.

Nota: Anche se una chiamata di servizio con una richiesta Recupera (retrieve) non viene considerata un passo di transazione secondaria, si può specificare una compensazione.

Per creare una maschera di collaborazione che supporti il comportamento transazionale, definire una *compensazione* per ogni passo di transazione secondaria. La compensazione è un'azione di annullamento logico: se l'esecuzione di un oggetto collaborazione non riesce, provoca un rollback delle operazioni eseguite precedentemente. Quando si verifica il rollback, l'ambiente di collaborazione runtime torna indietro lungo il percorso di esecuzione, eseguendo un passo di compensazione per ogni passo normale che è stato già eseguito e che ha definita la compensazione. Il rollback quindi riporta, in maniera logica, i dati allo stato in cui erano prima che la collaborazione transazionale avesse iniziato l'esecuzione.

Visto in Process Designer Express, un passo di transazione secondaria è una chiamata di servizio che richiede un'operazione come Crea, Aggiorna o Elimina. Queste operazioni risultano sempre in transazioni di modifica dati in un'applicazione. Per una chiamata di servizio è possibile specificare anche una compensazione che richiede un'operazione Recupera, sebbene la compensazione non sia richiesta perché non vengono modificati dati durante quel tipo di operazione. Una chiamata di servizio viene definita da un oggetto business con un'istruzione particolare, che la collaborazione invia ad un'altra collaborazione o ad un connettore. La compensazione per quell'operazione è un'altro oggetto business ed istruzione. Qualsiasi oggetto business e istruzione possono compensare per una chiamata di servizio.

La Tabella 41 elenca alcuni dei tipi comuni di compensazione.

Tabella 41. Esempi di compensazione

Azione	Compensazione
Creare un oggetto business	Eliminare un oggetto business
Eliminare un oggetto business	Creare un oggetto business
Aggiornare un oggetto business	Aggiornare un oggetto business, ripristinando i valori precedenti

La compensazione è supportata sia per le chiamate di servizio sincrone che per quelle asincrone in uscita. Se la collaborazione è transazionale e se l'istruzione della chiamata di servizio richiede una modifica di dati, è possibile specificare

l'operazione di compensazione che esegue il rollback della normale chiamata di servizio. Definire la compensazione come segue:

1. Se la finestra di dialogo Proprietà della chiamata di servizio non è aperta, visualizzarla facendo doppio clic sul simbolo della chiamata di servizio nel diagramma di attività.
2. Fare clic sulla casella di controllo Compensazione. I campi Porta, Istruzione e Variabile oggetto business del riquadro Chiamata servizio compensazione divengono attivi.
3. Nel riquadro Chiamata servizio compensazione selezionare la porta, l'istruzione e la variabile oggetto business da utilizzare nella chiamata di servizio di compensazione. Le chiamate di servizio di compensazione possono utilizzare la stessa porta, istruzione e variabile oggetto business della normale chiamata di servizio, oppure è possibile specificare delle impostazioni diverse.
4. Fare clic su Applica per salvare le modifiche.

Per ulteriori informazioni sulle collaborazioni transazionali, consultare "Utilizzo delle funzioni transazionali" a pagina 154.

Utilizzo di un attributo di correlazione

Un attributo di correlazione viene utilizzato per identificare una conversazione. Una *conversazione* è un'unità di comunicazione coerente fra due processi business. Poiché possono esistere più conversazioni quando due o più processi business comunicano fra loro, è necessario identificare una conversazione con un attributo di correlazione. Si può pensare all'attributo di correlazione come ad un UID di una conversazione. L'ID deve essere inizializzato quando viene avviata la conversazione e tutti i successivi partecipanti devono utilizzare questo ID quando vengono coinvolti nella conversazione.

InterChange Server Express supporta una sola conversazione per scenario. Se è necessaria più di una conversazione, si devono utilizzare più scenari all'interno della maschera di collaborazione oppure utilizzare più maschere di collaborazione. Inoltre, un attributo di correlazione può essere inizializzato solo una volta nello scenario.

Per poter utilizzare un attributo di correlazione è necessario verificare che seguenti condizioni siano soddisfatte:

- Sia stato aggiunto il supporto per i processi business di lunga durata quando è stata definita la maschera di collaborazione.
- Siano state create una o più variabili di maschera da utilizzare per catturare i valori degli attributi di correlazione. Per ogni attributo di oggetto business che si prevede di selezionare per la correlazione, è necessario avere una variabile di maschera univoca che può catturare quel valore.

Quando si utilizza un attributo di correlazione con una chiamata di servizio, l'oggetto business viene determinato automaticamente; si tratta sempre dell'oggetto business assegnato alla porta utilizzata per la chiamata di servizio.

Dopo aver inizializzato un attributo di correlazione in uno scenario, è possibile impostarlo per le chiamate di servizio in uscita (consultare "Impostazione degli attributi di correlazione" a pagina 142), associarlo sulle chiamate di servizio asincrone in ingresso (consultare "Associazione degli attributi di correlazione" a pagina 143) o effettuare entrambe le operazioni.

Inizializzazione degli attributi di correlazione: Il primo passo per l'utilizzo degli attributi di correlazione è l'inizializzazione. Quando si inizializza un attributo di

correlazione si specifica l'attributo oggetto business e la variabile di maschera da utilizzare per catturare il valore di quell'attributo. L'inizializzazione deve essere effettuata prima di impostare l'attributo di correlazione su una chiamata di servizio in uscita o di associarlo in un ambiente runtime.

L'inizializzazione può essere effettuata in un nodo di inizio, come chiamata di servizio asincrona in uscita oppure come chiamata di servizio sincrona.

Per definire ed inizializzare gli attributi di correlazione in una chiamata di servizio, eseguire i seguenti passi:

1. Verificare che Process Designer Express sia aperto e che la finestra Proprietà della chiamata di servizio sia visualizzata.
2. Fare clic su Avanzate per aprire la finestra di dialogo Proprietà avanzate della chiamata di servizio.
3. Fare clic su Serie di correlazioni. La variabile oggetto business viene definita automaticamente, come quella assegnata alla porta in uscita della chiamata di servizio.
4. Fare clic su Inizializza.
5. Fare clic sulla casella di controllo accanto ad ogni attributo di oggetto business che si desidera utilizzare per la correlazione.
6. Per ogni attributo selezionato, utilizzare l'elenco a discesa nella colonna Correlazione per selezionare una variabile di maschera per catturare e memorizzare il valore dell'attributo.
7. Fare clic su Applica.

La procedura per inizializzare una serie di correlazioni da un nodo di inizio è la stessa, ad eccezione della selezione della della casella di controllo Inizializza. Poiché un nodo di inizio non partecipa all'associazione degli attributi di correlazione, l'inizializzazione è implicita.

Impostazione degli attributi di correlazione: Dopo aver definito ed inizializzato una serie di correlazioni, è possibile assegnarla a qualsiasi chiamata di servizio in uscita. La chiamata di servizio, quindi, invia una richiesta che include la serie di correlazioni. E' possibile impostare gli attributi di correlazione su tante chiamate di servizio in uscita quante ne sono necessarie.

Per impostare gli attributi di correlazione su una chiamata di servizio in uscita, eseguire i seguenti passi.

1. Verificare che Process Designer Express sia aperto e che la finestra Proprietà della chiamata di servizio sia visualizzata per la chiamata di servizio in uscita.
2. Fare clic su Avanzate per aprire la finestra di dialogo Proprietà avanzate della chiamata di servizio.
3. Fare clic su Serie di correlazioni.
4. Fare clic sulla casella di controllo accanto ad ogni attributo di oggetto business che si desidera definire nella serie di correlazioni inizializzata.
5. Per ogni attributo selezionato, utilizzare l'elenco a discesa nella colonna Correlazione per selezionare la variabile di maschera utilizzata per catturare e memorizzare il valore dell'attributo. Questa variabile deve essere la stessa utilizzata quando è stata inizializzata la serie di correlazioni.
6. Fare clic su Applica.

Associazione degli attributi di correlazione: Dopo aver inizializzato una serie di correlazioni, è possibile configurare una chiamata di servizio asincrona in ingresso per ricevere qualunque risposta corrisponda a quella serie particolare di correlazioni.

Per specificare una serie di correlazioni in una chiamata di servizio asincrona in ingresso, eseguire i seguenti passi:

1. Verificare che Process Designer Express sia aperto e che la finestra Proprietà della chiamata di servizio sia visualizzata per la chiamata di servizio asincrona in ingresso.
2. Fare clic su Avanzate per aprire la finestra di dialogo Proprietà avanzate della chiamata di servizio.
3. Fare clic su Serie di correlazioni.
4. Fare clic sulla casella di controllo accanto ad ogni attributo di oggetto business che si desidera definire nella serie di correlazioni inizializzata.
5. Per ogni attributo selezionato, utilizzare l'elenco a discesa nella colonna Correlazione per selezionare la variabile di maschera utilizzata per catturare e memorizzare il valore dell'attributo. Questa variabile deve essere la stessa utilizzata quando è stata inizializzata la serie di correlazioni.
6. Fare clic su Applica.

Nel runtime, se la chiamata di servizio asincrona in ingresso ha degli attributi che corrispondono a quelli definiti nella serie di correlazioni, la chiamata di servizio viene richiamata dallo scenario. Se gli attributi non corrispondono, la chiamata di servizio viene instradata in un altro scenario o collaborazione che ha definita una serie di correlazioni corrispondente.

Gestione dei risultati

Quando viene eseguita una chiamata di servizio, lo scenario riceve due valori di ritorno: uno stato ed un oggetto business. La Tabella 42 descrive l'utilizzo di ciascuno.

Tabella 42. Valori di ritorno della chiamata di servizio

Restituito	Descrizione
Stato	Il collegamento di transizione normale in uscita dell'azione che ha generato la chiamata di servizio, testa la riuscita della chiamata di servizio. Un collegamento di transizione eccezione può cercare un'eccezione `ServiceCallException` per testare la non riuscita di una chiamata di servizio. Una chiamata di servizio può non riuscire a causa di problemi di trasporto come pure per problemi dell'applicazione. Poiché l'errore nel trasporto può causare la duplicazione di dati, è importante stabilire se la non riuscita di una chiamata di servizio è dovuta a problemi di trasmissione. Per ulteriori informazioni, consultare "Gestione di eccezioni di chiamate di servizio particolari" a pagina 191.
Oggetto business	Al termine di una chiamata di servizio, la variabile oggetto business, utilizzata dalla chiamata di servizio, contiene dei nuovi valori di dati, se la chiamata ha portato ad una qualunque modifica nell'applicazione.

- Dopo una chiamata di servizio Crea, non è necessario verificare il valore dell'oggetto business. Se la chiamata di servizio torna con esito positivo, l'operazione Crea è riuscita.
- Una richiesta Elimina (delete) non necessariamente porta alla reale eliminazione dei dati dell'applicazione. Poiché molte applicazioni non supportano

l'eliminazione, un connettore gestisce le istruzioni Delete secondo le regole della relativa applicazione. Ad esempio, un connettore potrebbe convertire una richiesta Elimina in una richiesta Aggiorna (update), aggiornando l'entità applicazione allo stato di Non attivo.

Considerazioni sulle prestazioni

Le prestazioni della collaborazione sono influenzate dal numero di chiamate di servizio e dalla dimensione degli oggetti business passati dalle chiamate di servizio. Poiché non è possibile modificare le dimensioni dell'oggetto business, tentare di ridurre il numero di chiamate di servizio effettuate dalla collaborazione.

Ad esempio, si supponga che lo scenario debba eseguire un'operazione su degli oggetti business secondari in un oggetto business gerarchico. In alcune situazioni può essere più efficace richiamare l'intero oggetto gerarchico e ripetere localmente le operazioni piuttosto che creare una chiamata di servizio per ogni oggetto business secondario che deve eseguire una chiamata di servizio.

Diagrammi secondari

Quando la logica del diagramma di attività diviene complessa, spesso torna utile eseguire una partizione della logica, separandone le unità distinte in *diagrammi secondari*. Ogni diagramma secondario viene associato ad un particolare diagramma principale.

La Figura 47 illustra uno scenario in cui il diagramma di attività principale contiene dei riferimenti a due diagrammi secondari, diagramma Retrieve e diagramma Delete.

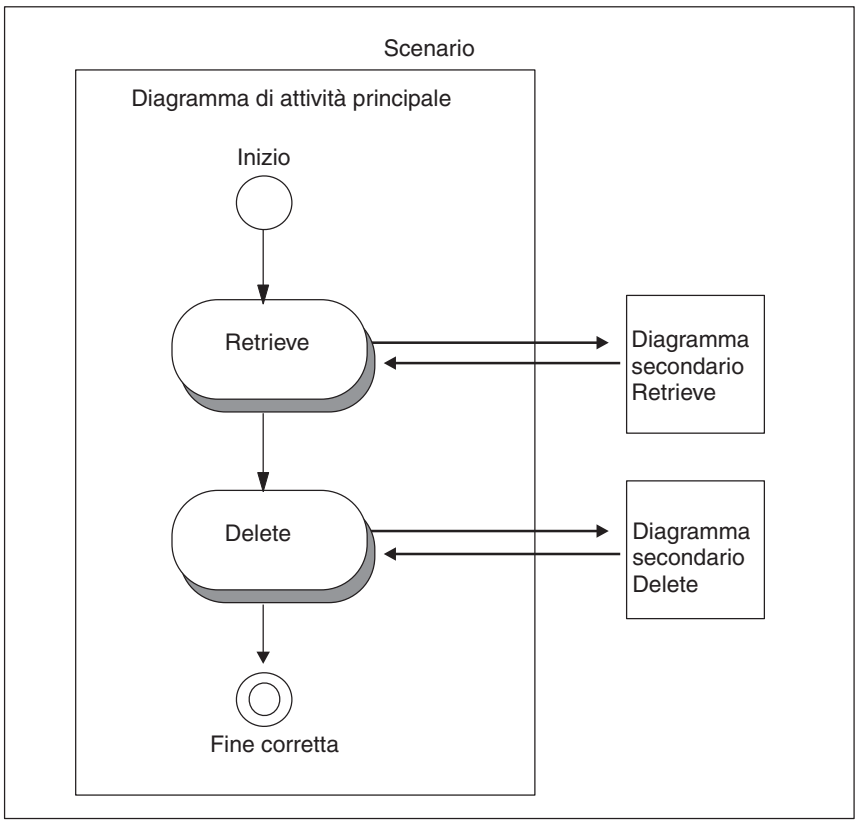


Figura 47. Scenario con due diagrammi secondari

Nota: Un iteratore è una forma specializzata di diagramma secondario. Per gli iteratori sono valide tutte le informazioni di base relative ai diagrammi secondari; per le informazioni che riguardano specificatamente gli iteratori, fare riferimento a "Iteratori" a pagina 149.

I diagrammi di attività in uno scenario sono disposti gerarchicamente. Tutti i diagrammi secondari e gli iteratori di uno scenario derivano dal dal diagramma di attività principale dello scenario. La Figura 48 illustra questa relazione. Il diagramma di attività in cui appare il simbolo di diagramma secondario viene denominato *diagramma principale* del diagramma secondario.

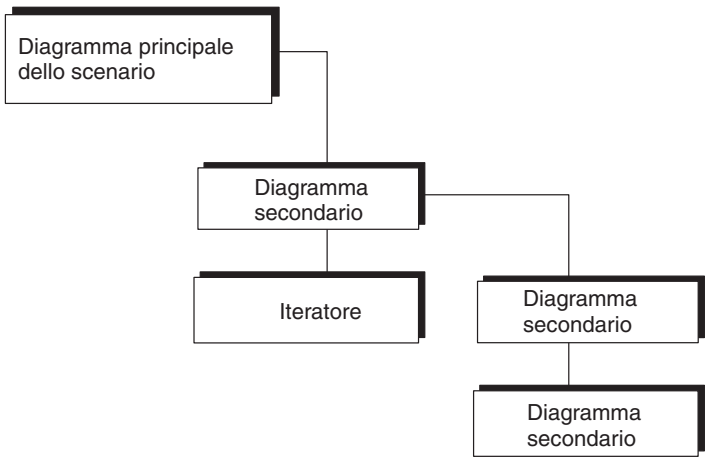


Figura 48. Relazione fra diagramma principale e diagrammi secondari

Un diagramma secondario ha accesso a tutte le proprietà della maschera di collaborazione e a tutte le variabili di scenario. La Tabella 43 riassume le differenze fra un diagramma secondario ed uno principale.

Tabella 43. Confronto fra i diagrammi principale e secondario

Argomento	Diagramma principale	Diagramma secondario
Modalità di creazione nella progettazione	Creato automaticamente quando si crea uno scenario	Controllato dal simbolo digramma secondario del diagramma principale
Causa dell'esecuzione al runtime	Avvia l'esecuzione quando l'ambiente di collaborazione runtime passa un evento di attivazione	Avvia l'esecuzione quando il percorso di esecuzione del diagramma principale porta all'esecuzione; non ha un evento di attivazione
In caso di eccezione non gestita o generata	Passa all'ambiente di collaborazione runtime	Passa al diagramma principale
Completamento nel runtime	Torna all'ambiente di collaborazione runtime	Torna al diagramma principale

Creazione di un diagramma secondario

Per aggiungere un diagramma secondario al diagramma di attività:

1. Nella barra degli strumenti Simboli, fare clic sul pulsante Diagramma secondario.
2. Fare clic su un diagramma di attività attivo per posizionare il simbolo del diagramma secondario.

   Viene visualizzato un identificativo univoco per il diagramma secondario nella struttura ad albero dello scenario, disposto gerarchicamente sotto il diagramma principale. La struttura ad albero dello scenario visualizza il nome nel seguente formato:

   (*UID*)

   Se si fornisce un'etichetta al diagramma secondario, la struttura dello scenario visualizza il nome nel seguente formato:

   etichetta (*UID*)

   L'UID è un identificativo univoco che è anche il nome dell'oggetto diagramma secondario della struttura dello scenario. Come per i UID degli altri simboli, è possibile scegliere se visualizzarlo o meno per il diagramma secondario. Per attivare e disattivare la visualizzazione dell'UID, utilizzare il menu di scelta rapida del nodo dello scenario nella struttura della maschera.

3. Fare doppio clic sul nome del diagramma secondario nella struttura dello scenario oppure fare clic con il tasto destro del mouse sul suo nodo nella finestra dell'editor di diagrammi e selezionare Apri diagramma secondario.

   Viene visualizzata una nuova finestra nell'area di lavoro, in cui è possibile definire un nuovo diagramma di attività.

Come per il diagramma di attività principale, il diagramma secondario inizia con un simbolo Inizio e termina con un simbolo Fine corretta, e, facoltativamente, con uno o più simboli Fine con errore. Un diagramma secondario può contenere tutti i componenti di un diagramma, inclusi i diagrammi secondari e gli iteratori.

Definizione di un diagramma secondario

Dopo che il diagramma secondario viene visualizzato nel diagramma di attività, è possibile definirne le proprietà nella finestra di dialogo Proprietà del diagramma secondario. Le proprietà di un diagramma secondario sono l'etichetta e la descrizione. Sono tutte facoltative.

Per definire le proprietà del diagramma secondario:

1. Visualizzare la finestra di dialogo Proprietà del diagramma secondario.

   E' possibile visualizzare questa finestra di dialogo in uno qualunque dei seguenti modi:

   - Fare doppio clic sul diagramma secondario selezionato.
   - Fare clic con il tasto destro del mouse sul diagramma secondario per richiamare il menu di scelta rapida e scegliere Proprietà.
   - Scegliere Proprietà dal menu a discesa Modifica.
   - Utilizzare la combinazione di tasti di accesso rapido `Ctrl + Invio`.
2. Facoltativamente, specificare l'etichetta e la descrizione relativa a quel diagramma secondario.

   L'etichetta rende il diagramma di attività più leggibile, apponendo al diagramma secondario del testo che è più descrittivo dell'UID. Il campo di descrizione è un luogo in cui immettere un commento, che descrive lo scopo del diagramma secondario.
3. Fare clic su Applica per salvare le proprietà del diagramma secondario. Fare clic su Ignora per annullare le proprietà. Fare clic su Chiudi per annullare la definizione del diagramma secondario.

Eliminazione di un diagramma secondario

Per eliminare un diagramma secondario, visualizzarne il diagramma principale o il diagramma secondario ed effettuare le seguenti operazioni:

1. Selezionare il simbolo del diagramma secondario per eliminarlo.
2. Scegliere l'opzione Elimina dal menu Modifica. In alternativa è possibile premere il tasto `Canc` (`Elimina`).

Se il diagramma principale è espanso nella struttura dello scenario, il nome del diagramma secondario scompare dalla struttura, quando viene eliminato.

Gestione dello stato di completamento del diagramma secondario

L'esecuzione di un diagramma principale risponde allo stato di esecuzione dei suoi diagrammi secondari. Sta allo sviluppatore del diagramma secondario deciderne il

suo stato di completamento, come pure il comportamento nella gestione delle eccezioni. Una collaborazione può intenzionalmente terminare un diagramma secondario in uno dei due seguenti modi:

- Posizionare il nodo Fine corretta al termine del percorso di esecuzione del diagramma secondario per indicare l'esito positivo dell'esecuzione del diagramma secondario.
- Posizionare un nodo Fine con errore al termine del percorso di esecuzione del diagramma secondario per indicare l'esito negativo dell'esecuzione del diagramma secondario.

Gestione di un esecuzione di diagramma secondario riuscita

Il nodo di completamento Fine corretta indica che l'esecuzione è terminata con esito positivo. Quando un diagramma secondario termina con un nodo Fine corretta, l'ambiente di collaborazione runtime termina il diagramma secondario e passa il controllo al diagramma principale. Il flusso del diagramma principale prosegue con il nodo successivo presente dopo il nodo del diagramma secondario. Di solito il nodo successivo è un nodo decisione che testa lo stato del diagramma secondario. Il nodo decisione può includere qualsiasi diramazione, fra quelle riportate di seguito, per testare l'esecuzione del diagramma secondario:

- Una diramazione normale che testa le condizioni booleane che è possibile impostare.

  Se l'esecuzione della collaborazione è in stato Normale, l'ambiente di collaborazione runtime valuta la condizione della diramazione normale. La diramazione predefinita viene eseguita se quella normale non viene risolta in true.
- Le diramazioni eccezione testano delle specifiche eccezioni prodotte.

  Se l'esecuzione della collaborazione è in stato Eccezione, l'ambiente di collaborazione runtime valuta le condizioni delle diramazioni eccezione.

Un diagramma secondario può terminare con esito positivo nei seguenti modi:

- Il diagramma secondario completa la sua attività e *non* rileva alcuna eccezione.

  L'esecuzione della collaborazione è in stato Normale. Quando il controllo passa al diagramma principale ed il nodo successivo è un nodo decisione, l'ambiente di collaborazione runtime valuta la diramazione normale.
- Il diagramma secondario rileva un'eccezione, la gestisce (senza innalzare l'eccezione) ed intenzionalmente termina con esito positivo.

  L'esecuzione della collaborazione è in stato Normale. Quando il controllo passa al diagramma principale ed il nodo successivo è un nodo decisione, l'ambiente di collaborazione runtime valuta la diramazione normale.
- Il diagramma secondario rileva un'eccezione, la gestisce innalzando l'eccezione al diagramma principale ed intenzionalmente termina con esito positivo.

  In questo caso, l'esecuzione della collaborazione è in stato Eccezione. Quando il controllo passa al diagramma principale ed il nodo successivo è un nodo decisione, l'ambiente di collaborazione runtime valuta le eventuali diramazioni eccezione. La diramazione eccezione porta ad un nodo azione che gestisce l'eccezione. Il modo migliore per gestire è di innalzarla di nuovo ed includerne il testo come causa.

  In una collaborazione con più livelli di diagrammi attività è necessario utilizzare esplicitamente il metodo `raiseException()` per innalzare un'eccezione attraverso ciascun livello nel diagramma di attività e fornire il testo originale dell'eccezione nell'elenco dei flussi non risolti. Ogni successiva chiamata `raiseException()` innalza l'eccezione e ne passa il testo originale. Quando l'esecuzione raggiunge il diagramma principale, l'ambiente di collaborazione runtime esegue la sua

operazione di gestione delle eccezioni, ad esempio scrivere dei log o, se la collaborazione è transazionale, avviare il rollback.

Nota: Lo sviluppatore può ottenere un completamento del diagramma secondario con esito positivo anche se incontra un'eccezione, purché la gestisca. Il completamento con esito positivo significa semplicemente che l'esecuzione del diagramma secondario ha raggiunto il simbolo Fine corretta e che è stato disponibile del codice per gestire le eventuali eccezioni rilevate.

Per informazioni relative al modo in cui creare un nodo decisione, consultare "Nodi decisione" a pagina 129. Per ulteriori informazioni su come implementare la gestione delle eccezioni, consultare Capitolo 7, "Gestione delle eccezioni", a pagina 179.

Gestione di un esecuzione di diagramma secondario non riuscita

Il nodo di completamento Fine con errore indica che l'esecuzione *non* è terminata con esito positivo. Quando un diagramma secondario termina con un nodo Fine con errore, l'ambiente di collaborazione runtime termina il diagramma secondario e l'intera collaborazione. Il controllo passa all'ambiente di collaborazione runtime, che crea una voce nella destinazione log della collaborazione e genera un flusso non risolto.

Un diagramma secondario può terminare con esito negativo nei seguenti modi:

- Il diagramma secondario rileva un'eccezione, la gestisce ed intenzionalmente termina con esito negativo. In questo caso, il diagramma secondario deve includere il testo dell'eccezione come causa dell'eccezione.
- Il diagramma secondario rileva un'eccezione non prevista che *non* gestisce ed intenzionalmente termina con esito negativo.

Ogni volte che un diagramma secondario termina con il nodo Fine con errore, l'ambiente di collaborazione runtime termina l'*intera* collaborazione. Per informazioni su come gestire le eccezioni incontrate in un diagramma secondario, consultare "Termine di un diagramma secondario o iteratore con esito positivo" a pagina 183. Per informazioni sul modo in cui l'ambiente runtime di collaborazione crea i flussi non risolti, consultare "Completamento del diagramma principale con esito positivo" a pagina 182.

Iteratori

Un *iteratore* è una forma specializzata di diagramma secondario che implementa loop o iterazioni. Utilizzare un iteratore per eseguire un'operazione:

- Su tutti gli attributi di un oggetto business
- Su tutti gli elementi di un vettore oggetti business

Un iteratore può anche essere utilizzato come loop. Qualsiasi valore richiesto per inizializzare, testare ed incrementare il loop deve essere fornito nella finestra di dialogo Proprietà dell'iteratore.

Un diagramma iteratore può richiamare dei diagrammi secondari o altri iteratori. Per elaborare gli oggetti business gerarchici o i vettori oggetto business gerarchici è necessaria una gerarchia di iteratori.

Quando l'esecuzione di un diagramma principale raggiunge il simbolo Iteratore, il controllo passa al diagramma di attività iteratore. La collaborazione ripete l'esecuzione del diagramma iteratore per ogni attributo contenuto nell'oggetto

business o per ogni oggetto business del vettore di oggetti business. Tramite la variabile iteratore specificata nella finestra di dialogo Proprietà dell'iteratore si ha accesso alla voce presente correntemente nell'iteratore.

Quando l'iteratore termina l'esecuzione, il controllo torna al diagramma principale.

Creazione di un iteratore

Per aggiungere un iteratore al diagramma di attività:

1. Nella barra degli strumenti Simboli, fare clic sul pulsante Iteratore.
2. Fare clic nell'area di lavoro per posizionare il simbolo Iteratore.

   nella struttura dello scenario viene visualizzato un identificativo univoco per l'iteratore, disposto gerarchicamente sotto il diagramma principale. La struttura ad albero dello scenario visualizza il nome nel seguente formato:

   (*UID*)

   Se si fornisce un'etichetta all'iteratore, la struttura dello scenario ne visualizza il nome nel seguente formato:

   etichetta (*UID*)

   L'UID è un identificativo univoco che è anche il nome dell'oggetto iteratore della struttura dello scenario. Come per i UID degli altri simboli, è possibile scegliere se visualizzarlo o meno per l'iteratore. Per attivare e disattivare la visualizzazione dell'UID, utilizzare il menu di scelta rapida del nodo dello scenario nella struttura della maschera.

Creazione delle variabili di iteratore

Un iteratore che utilizza gli attributi di un oggetto business o un vettore di oggetti business deve avere una variabile in ogni iteratore per conservare la voce in fase di elaborazione. Questa variabile di iteratore in realtà è una variabile di scenario che viene creata ed inizializzata nella finestra di dialogo Definizioni scenario. Creare le variabili di iteratore prima di definire le proprietà di un iteratore.

Se l'iteratore utilizza gli attributi di un oggetto business, è possibile utilizzare la variabile Object per conservare l'attributo corrente. Ad esempio, si potrebbe avere la seguente dichiarazione:

```
Object iterAttr = null;
```

Se l'iteratore sta utilizzando degli oggetti business di un vettore, è possibile utilizzare una variabile di tipo BusObj per conservare l'oggetto business corrente. Ad esempio:

```
BusObj iterBusObj = new BusObj("LineItem");
```

Se l'iteratore viene utilizzato come loop, non è necessario creare una variabile iteratore. Il sistema ne crea una automaticamente durante l'elaborazione. La variabile di indice loop può essere richiamata con l'API getCurrentLoopIndex().

Definizione di un iteratore

Una volta che l'iteratore viene visualizzato nel diagramma di attività, è possibile definirne le proprietà nella finestra di dialogo Proprietà dell'iteratore (vedere la Figura 49).

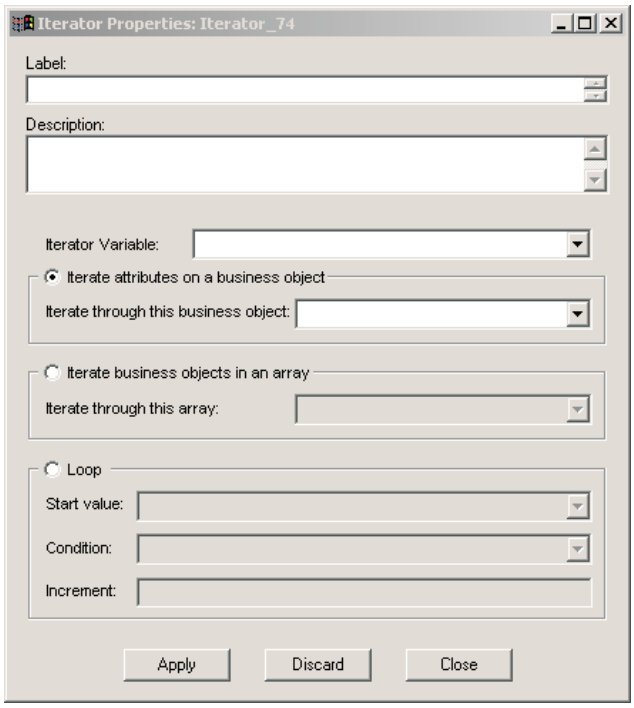


Figura 49. Finestra di dialogo Proprietà dell'iteratore

Aprire la finestra di dialogo Proprietà dell'iteratore facendo clic con il tasto destro del mouse sull'iteratore e selezionando Proprietà dal relativo menu di scelta rapida. E' possibile definire l'etichetta dell'iteratore e fornire una descrizione; entrambe queste proprietà sono facoltative. E' necessario, tuttavia, definire determinate altre proprietà, a seconda del tipo di iteratore che si desidera utilizzare. Le sezioni riportate di seguito descrivono i requisiti per la definizione di ciascun tipo di iteratore.

Utilizzo di un iteratore sugli attributi di un oggetto business

Se si desidera ripetere gli attributi di un oggetto business, effettuare le seguenti operazioni:

1. Utilizzare il campo Variabile iteratore per specificare la variabile che conterrà durante l'iterazione la voce da elaborare. L'elenco a discesa di questo campo contiene tutte le variabili di maschera e di scenario. E' possibile selezionare la variabile dall'elenco o inserirne il nome direttamente nel campo.
2. Fare clic sul pulsante di opzione Ripeti attributi in un oggetto business.
3. Utilizzare il campo Ripeti tramite questo oggetto business per specificare l'oggetto business che contiene gli attributi che si desiderano ripetere. Inserire direttamente il nome nel campo o utilizzare l'elenco a discesa per selezionare l'oggetto business.
4. Fare clic su Applica.

Dopo aver definito le proprietà di un iteratore è necessario modificare il relativo diagramma di attività per definire l'attività che l'iteratore svolge per ogni attributo che elabora. Aprire un diagramma attività di iteratore facendo doppio clic sul nome iteratore nella vista ad albero della maschera di Process Designer Express.

Utilizzo di un iteratore sugli oggetti business di un vettore

Se si desidera ripetere gli oggetti business di un vettore, effettuare le seguenti operazioni:

1. Utilizzare il campo Variabile iteratore per specificare la variabile che conterrà durante l'iterazione la voce da elaborare. L'elenco a discesa di questo campo contiene tutte le variabili di maschera e di scenario. E' possibile selezionare la variabile dall'elenco o inserirne il nome direttamente nel campo.
2. Fare clic sul pulsante di opzione Ripeti oggetti business in un vettore.
3. Utilizzare il campo Ripeti tramite questo vettore per specificare il vettore che si desidera ripetere. Inserire direttamente il nome nel campo o utilizzare l'elenco a discesa per selezionare il vettore.

   Utilizzare la seguente sintassi se si inserisce direttamente il valore nel campo:

   BusinessObjectVariable.AttributeName

   dove *BusinessObjectVariable* è il nome dell'oggetto business principale e *AttributeName* è il nome dell'attributo che rappresenta il vettore degli oggetti business secondari.

   Ad esempio, per ripetere gli oggetti business contenuti in un vettore rappresentato dagli attributi Items in un oggetto business contenuto nella variabile order, digitare `order.Items` nel campo Ripeti tramite questo vettore.
4. Fare clic su Applica.

Dopo aver definito le proprietà di un iteratore è necessario modificare il relativo diagramma di attività per definire l'attività che l'iteratore svolge per ogni oggetto business che elabora. Aprire un diagramma attività di iteratore facendo doppio clic sul nome iteratore nella vista ad albero della maschera di Process Designer Express.

Utilizzo di un iteratore come loop

Per definire un loop, effettuare le seguenti operazioni:

1. Fare clic sul pulsante di opzione Loop.
2. Utilizzare il campo Valore di inizio per specificare il valore iniziale della variabile counter. E' possibile utilizzare l'elenco a discesa per selezionare una variabile che contiene il valore oppure inserire direttamente il valore nel campo.
3. Nel campo Condizione, specificare la condizione che deve essere true perché il loop possa essere eseguito. L'elenco a discesa contiene tutte le variabili boolean definite nella maschera. Selezionare la variabile che si desidera utilizzare ed inserire la condizione direttamente nel campo.
4. Utilizzare il campo Incremento per specificare il metodo per incrementare il valore della variabile counter.
5. Fare clic su Applica.

Dopo aver definito le proprietà di un iteratore è necessario modificare il relativo diagramma di attività per definire l'attività che l'iteratore svolge per ogni attributo o oggetto business che elabora. Aprire un diagramma attività di iteratore facendo doppio clic sul nome iteratore nella vista ad albero della maschera di Process Designer Express.

Aggiunta di un'interruzione

Un'interruzione può essere aggiunta al diagramma di attività di un iteratore per forzare la terminazione prematura dell'iterazione. Quando il percorso di esecuzione dell'iteratore raggiunge il simbolo d'interruzione, l'iteratore termina ed il controllo torna al diagramma principale. Le interruzioni possono essere utilizzate con tutti i tipi di iteratori.

Posizionare un simbolo di interruzione nel diagramma di attività di un iteratore come segue:

1. Fare clic sul pulsante Interrompi nella barra degli strumenti Simboli.
2. Nel diagramma di attività, posizionare il cursore dove si desidera collocare l'interruzione e fare clic. Il simbolo di interruzione viene aggiunto al diagramma.

Figura 50. Simbolo Interruzione

Facoltativamente, ad un simbolo di interruzione è possibile aggiungere un'etichetta ed una descrizione. Fare doppio clic sul simbolo di interruzione nel diagramma di attività per aprire la finestra di dialogo Proprietà dell'interruzione e modificare le proprietà come si desidera.

Utilizzo delle altre funzioni della barra degli strumenti Simboli

La barra degli strumenti Simboli del diagramma contiene un pulsante Seleziona ed un pulsante Testo. Utilizzare il pulsante Testo per aggiungere una casella di testo al diagramma di attività, come descritto in "Utilizzo della funzione Casella di testo". Utilizzare il pulsante Seleziona per annullare un'operazione, come descritto in "Annullamento di un'operazione".

Utilizzo della funzione Casella di testo

E' possibile inserire una casella di testo in un diagramma di attività ed immettervi del testo oppure modificare il testo esistente. Per aggiungere del testo ad un diagramma di attività:

1. Nella barra degli strumenti Simboli, fare clic sul pulsante Testo.
2. Fare clic nell'area di lavoro per posizionare il simbolo Testo.

   Viene visualizzata una casella di testo con la parola "Testo" all'interno della casella.
3. Fare doppio clic sulla parola "Testo" ed inserire il testo desiderato.

E' possibile selezionare e trascinare la casella di testo per collocarla in un punto differente nel diagramma.

Annullamento di un'operazione

Se si cambia idea sull'inserimento di un particolare simbolo in un diagramma di attività, è possibile annullare l'inserimento. Per annullare un'operazione dopo aver fatto clic sul relativo pulsante nella barra degli strumenti Simboli del diagramma, premere il pulsante Seleziona, anch'esso contenuto nella stessa barra Simboli.

Per annullare un collegamento di transizione, utilizzare il tasto Esc. Ogni volta che si preme il tasto Esc viene annullato l'ultimo segmento del collegamento di transizione.

Ottenimento dei valori delle proprietà di configurazione della collaborazione

I nodi azione ed i nodi decisione di solito devono ottenere e calcolare i valori che un implementer o uno sviluppatore configurano per le proprietà di configurazione di un oggetto collaborazione. L'oggetto collaborazione eredita le sue proprietà di configurazione dalla maschera di collaborazione. Per ulteriori informazioni su come definire le proprietà della collaborazione, consultare "Definizione delle proprietà di configurazione della collaborazione (scheda Proprietà)" a pagina 94.

Per ottenere il valore di una proprietà, uno scenario richiama il metodo getConfigProperty(). Se la proprietà è un elenco di valori, uno scenario richiama getConfigPropertyArray(). Per ulteriori informazioni, consultare "Recupero di una proprietà di configurazione collaborazione" a pagina 211.

Utilizzo delle funzioni transazionali

Una **collaborazione transazionale** esegue il rollback, se incontra un errore che ne arresta la sua esecuzione. Perché il rollback abbia esito positivo, ogni scenario deve avere specificata la compensazione per ogni passo di transazione secondaria. Per impostare le proprietà transazionali di una collaborazione, è necessario eseguire i passi descritti nella Tabella 44.

Tabella 44. Definizione di una collaborazione transazionale

Passo di definizione	Descrizione	Per ulteriori informazioni
Assegnare un livello di transazione minimo per la maschera di collaborazione.	La collaborazione transazionale utilizza il livello minimo di transazione per stabilire quando eseguire il rollback della transazione.	"Specifica del livello di transazione minimo" a pagina 86
Specificare le compensazioni per i passi della transazione secondaria.	La collaborazione transazionale utilizza la compensazione definita per le sue transazioni secondarie per eseguire il rollback effettivo.	"Definizione della compensazione" a pagina 140

Termine del percorso di esecuzione

Ogni percorso di esecuzione di un diagramma di attività deve terminare con esito positivo o negativo.

Termine con esito positivo

Terminare la collaborazione con esito positivo significa che il percorso di esecuzione del diagramma di attività ha gestito con esito positivo l'evento di attivazione. L'intera esecuzione ha avuto esito positivo oppure, se si è verificata un'eccezione, il diagramma di attività l'ha gestita in modo tale da consentire alla collaborazione di terminare comunque con esito positivo. Per ulteriori informazioni, consultare Capitolo 7, "Gestione delle eccezioni", a pagina 179.

Per indicare il termine con esito positivo, posizionare il simbolo Fine corretta al termine del percorso di esecuzione. Quando l'ambiente di collaborazione runtime esegue Fine corretta, termina il percorso di esecuzione corrente e passa il controllo al successivo livello superiore di esecuzione (il diagramma principale), se ne esiste uno. Quando un diagramma secondario o iteratore raggiunge il nodo Fine corretta, passa il controllo al diagramma principale. Quando il diagramma di attività principale raggiunge il nodo Fine corretta, il controllo passa all'ambiente di collaborazione runtime, che esegue le proprie azioni di gestione errori.

Aggiunta di un simbolo Fine corretta

Per terminare un percorso di esecuzione con esito positivo, posizionare un simbolo Fine corretta nel diagramma di attività e collegarlo. Per aggiungere un simbolo Fine corretta ad un diagramma di attività:

1. Nella barra degli strumenti Simboli del diagramma, fare clic sul pulsante Fine corretta.
2. Fare clic nell'area di lavoro per posizionare il simbolo Fine corretta.

Definizione del simbolo Fine corretta

Dopo che il simbolo Fine corretta viene visualizzato nel diagramma di attività, è possibile definirne le proprietà nella finestra di dialogo Proprietà di Fine corretta. Le proprietà di un simbolo Fine corretta sono l'etichetta e la descrizione. Entrambe sono facoltative.

Per definire le proprietà di Fine corretta:

1. Visualizzare la finestra di dialogo Proprietà di Fine corretta.

   E' possibile visualizzare questa finestra di dialogo in uno qualunque dei seguenti modi:

   - Fare doppio clic sul nodo Fine corretta selezionato.
   - Fare clic con il tasto destro del mouse per richiamare il menu di scelta rapida e scegliere Proprietà.
   - Scegliere Proprietà dal menu a discesa Modifica.
   - Utilizzare la combinazione di tasti di accesso rapido `Ctrl + Invio`.

   Viene visualizzata la finestra di dialogo Proprietà di Fine corretta.
2. Facoltativamente, specificare l'etichetta e la descrizione di questo nodo azione.

   L'etichetta rende il diagramma di attività più leggibile, apponendo al simbolo Fine corretta del testo che è più descrittivo dell'UID. Il campo di descrizione è un luogo in cui immettere un commento, che descrive lo scopo del simbolo Fine corretta.
3. Fare clic su Applica per salvare le proprietà di Fine corretta. Fare clic su Ignora per annullare le proprietà. Fare clic su Chiudi per annullare questa definizione di Fine corretta.

Termine con esito negativo

Terminare la collaborazione con esito negativo significa che il diagramma di attività non è stato in grado di eseguirsi correttamente e l'esecuzione deve terminare. Per indicare un termine con esito negativo, posizionare il simbolo Fine con errore alla fine del percorso di esecuzione. Quando l'ambiente di collaborazione runtime esegue Fine con errore, termina l'intera collaborazione. Quando un diagramma secondario o un iteratore raggiunge il nodo Fine con errore, termina sia il diagramma secondario (o iteratore) che il diagramma principale. L'ambiente di collaborazione runtime allora esegue le sue azioni di gestione degli errori.

Nota: Il raggiungimento di un simbolo Fine con errore arresta *sempre* l'esecuzione della collaborazione. Terminare con Fine con errore, tuttavia, non implica automaticamente la non riuscita dell'evento di attivazione. Solo se l'esecuzione della collaborazione è in stato Eccezione l'ambiente di collaborazione runtime crea un flusso non risolto per l'evento di attivazione. Per ulteriori informazioni, consultare Capitolo 7, "Gestione delle eccezioni", a pagina 179.

Aggiunta di un simbolo Fine con errore

Per terminare un percorso di esecuzione con esito negativo, posizionare un simbolo Fine con errore nel diagramma di attività e collegarlo. Per aggiungere un simbolo Fine con errore ad un diagramma di attività:

1. Nella barra degli strumenti Simboli del diagramma, fare clic sul pulsante Fine con errore.
2. Fare clic nell'area di lavoro per posizionare il simbolo Fine con errore.

Definizione del simbolo Fine con errore

Dopo che il simbolo Fine con errore viene visualizzato nel diagramma di attività, è possibile definirne le proprietà nella finestra di dialogo Proprietà di Fine con errore. Le proprietà di un simbolo Fine con errore sono l'etichetta e la descrizione. Entrambe sono facoltative.

Per definire le proprietà di Fine con errore:

1. Visualizzare la finestra di dialogo Proprietà di Fine con errore.

   E' possibile visualizzare questa finestra di dialogo in uno qualunque dei seguenti modi:

   - Fare doppio clic sul nodo Fine con errore selezionato.
   - Fare clic con il tasto destro del mouse per richiamare il menu di scelta rapida e scegliere Proprietà.
   - Scegliere Proprietà dal menu a discesa Modifica.
   - Utilizzare la combinazione di tasti di accesso rapido `Ctrl + Invio`.
2. Facoltativamente, specificare l'etichetta e la descrizione di questo nodo azione.

   L'etichetta rende il diagramma di attività più leggibile, apponendo al simbolo Fine con errore del testo che è più descrittivo dell'UID. Il campo di descrizione è un luogo in cui immettere un commento, che descrive lo scopo del simbolo Fine con errore.
3. Fare clic su Applica per salvare le proprietà di Fine con errore. Fare clic su Ignora per annullare le proprietà. Fare clic su Chiudi per annullare questa definizione di Fine con errore.

Altre operazioni del diagramma di attività

Questa sezione fornisce le informazioni sulle seguenti operazioni aggiuntive che Process Designer Express fornisce su un diagramma di attività:

- "Apertura e chiusura dei diagrammi di attività"
- "Documentazione diagramma di attività" a pagina 157
- "Copia di un diagramma di attività" a pagina 157
- "Eliminazione all'interno di un diagramma di attività" a pagina 158

Apertura e chiusura dei diagrammi di attività

Process Designer Express fornisce la possibilità di aprire e chiudere un diagramma di attività.

Apertura di un diagramma di attività

Per aprire un diagramma di attività, è possibile eseguire una delle seguenti azioni:

- Scegliere Apri tutti i diagrammi dal menu Maschera.

  Apri tutti i diagrammi apre una finestra per ogni diagramma di attività definito per la maschera.

- Selezionare uno scenario, un diagramma secondari o un iteratore nella struttura dello scenario:
  - Dal menu Maschera, scegliere Apri diagramma per aprire una finestra per il diagramma di attività associato.
  - Fare doppio clic sul nome dello scenario, del diagramma secondario o dell'iteratore.
- Se il diagramma di attività è un digramma secondario o un iteratore, è possibile aprire il diagramma principale correlato facendo clic con il tasto destro del mouse nell'area di lavoro per visualizzare il menu di scelta rapida e scegliendo Apri diagramma principale.

Chiusura di un diagramma di attività

Per chiudere un diagramma di attività, è possibile eseguire una delle seguenti azioni:

- Scegliere Chiudi tutti i diagrammi dal menu Maschera.

  Quando è aperto almeno un diagramma di attività, Chiudi tutti diagrammi chiude tutte le finestre di diagrammi di attività.
- Selezionare il pulsante Chiudi nella parte superiore della finestra che visualizza lo scenario, il diagramma secondario o l'iteratore.

Documentazione diagramma di attività

Esistono due modi per documentare un diagramma di attività:

- Stampare una rappresentazione grafica di un diagramma, di un diagramma secondario o di un iteratore.
- Salvare il diagramma di attività completo in un file di testo.

Stampa di un diagramma

E' possibile stampare una rappresentazione grafica di ogni diagramma, diagramma secondario ed iteratore. Delle rappresentazioni lunghe si stampano su più pagine. Per stampare, effettuare le seguenti operazioni:

1. Aprire il diagramma — selezionarlo nella struttura ad albero della maschera e: fare clic con il tasto destro del mouse sul nome e fare clic su Apri diagramma dal menu di scelta rapida oppure selezionare Apri diagramma dal menu maschera.
2. Selezionare Stampa dal menu File oppure utilizzare la combinazione di tasti di accesso rapido `Ctrl+P`.

Salvataggio di un diagramma come file di testo

E' possibile salvare l'intero diagramma di attività come file di testo. A tale scopo:

1. Scegliere Salva diagramma come file di testo... dal menu Maschera.
2. Nella finestra di dialogo Salva diagramma come testo, specificare il nome file.

Copia di un diagramma di attività

non è possibile copiare un diagramma di attività intero separato dal suo scenario, ma è possibile copiarne il contenuto. Per copiare il contenuto di un diagramma di attività:

1. Visualizzare il diagramma di attività origine.
2. Selezionare gli oggetti che si desidera copiare.

- Per selezionare alcuni degli oggetti, tenere premuto il tasto sinistro del mouse mentre si sposta il cursore per definire l'area rettangolare che si desidera copiare. Rilasciare il tasto del mouse quando si è terminata la selezione.
- Per selezionare degli oggetti specifici uno alla volta, fare clic su un oggetto e tenendo premuto il tasto `Maiusc` fare clic sugli altri. Con il tasto `Maiusc` premuto, fare clic su un oggetto per rimuoverlo dalla selezione.
- Per selezionare tutti gli oggetti, nel menu Modifica, scegliere Seleziona tutto (tasti di accesso rapido `Ctrl+A`).

3. Nel menu Modifica, scegliere Copia (tasti di accesso rapido `Ctrl+C`).
4. Visualizzare il diagramma di attività di destinazione.
5. nel menu Modifica, scegliere Incolla (tasti di accesso rapido `Ctrl+V`).
6. Rivedere le definizioni dei simboli, se necessario, in modo che i riferimenti a proprietà, porte e variabili siano corretti.

Notare che la copia non è un processo iterativo. Se si copia e incolla un diagramma secondario o un iteratore, Process Designer Express crea un diagramma di attività vuoto nella struttura dello scenario, subordinato al diagramma corrente, per rappresentare il diagramma secondario o il diagramma iteratore. E' necessario copiare e incollare manualmente il contenuto dei diagrammi secondario e iteratore. Se viene selezionato un simbolo di interruzione in un nodo iteratore durante l'operazione di copia, può essere copiato solo in un altro nodo iteratore.

Importante: Non è possibile incollare un simbolo Inizio duplicato in un diagramma di attività. Quando si selezionano dei simboli da copiare e incollare, evitare di copiare il simbolo Inizio in un diagramma che ne contiene già uno. Se per i diagrammi si mantengono le impostazioni Preferenze utente predefinite, Process Designer Express posiziona automaticamente il simbolo Inizio in ogni nuovo diagramma. Per ulteriori informazioni, consultare "Modifica della visualizzazione: preferenze dell'utente" a pagina 201.

Eliminazione all'interno di un diagramma di attività

Per eliminare un simbolo all'interno di un diagramma di attività, selezionare il simbolo da eliminare ed eseguire una delle seguenti azioni:

- Dal menu Modifica scegliere Elimina.
- Premere il tasto `Canc` (`Elimina`).

Per eliminare l'intero diagramma di attività, eliminare lo scenario in cui è definito il diagramma di attività. Per ulteriori informazioni, consultare "Eliminazione di uno scenario" a pagina 104. Per uno scenario con diagrammi secondari o iteratori, è possibile eliminare il diagramma secondario o iteratore eliminando il relativo simbolo dal diagramma principale correlato.

Capitolo 6. Utilizzo di Activity Editor

Activity Editor è un'interfaccia grafica che consente di specificare una logica business di collaborazione senza dover scrivere del codice Java. Questo capitolo descrive l'interfaccia Activity Editor, illustra i componenti nelle definizioni attività, elenca i blocchi funzione supportati utilizzati per creare la logica business e fornisce un esempio di utilizzo di Activity Editor. Questo capitolo contiene le sezioni seguenti:

Avvio di Activity Editor

Avviare Activity Editor da un nodo azione in uno scenario maschera di collaborazione, come indicato di seguito:

1. Selezionare il nodo azione al quale si desidera aggiungere la logica business.
2. Aprire la finestra di dialogo Proprietà azione relativa al nodo azione effettuando una delle seguenti operazioni:
   - Facendo doppio clic sul nodo azione
   - Facendo clic con il tasto destro del mouse sul nodo azione e selezionando Proprietà dal menu di scelta rapida visualizzato
3. Fare clic su Modifica. Activity Editor viene aperto in una nuova finestra.

L'interfaccia Activity Editor

Le seguenti sezioni descrivono le viste, i menu, le barre degli strumenti ed i tasti di accesso rapido che fanno parte dell'interfaccia Activity Editor.

Viste di Activity Editor

Activity Editor ha due tipi di viste per l'aggiunta di frammenti di codice: Grafica e Java. Le sezioni che seguono descrivono ognuna delle due viste.

Vista Grafica di Activity Editor

La vista Grafica di Activity Editor consente di specificare la logica del nodo azione senza dover scrivere del codice Java o solo in minima parte. E' possibile, invece, trascinare i blocchi funzione nell'area di lavoro per rappresentare il flusso della definizione dell'attività. La Figura 51 a pagina 160 illustra la vista Grafica.

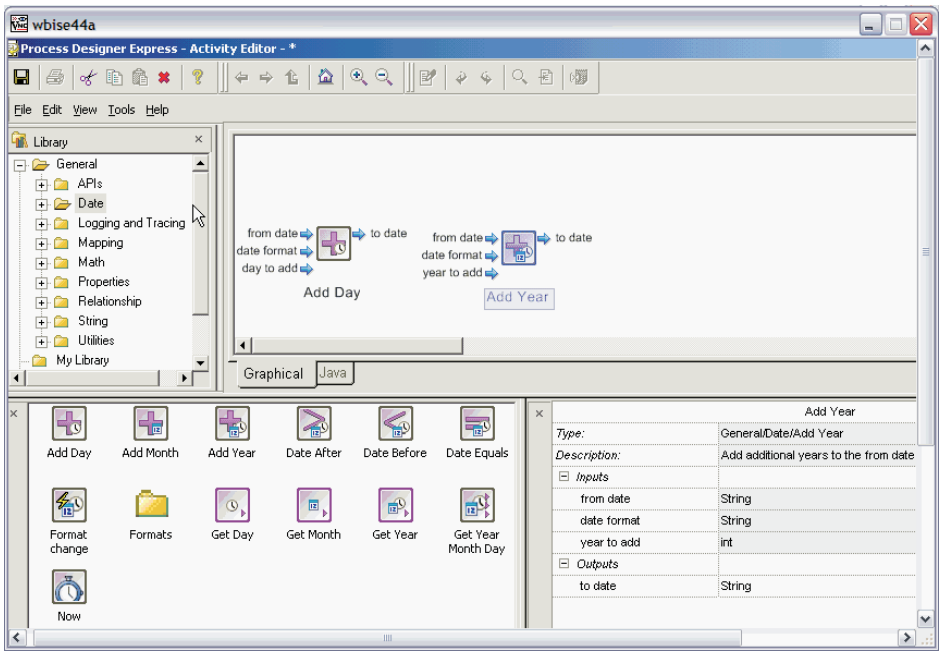


Figura 51. Activity Editor: vista Grafica

Nota: Le dimensioni dell'area di disegno dell'attività grafica sono limitate; tuttavia, è possibile disporre i componenti delle attività orizzontalmente, verticalmente o o diagonalmente per ottenere l'attività grafica desiderata.

La vista Grafica di Activity Editor dispone di quattro diverse finestre: Quaderno attività, Libreria, Contenuto e Proprietà.

- La finestra Quaderno attività — Questa finestra, di solito denominata area di disegno, è il luogo in cui si trascinano i blocchi funzione che formano la logica business del nodo azione.
- Finestra Libreria — Questa finestra contiene una vista ad albero dei blocchi funzione disponibili e, facoltativamente, i gruppi denominati. I blocchi funzione vengono organizzati in cartelle, in base al loro scopo (per ulteriori informazioni consultare "Blocchi funzione supportati" a pagina 168). Inoltre questa finestra contiene le seguenti cartelle:
  - Sistema — Questa cartella contiene gli elementi che possono essere aggiunti all'area di disegno. Gli elementi di sistema comprendono i commenti, le descrizioni, le etichette, le tag attività e le costanti. Per ulteriori informazioni sull'utilizzo di questi componenti, consultare "Blocco funzione Nuova costante" a pagina 167 e "Tag per le definizioni delle attività" a pagina 167).
  - Mia libreria — Questa cartella consente di personalizzare la finestra Libreria.
  - Raccolta personale — Questa cartella consente di creare una raccolta dei componenti utilizzati più frequentemente. In questa cartella è possibile collocare i normali blocchi funzione oppure creare un proprio gruppo di componenti riutilizzabili (per ulteriori informazioni consultare "Gruppi di componenti" a pagina 168).
  - Variabili — Questa cartella contiene le variabili globali disponibili per essere utilizzate nell'attività corrente. Di solito contiene le variabili degli oggetti business della porta, oltre agli altri oggetti business e alle variabili definiti nello scenario.
- Finestra contenuto — Questa finestra contiene un ampio elenco di icone dei blocchi funzione 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'icona blocco funzione nell'area di disegno per creare parte del flusso di attività.

- Finestra Proprietà — Questa finestra visualizza le proprietà del blocco funzione al momento selezionato. Le proprietà vengono visualizzate in una griglia all'interno della finestra. Alcune delle proprietà possono essere modificate direttamente nella finestra Proprietà, mentre altre sono in sola lettura.

Vista Java di Activity Editor

La vista Java può essere utilizzata per aggiungere del tradizionale codice Java ad un nodo azione. Utilizzare questa vista se si preferisce scrivere il codice invece di rappresentarlo graficamente oppure se la vista Grafica non fornisce tutte le funzioni necessarie per il frammento di codice.

Importante

Una volta immesso o modificato il codice nella vista Java, non è possibile utilizzare la vista Grafica. Ulteriore lavoro sul frammento di codice deve essere effettuato nella vista Java di Activity Editor oppure direttamente nella finestra Frammento di codice della finestra di dialogo Proprietà dell'azione.

Per semplificare la lettura e la comprensione del codice del programma, la vista Java visualizza il codice utilizzando uno schema basato sui colori, come descritto nella Tabella 45.

Tabella 45. Schema colori del frammento di codice

Colore	Tipo di codice
Verde	Commento
Rosa	Stringa letterale
Blu	Parola chiave

La Figura 52 a pagina 162 illustra la vista Java di Activity Editor.

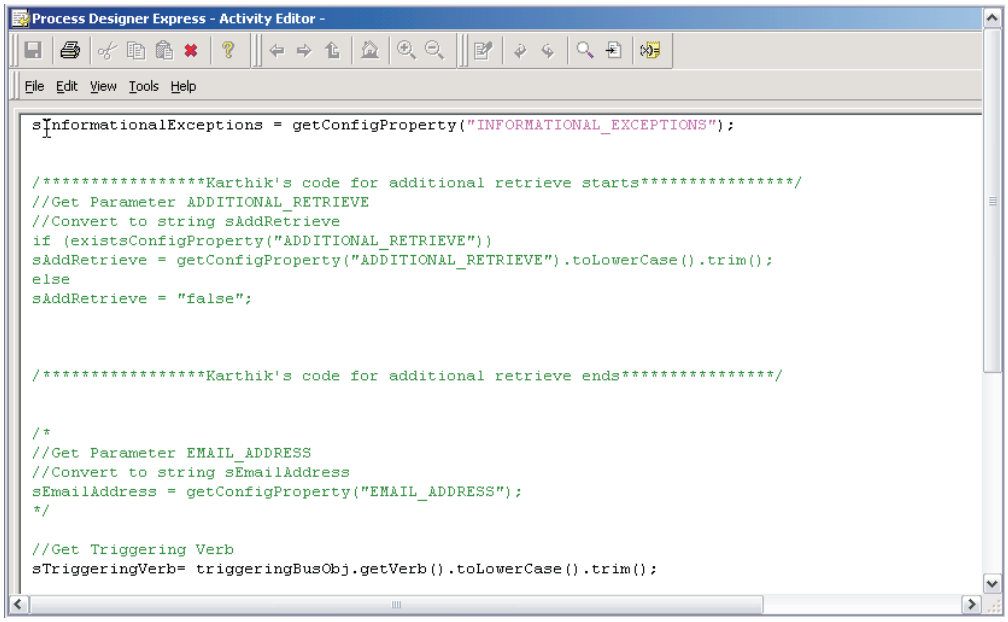


Figura 52. Activity Editor: vista Java

Menu di Activity Editor

Questa sezione descrive le funzioni disponibili dai menu di Activity Editor.

menu File

Il menu File di Activity Editor fornisce le seguenti opzioni:

- Salva — Salva l'attività in Process Designer Express.
- Imposta stampante — Apre la finestra di dialogo Imposta stampante per specificare le opzioni di stampa.
- Anteprima di stampa — cambia l'editor nella modalità di anteprima di stampa. Questa opzione di menu è disponibile solo nella vista Java.
- Stampa — Apre la finestra di dialogo Stampa per la stampa dell'attività corrente. Questa opzione di menu è disponibile solo nella vista Java.
- Chiudi — Chiude l'Activity Editor.

Menu Modifica

Il menu Modifica di Activity Editor fornisce le seguenti opzioni:

- Annulla — Annulla l'ultima modifica apportata e ripristina la versione precedente. Questa opzione di menu è disponibile solo nella vista Java.
- Ripeti — Ripristina una modifica che era stata precedentemente rimossa con il comando **Annulla**. Questa opzione di menu è disponibile solo nella vista Java.
- Taglia — Elimina la voce selezionata. La voce viene copiata negli appunti.
- Copia — Copia la voce selezionata negli appunti.
- Incolla — Inserisce l'oggetto presente al momento negli appunti nell'attività, nel punto in cui si trova il cursore.
- Elimina — Elimina l'oggetto selezionato.
- Seleziona tutto — Seleziona tutti gli oggetti contenuti nell'attività.
- Trova — Trova il testo specificato nell'area di modifica. Questa opzione di menu è disponibile solo nella vista Java.

- Sostituisci — Trova il testo specificato e lo sostituisce con il nuovo testo specificato. Questa opzione di menu è disponibile solo nella vista Java.
- Vai alla riga — Porta il cursore alla riga specificata. Questa opzione di menu è disponibile solo nella vista Java.

Menu Visualizza

Il menu Visualizza di Activity Editor fornisce le seguenti opzioni:

- Modalità progettazione — Alterna le modalità Progettazione e Vista rapida.
- Modalità vista rapida — Alterna le modalità Vista rapida e progettazione.
- Vai a — Apre il seguente menu secondario per lo spostamento all'interno dell'attività:
  - Indietro — Va indietro nella cronologia degli spostamenti.
  - Avanti — Va in avanti nella cronologia degli spostamenti.
  - Livello superiore — Visualizza il diagramma da un livello superiore.
  - Inizio — Porta al diagramma principale della vista Grafica.
- Zoom avanti — Ingrandisce il contenuto nell'editor.
- Zoom indietro — Riduce il contenuto nell'editor.
- Zoom — 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 strumenti — Apre il seguente menu secondario per la visualizzazione e la chiusura delle barre strumenti:
  - Standard — Attiva e disattiva la barra strumenti Standard.
  - Grafici — Attiva e disattiva la barra strumenti Grafici.
- Barra di stato — Attiva e disattiva la barra di stato.
- Preferenze — Apre la finestra di dialogo Preferenze per specificare il comportamento predefinito di Activity Editor.

Menu Strumenti

Il menu Strumenti di Activity Editor fornisce la seguente opzione:

- Converti — Converte l'attività corrente in codice Java ed apre la vista Java.

Menu Guida

Il menu Guida di Activity Editor fornisce le seguenti opzioni:

- Argomenti della guida — Apre gli argomenti della guida.
- Documentazione — Apre la documentazione di InterChange Server Express.

Menu di scelta rapida

Si accede al menu di scelta rapida di Activity Editor facendo clic con il tasto destro del mouse sull'area di disegno. Fornisce le seguenti opzioni:

- Nuova costante — Crea un nuovo componente 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 all'attività.

- Aggiungi a Raccolta personale — Crea un nuovo gruppo di componenti da riutilizzare nella finestra Libreria.

Barre degli strumenti di Activity Editor

Activity Editor dispone di due barre degli strumenti: la barra Standard e la barra Grafici. La barra strumenti Standard fornisce le funzioni di salvataggio e stampa delle attività, il taglio, la copia, l'inserimento e l'eliminazione di elementi nell'attività e l'accesso alla guida.

La barra strumenti Grafici fornisce le funzioni di spostamento all'interno delle attività. I pulsanti corrispondono alle voci di menu Visualizza —> Zoom e Visualizza —> Vai a.

Tasti di accesso rapido di Activity Editor

La Tabella 46 elenca le voci di menu di Activity Editor ed i relativi tasti di accesso rapido ad esse associati.

Tabella 46. Tasti di accesso rapido di Activity Editor

Menu	**Voce del menu**	**Tasto di accesso rapido**
Menu File	Salva	Ctrl+S
	Imposta stampante	Ctrl+Maiusc+P
	Anteprima di stampa	Non è disponibile un collegamento
	Stampa	Ctrl+P
	Chiudi	Non è disponibile un collegamento
Modifica	Taglia	Ctrl+X
	Copia	Ctrl+C
	Incolla	Ctrl+P
	Elimina	Canc
	Seleziona tutto	Ctrl+A
	Trova:	Ctrl+F
	Vai alla riga	Ctrl+G

Tabella 46. Tasti di accesso rapido di Activity Editor (Continua)

Menu	Voce del menu	Tasto di accesso rapido
Visualizza	Modalità progettazione	Non è disponibile un collegamento
	Modalità vista rapida	Non è disponibile un collegamento
	Vai a/Indietro	Alt+Freccia sinistra
	Vai a/Freccia destra	Alt+Freccia destra
	Vai a/Livello superiore	Non è disponibile un collegamento
	Vai a/Inizio	Alt+Home
	Zoom avanti	Ctrl++
	Zoom indietro	Ctrl+ -
	Zoom	Ctrl+M
	Finestra Libreria	Non è disponibile un collegamento
	Finestra Contenuto	Non è disponibile un collegamento
	Finestra Proprietà	Non è disponibile un collegamento
	Barre degli strumenti	Non è disponibile un collegamento
	Barra di stato	Non è disponibile un collegamento
	Preferenze	Ctrl+U
Strumenti	Converti	Ctrl+T
Guida	Argomenti della guida	F1
	Documentazione	Non è disponibile un collegamento

Definizioni delle attività

Activity Editor viene utilizzato per creare delle *definizioni di attività*, che specificano la logica business di ciascun nodo azione nella maschera di collaborazione. Ogni nodo azione ha associata una definizione di attività.

Blocchi funzione

Una definizione di attività si basa sui *blocchi funzione*. Un blocco funzione rappresenta una discreta parte di una definizione di attività, ad esempio una costante, una variabile o una determinata parte di una funzione (come un metodo di programmazione). Molti dei blocchi funzione in Activity Editor corrispondono a dei metodi individuali nell'API della collaborazione.

I blocchi funzione vengono posizionati nell'area di disegno trascinandoli dalla finestra Libreria o dalla finestra Contenuto. Una volta collocato nell'area di disegno, è possibile spostare il blocco funzione dove si desidera. Fare semplicemente clic sull'icona del blocco funzione nell'area di disegno per selezionarla e trascinarla nella posizione desiderata.

I blocchi funzione possono contenere 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 il blocco funzione viene trascinato nell'area di disegno, le relative porte di input e output sono rappresentate da delle frecce, come mostra la Figura 53.

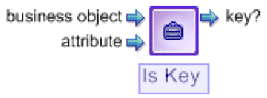


Figura 53. Un blocco funzione con le porte di input ed output

Queste porte servono da punti di connessione per il collegamento 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 (è possibile utilizzare l'opzione Visualizza —> Preferenze per nascondere i nomi).

Blocchi funzione dei servizi web

Qualsiasi servizio Web registrato può essere esportato da System Manager per essere utilizzato in Activity Editor. Il processo di esportazione converte ogni metodo del servizio Web in un blocco funzione, che viene quindi collocato nella cartella Mia libreria\Servizi web. Dopo essere stati esportati e convertiti in blocchi funzione, i metodi dei servizi web possono essere utilizzati nelle definizioni di attività. Utilizzarli nello stesso modo degli altri blocchi funzione — trascinarli nell'area di disegno e specificare gli input e gli output richiesti.

Per esportare un servizio Web e convertirne i metodi in blocchi funzione, effettuare le seguenti operazioni:

1. Verificare che System Manager sia aperto.
2. Individuare il servizio Web che si desidera esportare dalla ICL (Integration Component Library) appropriata.
3. Fare clic con il tasto destro del mouse sul nome del servizio Web per richiamare il menu di scelta rapida.
4. Fare clic su **Esporta in Activity Editor** dal menu di scelta rapida. Viene visualizzata la finestra di dialogo Esporta in Activity Editor.
5. Selezionare la casella di controllo accanto al servizio o servizi Web che si desidera esportare e fare clic su **Fine**. I metodi del servizio Web vengono convertiti in blocchi funzione ed esportati in Activity Editor.
6. Se si ha una sessione Activity Editor aperta, è necessario chiuderla e riavviare Activity Editor per rendere accessibili i nuovi blocchi funzione.

Collegamenti di connessione

I blocchi funzione sono collegati tramite dei *collegamenti di connessione*. I collegamenti di connessione definiscono il flusso di attività fra i vari componenti contenuti nella definizione di attività. Collegano la porta di output di un blocco funzione alla porta di input di un altro blocco funzione.

Nota: Le porte di output possono collegarsi a più collegamenti di connessione mentre le porte di input possono accettare solo un unico collegamento di connessione.

Per aggiungere un collegamento d connessione fra due blocchi funzione eseguire i seguenti passi:

- Fare clic e tenere premuto il tasto sinistro del mouse sulla porta di output del primo blocco funzione (blocco funzione A).
- Tenendo premuto il tasto sinistro del mouse, portare il cursore sulla porta di input del secondo blocco funzione (blocco funzione B).
- Rilasciare il tasto del mouse. E' stato posizionato un collegamento di connessione fra i due blocchi funzione. Viene rappresentato graficamente da una riga angolata a destra fra i due componenti.

Se una porta di input ha già un collegamento di connessione, viene sostituito dal nuovo collegamento.

Blocco funzione Nuova costante

Activity Editor dispone di una funzione Nuova costante che può essere trascinata nell'area di disegno per definire un valore costante che viene impostato ed utilizzato come input per altri blocchi funzione.

Il blocco funzione Nuova costante si trova nella cartella Sistema nelle finestre Libreria e Contenuto. La Figura 54 mostra l'aspetto del blocco funzione Nuova costante quando viene trascinato nell'area di disegno.

Figura 54. Blocco di funzione Nuova costante

La casella di modifica testo è visualizzata nella parte superiore del blocco funzione, in modo da poter inserire il valore della costante. (se si deve modificare il valore, fare clic all'interno del blocco funzione della costante ed immettere il nuovo valore). Notare che la costante contiene un'unica porta di output.

Nota: Il blocco funzione Costante è l'unico componente di definizione attività che accetta solo una singola riga per il valore. La costante viene convertita in un oggetto stringa Java ed il sistema non può convertire un valore costante di più righe. Se è assolutamente necessario un input su più righe, utilizzare la convenzione di programmazione ″\n″ per separare le righe nella costante. (Ad esempio, il valore ″line1\nline2″ indica che il sistema deve emettere il valore di output in due righe).

Tag per le definizioni delle attività

La cartella Sistema (che si trova nelle finestre Libreria e Contenuto) contiene i blocchi funzione per aggiungere alla definizione dell'attività un commento, una descrizione, un'etichetta e delle tag di attività. Queste tag facilitano l'identificazione di ogni attività o attività secondaria oppure servono da promemoria di 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 ed è possibile immettere il testo. Le tag mandano automaticamente a capo il testo delle righe troppo lunghe. E' anche possibile premere Invio, per immettere il testo in una nuova riga.

Se si desidera modificare le dimensioni di una tag, fare clic con il tasto sinistro del mouse sull'angolo in basso a destra della tag e, tenendo premuto il tasto, trascinare la tag fino alla dimensione desiderata. Le tag hanno un requisito minimo di dimensione e non possono essere ridotte oltre la dimensione minima.

Gruppi di componenti

E' possibile raggruppare una serie di blocchi funzione nell'area di disegno e salvarli per un utilizzo successivo in un'altra definizione di attività. In effetti, questo gruppo di componenti agisce come un unico blocco funzione.

Dopo aver creato il flusso di attività desiderato nell'area di disegno, eseguire i seguenti passi per salvare tutto il flusso o parte di esso come gruppo di componenti riutilizzabile:

1. Selezionare i blocchi funzione che si desidera raggruppare. Per selezionare più blocchi funzione, tenere premuto il tasto Ctrl.
2. Fare clic con il tasto destro del mouse sull'area di disegno per aprire il menu di scelta rapida.
3. Fare clic su Aggiungi a Raccolta personale. Viene visualizzata la finestra di dialogo Aggiungi a Raccolta personale.
4. Immettere un nome e (facoltativamente) una descrizione per il gruppo di componenti che si sta creando.
5. Selezionare l'icona che si desidera utilizzare per rappresentare il gruppo di componenti e fare clic su OK.

L'icona del nuovo gruppo di componenti viene aggiunta alla cartella Raccolta personale nelle finestre Libreria e Contenuto. E' possibile trascinare l'icona nell'area di disegno di qualsiasi definizione di attività all'interno del proprio scenario di collaborazione.

Blocchi funzione supportati

I blocchi funzione di Activity Editor sono organizzati nella cartella Generale della finestra Libreria e nelle corrispondenti cartelle della finestra Contenuto. La Tabella 47 descrive come sono organizzati i blocchi funzione.

Tabella 47. Organizzazione dei blocchi funzione

Cartella dei blocchi funzione	Descrizione	Per ulteriori informazioni
Generale\API\Oggetto business	Blocchi funzione da utilizzare con gli oggetti business.	Capitolo 11, "Blocchi funzione di oggetto business", a pagina 255
Generale\API\Oggetto business\Vettore	Blocchi funzione da utilizzare con i vettori Java nella classe BusObj.	Capitolo 11, "Blocchi funzione di oggetto business", a pagina 255
Generale\API\Oggetto business\Costanti	Blocchi funzione da utilizzare con le costanti Java nella classe BusObj.	Capitolo 11, "Blocchi funzione di oggetto business", a pagina 255
Generale\API\Vettore oggetti business	Blocchi funzione da utilizzare con i vettori dell'oggetto business.	Capitolo 12, "Blocchi funzione di vettore oggetto business", a pagina 281

Tabella 47. Organizzazione dei blocchi funzione (Continua)

Cartella dei blocchi funzione	Descrizione	Per ulteriori informazioni
Generale\API\Connessione database	Blocchi funzione per la creazione e la gestione di una connessione database.	Capitolo 14, "Blocchi funzione di connessione database", a pagina 309
Generale\API\Relazione d'identità	Blocchi funzione da utilizzare con le relazioni d'identità.	*Guida allo sviluppo delle mappe*
Generale\API\Mappe	Blocchi funzione per effettuare query ed impostare valori runtime necessari per l'esecuzione di una mappa.	*Guida allo sviluppo delle mappe*
Generale\API\Mappe\Costanti	Costanti dei blocchi funzione.	*Guida allo sviluppo delle mappe*
Generale\API\Mappe\Eccezione	Blocchi funzione per la creazione di nuovi oggetti eccezione in una mappa.	*Guida allo sviluppo delle mappe*
Generale\API\Partecipante	Blocchi funzione per l'impostazione ed il richiamo di valori dei partecipanti nelle relazioni d'identità.	*Guida allo sviluppo delle mappe*
Generale\API\Partecipante\Vettore	Blocchi funzione per la creazione e l'utilizzo di vettori di partecipanti.	*Guida allo sviluppo delle mappe*
Generale\API\Partecipante\Costanti	Costanti dei blocchi funzione da utilizzare con i partecipanti.	*Guida allo sviluppo delle mappe*
Generale\API\Relazione	Blocchi funzione per la manipolazione di istanze runtime delle relazioni.	*Guida allo sviluppo delle mappe*
Generale\Data	Blocchi funzione da utilizzare con le date.	Capitolo 18, "Blocchi funzione di data", a pagina 335
Generale\Data\Formati	Blocchi funzione per specificare i diversi formati di data.	Capitolo 18, "Blocchi funzione di data", a pagina 335
Generale\Registro e traccia	Blocchi funzione per la gestione dei messaggi di log e di traccia.	Capitolo 19, "Blocchi funzione di registro e traccia", a pagina 341
Generale\Registro e traccia\Errore log	Blocchi funzione per la formattazione dei messaggi di errore.	Capitolo 19, "Blocchi funzione di registro e traccia", a pagina 341
Generale\Registro e traccia\Informazioni log	Blocchi funzione per la formattazione dei messaggi informativi.	Capitolo 19, "Blocchi funzione di registro e traccia", a pagina 341
Generale\Registro e traccia\Avviso log	Blocchi funzione per la formattazione dei messaggi di avvertenza.	Capitolo 19, "Blocchi funzione di registro e traccia", a pagina 341
Generale\Registro e traccia\Traccia	Blocchi funzione per la formattazione dei messaggi di traccia.	Capitolo 19, "Blocchi funzione di registro e traccia", a pagina 341

Tabella 47. Organizzazione dei blocchi funzione (Continua)

Cartella dei blocchi funzione	Descrizione	Per ulteriori informazioni
Generale\Mappatura	Blocchi funzione per l'esecuzione di mappe in un contesto specifico.	*Guida allo sviluppo delle mappe*
Generale\Matematica	Blocchi funzione per le attività matematiche di base.	*Guida allo sviluppo delle mappe*
Generale\Proprietà	Blocchi funzione per il richiamo dei valori delle proprietà di configurazione.	*Guida allo sviluppo delle mappe*
Generale\Relazione	Blocchi funzione per la gestione e le query delle relazioni d'identità.	*Guida allo sviluppo delle mappe*
Generale\Stringa	Blocchi funzione per la manipolazione degli oggetti Stringa.	Capitolo 20, "Blocchi funzione di stringa", a pagina 349
Generale\Utilità	Blocchi funzione per la produzione e la rilevazione di eccezioni, per la creazione di loop, per lo spostamento di attributi e per l'impostazione di condizioni.	Capitolo 21, "Blocchi funzione di utilità", a pagina 357
Generale\Utilità\Vettore	Blocchi funzione da utilizzare con gli oggetti Vettore.	Capitolo 21, "Blocchi funzione di utilità", a pagina 357

Esempio: modifica di un formato di data

Questo esempio descrive come utilizzare Activity Editor per modificare il formato della data dell'attributo origine ed assegnare il valore riformattato all'attributo di destinazione. In questo esempio, l'attributo origine è QuoteSchedule.ExpireDate e l'attributo do destinazione è Invoice.PostingDate. Il formato della data originale è aaaaMMgg ed il formato aggiornato della data è aaaa-MM-gg. Questo esempio presuppone che gli oggetti business e gli attributi siano stati già creati e dichiarati nello scenario della maschera di collaborazione.

Sono necessari i seguenti passi per modificare il formato della data dell'attributo origine e quindi assegnarlo all'attributo di destinazione:

1. Verificare che Activity Editor sia aperto.
2. Trascinare il blocco funzione della variabile QuoteSchedule.ExpireDate nell'area di disegno. (I blocchi funzione che rappresentano gli oggetti business, gli attributi e le variabili disponibili in uno scenario si trovano nella cartella Variabili delle finestre Libreria e Contenuto).
3. Trascinare il blocco funzione Modifica formato nell'area di disegno a destra del blocco funzione QuoteSchedule.ExpireDate, come illustrato nella Figura 55 a pagina 171.

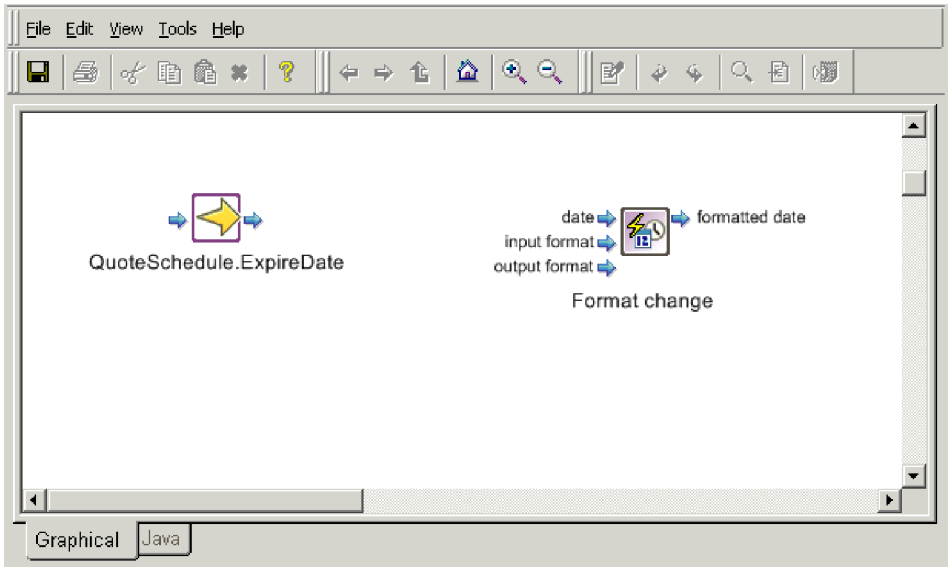


Figura 55. Posizionamento del blocco funzione Modifica formato

4. Creare un collegamento di connessione fra la porta di output del blocco funzione QuoteSchedule.ExpireDate e l'input Data del blocco funzione Modifica formato.
5. Trascinare la costante del blocco funzione aaaaMMgg nell'area di disegno, posizionandola sotto i blocchi funzione QuoteSchedule.ExpireDate e Modifica formato. Questo blocco funzione rappresenta il formato corrente dell'attributo QuoteSchedule.ExpireDate.
6. Creare un collegamento di connessione fra la porta di output del blocco funzione aaaaMMgg e l'input Formato input del blocco funzione Modifica formato, come illustrato nella Figura 58 a pagina 172.

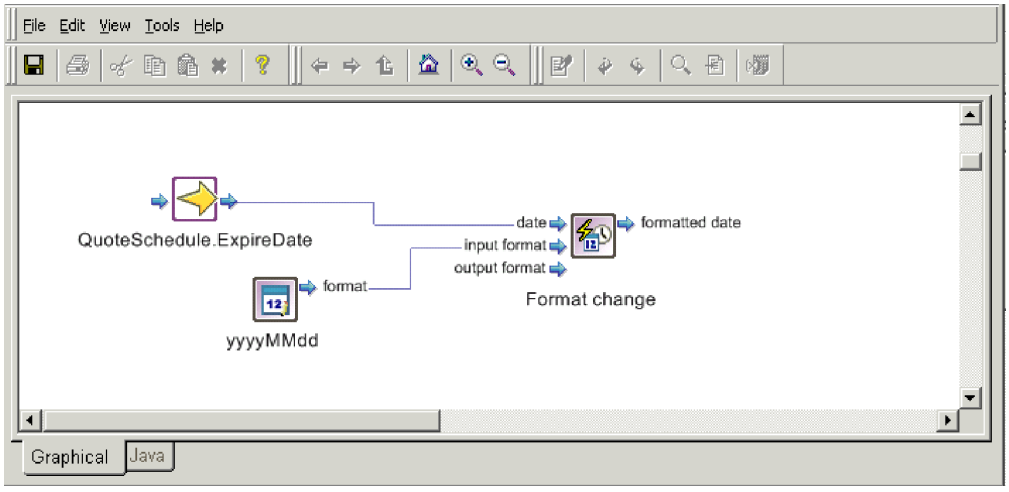


Figura 56. Specifica del formato data di input

7. Trascinare la costante del blocco funzione aaaa-MM-gg nell'area di disegno, posizionandola accanto al blocco funzione aaaaMMgg. Questo blocco funzione rappresenta il nuovo formato dell'attributo QuoteSchedule.ExpireDate.

8. Creare un collegamento di connessione fra la porta di output del blocco funzione aaaa-MM-gg e l'input del formato output del blocco funzione Modifica formato, come illustrato nella Figura 58.

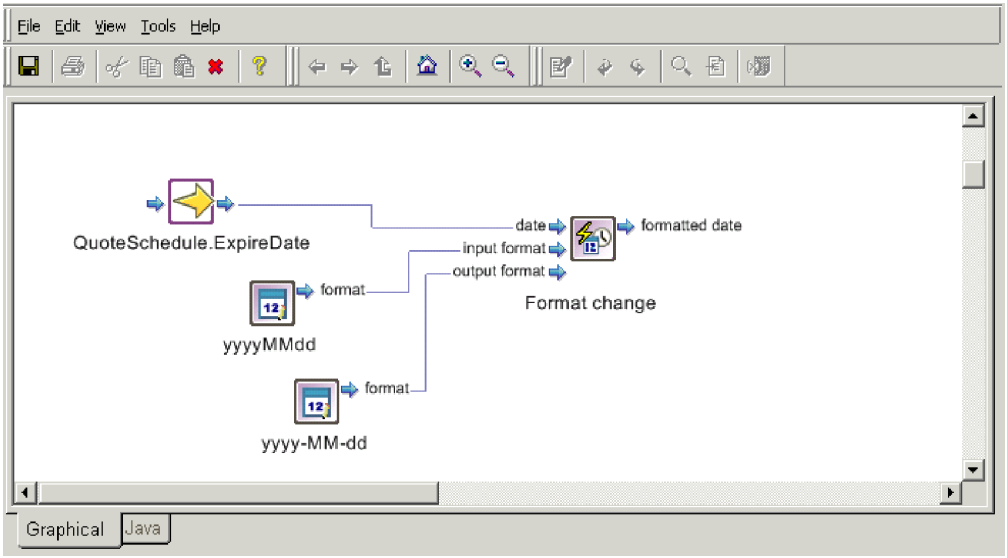


Figura 57. Specifica del formato data di output

9. Trascinare il blocco funzione Invoice.PostingDate nell'area di disegno; questo è l'attributo di destinazione. Posizionarlo a destra del blocco funzione Modifica formato.
10. Per assegnare l'output del blocco funzione Modifica formato all'attributo Invoice.PostingDate, creare un collegamento di connessione fra la porta di output del blocco funzione Modifica formato e la porta di input di Invoice.PostingDate, come illustrato nella Figura 58.

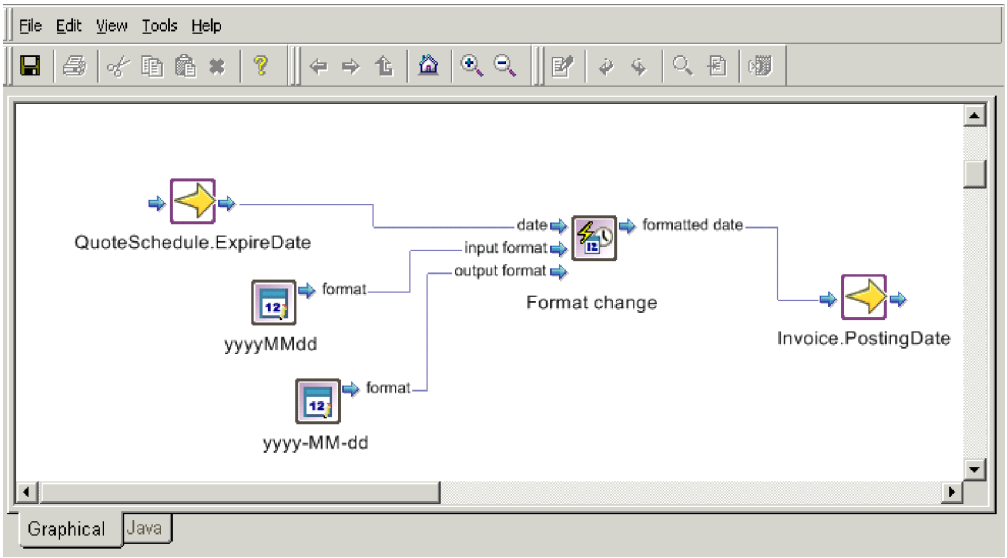


Figura 58. Assegnazione dell'output all'attributo di destinazione

11. Salvare la definizione dell'attività facendo clic su File —> Salva.

Esempio: creazione di un oggetto business duplicato

Il seguente esempio descrive come duplicare un oggetto business. In questo esempio, l'oggetto originale è l'oggetto business di attivazione (triggeringBusObj) ed il duplicato sarà l'oggetto business per una variabile di porta denominata inPort (inPortBusObj). Entrambi gli oggetti business si trovano nella cartella Variabili delle finestre Libreria e Contenuto.

Per creare l'oggetto business duplicato di questo esempio:

1. Verificare che Activity Editor sia aperto.
2. Trascinare la variabile triggeringBusObj nell'area di disegno.
3. Trascinare il blocco funzione Duplicato nell'area di disegno, posizionandolo a destra del blocco funzione triggeringBusObj.

   Nota: Il blocco funzione Duplicato si trova nella cartella Generale\API\Oggetto business nelle finestre Libreria e Contenuto.
4. Creare un collegamento di connessione fra la porta di output della variabile triggeringBusObj e l'input Originale del blocco funzione Duplicato.
5. Trascinare la variabile inPortBusObj nell'area di disegno, posizionandola a destra del blocco funzione Duplicato.
6. Per assegnare il valore dell'oggetto business originale al nuovo oggetto business, creare un collegamento di connessione fra l'output duplicato del blocco funzione Duplicato e l'input della variabile inPortBusObj.

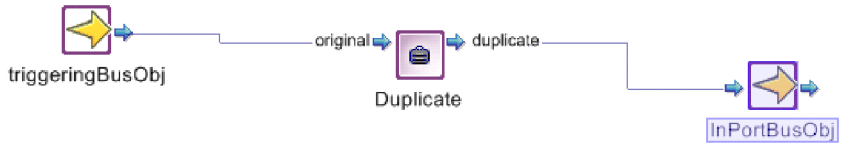


Figura 59. Duplicazione di un oggetto business

7. Salvare la definizione dell'attività facendo clic su File —> Salva.

Importazione di pacchetti Java ed altro codice personalizzato

Process Designer Express fornisce un modo per importare i pacchetti Java ed altro codice personalizzato, come descritto nelle sezioni che seguono.

Importazione di librerie Jar come blocchi funzione di attività

Oltre ad utilizzare i blocchi funzione standard forniti da Activity Editor, Process Designer Express consente di importare una propria libreria Java da utilizzare come blocchi funzione in Activity Editor. L'importazione di librerie Jar personalizzate nelle impostazioni delle attività consentirà l'utilizzo di qualsiasi metodo pubblico contenuto nella libreria Jar come blocchi funzione in Activity Editor.

Passi per l'importazione di librerie Jar come blocchi funzione di attività

Prima di iniziare: esportare le classi Java in un file .jar.

Per importare una libreria Jar in Activity Editor:

1. Aprire la vista Impostazioni attività in System Manager facendo clic su Finestra > Mostra vista > Altro e selezionando Impostazioni attività dalla categoria System Manager.
2. Fare clic con il tasto destro del mouse su Librerie BuildBlock e selezionare Aggiungi libreria. La Figura 60 mostra la vista Impostazioni attività per l'aggiunta di una libreria Jar personalizzata.

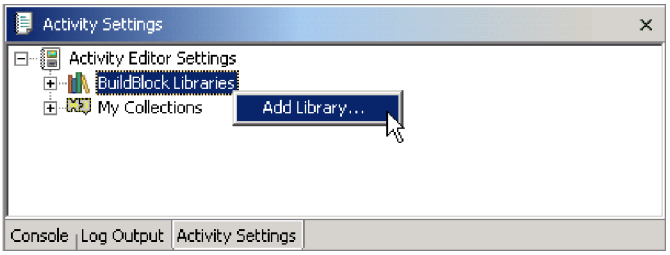


Figura 60. Vista Impostazioni attività

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 in Librerie BuildBlock nella vista Impostazioni attività.

   Suggerimento: Dopo aver importato i propri file .jar personalizzati in Impostazioni attività, quando verranno compilate le 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 suo CLASSPATH includa il file custom.jar. Per informazioni sulla configurazione di InterChange Server Express per l'importazione di file .jar personalizzati, consultare "Importazione di classi fornite da terzi in InterChange Server Express" a pagina 176.
4. Riavviare Process Designer Express.

   Regola: Dopo aver modificato una qualsiasi impostazione della vista Impostazioni attività, è necessario riavviare Process Designer Express perché la modifica diventi attiva in Activity Editor.

Risultato: Quando si apre Activity Editor, la libreria Jar sarà elencata nella finestra Libreria, sotto Mia libreria in Activity Editor. Per impostazione predefinita, i blocchi funzione disponibili personalizzati vengono elencati in base alla struttura del relativo pacchetto. E' possibile utilizzarli in un'attività allo 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 modificare le proprietà della libreria Jar:

- Visualizzare la finestra Proprietà della libreria Jar personalizzata facendo clic con il tasto destro del mouse sulla libreria Jar personalizzata elencata in 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 e nel pacchetto Java 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 Mia libreria 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 Mia libreria. Questa opzione di solito è utile quando i metodi Java della libreria Jar personalizzata sono contenuti in una classe Java che si trova in un pacchetto nidificata sotto molti livelli. Abilitando questa opzione la struttura ad albero secondaria di Mia libreria viene organizzata meglio in Activity Editor.

La Figura 61 mostra la finestra di dialogo per la personalizzazione della visualizzazione della libreria Jar.

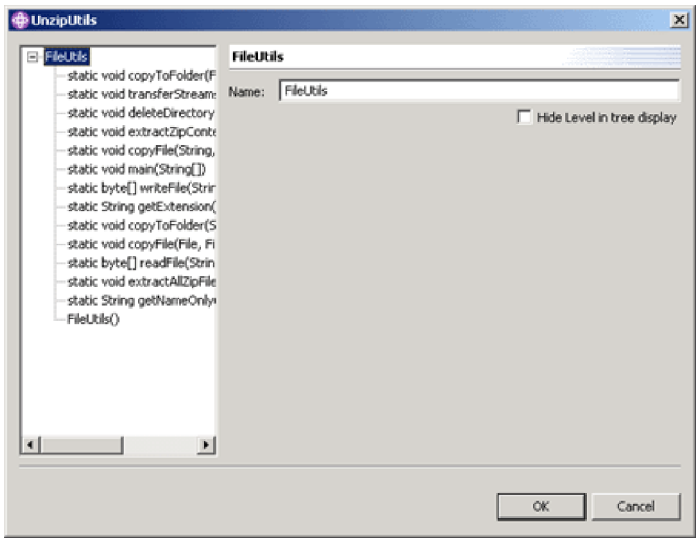


Figura 61. Finestra di dialogo Proprietà per la personalizzazione della visualizzazione della libreria Jar

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: Quando si importa un'icona da utilizzare per il blocco funzione, deve essere di 32 pixel per 32 pixel come dimensione e deve essere in formato .bmp. La profondità del colore dell'icona può essere fino a 24 bit.

La Figura 62 mostra la finestra di dialogo Proprietà per la personalizzazione della visualizzazione dei blocchi funzione della libreria Jar.

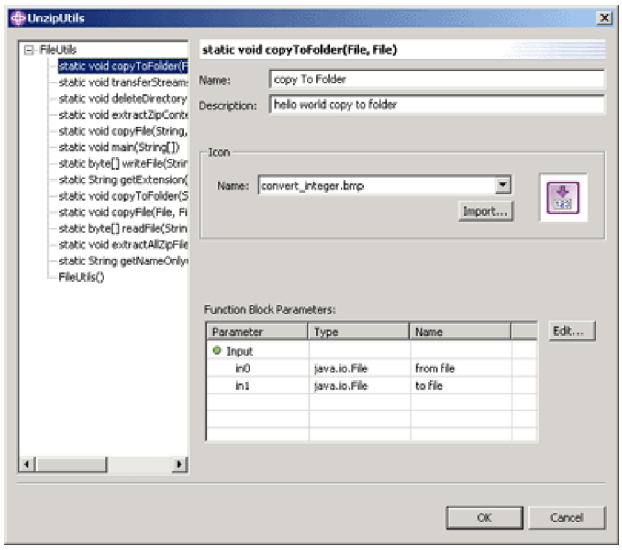


Figura 62. Finestra di dialogo Proprietà per la personalizzazione della visualizzazione dei blocchi funzione della libreria Jar

Regola: dopo qualsiasi modifica alle impostazioni della vista Impostazioni attività è necessario riavviare Process Designer Express perché la modifica diventi attiva in Activity Editor.

Importazione di classi fornite da terzi in InterChange Server Express

Se le classi importate sono contenute in un pacchetto fornito da terze parti, piuttosto che nel pacchetto JDK, per configurare la compilazione del server è necessario aggiungerle al percorso delle classi importate nella variabile JCLASSES.

Raccomandazione: Utilizzare qualche tipo di meccanismo per differenziare in JCLASSES le classi standard da quelle personalizzate.

Esempio: è possibile creare una nuova variabile per contenere solo le classi personalizzate ed aggiungerla JCLASSES effettuando le seguenti operazioni:

1. Creare una nuova proprietà, ad esempio DEPENDENCIES.
2. Collocare il file CwMacroUtils.jar in una sua propria directory.

   Esempio: Creare una directory dependencies nella 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 nella 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="%ProductDirS%"\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 modificare la preferenza Compilatore Classpath:

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 della preferenza 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 ricevere un messaggio di errore che indica che il software non ha trovato la classe personalizzata. Se questo si verifica:

- 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 di 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, include il percorso %ProductDir%\lib.

Capitolo 7. Gestione delle eccezioni

Un' *eccezione* rappresenta una situazione di errore che, se non gestita esplicitamente nel diagramma delle attività, può interrompere l'esecuzione della collaborazione. L'obiettivo della gestione delle eccezioni è di assicurare quanto segue:

- Se possibile, la condizione di errore che ha causato l'eccezione viene corretta o ne viene ridotto l'ambito, in modo da poter continuare l'esecuzione della collaborazione.
- Se la condizione di errore non può essere corretta e lo scenario deve terminare con esito negativo, la collaborazione deve essere interrotta. In questo caso, la collaborazione deve fornire più informazioni possibile sulla causa della condizione di errore. Queste informazioni sono utili all'amministratore per stabilire come correggere l'istanza di errore e prevenire ulteriori future ricorrenze dell'errore.

Pertanto è importante capire come vengono gestite le eccezioni, sia nella maschera della collaborazione che nell'ambiente di collaborazione runtime. Questa sezione fornisce le seguenti informazioni sulla gestione delle eccezioni:

Descrizone dell'eccezione di collaborazione

L'API della collaborazione fornisce un *oggetto eccezione* per rappresentare un'eccezione che si verifica in una collaborazione. Come mostra la Tabella 48, questo oggetto eccezione contiene le informazioni sulla condizione che ha causato l'eccezione.

Tabella 48. L'oggetto eccezione CollaborationException

Tipo di eccezione
Tipo secondario di eccezione
Messaggio
Numero messaggio

Questo oggetto eccezione è un'istanza della classe `CollaborationException`, che è un'estensione della classe Java `Exception`. La Tabella 49 mostra i metodi accessor che la classe `CollaborationException` fornisce per ottenere le informazioni nell'oggetto eccezione.

Tabella 49. Informazioni contenute oggetto eccezione

Membro	**Metodo del dispositivo di accesso**
Tipo di eccezione	`getType()`
Tipo secondario di eccezione	`getSubType()`
Testo del messaggio	`getMessage()`, `toString()`
Numero messaggio	`getMsgNumber()`

Nota: Quando si verifica un'eccezione, l'ambiente di collaborazione runtime popola una variabile di sistema denominata `currentException` con le

informazioni sull'eccezione. La variabile currentException è un'istanza della classe CollaborationException. Quindi è possibile utilizzare i metodi contenuti nella Tabella 49 a pagina 179 per ottenere le informazioni sull'eccezione dalla variabile currentException.

Per identificare la causa dell'eccezione della collaborazione, l'oggetto eccezione include uno dei *tipi di eccezione* elencati nella Tabella 50. I tipi di eccezione sono dei valori di stringhe per i quali sono state dichiarate delle costanti Java statiche.

Tabella 50. Tipi di eccezione

Costante del tipo di eccezione	Descrizione
AnyException	Qualsiasi tipo di eccezione. Se esistono due diramazioni di eccezione — una che testa un tipo specifico di eccezione ed una che testa AnyException — la diramazione che testa il tipo di eccezione specifico viene verificata per prima. Se l'eccezione corrente non corrisponde all'eccezione specifica, viene elaborata prima la diramazione che testa AnyException.
AttributeException	Problema di accesso attributo, ad esempio, la collaborazione denominata getDouble() su un attributo String o denominata getString() su un attributo nonexistent.
JavaException	Problema con del codice Java che non fa parte dell'API della collaborazione.
ObjectException	Oggetto business non valido passato ad un metodo.
OperationException	La chiamata di servizio non è stata configurata correttamente e non è stato possibile inviarla.
ServiceCallException	La chiamata di servizio non è riuscita per cause esterne alla collaborazione. Ad esempio, un connettore o un'applicazione non sono disponibili oppure si è verificata un'interruzione di rete.
SystemException	Qualsiasi errore interno nel sistema InterChange Server Express.
TimeoutException	La chiamata di servizio in ingresso sincrona o asincrona ha superato il tempo limite.
TransactionException	Errore relativo al comportamento di transazione di una collaborazione di transazione. Ad esempio, il rollback non è riuscito o la collaborazione non ha potuto applicare alcuna compensazione.

Nota: Quando si definisce una sezione di eccezione in un nodo decisione, si specifica il tipo di eccezione da controllare nella condizione della sezione di eccezione. Per ulteriori informazioni, consultare "Rilevazione dell'eccezione" a pagina 185.

Alcuni di questi tipi di eccezione hanno numerose situazioni che potrebbero causarli. Per questi tipi di eccezione, l'oggetto eccezione spesso include un *tipo di eccezione secondario*, che fornisce ulteriori informazioni sulle cause dell'eccezione. I due principali tipi di eccezione che utilizzano dei tipi secondari di eccezione sono JavaException e ServiceCallException. Per ulteriori informazioni, vedere la descrizione di "getSubType()" a pagina 450.

Modalità di elaborazione delle eccezioni

Quando si esegue una collaborazione, può essere in uno dei due seguenti *stati di esecuzione*:

- Lo stato Normale indica una delle seguenti condizioni:
  - Non si sono verificate eccezioni.
  - Si è verificata un'eccezione ma la maschera di collaborazione l'ha rilevata.
- Lo stato Eccezione indica che si è verificata un'eccezione e *non* è stata gestita all'interno della maschera di collaborazione. Una collaborazione entra in stato di eccezione quando si verifica una delle seguenti condizioni:
  - Si verifica un'eccezione di collaborazione, ad esempio una delle seguenti condizioni:
    - Una chiamata di servizio ha esito negativo; vale a dire, si verifica un'eccezione di collaborazione con tipo eccezione `ServiceCallException`.
    - Si verifica un'eccezione Java; vale a dire, si verifica un'eccezione di collaborazione con tipo eccezione `JavaException`.
  - La maschera di collaborazione richiama il metodo `raiseException()`, generando un'eccezione di collaborazione con qualsiasi tipo di eccezione valido.

La collaborazione può alternarsi fra questi due stati durante l'esecuzione. Entra in stato eccezione quando si verifica un eccezione o viene eseguito `raiseException()`. Ritorna allo stato Normale quando la maschera di collaborazione rileva l'eccezione con una diramazione di eccezione. A prescindere dalla gestione eccezioni che un diagramma di attività può (o può non) eseguire, l'ambiente di collaborazione runtime continua l'esecuzione della logica del diagramma, dopo che si è verificata un'eccezione. Questa logica terminerà con un nodo Fine corretta o Fine con errore. L'ambiente di collaborazione runtime utilizza lo stato di esecuzione della collaborazione per stabilire se creare un flusso non risolto, una volta terminata la collaborazione. Per ulteriori informazioni sul termine dei nodi, consultare "Termine del percorso di esecuzione" a pagina 154. Per ulteriori informazioni sui flussi non risolti, consultare "Elaborazione dello stato Eccezione" a pagina 182.

Elaborazione dello stato Normale

Quando l'ambiente di collaborazione runtime sta eseguendo con esito positivo una collaborazione (come definito dalla logica nel diagramma di attività), l'esecuzione della collaborazione è in stato Normale. I possibili modi per terminare un percorso di esecuzione includono:

- Un nodo Fine corretta

  L'ambiente di collaborazione runtime arresta l'esecuzione del diagramma corrente e passa il controllo al successivo livello superiore di esecuzione:
  - Se il nodo Fine corretta termina il il diagramma di attività principale, l'ambiente di collaborazione runtime termina la collaborazione. Quando l'esecuzione della collaborazione è nello stato Normale, l'ambiente di collaborazione runtime *non* crea un flusso non risolto.
  - Se il nodo Fine corretta termina un diagramma secondario o iteratore, l'ambiente di collaborazione runtime effettua le seguenti operazioni:
    - Termina il livello di esecuzione corrente e passa il controllo al diagramma principale.

- Continua l'esecuzione del diagramma principale come definito dalla sua logica. Quando, tuttavia, l'esecuzione della collaborazione e nello stato Normale, l'ambiente di collaborazione runtime *non* controlla la presenza di eventuali diramazioni di eccezioni.

- Un nodo Fine con errore

  L'ambiente di collaborazione runtime interrompe l'esecuzione della collaborazione ed esegue i passi per una collaborazione in stato Normale (consultare "Termine con esito negativo" a pagina 155).

Elaborazione dello stato Eccezione

Quando l'esecuzione della collaborazione entra nello stato Eccezione, l'ambiente di collaborazione runtime *non* arresta l'esecuzione. Continua, invece, l'esecuzione come definito dalla logica del diagramma di attività, esattamente come fa per l'esecuzione in stato Normale. I possibili modi per terminare questo percorso di esecuzione includono:

- Un nodo Fine corretta

  L'ambiente di collaborazione runtime arresta l'esecuzione del diagramma corrente e passa il controllo al successivo livello superiore di esecuzione, che può essere:

  – Se il nodo Fine corretta termina il diagramma principale, il successivo livello superiore di esecuzione è l'ambiente di collaborazione runtime. Per ulteriori informazioni, consultare "Completamento del diagramma principale con esito positivo".

  – Se il nodo Fine corretta termina un diagramma secondario o iteratore, il successivo livello superiore è il diagramma principale. Per ulteriori informazioni, consultare "Termine di un diagramma secondario o iteratore con esito positivo" a pagina 183.

- Un nodo Fine con errore

  L'ambiente di collaborazione runtime arresta l'esecuzione della collaborazione. Quando l'esecuzione della collaborazione è in stato Eccezione, l'ambiente di collaborazione runtime come passo successivo gestisce l'eccezione. Per informazioni sui passi di gestione eccezioni, consultare "Completamento del diagramma principale con esito positivo".

Completamento del diagramma principale con esito positivo

Quando il nodo Fine corretta termina il diagramma principale, l'ambiente di collaborazione runtime termina la collaborazione: Se l'esecuzione della collaborazione è in stato Eccezione l'ambiente runtime esegue il seguenti passi per gestire eccezione:

1. Registrare un errore nella destinazione log della collaborazione, che può essere l'output standard (STDOUT) o un file log, a seconda di come è configurata la destinazione log in InterChange Server Express's.
   - Se la collaborazione *non* è transazionale, l'ambiente di collaborazione runtime registra un errore.
   - Se la collaborazione è transazionale, l'ambiente di collaborazione runtime ne esegue il rollback, eseguendo i passi di compensazione della collaborazione.

     Se l'eccezione si verifica durante il rollback, l'ambiente di collaborazione runtime termina la collaborazione e registra l'errore. A questo punto, l'oggetto di collaborazione è in uno stato "in dubbio". Un amministratore può risolvere manualmente lo stato transazionale dell'oggetto di collaborazione eseguendo o ignorando i rimanenti passi di compensazione.
2. Creare un *flusso non risolto* per la collaborazione non riuscita.

Quando una collaborazione termina con la sua esecuzione in stato di Eccezione, lascia un flusso non risolto che include:

- Un **evento non riuscito**, che è l'evento originale (oggetto business e istruzione) che ha attivato la collaborazione non riuscita
- Un messaggio di eccezione che descrive la causa dell'errore

L'ambiente di collaborazione runtime invia il flusso non risolto alla coda di reinoltro eventi di InterChange Server Express, dove l'amministratore del può analizzarlo e valutarne un possibile reinoltro. Lo strumento Flow Manager fornisce all'amministratore l'accesso alla coda di reinoltro eventi. L'amministratore può esaminare le informazioni sui flussi non risolti, come il nome della collaborazione terminata ed un messaggio che descrive la condizione di errore.

Nota: per ulteriori informazioni sull'utilizzo di Flow Manager, consultare il manuale *System Administration Guide*.

3. Gestore l'eccezione, salvare l'evento non riuscito e tornare allo scenario principale. In un secondo momento, l'amministratore può inoltrare di nuovo gli eventi non riusciti alla collaborazione per essere elaborati. Consultare "Salvataggio di un evento non riuscito" a pagina 78.

Per impostazione predefinita, l'ambiente di collaborazione runtime associa un messaggio di eccezione molto semplice ad un flusso non risolto:

```
Scenario non riuscito.
```

Questo messaggio di eccezione predefinito *non* fornisce all'amministratore molte informazioni con le quali analizzare la causa del flusso non risolto. Tuttavia, se si codifica la maschera d collaborazione a generare l'eccezione, è possibile fornire un messaggio di eccezione con più informazioni sull'effettiva condizione di errore che si è verificata. Quando l'ambiente di collaborazione runtime gestisce l'eccezione, è possibile associare questo messaggio di eccezione più dettagliato al flusso non risolto. Per ulteriori informazioni, consultare "Innalzamento dell'eccezione" a pagina 187.

Termine di un diagramma secondario o iteratore con esito positivo

Quando un nodo Fine corretta termina un diagramma secondario (o iteratore), e l'esecuzione della collaborazione è in stato Eccezione, l'ambiente di collaborazione runtime effettua le seguenti operazioni:

1. Passa il controllo al diagramma principale. Il diagramma principale è il diagramma che include il nodo diagramma secondario (o iteratore).
2. Verificare l'eventuale presenza di diramazioni di eccezione nel nodo decisione del diagramma principale che collegano il diagramma secondario (o iteratore) ad un nodo di gestione eccezioni. Eseguire una delle seguenti azioni:
   - Se esiste una diramazione di eccezione che rileva l'eccezione corrente, l'ambiente di collaborazione runtime passa il controllo al nodo di gestione eccezioni indicato dalla diramazione eccezione. Quando il nodo di gestione eccezioni viene completato, l'esecuzione continua come definito dalla diramazione del diagramma di attività che contiene il nodo di gestione eccezioni.

     Nota: Quando la diramazione eccezione viene eseguita, l'esecuzione della collaborazione cambia allo stato Normale. Pertanto, a meno che la maschera di collaborazione non presenti l'eccezione in qualche punto del percorso di esecuzione, l'ambiente di collaborazione runtime *non*

crea un flusso non risolto per la collaborazione. Per informazioni su come implementare il codice di gestione eccezioni, consultare "Modalità di gestione delle eccezioni".

- Se nessuna diramazione di eccezione rileva l'eccezione corrente, l'ambiente di collaborazione runtime continua l'esecuzione del diagramma principale come definito dalla relativa logica. L'esecuzione della collaborazione, tuttavia, sarà ancora nello stato Eccezione. A meno che qualche altro livello di esecuzione non rilevi l'eccezione, l'ambiente di collaborazione runtime crea un flusso non risolto per la collaborazione quando questa termina.

L'ambiente di collaborazione runtime continua l'esecuzione della logica del diagramma principale finché il diagramma non termina in Fine corretta o Fine con errore. Finché l'esecuzione della collaborazione resta in stato Eccezione, l'ambiente runtime gestisce l'eccezione quando termina la collaborazione. Fin quando ogni livello di esecuzione termina con Fine corretta, l'esecuzione passa al successivo livello superiore finché non giunge al diagramma principale. A meno che il diagramma principale non rilevi l'eccezione, questa collaborazione termina ed il controllo passa all'ambiente di collaborazione runtime.

Modalità di gestione delle eccezioni

Esistono due categorie di eccezioni che una collaborazione può gestire:

- Eccezioni del processo business

  Le eccezioni del processo business sono generate dal codice che utilizza i metodi Api di collaborazione. Ad esempio, un'eccezione di processo può verificarsi quando lo scenario imposta il valore di un attributo di oggetto business, invia una richiesta di chiamata di servizio ad un connettore, e così via. Per informazioni su come gestire determinate eccezioni di chiamata di servizio, consultare "Gestione di eccezioni di chiamate di servizio particolari" a pagina 191.

- Eccezioni Java native

  Le eccezioni Java provengono dal codice utente che utilizza i metodi Java nativi. Quando si verifica questo tipo di eccezione, è possibile produrre un'eccezione di collaborazione il cui tipo di eccezione è `JavaException` ed il cui tipo secondario contiene la articolare eccezione Java. L'ambiente di collaborazione runtime rileva e gestisce le eccezioni Java che derivano dal proprio codice.

Quando si verifica un'eccezione, la gestione dell'eccezione ad un dato livelli di gerarchia del diagramma di attività può gestire l'eccezione in uno dei seguenti modi:

- *Non* rilevare l'eccezione al livello di esecuzione corrente.
- Rilevare l'eccezione con una diramazione eccezione in u nodo decisione.

Non rilevazione dell'eccezione

Se il diagramma di attività *non* rileva esplicitamente l'eccezione con una diramazione eccezione, l'esecuzione della collaborazione resta in stato eccezione (in cui è entrata quando si è verificata l'eccezione). L'ambiente di collaborazione runtime *non* arresta l'esecuzione nel diagramma quando si verifica l'eccezione. L'esecuzione, invece, continua secondo la logica del diagramma di attività, terminando in un nodo Fine corretta e Fine con errore:

- Se il percorso dell'esecuzione termina in Fine con errore, l'ambiente di collaborazione runtime termina la collaborazione e crea un flusso non risolto.

Pertanto, se si desidera rilevare un'eccezione nel diagramma secondario ma si desidera anche passare il nodo Fine con errore, e necessario assicurarsi che il codice rilevi l'eccezione nel diagramma secondario (prima del nodo Fine con errore).

- Se il percorso di esecuzione termina con Fine corretta, la collaborazione passa il controllo al successivo livello superiore.

  In un diagramma di attività gerarchico, se non si utilizza una diramazione eccezione per rilevare un'eccezione ad un livello di esecuzione, è possibile utilizzare Fine corretta per passare il controllo al successivo diagramma di livello superiore. In questo diagramma di livello superiore è possibile rilevare l'evento e gestire l'eccezione oppure innalzarla al successivo livello superiore.

  Se la maschera di collaborazione non ha rilevato l'eccezione *da nessuna parte* nel suo percorso di esecuzione, l'esecuzione resta in stato Eccezione. In questo caso l'ambiente di collaborazione runtime gestisce ancora l'eccezione, come descritto in "Elaborazione dello stato Eccezione" a pagina 182. Poiché la maschera di collaborazione non ha mai rilevato l'eccezione, l'ambiente di collaborazione runtime deve includere il proprio messaggio di eccezione predefinito (`Scenario non riuscito.`) con il flusso non risolto.

Rilevazione dell'eccezione

Una maschera di collaborazione può includere la gestione delle eccezioni nel suo diagramma di attività tramite le *diramazioni di eccezione*, che sono delle sezioni all'interno di un nodo di decisione il cui tipo di diramazione è impostato su `Eccezione`. Il nodo decisione collega il simbolo dell'azione ai possibili risultati della decisione. Una diramazione di eccezione instrada il simbolo azione in cui si verifica l'eccezione al simbolo azione in cui viene gestita l'eccezione. Una diramazione di eccezione include una condizione di errore, che specifica il tipo di eccezione che quella diramazione di eccezione rileva. La Tabella 50 a pagina 180 elenca i tipi di eccezione di collaborazione dai quali si può scegliere quando si definisce la condizione di eccezione.

Nota: Per ulteriori informazioni su come aggiungere una diramazione di eccezione ad un diagramma di attività, consultare "Definizione di una diramazione eccezione" a pagina 132.

Quando si verifica un'eccezione, l'ambiente di collaborazione runtime compila la variabile di sistema `currentException`. Per stabilire se seguire la diramazione di eccezione, l'ambiente di collaborazione runtime valuta la condizione di eccezione della diramazione di eccezione confrontandone il tipo contenuto nella condizione della diramazione e quello della variabile di sistema `currentException`:

- Se i due tipi di eccezione sono uguali, la condizione di eccezione è *true* ed il diagramma di attività *rileva* l'eccezione.

  L'ambiente di collaborazione runtime cambia lo stato dell'esecuzione della collaborazione in Normale e passa il controllo al simbolo azione a cui punta la diramazione di eccezione. Il simbolo azione può contenere il codice di gestione eccezioni per gestire il tipo di eccezione specificato nella condizione di eccezione. Questo codice può accedere alla variabile di sistema `currentException` per ottenere le informazioni sull'eccezione.

- Se i tipi di eccezione *non* corrispondono, la condizione di eccezione è false e il diagramma di attività *non rileva* l'eccezione.

  L'esecuzione passa alla diramazione predefinita del nodo decisione (se ne esiste una) e quindi al successivo simbolo azione nella logica del diagramma di attività. A meno che l'eccezione non venga gestita nella diramazione predefinita,

questa situazione comporta che l'eccezione *non* viene gestita a questo livello del diagramma di attività. Significa anche che l'esecuzione della collaborazione resta in stato Eccezione.

Nota: L'ambiente di collaborazione runtime verifica la presenza di eventuali diramazioni di eccezione *solo* quando l'esecuzione della collaborazione è in stato Eccezione.

Un nodo decisione può avere un massimo di sette diramazioni. Pertanto può fornire la gestione eccezioni per molti tipi di eccezione. Ogni diramazione di eccezione può specificare un tipo diverso di eccezione nella relativa condizione è può puntare ad un nodo di gestione eccezione relativo a quel tipo. In alternativa, è possibile gestire *tutti* i tipi di eccezione in un'unica diramazione eccezione che specifica il tipo eccezione `AnyException` nella condizione eccezione.

Dopo che la maschera di collaborazione ha rilevato l'eccezione e l'esecuzione è passata al nodo di gestione eccezioni, la maschera di collaborazione può gestirla nei seguenti modi:

- Procedere con la logica dello scenario verso un completamento con esito positivo o negativo.
- Innalzare l'eccezione al successivo livello superiore nel diagramma di attività.

Continuazione con la logica dello scenario

Per procedere con la logica dello scenario, effettuare i seguenti passi nel nodo di gestione eccezioni:

1. Elaborare l'eccezione in modo tale da *non* implicare il passaggio dell'eccezione al livello successivo.

   Nel nodo al quale punta la diramazione di eccezione è possibile includere il codice che elabora l'eccezione. La Tabella 51 elenca alcuni possibili passi di elaborazione.
2. Terminare il percorso di esecuzione del nodo di attività con un nodo Fine corretta o Fine con errore.

   Finché il nodo di gestione eccezione *non* innalza l'eccezione al successivo livello superiore (con il metodo `raiseException()`), l'esecuzione di collaborazione resta in stato Normale. Quindi l'ambiente di collaborazione runtime *non* crea un flusso non risolto quando l'esecuzione di collaborazione viene completata. Per ulteriori informazioni su come l'ambiente di collaborazione runtime elabora questi nodi di terminazione, consultare "Elaborazione dello stato Normale" a pagina 181.

La Tabella 51 elenca alcuni dei possibili passi di elaborazione che il nodo di gestione eccezioni può eseguire. Nessuno di questi passi modifica lo stato di esecuzione della collaborazione. Quindi l'esecuzione della collaborazione resta in stato Normale.

Tabella 51. Possibili passi di elaborazione per la gestione delle eccezioni

Passi di gestione eccezione	Metodi	Per ulteriori informazioni
Registrare un messaggio nella destinazione log della collaborazione.	`logError()`, `logWarning()`, `logInfo()`	"Registrazione dei messaggi" a pagina 207
Ottenere le informazioni sull'eccezione.	Metodi della classe `CollaborationException`.	Tabella 49 a pagina 179

Ad esempio, il metodo `logError()` registra gli errori nella destinazione log della collaborazione. Questa destinazione può essere un output standard (STDOUT) oppure un file log, se configurato. Questo metodo inoltre invia il messaggio di errore ad un destinatario di email. La maschera di collaborazione può utilizzare questo metodo per registrare degli errori che l'amministratore può esaminare. Il frammento di codice riportato di seguito estrae le informazioni sull'eccezione dalla variabile `currentException` con i metodi `getMessage()` e `getMsgNumber()` della classe `CollaborationException`. Utilizza, quindi le informazioni per formattare l'errore da inviare alla destinazione log della collaborazione con una chiamata a `logError()`.

```
// extract exception information
sMessage = currentException.getMessage();
imsgNumber = currentException.getMsgNumber();

// log message and send email (if configured)
logError(imsgNumber, sMessage, ...);
```

Per ulteriori informazioni, leggere la descrizione del metodo `logError()` in "logError(), logInfo(), logWarning()" a pagina 377. Tenere presente che inviare semplicemente un errore alla destinazione log è spesso insufficiente per associare un chiaro messaggio di eccezione al flusso non risolto. Ad esempio, si verifica un'eccezione, una diramazione eccezione la rileva eil nodo di gestione eccezioni registra semplicemente l'errore e termina con esito negativo. In questo caso, il flusso non risolto della collaborazione non riuscita include l'evento con esito negativo ma il suo messaggio di eccezione è semplicemente il messaggio predefinito dell'ambiente di collaborazione runtime (`Scenario non riuscito.`).

Innalzamento dell'eccezione

Il metodo `raiseException()` innalza un'eccezione di collaborazione al successivo livello superiore dell'esecuzione. Quando l'ambiente di collaborazione runtime esegue la chiamata `raiseException()`, modifica lo stato di esecuzione della collaborazione in Eccezione e continua con la logica del diagramma di attività. Per innalzare l'eccezione al successivo livello superiore nel diagramma di attività eseguire i seguenti passi nel nodo di gestione eccezioni:

1. Acquisire le informazioni sull'eccezione dall'eccezione corrente per includerle nell'eccezione aumentata.

   Nel nodo di gestione eccezioni, è possibile utilizzare i metodi della classe `CollaborationException` per estrarre le informazioni sull'eccezione dalla variabile di sistema `currentException`.

   Nota: E' importante estrarre il messaggio dalla variabile `currentException` in modo da poterlo includere nell'eccezione innalzata. Facendo questo, il messaggio può essere disponibile per l'ambiente di collaborazione runtime quando associa un messaggio di eccezione al flusso non risolto.

2. Includere una chiamata al metodo `raiseException()` per generare l'eccezione da innalzare.

   Quando l'ambiente di collaborazione runtime esegue una chiamata a `raiseException()`, cambia l'esecuzione della collaborazione allo stato Eccezione. La chiamata `raiseException()` fornisce l'eccezione da innalzare al successivo livello superiore di esecuzione.

3. Terminare il percorso di esecuzione della diramazione che contiene il codice di gestione eccezioni con un nodo Fine corretta o Fine con errore.
   - Se si termina il percorso di esecuzione in Fine corretta, si innalza l'eccezione al successivo livello superiore dell'esecuzione, dove può essere rilevata o innalzata al successivo livello superiore. Quando si utilizza questo metodo ad

ogni livello di esecuzione, il codice della collaborazione può innalzare al livello principale le eccezioni catturate, che può effettuare la determinazione finale della gestione errori.

- Se si termina il percorso di esecuzione in Fine con errore, si innalza l'eccezione all'ambiente di collaborazione runtime, che include il messaggio di eccezione come parte del flusso non risolto.

Poiché è stata innalzata l'eccezione, l'esecuzione della collaborazione è in stato Eccezione. Finché ogni livello di esecuzione innalza l'eccezione (con `raiseException()`), l'esecuzione della collaborazione resta in stato Eccezione. Pertanto, l'ambiente di collaborazione runtime crea un flusso non risolto quando termina l'esecuzione della collaborazione. Per informazioni su come l'ambiente di collaborazione runtime elabora questi nodi di terminazione, consultare "Elaborazione dello stato Eccezione" a pagina 182.

Per capire come poter utilizzare i metodi `raiseException()` e `logError()` nelle maschere di collaborazione per gestire le eccezioni, considerare l'esempio riportato di seguito. Un diagramma principale della collaborazione richiama un diagramma secondario A, che a sua volta richiama il diagramma secondario B. Il diagramma secondario B effettua una chiamata di servizio, che potrebbe risultare in un'eccezione. Pertanto, il diagramma secondario contiene un nodo azione che richiama una chiamata di servizio. il nodo azione si collega ad un nodo decisione con una diramazione di eccezione che verifica la presenza di eccezioni di chiamate di servizio. Se si verifica un'eccezione di chiamata di servizio, la diramazione di eccezione si collega ad un nodo azione con il codice per la gestione di eccezioni.

Quando si verifica un'eccezione, l'ambiente di collaborazione runtime modifica cambia l'esecuzione della collaborazione allo stato Eccezione e valuta la condizione di eccezione di tutte le diramazioni eccezione associate alla chiamata di servizio nel diagramma secondario B. Se una condizione di diramazione eccezione si risolve in true, la diramazione di eccezione rileva l'eccezione ed il controllo passa al nodo di gestione eccezioni a cui punta la diramazione eccezione. Quando la diramazione eccezione rileva l'eccezione, l'esecuzione della collaborazione torna allo stato Normale.

La Figura 63 mostra il frammento di codice che esegue la gestione eccezione per un'eccezione di chiamata di servizio nel diagramma secondario B.

```
// exception handling in subdiagramB
sMessage = currentException.getMessage();
sType = currentException.getType();

// raise the exception to subdiagramA
raiseException(sType, 2345, parameter1, parameter2, sMessage);
}
```

Figura 63. Gestione dell'eccezione di chiamata di servizio nel diagramma secondario B

Il codice nel nodo di gestione eccezioni esegue i seguenti passi:

1. Controlla la variabile di sistema `currentException` per informazioni sull'eccezione.

   Il codice ottiene il messaggio di eccezione ed il tipo di eccezione da `currentException` e li salva in due variabili string (rispettivamente `sMessage` e `sType`).
2. Innalza l'eccezione al diagramma principale (in questo caso il diagramma secondario A).

Una volta raccolte le informazioni sull'eccezione, il codice richiama il metodo `raiseException()` per innalzare l'eccezione al diagramma secondario A. Questo formato del metodo `raiseException()` riceve le informazioni sull'eccezione: il tipo di eccezione ed un messaggio di errore (2345) con i relativi tre parametri di messaggio. I parametri del messaggio includono il messaggio di eccezione che il codice ha ottenuto dalla variabile di sistema `currentException`. La chiamata `raiseException()` quindi crea un'eccezione che contiene queste informazioni. Modifica anche l'esecuzione della collaborazione allo stato Eccezione.

Nota:

a. Il numero di parametri messaggio che `raiseException()` fornisce per un messaggio dipende dal particolare formato messaggio nel file di messaggi della collaborazione.
b. E' possibile specificare gli stessi tipi di eccezione quando si genera l'eccezione nel codice di gestione eccezioni con il metodo `raiseException()` che si possono specificare quando si definisce la condizione di eccezione nella diramazione eccezione. Per un elenco di tipi di eccezione, consultare Tabella 50 a pagina 180.

Dopo l'esecuzione di `raiseException()`, l'azione che esegue l'ambiente di collaborazione runtime dipende dal nodo di terminazione che termina il percorso di esecuzione. Se il percorso di esecuzione termina in un nodo Fine corretta, l'ambiente di collaborazione runtime esegue i seguenti passi:

- Passa il controllo al diagramma secondario A.
- Cerca un nodo decisione con una diramazione eccezione che rileva questa eccezione (perché lo stato di esecuzione è Eccezione). Questo nodo decisione collegherebbe al nodo del diagramma secondario B e la relativa diramazione eccezione collegherebbe al nodo di gestione eccezioni appropriato.

Nota: Se il percorso di esecuzione termina in u nodo Fine con errori, l'ambiente di collaborazione runtime termina l'intera collaborazione. Poiché lo stato di esecuzione è Eccezione, l'ambiente runtime crea un flusso non risolto utilizzando le informazioni eccezione nell'eccezione che ha generato la chiamata `raiseException()`.

Se nel diagramma secondario esiste un nodo decisione con diramazioni eccezione, ambiente di collaborazione runtime valuta ogni condizione di diramazione eccezione. Se questa condizione di eccezione si risolve in true, il diagramma secondario A rileva l'eccezione che ha generato il diagramma secondario B. L'esecuzione della collaborazione cambia a stato Normale ed il controllo passa al nodo di gestione eccezioni, che gestisce l'eccezione innalzandole al diagramma principale con la seguente chiamata a `raiseException()`:

```
// exception handling in subdiagramA: raise the exception to main diagram
raiseException(currentException);
```

Questo formato del metodo `raiseException()` innalza solo l'oggetto eccezione che riceve come argomento. *Non* crea un'eccezione dalle informazioni che gli sono state passate. In questo caso, non è necessario che `raiseException()` crei un'eccezione perché il codice di gestione eccezioni nel diagramma secondario B (Figura 63 a pagina 188) ha già creato l'eccezione con le appropriate informazioni eccezione. L'eccezione che il diagramma secondario A contiene nella sua variabile `currentException` è la stessa innalzata dal diagramma secondario B. Al termine di `raiseException()`, l'esecuzione di collaborazione continua secondo la logica nel diagramma secondario A. Se questa diramazione del diagramma secondario A termina in Fine corretta, l'ambiente di collaborazione runtime termina il

diagramma secondario A e passa il controllo al relativo diagramma principale. Pertanto, l'oggetto eccezione `raiseException()` genera (l'oggetto eccezione `currentException` nel diagramma secondario A) ora è innalzato al diagramma principale.

Nota: Se questa diramazione di gestione eccezioni del diagramma secondario A è terminata con Fine con errore, la collaborazione termina e l'oggetto eccezione viene innalzato all'ambiente di collaborazione runtime, che include il relativo messaggio di eccezione come parte del flusso non risolto.

L'esecuzione della collaborazione correntemente è nel nodo diagramma secondario A nel diagramma principale. L'esecuzione della collaborazione correntemente è in stato eccezione perché il nodo gestione eccezioni nel diagramma secondario A ha richiamato `raiseException()`. Pertanto l'ambiente di collaborazione runtime verifica nel diagramma principale la presenza di eventuali diramazioni eccezione che rilevano l'eccezione innalzata. Queste ramificazioni eccezione si troverebbero in un nodo decisione che collega la chiamata al diagramma secondario A ad uno o più nodi azione di gestione eccezioni. Se una condizione di diramazione eccezione si risolve in true, il diagramma principale rileva l'eccezione innalzata dal diagramma secondario A. L'esecuzione della collaborazione cambia a stato Normale ed il controllo passa al nodo di gestione eccezioni, che può eseguire gli appropriati passi di gestione eccezioni di livello superiore.

Come esempio, si supponga che il nodo di gestione eccezioni nel diagramma principale esegua i seguenti passi:

1. Verifica se la proprietà di configurazione `SEND_EMAIL` della collaborazione è impostato su `tutti` oppure su un elenco delimitato da virgole di numeri messaggio.

   Tutti gli oggetti di collaborazione possono indicare i destinatari di email per gli errori che `logError()` invia alla destinazione log. Se la collaborazione si basa su CollaborationFoundation, può sfruttare l'ulteriore funzionalità che fornisce la proprietà di collaborazione `SEND_EMAIL`. Se l'oggetto di collaborazione è configurato ad inviare email *e* `SEND_EMAIL` viene impostato su `tutti` oppure su un elenco di numeri di messaggio, la collaborazione invia email ai destinatari specificati quando si verifica un qualunque errore (`SEND_EMAIL` è `tutti`) o un errore specificato (`SEND_EMAIL` fornisce un elenco di numeri di messaggi).

   Se queste condizioni vengono soddisfatte, il nodo di gestione eccezioni deve chiamare `logError()` per registrare l'errore ed inviare l'email al destinatario specificato. Pertanto, il codice deve prima recuperare le informazioni sull'eccezione da includere nel messaggio di errore provenienti dall'eccezione corrente.

   Nota: la proprietà di configurazione `SEND_EMAIL` è una funzione di CollaborationFoundation. Se la collaborazione si basa su CollaborationFoundation, può eseguire questo controllo della proprietà `SEND_EMAIL`. Altrimenti questa proprietà di configurazione non è definita.
2. Inviare il messaggio di eccezione alla destinazione log della collaborazione e come messaggio email, se appropriato.

   Se l'oggetto collaborazione è configurato per inviare email, il metodo `logError()` invia automaticamente il messaggio di errore al destinatario email specificato. Questa diramazione utilizza il metodo `logError()` per inviare l'eccezione alla destinazione log della collaborazione (output standard o un file log).

Il seguente frammento di codice nel nodo di gestione eccezioni del diagramma principale esegue questi passi:

```
// exception handling in main diagram

// determine if SEND_EMAIL is set to "all" or a message-number list;
// if so, obtain exception information from the current exception
sMessage = currentException.getMessage();
imsgNumber = currentException.getMsgNumber();

// log message and send email
logError(imsgNumber, sMessage, ...);

// raise the exception to collaboration run-time environment
raiseException(currentException);
```

Quando l'ambiente di collaborazione runtime esegue questa chiamata `raiseException()`, effettua i seguenti passi:

- Imposta l'esecuzione della collaborazione sullo stato Eccezione.
- Procede con il percorso di esecuzione per stabilire come terminare l'esecuzione del diagramma principale.
- Poiché l'esecuzione della collaborazione è in stato eccezione, crea un flusso non risolto quando termina la collaborazione. Ottiene il messaggio di errore dall'eccezione e l'associa al flusso non risolto.
- Invia il flusso non risolto alla coda di flussi non risolti.

Quando l'amministratore visualizza i flussi non risolti, lo strumento Flow Manager mostra il messaggio del flusso non risolto (ottenuto dall'eccezione quando è stata inizialmente prodotta, nel diagramma secondario B).

Gestione di eccezioni di chiamate di servizio particolari

Quando una collaborazione invia una richiesta di oggetto business alla relativa applicazione di destinazione, l'ambiente di collaborazione runtime indica gli eventuali errori producendo un'eccezione di collaborazione con il tipo eccezione `ServiceCallException`. Tuttavia, la causa di un errore di chiamata di servizio può essere ambigua. Una chiamata di servizio potrebbe non riuscire per i seguenti motivi:

- Correlati all'applicazione o alla logica — Ad esempio, si verifica un problema perché l'entità che la collaborazione ha tentato di recuperare non esiste nell'applicazione. In questo caso, l'applicazione non crea o modifica l'entità richiesta.
- Correlati ai trasporti — Ad esempio, si verifica un problema durante la trasmissione dell'oggetto business fra la collaborazione e l'applicazione. In questo caso, l'applicazione potrebbe aver elaborato la richiesta ma la trasmissione dello stato di ritorno non riesce a raggiungere la collaborazione.

Questa sezione fornisce le informazioni su come gestire le seguenti condizioni di errore di chiamata di servizio nella maschera di collaborazione:

- "Chiamate di servizio e richieste exactly-once (esattamente una volta)" a pagina 192
- "Richieste di chiamate di servizio non inviate" a pagina 194

Chiamate di servizio e richieste exactly-once (esattamente una volta)

Il potenziale per la duplicazione di dati può verificarsi in una delle seguenti situazioni:

- Se l'esito negativo della chiamata di servizio si verifica durante la trasmissione dell'oggetto business dall'applicazione alla collaborazione e l'oggetto collaborazione è terminato o era in corso di completamento di una o più chiamate di servizio, il reinoltro della richiesta durante il processo di ripristino potrebbe causare la duplicazione dei dati.
- Se InterChange Server Express si chiude mentre un oggetto collaborazione stava elaborando una chiamata di servizio. In questo caso, il flusso viene considerato come In corso ed il processo di ripristino riesegue l'intero scenario. Per le collaborazioni transazionali, la chiamata di servizio non riuscita viene eseguita solo dopo che sono stati eseguiti tutti i passi di compensazione.

La prevenzione della duplicazione di dati a causa di eccezioni correlate al trasporto devono essere gestire in entrambi i seguenti punti:

- Runtime della collaborazione
- Ripristino all'avvio

Le prossime sezioni descrivono come gestire le eccezioni correlate al trasporto.

Gestione delle eccezioni runtime correlate al trasporto

Per impedire la duplicazione dei dati a causa di un errore di trasporto che si verifica al runtime della collaborazione, codificare la maschera di collaborazione per distinguere, dopo ogni chiamata di servizio, fra un'eccezione correlata al trasporto ed una non collegata al trasporto. Per verificare un eccezione correlata al trasporto, utilizzare il tipo secondario `ServiceCallTransportException` del tipo eccezione `ServiceCallException`. Questo tipo secondario di eccezione indica che si è verificato un errore nel trasporto e che non può essere stabilito con certezza se la richiesta ha raggiunto l'applicazione.

Importante: Un sottoinsieme di tipi di eccezione che prima venivano rappresentati dal tipo secondario `AppUnknown` di `ServiceCallException` vengono ora rappresentati dal tipo secondario `ServiceCallTransportException`. Pertanto, una maschera di collaborazione che verifica specificatamente il tipo secondario `AppUnknown` ora deve cercare anche `ServiceCallTransportException`. Poiché il tipo secondario `AppUnknown` non gestisce più le eccezioni di trasporto, l'oggetto collaborazione non catturerà le eccezioni di trasporto se verifica tutti i tipi secondari tranne `ServiceCallTransportException`.

Gestire un'eccezione correlata al trasporto nel nodo subito dopo la diramazione eccezione; vale a dire, il nodo al quale punta la diramazione di eccezione. Codificare il nodo di gestione eccezioni, che rileva un eccezione, con il tipo secondario di eccezione `ServiceCallException` per fornire un ulteriore Recupera chiamata di servizio che recupera dall'applicazione di destinazione l'oggetto business della richiesta e stabilisce se l'applicazione ha creato o modificato con esito positivo l'oggetto. Se l'oggetto *non* è stato creato o modificato con esito positivo, codificare la collaborazione a ritentare la richiesta.

Il frammento di codice che segue fornisce la gestione eccezioni per un'eccezione correlata al trasporto. Utilizza il metodo `getSubType()` per recuperare il tipo secondario di eccezione dalla variabile di sistema `currentException`. Se questo tipo

secondario di eccezione è ServiceCallTransportException, il codice di gestione eccezioni deve eseguire un retrieve per stabilire se i dati sono cambiati nell'applicazione come risultato della richiesta di chiamata di servizio.

```
if (currentException.getType().equals(ServiceCallException)) {
    if (currentException.getSubType().equals(
        ServiceCallTransportException))
    {
        //Perform a retrieve to determine whether data changed
    //in the application reflecting the ICS business object request
    }
    else
        raiseException(ServiceCallException, ...);
}
```

Nota: Verificare la presenza di un'eccezione correlata al trasporto è particolarmente importante se la collaborazione si collega all'applicazione di destinazione su una rete inaffidabile. In tal caso è importante ritentare ogni volta che questa eccezione potrebbe verificarsi. Poiché il codice può cercare specificatamente le eccezioni causate da errori di trasporto, c'è un impatto sulle prestazioni notevolmente inferiore che se il codice eseguisse un nuovo tentativo su *tutte* le eccezioni.

Gestione delle eccezioni di ripristino all'avvio correlate al trasporto

Per impedire la duplicazione di dati per un errore provocato da una chiusura improvvisa di InterChange Server Express mentre era in corso una chiamata di servizio di collaborazione, è possibile effettuare una delle seguenti operazioni:

- Rendere transazionale la collaborazione, fornendo la compensazione per ogni chiamata di servizio.
- Configurare una collaborazione non transazionale per la persistenza della chiamata di servizio in transito.

Collaborazioni transazionali: Quando InterChange Server Express ripristina una collaborazione transazionale che specifica la compensazione di ogni chiamata di servizio, include il rollback della collaborazione prima di inoltrare di nuovo la richiesta non riuscita. A causa del rollback, la duplicazione dei dati per un errore risultante da una chiusura improvvisa del server non è un problema. Per ulteriori informazioni, consultare "Utilizzo delle funzioni transazionali" a pagina 154.

Collaborazioni non transazionali: Il ripristino di una collaborazione non transazionale *non* include il rollback della collaborazione prima di inviare di nuovo la richiesta non riuscita. Pertanto, la duplicazione di dati può essere un problema. Per evitare la duplicazione di dati per una collaborazione non transazionale, configurare l'oggetto collaborazione per la persistenza Chiamata di servizio in transito. Nella finestra di dialogo Proprietà della collaborazione, contrassegnare la casella di controllo denominata:

```
Mantieni chiamata del servizio in stato di transito
```

Per la compatibilità a ritroso con le collaborazioni esistenti e perché le collaborazioni transazionali *non* beneficiano dalla memorizzazione persistente di ogni stato di chiamata di servizio, l'impostazione predefinita per la persistenza della chiamata di servizio in transito è `non attiva`; vale a dire, la casella `Mantieni chiamata del servizio in stato di transito` non è contrassegnata per un oggetto di collaborazione.

Quando un oggetto di collaborazione configurato per la persistenza entra in una chiamata di servizio, InterChange Server Express mantiene lo stato della richiesta

finché è stata completata. Se il server si chiude mentre la collaborazione sta eseguendo una chiamata di servizio, lo stato della chiamata di servizio non riuscita viene visualizzato in Flow Manager, con il seguente stato:

- Uno stato evento "Chiamata di servizio in transito"
- Un messaggio con il seguente testo:

  ```
  Chiamata di servizio in transito
  ```

In questo caso, l'amministratore deve manualmente stabilire se la richiesta della collaborazione è stata elaborata con esito positivo prima dell'errore di trasporto. Se la richiesta *non* ha avuto esito positivo, l'amministratore deve inoltrarla di nuovo. Per ulteriori informazioni, consultare il manuale *System Administration Guide*.

Nota: se una collaborazione *non* è configurata per la persistenza della chiamata di servizio in transito, tutti gli errori e le eccezioni durante il runtime che provocano la non riuscita del flusso hanno uno stato evento "Non riuscito" nella vista Flussi non risolti.

Richieste di chiamate di servizio non inviate

Per verificare che la chiamata di servizio sia stata inviata all'applicazione, codificare la maschera di collaborazione a controllare dopo ogni chiamata di servizio. Per verificare che la richiesta sia stata inviata, utilizzare il tipo secondario `AppRequestNotYetSent` del tipo eccezione `ServiceCallException`. In caso di agent connettore parallelo, questo tipo secondario di eccezione indica che la richiesta è stata messo in coda nel master agent ma non è mai stata distribuita all'applicazione; quindi è possibile inviare di nuovo la richiesta.

Gestire un'eccezione di chiamata di servizio non inviata nel nodo subito dopo la diramazione eccezione; vale a dire, il nodo al quale punta la diramazione di eccezione. Codificare la maschera della collaborazione al reinoltro della richiesta. Il frammento di codice che segue fornisce la gestione eccezioni per una richiesta di chiamata di servizio non inviata. Utilizza il metodo `getSubType()` per recuperare il tipo secondario di eccezione dalla variabile di sistema `currentException`. Se il tipo secondario di eccezione è `AppRequestNotYetSent`, il codice deve inoltrare di nuovo l'evento tornando al nodo azione con la chiamata di servizio.

```
if (currentException.getType().equals(ServiceCallException))
{
    if (currentException.getSubType().equals(
        AppRequestNotYetSent))
    {
    // Resubmit the event by returning execution to the action node
    // with the service call.
    }
    else
        raiseException(ServiceCallException, ...);
}
```

Importante: L'ambiente di collaborazione runtime *non* imposta un valore per il tipo secondario `AppRequestNotYetSent`se la proprietà connettore `ControllerStoreAndForward` del connettore di destinazione viene impostata su `true`. Se questa proprietà di connettore viene impostata su `false`, la collaborazione deve verificare questo tipo secondario ed inviare di nuovo la richiesta.

Eccezioni dalle API di collaborazione

Quando si progetta una maschera di collaborazione, è possibile includere dei nodi decisione con delle diramazioni eccezione per rilevare le eccezioni prodotte dai metodi dell'API di collaborazione. Per i metodi che generano un'eccezione `CollaborationException`, la descrizione di riferimento ha una sezione intitolata Eccezioni, che visualizza quando vengono prodotte delle eccezioni da quel metodo.

Capitolo 8. Opzioni dell'area di lavoro e del layout

Questo capitolo contiene le informazioni sulle opzioni relative alla disposizione dei simboli e alla personalizzazione dell'area di lavoro quando si modifica un diagramma di attività.

Per ulteriori informazioni sulla personalizzazione del layout della finestra principale di Process Designer Express consultare "Personalizzazione della finestra principale" a pagina 24.

Allineamento dei simboli

Le operazioni di allineamento nella barra degli strumenti Allineamento (vedere la Figura 64) riposizionano due o più simboli per allineare dei margini o dei centri specifici. La barra degli strumenti Allineamento diviene attiva quando nel diagramma delle attività è selezionato più di un simbolo.

L'ordine di tutte le operazioni di allineamento è il seguente:

1. Selezionare un simbolo da utilizzare come "base" (o ancoraggio) dell'allineamento.
2. Tenendo premuto il tasto Maiusc, selezionare gli altri simboli che si desidera allineare al primo.
3. Nella barra strumenti Allineamento, fare clic sull'operazione che si desidera eseguire. Consultare Figura 64.

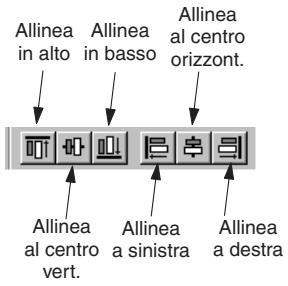


Figura 64. Barra strumenti Allineamento

Allineamento dei margini

L'allineamento dei margini di più simboli allinea il margine specificato di ciascun simbolo ad una linea immaginaria che segue il margine specificato del simbolo del modello. Le operazioni di allineamento del margine includono: Allinea in alto, Allinea in basso, Allinea a sinistra e Allinea a destra.

Ad esempio, la Figura 65 illustra il risultato dell'allineamento di un simbolo Fine corretta e di un simbolo Azione.

Figura 65. Allineamento margini inferiori

Per il simbolo Fine corretta, l'etichetta ed il simbolo insieme formano un oggetto la cui parte inferiore viene allineata con la parte inferiore del simbolo Azione.

Per allineare il margine superiore, inferiore, sinistro o destro di un gruppo di simboli:

1. Fare clic sul simbolo (base o ancoraggio) sul quale si desiderano allineare gli altri.
2. Tenendo premuto il tasto `Maiusc`, fare clic su uno o più simboli addizionali o gruppi di simboli.
3. Sulla barra strumenti Allineamento, fare clic sul pulsante Allinea in alto, Allinea in basso, Allinea a sinistra o Allinea a destra.

   Tutti i simboli si allineano sulla destinazione.

Allineamento al centro

E' possibile centrare i simboli lungo una linea verticale o orizzontale immaginaria disegnata al centro del primo simbolo selezionato. Ogni simbolo viene centrato orizzontalmente o verticalmente lungo quella linea. Le operazioni di allineamento centrale includono Allinea al centro verticalmente e Allinea al centro orizzontalmente.

La linea tratteggiata nella Figura 66 descrive l'operazione Allinea al centro orizzontalmente: l'allineamento dei centri verticali dei due simboli.

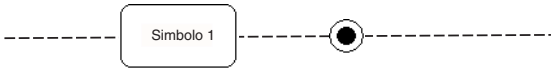


Figura 66. Operazione Allinea al centro verticalmente

La linea tratteggiata nella Figura 67 illustra L'operazione Allinea al centro orizzontalmente: l'allineamento dei centri orizzontali di due simboli.

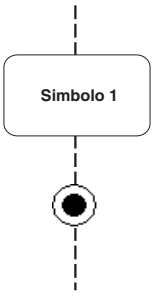


Figura 67. Operazione Allinea al centro orizzontalmente

Per allineare i centri:

1. Selezionare il simbolo o la serie preraggruppata di simboli di cui si desidera utilizzare il centro come base o ancoraggio.
2. Tenendo premuto il tasto `Maiusc`, selezionare gli altri simboli o gruppi di simboli che si desidera allineare.
3. Nella barra strumenti Allineamento, fare clic si Allinea al centro orizzontalmente o Allinea al centro verticalmente.

Ad esempio, questi sono due simboli prima dell'allineamento dei loro centri orizzontali:

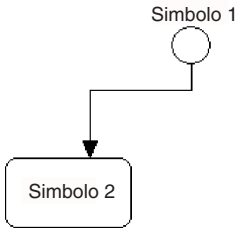


Figura 68. Simboli non allineati

Questi sono gli stessi simboli con i relativi centri orizzontali allineati:

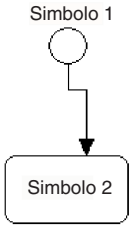


Figura 69. Simboli allineati

Spostamento di simboli

Le operazioni di "spostamento" nella barra degli strumenti Spostamento (vedere la Figura 70) consentono di spostare leggermente i simboli selezionati di un diagramma di attività. La barra degli strumenti Spostamento diviene attiva quando viene selezionato un simbolo nel diagramma di attività.

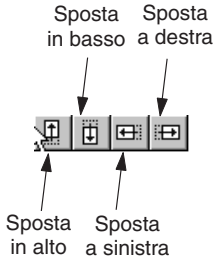


Figura 70. Barra degli strumenti Spostamento

Per controllare lo spostamento più preciso dei simboli:

1. Fare clic sul simbolo che si desidera spostare.
2. Tenendo premuto il tasto `Maiusc`, selezionare gli altri simboli o gruppi di simboli che si desidera spostare.
3. Sulla barra strumenti Spostamento, fare clic sul pulsante Sposta in alto, Sposta in basso, Sposta a sinistra o Sposta a destra.

   Per impostazione predefinita questi comandi spostano i componenti selezionati nella direzione specificata di una unità pixel. Per spostare in una volta di cinque unità pixel i componenti selezionati, tenete premuto il tasto Maiusc.

Zoom o panoramica sui simboli

Le operazioni nella barra degli strumenti Zoom/Panoramica (vedere la Figura 71) consentono di effettuare lo zoom o la panoramica dei simboli selezionati di un diagramma di attività. La barra degli strumenti Zoom/Panoramica si attiva quando viene selezionato un simbolo nel diagramma di attività.

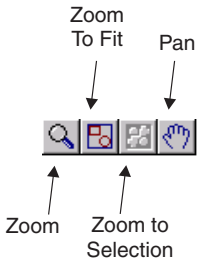


Figura 71. Barra degli strumenti Zoom/Panoramica

La barra degli strumenti Zoom/Panoramica fornisce le seguenti operazioni:

- Zoom: questa operazione consente di ingrandire i simboli selezionati nel diagramma. Per eseguire uno zoom, fare clic e tenere premuto il tasto sinistro del mouse finché si è in modalità zoom. Quando si tiene premuto il tasto del mouse e si trascina in modalità zoom, la selezione zoom traccia un rettangolo per indicare l'area in cui si sta effettuando lo zoom. Una volta posizionati nell'area del diagramma, rilasciare il tasto del mouse per selezionare l'area per lo zoom.
- Adatta alla pagina: questa operazione imposta l'ingrandimento del diagramma in modo da rendere visibili nel diagramma tutti i simboli.
- Adatta a selezione: questa operazione consente di effettuare lo zoom su dei simboli selezionati. Selezionare il simbolo e fare clic su Adatta a selezione per ingrandire il simbolo fino al bordo.
- Panoramica: questa operazione consente di effettuare una "panoramica" o spostarsi ad aree diverse del diagramma. Dopo aver fatto clic su Panoramica, il puntatore diviene una mano. Quindi si fa clic e si trascina il mouse per spostarsi nel diagramma.

Utilizzo della griglia dell'area di lavoro

Process Designer Express visualizza una griglia nell'area di lavoro di dell'editor di diagrammi per facilitare l'allineamento dei simboli. Nella finestra di dialogo Proprietà della griglia (vedere la Figura 72), è possibile impostare le seguenti opzioni della griglia:

- Impostare la griglia in modo che sia visibile o non visibile.

  E' possibile allineare manualmente i simboli per la griglia, se è visibile. Per impostazione predefinita, la griglia non è visibile.
- Attivare "Blocca sulla griglia".
- Impostare il colore della griglia.
- Impostare la spaziatura della griglia.

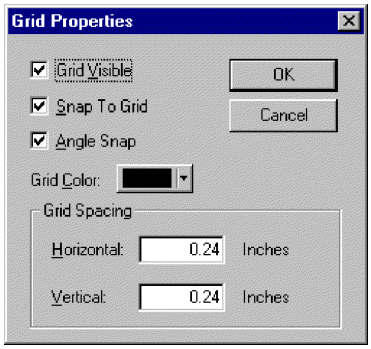


Figura 72. Finestra di dialogo Proprietà della griglia

Per modificare le proprietà della griglia:

1. Verificare che sia aperta la finestra dell'editor di diagrammi.
2. Fare clic su Visualizza —> Proprietà griglia per aprire la finestra di dialogo Proprietà della griglia.
3. Regolare, a seconda delle esigenze, qualunque aspetto della griglia fra quelli elencati di seguito:
   - Se la griglia deve essere visibile o meno
   - Se impostare lo spostamento del simbolo in modo che si agganci alle righe della griglia
   - Il colore della griglia
   - La spaziatura della griglia

     E' possibile regolare la dimensione dei riquadri della griglia secondo le misure preferite. Si può regolare la dimensione della griglia anche se questa è invisibile. In Spaziatura della griglia, immettere un numero per la larghezza e l'altezza di ogni riquadro della griglia, in pollici.

   La selezione dell'ancoraggio angolare non ha effetto sui simboli nel diagramma dell'attività.
4. Fare clic su OK per salvare le opzioni della griglia. Fare clic su Annulla per annullare le modifiche. Entrambe le opzioni chiudono la finestra di dialogo.

E' possibile anche controllare le seguenti opzioni di griglia, direttamente dal menu Visualizza:

- L'opzione Griglia controlla la visibilità della griglia.
- L'opzione Blocca sulla griglia controlla se i simboli devono spostarsi su una riga della griglia.

Modifica della visualizzazione: preferenze dell'utente

Process Designer Express fornisce diversi modi per modificare la visualizzazione. Accedere alla finestra delle preferenze utente facendo clic su Visualizza —> Preferenze, oppure utilizzando la combinazione di tasti di accesso rapido `Ctrl+U`.

La finestra Preferenze utente è composta da tre schede che consentono le seguenti personalizzazioni:

- "Modifica della visualizzazione generale" a pagina 202
- "Modifica della visualizzazione diagramma" a pagina 202
- "Modifica del colore dei simboli e dei collegamenti" a pagina 203

Modifica della visualizzazione generale

La scheda Generale della finestra Preferenze utente consente di:

- Abilitare e disabilitare la visualizzazione delle schede di esercitazione in fondo all'area di lavoro di Process Designer Express's. La Figura 4 a pagina 16 illustra un'area di lavoro che visualizza tre schede di lavoro.

  Nota: La modifica di questa visualizzazione diviene effettiva solo quando viene riavviato Process Designer Express.
- Abilitare e disabilitare la convalida del contenuto. La disabilitazione della convalida del contenuto è utile quando una maschera è danneggiata ma si desidera esaminarla ugualmente. La disabilitazione della convalida del contenuto, tuttavia, non garantisce che Process Designer Express sarà in grado di aprire la maschera danneggiata.
- Abilitare e disabilitare la compressione del contenuto della maschera. Questa opzione non è configurabile dall'utente. Viene fornita per visualizzare l'impostazione di compressione corrente. Il contenuto compresso può migliorare la velocità di trasferimento dati client-server ed i requisiti di memoria.
- Visualizzare o nascondere la sezione Importazioni standard e la sezione Porte ed eventi di attivazione contenute nella scheda Dichiarazioni definizione maschera.
- Impostare i colori di ciascuna sezione della scheda Dichiarazioni definizioni maschere e della finestra Variabili scenario nella finestra di dialogo Definizioni scenario.

La Figura 73 mostra la scheda Generale della finestra Preferenze utente.

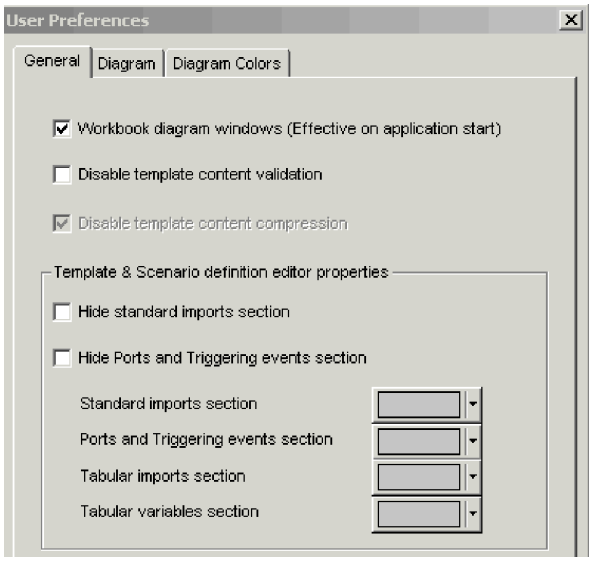


Figura 73. Finestra di dialogo Preferenze utente: scheda Generale

Modifica della visualizzazione diagramma

La scheda Diagramma della finestra Preferenze utente consente di:

- Abilitare e disabilitare la visualizzazione automatica di un nodo di inizio in un nuovo diagramma vuoto. La disabilitazione di questa visualizzazione richiede l'immissione manuale da parte dello sviluppatore del nodo d'inizio in ogni diagramma.

- Abilitare e disabilitare la visualizzazione dei punti di connessione sui nodi del diagramma di attività. La visualizzazione dei punti di connessione semplifica l'aggiunta di un collegamento di transizione fra due nodi.
- Abilitare e disabilitare la visualizzazione di sezioni non utilizzate in un nodo decisione.
- Abilitare la modifica in loco dei frammenti di codice nella finestra di dialogo Proprietà azione. Se queste opzioni sono abilitate, è possibile modificare i frammenti di codice direttamente nella finestra di dialogo Proprietà azione, invece di utilizzare Activity Editor.

La Figura 74 mostra la scheda Diagramma della finestra Preferenze utente.

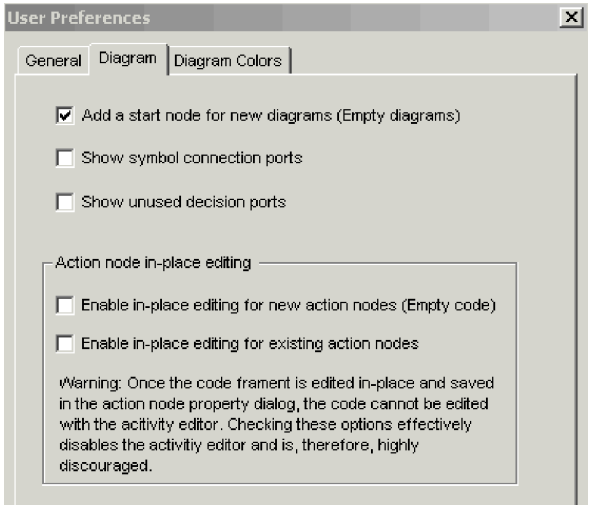


Figura 74. Finestra di dialogo Preferenze utente: scheda Diagramma

Modifica del colore dei simboli e dei collegamenti

La scheda Colori diagramma nella finestra Preferenze utente consente di modificare il colore di visualizzazione dei simboli e dei collegamenti in un diagramma di attività. Per ogni simbolo o collegamento, premere la freccia relativa al campo Riempimento o Riga ed effettuare la selezione dall'elenco a discesa delle opzioni. La Figura 72 mostra la finestra di preferenze di Colori diagramma.

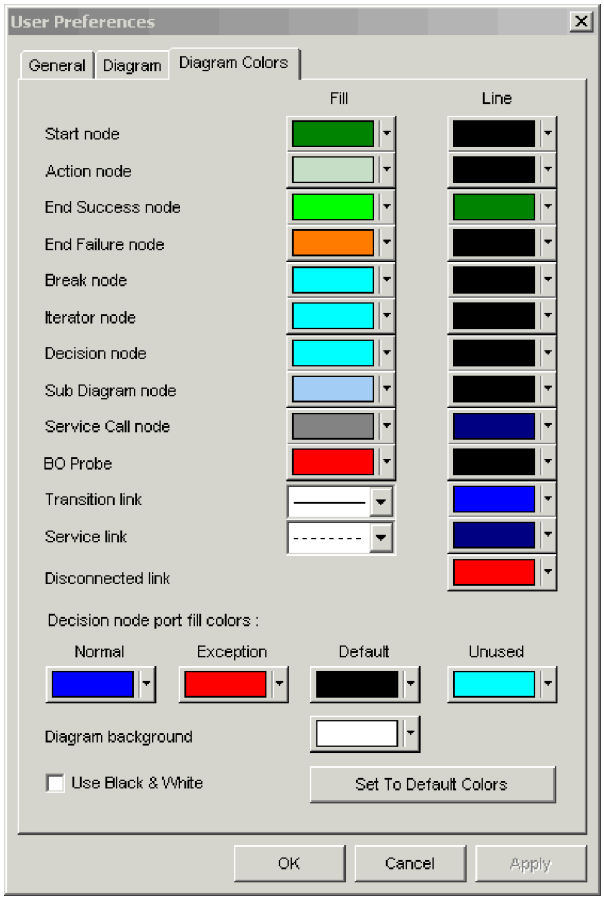


Figura 75. Finestra di dialogo Preferenze utente: scheda Colori diagramma

Allineamento degli scenari

A volte, durante lo sviluppo di una maschera, uno scenario può diventare talmente disallineato da richiedere una significativa quantità di tempo per riallineamento. (Vedere la Figura 76). Come si può notare, i vari nodi di decisione, i collegamenti di transizione e gli altri oggetti sono pesantemente disallineati in questo diagramma. Per correggere l'allineamento sarebbe necessaria una discreta quantità di lavoro manuale.

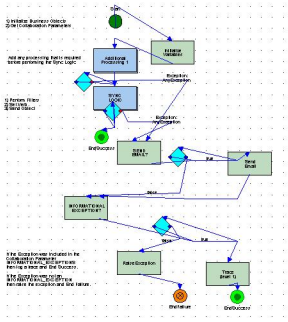


Figura 76. Uno scenario disallineato

Per allineare uno scenario, utilizzare l'opzione Disposizione automatica del menu Visualizza (vedere la Figura 77). La selezione di questa opzione allinea gli oggetti dello scenario in modo da essere visualizzati correttamente in Process Designer Express (vedere la Figura 78).

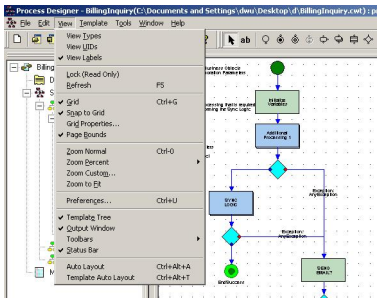


Figura 77. menu Visualizza

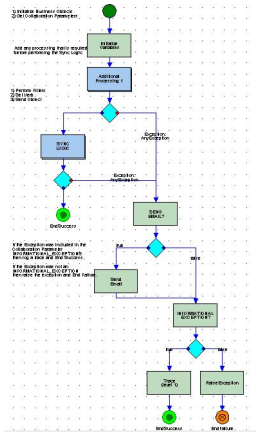


Figura 78. Scenario dopo aver applicato la disposizione automatico.

Per allineare contemporaneamente tutti gli scenari contenuti in una maschera, selezionare l'opzione Disposizione automatica maschera del menu Visualizza.

Disabilitazione della visualizzazione delle finestre delle proprietà dei simboli

Quando si fa doppio clic su un simbolo, le proprietà vengono visualizzate in una piccola finestra. Quando si fa clic su un altro simbolo, il contenuto della finestra visualizza le proprietà del simbolo appena selezionato.

Se si preferisce non visualizzare la finestra delle proprietà del simbolo, chiudere la finestra.

Capitolo 9. Esempi e suggerimenti di codifica

Questo capitolo descrive le modalità di programmazione per tipi specifici di attività di collaborazione.

Operazioni sulla collaborazione

Questa sezione descrive le operazioni che interessano la collaborazione nel suo insieme. Le operazioni sono le seguenti:

Ogni maschera di collaborazione deve essere associata ad un file di messaggi. Il *file di messaggi* contiene il testo dei messaggi di eccezione e di log della collaborazione. Un numero univoco identifica ciascun messaggio nel file di messaggi. Il testo del messaggio può contenere anche i segnaposti per le variabili.

Quando la collaborazione richiama un metodo che visualizza un particolare messaggio, passa al metodo il numero di identificazione del messaggio e gli eventuali 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 nei segnaposti delle variabili nel testo del messaggio.

Ad esempio, un file di messaggi di collaborazione potrebbe contenere un messaggio identificato dal numero 23, il cui testo include due segnaposti, contrassegnati come {1} e {2}:

```
23
Customer ID {1} could not be changed: {2}
```

Quando la collaborazione deve visualizzare o registrare nel log questo messaggio, passa al metodo opportuno, ad esempio `raiseException()`, il numero di identificazione del messaggio (23) e due parametri aggiuntivi, il numero corrispondente a Customer ID (6701) e una variabile String che contiene un testo esplicativo, ad esempio `greater than maximum length`. Il metodo individua il messaggio, sostituisce i valori dei parametri ai segnaposti e visualizza, o registra nel log, il seguente messaggio:

```
Customer ID 6701 could not be changed: greater than maximum length
```

Registrazione dei messaggi

Una maschera di collaborazione può registrare nel log un messaggio quando si verifica una situazione significativa per un amministratore. Per registrare un messaggio, utilizzare i metodi logInfo(), logWarning() e logError() nella maschera di collaborazione. Ogni metodo è associato ad un diverso livello di severità del messaggio. Nella Tabella 52 a pagina 208 sono riportati i livelli di severità e i relativi metodi associati.

Tabella 52. Livelli di messaggio per i metodi di log

Metodo	Livello di severità	Descrizione
`logInfo()`	Informazioni	Solo informativo. L'utente non deve intraprendere alcuna azione.
`logWarning()`	Avvertenza	Riporta le informazioni relative ad un problema. Non utilizzare questo livello per problemi che richiedono un'azione da parte dell'utente.
`logError()`	Errore	Indica un problema grave che richiede un'approfondimento da parte dell'utente.

Il testo del messaggio inviato da questi metodi alla destinazione di log sono prefissati dal livello di severità.

Questa sezione fornisce le seguenti informazioni sulla registrazione dei messaggi:

- "Utilizzo di un file di messaggi"
- "Principi di registrazione dei messaggi"

Utilizzo di un file di messaggi

Ogni maschera di collaborazione deve disporre di un file di messaggi contenente i messaggi di log. Quando una collaborazione registra un errore, il testo del messaggio di errore può provenire dal file di messaggi della collaborazione. Nel seguente esempio, viene registrato un messaggio di errore il cui testo è contenuto nel file di messaggi della collaborazione:

```
logError(10, customer.get("LName"), customer.get("FName");
```

Il testo del messaggio di errore 10 presenta due parametri e si trova nel file di messaggi con il seguente formato:

```
10
Credit report error for {1} {2}.
```

Quando viene eseguito il metodo logError(), questo ottiene il testo del messaggio 10 dal file di messaggi, sostituisce i parametri 1 e 2 del messaggio con il cognome e nome del cliente ed aggiunge al messaggio il prefisso di severità ″Error″. Quindi scrive il messaggio di errore nella destinazione di log della collaborazione.

Ad esempio, il messaggio registrato per il cliente John Davidson, sarà:

```
Error: Credit report error for Davidson John.
```

Se la collaborazione è stata configurata per la notifica e-mail, logError() invia il messaggio di errore anche al destinatario (o ai destinatari) e-mail prestabilito. Per informazioni sull'impostazione di un file di messaggi, fare riferimento a Capitolo 10, "Creazione del file di messaggi", a pagina 247.

Principi di registrazione dei messaggi

Quando si creano i messaggi, occorre valutare come gli amministratori utilizzeranno le funzioni di registrazione log.

Assegnazione dei livelli di severità: E' importante assegnare correttamente i livelli di errore ai messaggi. La funzione di notifica e-mail di InterChange Server Express invia un messaggio alla persona designata, di solito un amministratore, quando rileva che è stato generato un messaggio di errore o errore grave (`logError()`). Gli amministratori possono utilizzare la funzione di notifica e-mail di InterChange Server Express ed eventualmente associare l'invio di un messaggio ad un cercapersone quando si verifica un errore. Assegnando correttamente il livello di errore ai messaggi, si può ridurre il numero di messaggi critici.

Correzione dei messaggi: E' possibile modificare il testo di un messaggio in qualsiasi momento per chiarire o implementare il testo. Tuttavia, quando si assegna un numero al messaggio per un determinato tipo di errore, è importante *non* riassegnare il numero successivamente. Molti amministratori utilizzano script per filtrare i messaggi del log, e tali script si basano sui numeri dei messaggi. Pertanto, è importante che il numero presente nel file di messaggi non assuma un altro significato. Se questo avviene, gli utenti potrebbero non ricevere messaggi importanti oppure ricevere messaggi non desiderati.

Utilizzo dei messaggi informativi: E' possibile utilizzare il metodo `logInfo()` per creare messaggi temporanei per il debug. Tuttavia, è necessario assicurarsi che le chiamate ai metodi di debug siano rimosse al termine della fase di sviluppo.

Si sconsiglia di utilizzare il metodo `logInfo()` per documentare le normali operazioni della collaborazione. L'utilizzo di questo metodo può sovraccaricare il file di log dell'amministratore con messaggi non rilevanti. Utilizzare invece il metodo `trace()` per fornire all'amministratore informazioni dettagliate per il debug.

Aggiunta di messaggi di traccia

E' possibile aggiungere i messaggi di traccia alla maschera di collaborazione in modo che, al momento dell'esecuzione di un oggetto di collaborazione, sia generata una descrizione dettagliata delle sue azioni. I messaggi di traccia sono utili per il debug e per la risoluzione dei problemi da parte degli amministratori.

Il messaggi di traccia differiscono dai messaggi di log in quanto i messaggi di traccia vengono soppressi per impostazione predefinita, mentre i messaggi di log non possono essere rimossi. I messaggi di traccia sono generalmente più dettagliati e si prevede di visualizzarli solo in particolari circostanze, quando il livello di traccia dell'oggetto di collaborazione viene impostato intenzionalmente ad un numero maggiore di zero. I messaggi di traccia e i messaggi di log possono essere inviati su file diversi.

Per una collaborazione sono presenti due tipi di messaggi di traccia:

- Messaggi di traccia generati dalla collaborazione, che sono codificati nella maschera di collaborazione
- Messaggi di traccia generati da InterChange Server Express o generati dal sistema in ambiente di collaborazione runtime

Utilizzare la finestra di dialogo Proprietà oggetto di collaborazione in System Manager per impostare i livelli di traccia per entrambi i tipi di messaggi di traccia.

Lo sviluppatore della maschera di collaborazione crea i livelli per i quali è richiesta la traccia generata dalla collaborazione, come descritto nella successiva sezione. I livelli di traccia generati dal sistema sono gli stessi per tutti gli oggetti di collaborazione. Questi sono descritti in "Messaggi di traccia generati da InterChange Server Express" a pagina 211.

Messaggi di traccia generati dalla collaborazione

E' possibile aggiungere messaggi di traccia ad una maschera di collaborazione per registrare le operazioni specifiche della collaborazione. Di seguito sono riportati alcuni esempi di informazioni che possono essere scritti sul file di traccia dalla collaborazione:

- I valori chiave di un oggetto business quando la collaborazione entra o esce da un particolare nodo azione.

- Il valore di una opzione di configurazione quando viene recuperata da una collaborazione.
- La decisione di eseguire un particolare ramo del percorso di esecuzione.
- Il codice di eccezione restituito da una chiamata di servizio.
- Il valore di ciascun attributo di un oggetto business quando entra o esce da un particolare nodo azione, iteratore o diagramma secondario.

Assegnazione dei livelli di traccia: Ogni messaggio di traccia deve essere associato con un **livello di traccia** compreso tra 1 e 5. Il livello di traccia normalmente è correlato ad un livello di dettaglio: i messaggi di livello 1 normalmente contengono meno dettagli rispetto ai messaggi di livello 2, che contengono meno dettagli rispetto ai messaggi di livello 3, ecc. Pertanto, se si attiva la traccia a livello 1, sono visualizzati i messaggi che contengono meno dettagli rispetto ai messaggi del livello 5. Tuttavia, è possibile assegnare i livelli secondo le modalità ritenute più opportune. Di seguito sono riportati alcuni consigli:

- E' possibile assegnare lo stesso livello a tutti i messaggi di traccia.
- E' possibile assegnare i livelli di traccia in base al livello di dettaglio, come nell'ambiente di collaborazione runtime.
- E' possibile assegnare i livelli di messaggio in base all'oggetto business interessato: messaggi di traccia di livello 1 per un determinato oggetto business, messaggi di traccia di livello 2 per un altro oggetto business, ecc.

Quando si attiva la traccia per un determinato livello, sono visualizzati i messaggi associati al livello specificato e tutti quelli associati ai livelli inferiori. Ad esempio, la traccia a livello 2 visualizza i messaggi di livello 2 e 1.

Suggerimenti: Assicurarsi di riportare nella documentazione i livelli di traccia, in modo che gli utenti possano sapere quale livello utilizzare in base alle esigenze.

Generazione di un messaggio di traccia: Di seguito viene riportato un esempio di messaggio e la chiamata al metodo che genera il messaggio. Il messaggio che si trova nel file di messaggi è il seguente:

```
20
Configuration property DO_VERIFICATION = {1}
```

La chiamata al metodo ottiene il valore della proprietà di configurazione DO_VERIFICATION, quindi utilizza il valore per sostituire il parametro nel messaggio. Il codice presente nella collaborazione è il seguente, e il messaggio viene visualizzato quando l'utente imposta il livello di traccia 3:

```
String validateProp = getConfigProperty("DO_VERIFICATION");
trace(3, 20, validateProp);
```

Il seguente esempio ottiene il valore dell'attributo Salary dell'oggetto business Employee e prende una decisione in base al valore di Salary. Il messaggio che si trova nel file di messaggi è il seguente:

```
15
Salary {1} {2}
```

L'esempio invia un messaggio di traccia che riporta il valore di Salary e il percorso scelto.

```
int newsalary = employee.getInt("Salary");
String sal = Integer.toString(newsalary);
if (newsalary <150000)
    {
     trace (3, 15, sal, "do extra check");
    }
else
    {
     trace (3, 15, sal, "take normal path");
    }
```

Messaggi di traccia generati da InterChange Server Express

L'ambiente di collaborazione runtime di InterChange Server Express dispone di un componente di traccia che fornisce i messaggi relativi all'esecuzione di una collaborazione.

Il componente di traccia dell'ambiente runtime utilizza sei numeri per rappresentare i livelli di traccia. Il primo livello, zero, è l'impostazione predefinita e indica che non viene eseguita la traccia. I livelli da 1 a 5 indicano livelli di dettaglio crescenti. Una traccia di livello 1 fornisce il dettaglio minore e la traccia di livello 5 fornisce il dettaglio maggiore.

Per attivare la traccia, modificare il livello di traccia dell'oggetto di collaborazione da zero ad un numero superiore. Singoli messaggi di traccia sono associati a ciascun livello. La Tabella 53 descrive i tipi di messaggi che sono visualizzati per ciascun livello.

Tabella 53. Livelli di traccia per traccia generata dal sistema

Livello	Traccia generata dal sistema InterChange Server Express
0	Nessuna.
1	La ricezione di un evento di attivazione e l'avvio di uno scenario.
2	L'avvio e il completamento di uno scenario, riporta sia l'esecuzione che il rollback.
3	L'esecuzione di un nodo azione.
4	L'invio di un oggetto business in una chiamata di servizio e la ricezione di una risposta.
5	Una versione dettagliata della traccia di livello 4. Questa traccia stampa il valore di tutti gli attributi dell'oggetto business inviato e ricevuto.

Recupero di una proprietà di configurazione collaborazione

Per recuperare la proprietà di configurazione collaborazione, utilizzare il metodo `getConfigProperty()`.

Il seguente esempio mostra come una collaborazione utilizza una proprietà di configurazione per determinare il percorso del codice.

```
if (getConfigProperty("CONVERT_NEGQTY").equals("true")) {
// take this code path
}
else {
// take this code path
}
```

Per confrontare un valore di proprietà con un valore specifico, utilizzare sempre il metodo `equals()`, come mostrato nell'esempio. Non utilizzare l'operatore di eguaglianza condizionale `==`, che verifica se due variabili fanno riferimento allo stesso oggetto, invece di verificare che due oggetti contengono gli stessi valori. (Per un elenco completo degli operatori disponibili, consultare Tabella 37 e Tabella 38).

Notare che per il valore viene considerata la distinzione maiuscolo/minuscolo. Il formato maiuscolo/minuscolo del parametro di configurazione deve essere identico al formato indicato nel codice che verifica l'eguaglianza. Nel precedente esempio, il valore ″True″ non rispetterebbe l'eguaglianza.

Una proprietà di configurazione può anche essere costituita da un vettore di valori, separati da punti e virgola. Per ulteriori informazioni, fare riferimento a "getConfigPropertyArray()" a pagina 370.

Riutilizzo delle istanze di un oggetto di collaborazione

Generalmente, InterChange Server Express crea un'istanza di un oggetto di collaborazione per ogni evento di attivazione. Quando l'istanza completa la gestione dell'evento di attivazione, il sistema libera le risorse e le restituisce al pool di risorse libere Java. Tuttavia, JDK non sempre elimina le istanze in modo efficiente, provocando un utilizzo eccessivo di memoria.

Per ridurre l'utilizzo di memoria, InterChange Server Express utilizza l'opzione Riutilizzo istanza della collaborazione, che consente al sistema di riutilizzare un'istanza di una oggetto di collaborazione mettendola in cache e utilizzandola di nuovo per lo stesso tipo di oggetto di collaborazione in un momento successivo. Se InterChange Server Express riesce a riutilizzare un'istanza di collaborazione esistente, è possibile evitare:

- Il sovraccarico della creazione di istanze per gli oggetti di collaborazione
- La dipendenza dal programma di raccolta dati da eliminare di JDK per la gestione della memoria

Il sistema utilizza automaticamente l'opzione Riutilizzo istanza della collaborazione se la maschera di collaborazione rispetta *entrambi* i seguenti requisiti:

- La collaborazione *non* contiene variabili (globali) di maschera.
- La collaborazione è stata compilata con una versione di Process Designer Express *successiva* a 3.0.

Se queste condizioni *non* sono rispettate, l'opzione Riutilizzo istanza della collaborazione non viene utilizzata. Quindi, per trarre vantaggio da questa opzione, evitare di utilizzare variabili (globali) di maschera nel codice della maschera di collaborazione. Una variabile di maschera è una variabile dichiarata con ambito riferito all'intera maschera di collaborazione. Una variabile di maschera viene dichiarata nell'area con etichetta "Variabili globali:" nella scheda Dichiarazioni della finestra Definizioni.

Se la collaborazione richiede variabili di maschera e si desidera comunque utilizzare l'opzione Riutilizzo istanza della collaborazione, assicurarsi che la maschera di collaborazione rispetti i seguenti requisiti di programmazione:

- Evitare di inizializzare le variabili di maschera al momento della dichiarazione. Assicurarsi, invece, che tutte le variabili di maschera siano inizializzate nel primo nodo della maschera di collaborazione.
- Se la maschera di collaborazione utilizza una variabile di maschera per fare riferimento ad un oggetto LOB, assicurarsi che questa variabile sia reimpostata a `null`:
  - prima dell'uscita dalla collaborazione
  - prima che sia generata un'eccezione

Importante

Una maschera di collaborazione, che contiene variabili di maschera che *non* sono inizializzate al primo nodo, non può essere riutilizzata perché i valori delle variabili nell'istanza di oggetto di collaborazione in cache vengono mantenuti quando si riutilizza l'istanza. Quando l'istanza di collaborazione in cache viene riutilizzata e inizia l'esecuzione, ogni variabile di maschera contiene il valore presente alla fine del precedente utilizzo dell'istanza di collaborazione.

Dopo avere scritto il codice della maschera di collaborazione con l'inizializzazione corretta delle variabili di maschera, eseguire le seguenti attività per abilitare l'opzione Riutilizzo istanza della collaborazione:

1. Definire una proprietà di configurazione specifica denominata `EnableInstanceReuse` e impostare il relativo valore predefinito a `true` o `false`.

   Le proprietà di configurazione specifiche della collaborazione sono definite nella finestra Definizioni maschera. Impostare il valore predefinito di `EnableInstanceReuse` in funzione del comportamento desiderato degli oggetti di collaborazione:

   - Per forzare il riutilizzo dell'istanza per *tutti* gli oggetti di collaborazione della maschera di collaborazione, impostare il valore predefinito di `EnableInstanceReuse` su `true`.
   - Per forzare il riutilizzo dell'istanza solo per particolari oggetti di collaborazione della maschera di collaborazione, impostare il valore predefinito di `EnableInstanceReuse` su `false`.
2. Assicurarsi che ogni oggetto di collaborazione da riutilizzare abbia la propria proprietà di collaborazione `EnableInstanceReuse` impostata su `true`.

   I valori delle proprietà di configurazione specifiche della collaborazione sono impostate nella scheda Proprietà della finestra Proprietà oggetto di collaborazione di System Manager. Per ulteriori informazioni, consultare il manuale *System Administration Guide*.

Se non è possibile modificare il codice della collaborazione in modo da rispettare i requisiti di programmazione precedentemente indicati, *non* utilizzare l'opzione Riutilizzo istanza della collaborazione. Per disabilitare questa opzione, non definire la proprietà di configurazione collaborazione `EnableInstanceReuse` per la maschera di collaborazione.

Nota: Per attivare l'opzione Riutilizzo istanza della collaborazione è necessario arrestare e avviare l'oggetto di collaborazione. L'opzione diventa effettiva solo dopo la successiva riattivazione della collaborazione.

Il software utilizza una cache, denominata *cache di istanza di collaborazione*, per contenere le istanze degli oggetti di collaborazione. La dimensione della cache di istanza di collaborazione deriva dal valore di "Numero massimo di eventi simultanei", che viene configurato nella scheda Generale della finestra Proprietà oggetto di collaborazione di System Manager. Potrebbe essere necessario ridimensionare la cache di istanza di collaborazione, in base al tipo di modello di elaborazione flusso utilizzato per eseguire le collaborazioni: attivazione da evento o attivazione da chiamata.

Il ridimensionamento della cache di istanza di collaborazione richiede la definizione di una proprietà di configurazione collaborazione denominata `CollaborationInstanceCacheSize`. Questa proprietà viene definita insieme alle altre proprietà di collaborazione nell'area "Proprietà" della scheda Generale nella finestra Definizioni maschera. Dopo aver definito `CollaborationInstanceCacheSize`, impostarne il valore predefinito compatibilmente con il numero di istanze di collaborazione. Per ulteriori informazioni, fare riferimento all'opzione Riutilizzo istanza della collaborazione nel manuale *System Administration Guide*.

Richiamo di una mappa nativa

Normalmente, il richiamo di mappe e mappe secondarie viene effettuato solo all'interno di una mappa. Tuttavia, a volte la collaborazione potrebbe avere la necessità di richiamare una mappa nativa di InterChange Server Express direttamente. Il richiamo di una mappa nativa da una collaborazione offre diversi vantaggi:

- Facilità di trasformazioni generiche
- Conversione tra diverse strutture di applicazioni

Il richiamo della mappa nativa da una maschera di collaborazione prevede diverse fasi:

- "Creazione di una proprietà di collaborazione per il nome mappa"
- "Inizializzazione della collaborazione"
- "Richiamo della mappa" a pagina 215
- "Valorizzazione della variabile di collaborazione" a pagina 217

Creazione di una proprietà di collaborazione per il nome mappa

E' possibile creare una proprietà di collaborazione che contiene il nome della mappa da richiamare. Questo non è richiesto, ma consente di evitare la ricompilazione del codice di collaborazione se il nome della mappa viene modificato. In questo caso, se si modifica il nome della mappa, si deve semplicemente modificare il valore della proprietà di collaborazione. Ad esempio, è possibile definire una proprietà di collaborazione denominata `MAP_NAME` per contenere il nome della mappa da richiamare. Le proprietà di collaborazione sono definite nella finestra Definizioni maschera. Per ulteriori informazioni, consultare "Definizione delle proprietà di configurazione della collaborazione (scheda Proprietà)" a pagina 94.

Nota: Utilizzare il metodo `getConfigProperty()` per ottenere il valore di questa proprietà di collaborazione all'interno del codice della maschera di collaborazione.

Inizializzazione della collaborazione

L'inizializzazione della maschera di collaborazione per il richiamo della mappa comporta l'importazione delle classi Java dell'API di mappatura nella maschera di

collaborazione. Le mappe di InterChange Server Express richiedono alcune classi Java per essere eseguite. Alcune di queste classi *non* sono incluse automaticamente in una maschera di collaborazione. Affinché sia possibile eseguire la mappa, è necessario importare esplicitamente i seguenti pacchetti e classi della mappa:

- Classe `CxCommon.CxExecutionContext`
- Pacchetto `CxCommon.Exceptions`
- Pacchetto `CxCommon.Dtp`
- Pacchetto `CxCommon.BaseRunTimes`
- Pacchetto `DLM`

Importare ciascun elemento nella sezione di importazione della scheda Dichiarazioni nella finestra Definizioni maschera. Ad esempio, aggiungere le seguenti voci nella tabella di importazione per importare le classi di mappa nella maschera di collaborazione:

```
CxCommon.CxExecutionContext
CxCommon.Exceptions.*
CxCommon.Dtp.*
CxCommon.BaseRunTimes.*
DLM.*
```

Nota: Assicurarsi di rispettare la sintassi con ".*" dopo il nome del pacchetto per importare *tutte* le classi presenti nel pacchetto.

Per informazioni generali sulle modalità di importazione delle classi Java, consultare "Importazione di pacchetti Java" a pagina 89.

Richiamo della mappa

Per richiamare una mappa, utilizzare il metodo `runMap()` della classe dell'API di mappatura, `DtpMapService`. Il metodo `runMap()` richiede che le seguenti informazioni siano passate come argomenti:

- Il nome della mappa da eseguire
- Il tipo di mappa, sempre rappresentato come `CWMAPTYPE` per le mappe native di InterChange Server Express
- Un vettore di oggetti business di input, che contiene gli oggetti business di origine per la mappa
- Un contesto di esecuzione della mappa

Quindi, è necessario inizializzare queste informazioni all'interno della collaborazione *prima* della chiamata a `runMap()`.

Ottenimento del nome della mappa: Per passare il nome della mappa da eseguire, è possibile inserire direttamente nel codice il nome della mappa nella chiamata a `runMap()`. Tuttavia, per una maggiore flessibilità, si può utilizzare una proprietà della collaborazione per contenere il nome della mappa. Questo richiede la seguente procedura:

- Impostare una proprietà di collaborazione, come descritto in "Creazione di una proprietà di collaborazione per il nome mappa" a pagina 214
- Utilizzare il metodo `getConfigProperty()` per ottenere il valore di questa proprietà di collaborazione all'interno del codice della maschera di collaborazione.

La Figura 79 mostra una riga di codice che ottiene il nome della mappa memorizzato nella proprietà di collaborazione `MAP_NAME`.

```
String map_name = getConfigProperty("MAP_NAME");
```

Figura 79. Ottenimento del nome della mappa da eseguire

Nota: In questo esempio di suppone che sia stata creata la proprietà di collaborazione MAP_NAME e che ad essa sia stato assegnato il nome corretto della mappa da eseguire.

Inizializzazione del vettore di input: Il metodo runMap() richiede un vettore di input, che contiene gli oggetti business di origine per la mappa. Normalmente, una mappa trasforma un singolo oggetto business di origine. Per questo tipo di mappa, il matrice di input è composto da un solo elemento. Quando si richiama una mappa all'interno di una collaborazione, generalmente si vuole collocare una copia dell'oggetto business di attivazione in questo vettore di input. Quindi, questo vettore di input viene passato come terzo argomento a runMap().

La Figura 80 mostra una riga di codice che inizializza il vettore di input con una copia dell'oggetto business di attivazione della collaborazione.

```
BusObj[] sourceBusObjs = { inputBusObj };
```

Figura 80. Inizializzazione del vettore di input con l'oggetto business di attivazione

Preparazione del contesto di esecuzione della mappa: Un'istanza di mappa viene eseguita in un contesto di esecuzione specifico della mappa, che contiene le informazioni necessarie alla mappa, come indicato di seguito:

- Il contesto di chiamata indica la condizione che ha attivato questo richiamo della mappa. I contesti di chiamata sono rappresentati come costanti predefinite nella classe MapExeContext.
- L'oggetto business della richiesta originale è una copia dell'oggetto business associato con questo richiamo della mappa.

L'API di mappatura rappresenta un contesto di esecuzione di mappa come oggetto MapExeContext. All'interno del codice della mappa, è sempre possibile ottenere il contesto di esecuzione della mappa dalla variabile generata dal sistema, cwExecCtx. Tuttavia, questa variabile generata dal sistema non è accessibile dalla maschera di collaborazione. La collaborazione deve eseguire le seguenti operazioni:

- Creare un'istanza dell'oggetto MapExeContext con il costruttore MapExeContext().
- Assegnare l'oggetto MapExeContext al contesto di esecuzione globale.

  La classe CxExecutionContext rappresenta il contesto di esecuzione globale della collaborazione. Pertanto, per inizializzare il contesto di esecuzione della mappa, è necessario eseguire le seguenti operazioni:

  – Creare un'istanza dell'oggetto CxExecutionContext con il costruttore CxExecutionContext().
  – Asssegnare l'oggetto MapExeContext all'oggetto CxExecutionContext con il metodo setContext().
- Fornire all'oggetto MapExeContext il contesto di chiamata.

  Il metodo setInitiator() della classe MapExeContext imposta il contesto di chiamata (in precedenza veniva definito "iniziatore mappa"). Per ulteriori informazioni sui contesti di esecuzione di mappa e sui metodi della classe MapExeContext, consultare il manuale *Guida allo sviluppo delle mappe*.

La Figura 81 mostra un frammento di codice che inizializza il contesto di esecuzione della mappa con il contesto di chiamata EVENT_DELIVERY (convertendo

da un oggetto business specifico di applicazione ad un oggetto business generico) e un oggetto business della richiesta originale dell'oggetto business di attivazione.

```
// Instantiate objects for the map execution context and the global
// execution context
map_exe_context = new MapExeContext();
global_exe_context = new CxExecutionContext();

// Assign the map execution context to the global execution context
global_exe_context.setContext(
   CxExecutionContext.MAPCONTEXT,
   map_exe_context);

// Initialize the map execution context
map_exe_context.setInitiator(MapExeContext.EVENT_DELIVERY);
```

Figura 81. Inizializzazione del contesto di esecuzione della mappa

Richiamo del metodo runMap(): Dopo che la maschera di collaborazione ha inizializzato le informazioni di mappa, può effettuare la chiamata alla mappa. La chiamata al metodo `runMap()` richiede due operazioni:

- Richiamare il metodo `runMap()`.

  Questo metodo è un metodo statico all'interno della classe API di mappatura, `DtpMapService`. Quindi, non è necessario creare un'istanza di `DtpMapService`.
- Fornire un vettore di output per i valore restituiti da `runMap()`.

  Oltre al vettore di input, il metodo `runMap()` richiede anche un vettore di output, che viene valorizzato da `runMap()` con gli oggetti business di destinazione della mappa e restituito al codice chiamante. Normalmente, una mappa genera un singolo oggetto business di destinazione. Per questo tipo di mappa, il vettore di output è composto da un solo elemento.

La Figura 82 mostra la chiamata a `runMap()` per eseguire la mappa con le seguenti caratteristiche:

- Il nome della mappa è identificato dalla proprietà di collaborazione `MAP_NAME` (Figura 79).
- L'oggetto business di destinazione della mappa è l'oggetto business di attivazione della collaborazione (Figura 80).
- Il contesto di esecuzione della mappa viene inizializzato ad un contesto di chiamata `EVENT_DELIVERY` e l'oggetto business di attivazione come oggetto business della richiesta originale (Figura 81).

```
BusObj[] destinationBusObjs = DtpMapService.runMap(
   map_name,
   CWMAPTYPE,
   sourceBusObjs,
   global_exe_context);
```

Figura 82. Chiamata a runMap() per eseguire la mappa

Valorizzazione della variabile di collaborazione

Gli oggetti business di destinazione della mappa sono disponibili nel vettore di output restituito da `runMap()`. Per inviare un oggetto business di destinazione al di fuori della collaborazione, è necessario copiarlo dal vettore di output in una variabile di collaborazione opportuna. Questa variabile di collaborazione è normalmente associata con la porta `A` della collaborazione. Pertanto, l'oggetto business di destinazione viene di solito copiato nella variabile di collaborazione `ToBusObj`.

La Figura 83 utilizza il metodo `BusObj.copy()` per copiare il singolo oggetto business di destinazione restituito dalla chiamata a `runMap()` nella Figura 82 alla variabile di collaborazione `ToBusObj`.

```
ToBusObj.copy(destinationBusObjs[0]);
```

Figura 83. Valorizzazione della variabile di collaborazione

Operazioni sugli oggetti business

Questa sezione descrive le operazioni relative alla gestione degli oggetti business e dei relativi valori. Le operazioni sono:

Creazione di un nuovo oggetto business

Utilizzare il metodo del costruttore `new` per creare un nuovo oggetto business.

Sintassi

```
new BusObj(String busObjType)
```

Parametri

busObjType Il nome di una definizione di oggetto business

Valore di ritorno

Un oggetto di tipo `BusObj`.

Eccezioni

`ObjectException` – Generata se l'argomento dell'oggetto business non è valido.

Note

Questo metodo crea un oggetto business privo di valori. Il valori sono impostati quando si valorizzano gli attributi.

Se un attributo nel nuovo oggetto business viene definito come oggetto business secondario o vettore oggetti business secondario, è necessario creare esplicitamente l'oggetto business secondario ed associarlo all'attributo; per ulteriori informazioni, fare riferimento a "Creazione di un oggetto business secondario in un nuovo oggetto business" a pagina 219.

Esempio

Il seguente esempio utilizza la definizione di oggetto business Customer per creare un nuovo oggetto business denominato `destinationCustomer`. Il nuovo oggetto business viene creato, ma non presenta valori di attributi.

```
BusObj destinationCustomer = new BusObj("Customer");
```

Creazione di un oggetto business secondario in un nuovo oggetto business

Quando si crea un nuovo oggetto business, un attributo che viene definito per contenere un oggetto business secondario di cardinalità 1 o n non presenta alcun valore. Per assegnare un valore a tale attributo, è necessario creare esplicitamente un oggetto business secondario o un vettore oggetti business secondario e poi associarlo all'attributo. Questa sezione illustra come eseguire questa operazione per un singolo oggetto business secondario o per un vettore.

Creazione di un singolo oggetto business secondario

Di seguito viene illustrato prima come creare un nuovo oggetto business, quindi come creare un oggetto business secondario contenuto in uno dei relativi attributi:

1. Utilizzare il metodo `new` per creare un oggetto business principale.
2. Utilizzare il metodo `new` per creare un oggetto business secondario del tipo per il quale è stato definito l'attributo.
3. Utilizzare metodo `BusObj.set()` per impostare il valore dell'attributo nell'oggetto principale sul nuovo oggetto business secondario.

Il seguente esempio illustra la creazione di un nuovo oggetto business Invoice, che presenta un attributo denominato `SoldToAddressAttribute`. Questo attributo contiene l'indirizzo del cliente ed è un oggetto business di tipo Address. L'esempio mostra l'associazione tra un oggetto business principale e l'oggetto business secondario.

```
// Declarations
BusObj invoice = new BusObj("Invoice");
// Create child business object in invoice
invoice.set("SoldToAddressAttribute", new BusObj("Address"));
```

Se la collaborazione richiede la gestione dell'oggetto business secondario, l'esempio è il seguente:

```
// Declarations
BusObj invoice = new BusObj("Invoice");
BusObj soldToAddress = new BusObj("Address");
// Manipulate child business object soldToAddress
// Associate child business object soldToAddress with parent invoice
invoice.set("SoldToAddressAttribute", soldToAddress);
```

Creazione di un vettore oggetti business secondario

In questa sezione, viene illustrato prima come creare un nuovo oggetto business, quindi come creare un vettore oggetti business secondario contenuto in uno dei relativi attributi:

1. Utilizzare il metodo `new` per creare un oggetto business. Questo rappresenta l'oggetto business principale.
2. Per l'attributo dell'oggetto principale che è stato definito per contenere un oggetto business con cardinalità uguale a n, creare un oggetto business del tipo specificato dall'attributo.
3. Impostare il valore dell'attributo dell'oggetto principale al nuovo oggetto business singolo.

4. Dichiarare un oggetto BusObjArray, ottenere il valore dell'attributo e assegnarlo al vettore.

Quindi è possibile utilizzare i metodi della classe BusObjArray per aggiungere elementi o per eseguire altre operazioni sul vettore oggetti business.

Il seguente esempio illustra la creazione di un nuovo oggetto business Bill of Materials, la creazione di un vettore oggetti business secondario per il relativo attributo LineItems e il posizionamento di ulteriori oggetti business nel vettore.

```
// Declarations
BusObj bom = new BusObj("Bill_Of_Materials");
BusObjArray lineItemArray = null;
BusObj singleLineItem = new BusObj ("LineItem");
// Create first child item
bom.set("LineItemsAttribute", singleLineItem);
//If there are additional line items, do this once
lineItemArray = bom.getBusObjArray("LineItemAttribute");
// Now do this for each additional child item
lineItemArray.addElement(new BusObj("singleLineItem"));
```

Copia dell'evento di attivazione

La prima azione di ogni scenario dovrebbe gestire l'attivazione flusso dello scenario. Process Designer Express dichiara automaticamente una variabile di tipo BusObj denominata triggeringBusObj; questa variabile contiene l' attivazione flusso (evento di attivazione o chiamata di accesso di attivazione) che determina l'esecuzione dello scenario.

Inoltre, Process Designer Express dichiara automaticamente le variabili di maschera BusObj per ciascuna porta definita. Il nome della variabile BusObj corrisponde al nome della porta, con accodato BusObj.

Ad esempio, si supponga che l'evento di attivazione per uno scenario sia Customer.Create. La porta che riceve l'evento di attivazione è denominata SourceCust. Process Designer Express dichiara automaticamente una variabile denominata SourceCustBusObj, combinando il nome della porta con il suffisso BusObj. In questo caso, Process Designer Express visualizza la seguente dichiarazione di variabile:

```
BusObj SourceCustBusObj = new BusObj("Customer");
```

E' possibile aggiungere il seguente frammento di codice per copiare l'evento di attivazione in SourceCustBusObj.

```
SourceCustBusObj.copy(triggeringBusObj);
```

Molte collaborazioni sono attivate da più tipi di oggetti business. E' necessario determinare il tipo di oggetto business prima di poter creare un oggetto business per contenere l'attivazione flusso. Utilizzare il metodo BusObj.getType() sull'attivazione flusso, prima per accertarne il tipo, poi per creare il tipo di oggetto

business appropriato e infine per copiare l'attivazione flusso al nuovo oggetto business creato.

```
sourceBusObj = new BusObj(triggeringBusObj.getType());
sourceBusObj.copy(triggeringBusObj);
```

Suggerimenti: Si consiglia di copiare la variabile `triggeringBusObj` in un'altra variabile prima di eseguire qualsiasi operazione su di essa.

Copia o duplicazione di un oggetto business

Per spostare i valori di un oggetto business ad un altro si possono utilizzare due metodi: `copy()` e `duplicate()`. Entrambi i metodi gestiscono l'intera gerarchia di un oggetto business, copiando l'oggetto business principale e tutti i relativi oggetti secondari. La Tabella 54 descrive questi metodi.

Tabella 54. Confronto dei metodi copy() e duplicate()

Metodo	Descrizione
`copy()`	Imposta tutti i valori degli attributi di un oggetto business esistente su quelli di un altro oggetto business.
`duplicate()`	Crea un nuovo oggetto business clonando un oggetto business esistente. Riproduce sia i valori degli attributi che l'istruzione. Il valore di ritorno di questo metodo deve essere assegnato ad una variabile.

La differenza tra i due metodi è principalmente nella pre-esistenza di un'oggetto business nel momento in cui si immettono i valori copiati.

Copia

Prima di utilizzare il metodo `copy()`, l'oggetto business in cui copiare i valori deve già esistere. Utilizzare il metodo `new` per creare l'oggetto business se non esiste.

Il seguente esempio copia i valori degli attributi contenuti in un oggetto business Customer ricevuto dall'applicazione di origine in un oggetto business da inviare all'applicazione di destinazione:

```
BusObj destination = new BusObj("Customer");
destination.copy(sourceBusObj);
```

Nota: Il metodo `copy()` copia l'intero oggetto business, inclusi tutti gli oggetti business secondari e i vettori di oggetti business secondari. Questo metodo *non* imposta un riferimento all'oggetto copiato. Al contrario, clona tutti gli attributi, ovvero crea copie separate degli attributi.

Duplicazione

Al contrario del metodo `copy()`, `duplicate()` crea un clone completo di un oggetto business esistente e lo restituisce.

Il seguente esempio clona l'oggetto business di origine e lo assegna ad una variabile denominata `destination`:

```
BusObj destination = sourceBusObj.duplicate();
```

Utilizzo dei valori degli attributi

Le collaborazioni spesso recuperano i valori degli attributi contenuti negli oggetti business ricevuti.

Se una collaborazione deve effettuare un'operazione su un valore di attributo ricevuto, deve assicurarsi che il valore dell'attributo non sia null prima di utilizzarlo (consultare "Verifica dei valori null" a pagina 222).

Verifica dei valori null

Per verificare se il valore di un attributo è null, una collaborazione richiama `BusObj.isNull(attribute)`. Generalmente la presenza di un valore null è dovuta a uno dei seguenti motivi:

- L'attributo non è mai stato impostato.

  Al momento della creazione di un oggetto business, tutti gli attributi presentano il valore null e lo mantengono fino a quando questo non viene esplicitamente impostato. Questo è valido anche per gli oggetti business secondari e per i vettori oggetto business secondari.
- L'attributo è stato impostato esplicitamente su `null` dal metodo `BusObj.set()`.
- Durante il processo di mappatura, nell'oggetto business di input non era presente alcun valore da associare a questo attributo.

Il seguente esempio di codice imposta un valore in un oggetto business Billing, utilizzando i dati ricevuti in un oggetto business Order. Se i dati di Order ricevuti sono null o vuoti, il codice imposta l'attributo su "Unknown".

```
//Check whether ProductLine is null or blank; if so, default it
if (order.isNull("ProductLine") || order.isBlank("ProductLine"))
     {
     logInfo("Setting ProductLine to default Unknown");
     order.set("ProductLine", "Unknown");
     }
//Now set Billing object's equivalent attribute to the same value
billing.set("ProductLine", order.get("ProductLine");
```

Il metodo `isNull()` può essere utilizzato per tutti i tipi di attributi, inclusi quelli che contengono oggetti business secondari o vettori oggetto business secondari.

Confronto di un valore di attributo con un valore noto

Una collaborazione può utilizzare il metodo `equals()` del linguaggio di programmazione Java per verificare il valore di un attributo rispetto ad uno specifico valore atteso. Il metodo `equals()` confronta il valore atteso con il valore dell'attributo recuperato, come mostrato nell'esempio seguente. Il metodo `equals()` viene chiamato per l'oggetto noto e lo confronta con l'attributo sconosciuto.

In questo esempio, Smith è il valore atteso nell'attributo LName.

```
String name = "Smith";
boolean checkName=name.equals(CustBusObj.get("LName"));
```

Recupero di un attributo

Questa sezione illustra i diversi tipi di operazioni per il recupero degli attributi. Queste operazioni sono presentate in ordine di complessità crescente:

- Recupero di un valore di attributo con tipo base
- Recupero di un attributo in un oggetto business secondario

Recupero di un attributo con tipo base: I metodi `BusObj.get()` recuperano i valori di attributo che appartengono ai tipi base. I tipi base per un attributo sono i tipi primitivi supportati, come mostrato nella Tabella 55.

Tabella 55. Recupero di attributi con tipi di dati base

Tipo di dati base	Metodo per l'impostazione dell'attributo
`boolean`	`getBoolean()`
`double`	`getDouble()`
`float`	`getFloat()`
`int`	`getInt()`
`long`	`getLong()`
`Object`	`get()`
`LongText`	`getLongText()`
`String`	`getString()`

Il seguente esempio recupera il contenuto dell'attributo Credit-Limit, di tipo `int`.

```
int creditLimit = customer.getInt("Credit-Limit");
```

Se non si conosce il tipo di dati dell'attributo, utilizzare il modulo del metodo `get()` che recupera un tipo di dati `Object` Java.

Nota: Il metodo `get()` restituisce una copia dell'attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito da `get()`. Ogni volta che si utilizza questo metodo, viene restituita una nuova copia (clone) dell'attributo.

Recupero di un attributo da un oggetto business secondario: Se un attributo è di tipo oggetto business, la cardinalità definita nella definizione dell'oggetto business specifica se l'attributo contiene un singolo oggetto business o un vettore. Se la cardinalità è:

- 1, l'attributo contiene un singolo oggetto business. Assegnare il valore restituito da `getBusObj()` ad un oggetto `BusObj`.
- n, l'attributo contiene un vettore di oggetti business. Assegnare il valore restituito da `getBusObjArray()` ad un oggetto `BusObjArray`.

Il seguente esempio recupera l'oggetto business Address con cardinalità singola contenuto nell'attributo SoldToAddress di un oggetto business Bill of Materials.

```
BusObj addr = new BusObj("Address");
addr = bom.getBusObj("SoldToAddress");
```

Il seguente esempio ricerca gli indirizzi di vendita (SoldTo) negli Stati Uniti. La struttura dell'oggetto business è la seguente:

- BusOrg è l'oggetto business di livello principale.
- BusOrg presenta un attributo denominato SoldToSite, che è un oggetto business a cardinalità multipla.
- SoldToSite presenta un attributo denominato SoldToAddress, che è un oggetto business a cardinalità singola.
- SoldToAddress contiene un attributo denominato CountryName.

```
//Look for sold-to address in the US
//Start with the busOrg business object
//Get the child business object array in the "SoldToSite" attribute
if (!busOrg.isNull("SoldToSite"))
    {
    BusObjArray siteAddArray = busOrg.getBusObjArray("SoldToSite");
    //
    //String to compare with sold-to country name
    String countryName = "USA";
    //Get size of child business object array
    int count = siteAddArray.size();
    //
    //For each business object in the array get the SoldToAddress
    //attribute, which is a business object, and compare its
    //SoldToCountryName attribute to the string "USA"
    //
    for (int i = 0; i < count ; i ++)
        {
         BusObj siteAddr = siteAddArray.elementAt( i );
         if (!siteAddr.isNull("SoldToAddress"))
            {
            BusObj soldToAddress =
                siteAddr.getBusObj("SoldToAddress");
            if (countryName.equalsIgnoreCase(
                    soldToAddress.getString("SoldToCountryName")))
               {
               //do something
               }
            }//end if

        }//end for
    }//end if
```

Impostazione dei valori di attributo

Questa sezione illustra tre tipi di operazioni per l'impostazione degli attributi. Queste operazioni sono presentate in ordine di complessità crescente:

- Impostazione di un valore di attributo con tipo base
- Impostazione di un attributo in un oggetto business secondario singolo
- Impostazione di un attributo in un oggetto business secondario che fa parte di un vettore oggetti business

Impostazione di un attributo con tipo base

I metodi `BusObj.set()` impostano i valori di attributo che appartengono ai tipi base. I tipi base per un attributo sono i tipi primitivi supportati: `boolean`, `double`, `float`, `int`, `long`, `Object` e `String`.

Tutti i metodi `set()` hanno lo stesso nome ma firme differenti: il primo parametro è sempre una stringa (String) che contiene il nome dell'attributo il cui valore deve essere impostato. Il secondo parametro è una variabile o una costante di tipo primitivo, un oggetto String o un tipo Object. Qualsiasi sia il tipo della variabile si richiama sempre lo stesso metodo `set()`; il metodo è in grado di accettare una variabile di qualsiasi tipo. Il compilatore determina quale variante del metodo utilizzare.

Il seguente esempio imposta i valori di due attributi: `FirstName` e `Salary`.

```
Customer.set("FirstName", "Sue");
Customer.set("Salary", 30500);
```

E' possibile utilizzare i metodi `BusObj.get()` e `BusObj.set()` per copiare i valori degli attributi da un oggetto business ad un altro. Il seguente esempio riceve le variabili String `HeaderId` e `ServiceId` dall'oggetto business di origine ed imposta gli attributi `HeaderId` e `SalesNum` dell'oggetto business di destinazione a tali valori.

```
DestinationBusObj.set(
    "HeaderId", SourceBusObj.getString("HeaderId"));
DestinationBusObj.set(
    "SalesNum", SourceBusObj.getString("ServiceId"));
```

Nota: Il metodo `set()` imposta un riferimento di oggetto al valore quando il valore viene assegnato all'attributo. Il valore dell'attributo *non* viene clonato dall'oggetto business di origine. Di conseguenza, le modifiche al valore nell'oggetto business di origine vengono effettuate anche all'attributo nell'oggetto business che richiama `set()`.

Impostazione di un attributo in un oggetto business secondario singolo

Per impostare un attributo ad un oggetto business secondario la cui cardinalità è 1, è necessario prima ottenere un handle per l'oggetto business secondario.

Utilizzare metodo `BusObj.getBusObj()` per ottenere un handle o un riferimento all'oggetto business secondario e assegnare i risultati ad una variabile di tipo `BusObj`. Quindi chiamare il metodo `BusObj.set()`; questo richiama la versione del metodo che corrisponde al tipo di dati dell'attributo.

L'esempio seguente si basa sulla Figura 84. L'oggetto business `Customer` presenta un attributo denominato `Address`, che rappresenta un riferimento ad un oggetto business secondario denominato `CustAddress`.

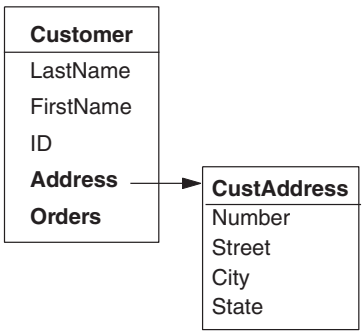


Figura 84. Oggetto business secondario singolo

L'esempio di codice riportato esegue le seguenti operazioni:

1. Dichiara una variabile denominata `addr`
2. Recupera l'attributo `Address` dell'oggetto business `Customer` e lo assegna a `addr`
3. Imposta l'attributo `City`

```
BusObj addr = Customer.getBusObj("Address"); addr.set("City", "SF");
```

Impostazione di un attributo in un vettore di oggetti business secondari

Per impostare un attributo in un vettore di oggetti business secondari (un attributo con cardinalità uguale a n), utilizzare il metodo `BusObj.getBusObjArray()` ed assegnare i risultati ad una variabile di tipo `BusObjArray`. Quindi, utilizzare i metodi `BusObjArray` su tale variabile.

Gli esempi di codice riportati di seguito si basano sulla struttura mostrata.

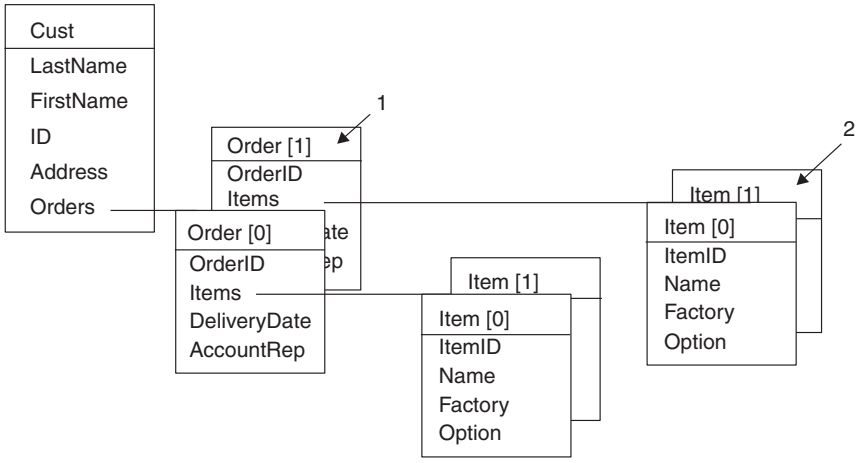


Figura 85. Vettori di oggetti business secondari

Il seguente esempio imposta l'attributo `OrderID` dell'oggetto business contrassegnato con 1 nella Figura 85.

```
BusObjArray[] orders =
     cust.getBusObjArray("Orders").elementAt(1);
orders[1].set("OrderID", "x1234");
```

Il seguente esempio imposta l'attributo `Factory` dell'oggetto business contrassegnato con 2 nella Figura 85.

```
BusObjArray items = orders[1].getBusObjArray("Items");
BusObj item = items.elementAt(1);
item.set("Factory", "MyCompany");
```

Impostazione di un valore di attributo a null

Il seguente esempio imposta il valore dell'attributo Total dell'oggetto business Order su null:

```
order.set("Total", null);
```

E' possibile utilizzare questa tecnica per impostare qualsiasi tipo di attributo su null, se il valore dell'attributo è un tipo base, un tipo `BusObj` o un tipo `BusObjArray`. Tuttavia, non è possibile utilizzarla per impostare su null oggetti business secondari in un vettore.

Esecuzione di query sul database

Durante l'esecuzione di una collaborazione, potrebbe essere necessario ottenere informazioni da un database, ad esempio un database relazionale. Per ottenere o modificare le informazioni presenti in un database, si effettuano query sulle tabelle del database. Una **query** è una richiesta, generalmente sotto forma di un'istruzione SQL (Structured Query Language), che viene inviata al database per essere eseguita. La Tabella 56 mostra le attività richieste per eseguire una query sul database.

Nota: E' possibile accedere a tutti i database esterni supportati da InterChange Server Express con JDBC mediante il driver thin di tipo 4 e un driver MS-SQL Server di tipo 4 di InterChange Server Express.

Tabella 56. Fasi di esecuzione di una query

Attività di esecuzione di una query		Per ulteriori informazioni
1.	Ottenere una connessione (un oggetto `CwDBConnection`) al database.	"Ottenimento di una connessione" a pagina 228
2.	Mediante l'oggetto `CwDBConnection`, inviare query e gestire le transazioni nel database.	"Esecuzione della query" a pagina 228"Gestione della transazione" a pagina 240
3.	Rilasciare la connessione.	"Rilascio di una connessione" a pagina 244

Suggerimenti: Uno dei possibili usi delle query di database è la gestione delle chiamate di servizio con latenza prolungata. Dopo che la collaborazione ha effettuato una richiesta di servizio che può richiedere un tempo elevato, la collaborazione salva il contesto di esecuzione utilizzando una connessione al database e poi termina.

La risposta effettiva alla chiamata di servizio, che può richiedere ore o persino giorni, ritorna come nuovo evento e attiva un'altra collaborazione, che ripristina il contesto di esecuzione appropriato dal database e riprende l'esecuzione del processo business.

Ottenimento di una connessione

Per poter eseguire query nel database, è necessario prima ottenere una connessione al database con il metodo `getDBConnection()` della classe `BaseCollaboration`. Per identificare la connessione da ottenere, specificare il nome del pool di connessioni che contiene questa connessione. Tutte le connessioni in un particolare pool di connessioni sono relative allo stesso database. Il numero di connessioni presenti in un pool viene determinato durante la configurazione del pool di connessioni. E' necessario determinare il nome del pool di connessioni che contiene le connessioni per il database su cui eseguire query.

Importante: Le connessioni sono aperte quando si avvia InterChange Server Express o dinamicamente quando si configura un nuovo pool di connessioni. Pertanto, il pool di connessioni che contiene il database desiderato, deve essere configurato prima dell'esecuzione dell'oggetto di collaborazione che richiede la connessione. I pool di connessioni sono configurati all'interno di System Manager. Per ulteriori informazioni, consultare il manuale *System Implementation Guide*.

Nella Figura 86, la chiamata a `getDBConnection()` ottiene una connessione al database associato con le connessioni nel pool `CustDBConnPool`.

```
CwDBConnection connection = getDBConnection("CustDBConnPool");
```

Figura 86. Ottenimento di una connessione da un pool di connessioni

La chiamata a `getDBConnection()` restituisce un oggetto `CwDBConnection` nella variabile `connection`, che può essere poi utilizzata per accedere al database associato alla connessione.

Suggerimenti: Il metodo `getDBConnection()` fornisce un ulteriore modulo che consente di specificare il modello di programmazione transazione per la connessione. Per ulteriori informazioni, consultare "Gestione della transazione" a pagina 240.

Esecuzione della query

La Tabella 57 mostra le modalità con le quali eseguire query SQL con i metodi della classe `CwDBConnection`.

Tabella 57. Esecuzione di 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 l'esecuzione iniziale, l'istruzione SQL viene salvata in formato compilato eseguibile, in modo che le successive esecuzioni possano utilizzare il formato precompilato.	`"executePreparedSQL()" a pagina 427`
Procedura memorizzata	Una procedura definita dall'utente che contiene istruzioni SQL e logica condizionale	`executeSQL() "executePreparedSQL()" a pagina 427 executeStoredProcedure()`

Esecuzione di query statiche

Il metodo `executeSQL()` invia una query statica al database per l'esecuzione. Una **query statica** è un'istruzione SQL inviata come stringa al database, che analizza la stringa ed esegue l'istruzione SQL risultante. Questa sezione illustra le modalità di invio di diversi 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 i dati da una o più tabelle. Per inviare un'istruzione SELECT al database per l'esecuzione, specificare una rappresentazione in formato stringa della SELECT come argomento del metodo `executeSQL()`. Ad esempio, la seguente chiamata a `executeSQL()` invia una SELECT di un valore di colonna dalla tabella `customer`:

```
connection.executeSQL(
   "select cust_id from customer where active_status = 1");
```

Nota: Nel codice sopra riportato la variabile `connection` è un oggetto `CwDBConnection` ottenuto da una precedente chiamata al metodo `getDBConnection()` (consultare Figura 86).

E' possibile anche inviare un'istruzione SELECT che contiene parametri utilizzando il secondo modulo del metodo `executeSQL()`. Ad esempio, la seguente chiamata a `executeSQL()` esegue le stesse attività del precedente esempio, ma passa lo stato attivo come parametro per l'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 di dati che soddisfano le condizioni nella clausola WHERE della SELECT. Ogni riga contiene i valori delle colonne specificate nell'istruzione SELECT. I dati restituiti possono essere visualizzati come un vettore a due dimensioni di queste righe e colonne.

Suggerimenti: La sintassi dell'istruzione SELECT deve essere valida per il particolare database a cui si accede. Consultare la documentazione del database per verificare la sintassi esatta dell'istruzione SELECT.

Per accedere ai dati restituiti, eseguire le seguenti operazioni:

1. Ottenere una riga di dati.
2. Ottenere i valori delle colonne, uno alla volta.

La Tabella 58 mostra i metodi della classe `CwDBConnection` che forniscono accesso alle righe dei dati della query restituita.

Tabella 58. Metodi CwDBConnection di accesso alle righe

Attività di accesso alle righe	Metodo CwDBConnection
Verificare l'esistenza di una riga.	`hasMoreRows()`
Ottenere una riga di dati.	`nextRow()`

Controllare il ciclo di ricezione delle righe con il metodo `hasMoreRows()`. Terminare il loop di ricezione delle righe quando `hasMoreRows()` restituisce `false`. Per ottenere una riga di dati, utilizzare il metodo `nextRow()`. Questo metodo restituisce i valori di colonna selezionati come elementi in un oggetto `Vector` Java. E' possibile quindi utilizzare la classe `Enumeration` per accedere ai singoli valori di colonna. Le classi `Vector` e `Enumeration` si trovano nel pacchetto `java.util`.

La Tabella 59 mostra i metodi Java per accedere alle colonne di un riga di query restituita.

Tabella 59. Metodi Java per accedere ai valori di colonna

Attività di accesso alle colonne	Metodo Java
Determinare il numero di colonne.	`Vector.size()`
Eseguire il casting di Vector a Enumeration.	`Vector.elements()`
Verificare l'esistenza di una colonna.	`Enumeration.hasMoreElements()`
Ottenere una colonna di dati.	`Enumeration.nextElement()`

Controllare il ciclo di ricezione dei valori di colonna con il metodo `hasMoreElements()`. Terminare il loop di ricezione delle colonne quando `hasMoreElements()` restituisce `false`. Per ottenere un valore di colonna, utilizzare il metodo `nextElement()`.

Il seguente esempio di codice riceve un'istanza della classe `CwDBConnection`, che è una connessione al database che contiene le informazioni sui clienti. Quindi esegue un'istruzione SELECT che restituisce una sola riga, che contiene una sola colonna, il nome dell'azienda (company_name) "`CrossWorlds`" per l'id cliente (cust_id) 20987:

```
CwDBConnection connectn = null;
Vector theRow = null;
Enumeration theRowEnum = null;
String theColumn1 = null;

try
   {
   // Obtain a connection to the database
   connectn = getDBConnection("sampleConnectionPoolName");
   }

catch(CwDBConnectionFactoryException e)
   {
   System.out.println(e.getMessage());
   throw e;
   }

// Test for a resulting single-column, single-row, result set
try {
   // Send the SELECT statement to the database
   connectn.executeSQL(
      "select company_name from customer where cust_id = 20987");

   // Loop through each row
   while(connectn.hasMoreRows())
      {
      // Obtain one row
      theRow = connectn.nextRow();
      int length = 0;
      if ((length = theRow.size())!= 1)
         {
         return methodName + "Expected result set size = 1," +
            " Actual result state size = " + length;
```

```
         }

      // Get column values as an Enumeration object
      theRowEnum = theRow.elements();

      // Verify that column values exist
      if (theRowEnum.hasMoreElements())
         {
         // Get the column value
         theColumn1 = (String)theRowEnum.nextElement();
         if (theColumn1.equals("CrossWorlds")==false)
            {
            return "Expected result = CrossWorlds,"
               + " Resulting result = " + theColumn1;
            }
         }
      }
   }

// Handle any exceptions thrown by executeSQL()
catch(CwDBSQLException e)
   {
   System.out.println(e.getMessage());
   }
```

Il seguente esempio mostra un frammento di codice per un'istruzione SELECT che restituisce più righe e ogni riga contiene due colonne, l'id cliente (cust_id) e il nome dell'azienda (company_name) associato:

```
CwDBConnection connectn = null;
Vector theRow = null;
Enumeration theRowEnum = null;
Integer theColumn1 = 0;
String theColumn2 = null;

try
   {
   // Obtain a connection to the database
   connectn = getDBConnection("sampleConnectionPoolName");
   }

catch(CwDBConnectionFactoryException e)
   {
   System.out.println(e.getMessage());
   throw e;
   }

// Code fragment for multiple-row, multiple-column result set.
// Get all rows with the specified columns, where the
// specified condition is satisfied
try
   {
   connectn.executeSQL(
"select cust_id, company_name from customer where active_status = 1");

   // Loop through each row
   while(connectn.hasMoreRows())
      {
      // Obtain one row
      theRow = connectn.nextRow();

      // Obtain column values as an Enumeration object
      theRowEnum = theRow.elements();
      int length = 0;
      if ((length = theRow.size()) != 2)
      {
         return "Expected result set size = 2," +
```

```
            " Actual result state size = " + length;
      }
      // Verify that column values exist
      if (theRowEnum.hasMoreElements())
         {
         // Get the column values
         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, *non* è normalmente necessario eseguire la gestione transazioni per le istruzioni SELECT.

Esecuzione di query statiche che modificano dati: Le istruzioni SQL che modificano i dati in una tabella di database sono le seguenti:

- INSERT aggiunge nuove righe ad una tabella di database.
- UPDATE modifica le righe esistenti di una tabella di database.
- DELETE elimina le righe da una tabella di database.

Per inviare una di queste istruzioni come query statica al database per l'esecuzione, specificare una rappresentazione in formato stringa della istruzione come argomento del metodo `executeSQL()`. Ad esempio, la seguente chiamata a `executeSQL()` invia una 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 sopra riportato 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 modificate o aggiunte con il metodo `getUpdateCount()`.

Importante: Dal momento che le istruzioni INSERT, UPDATE e DELETE modificano il contenuto del database, è necessario applicare la gestione transazioni per queste istruzioni. Per ulteriori informazioni, consultare "Gestione della transazione" a pagina 240.

Esecuzione di una procedura memorizzata statica: E' possibile utilizzare il metodo `executeSQL()` per eseguire una chiamata ad una procedura memorizzata se si verificano *entrambe* le condizioni riportate di seguito:

- La procedura memorizzata *non* utilizza parametri OUT.

  Se la procedura memorizzata utilizza parametri OUT, si *deve* utilizzare `executeStoredProcedure()` per eseguirla.
- La procedura memorizzata viene richiamata una sola volta.

  Il metodo `executeSQL()` *non* salva l'istruzione preparata per la chiamata alla procedura memorizzata. Quindi, se si richiama la stessa procedura memorizzata più volte, (ad esempio, in un loop), l'utilizzo di `executeSQL()` può rallentare

l'esecuzione rispetto ad un metodo che salva l'istruzione preparata: `executePreparedSQL()` o `executeStoredProcedure()`.

Per ulteriori informazioni, consultare "Esecuzione di procedure memorizzate" a pagina 235.

Esecuzione di query preparate

Il metodo "executePreparedSQL()" a pagina 427 invia una query preparata al database per l'esecuzione. Una **query preparata** è un'istruzione SQL che è già stata precompilata nel formato eseguibile utilizzato dal database. La prima volta che `executePreparedSQL()` invia una query al database, la invia sotto forma di stringa. Il database riceve la query, la compila in formato eseguibile analizzando la stringa ed esegue l'istruzione SQL risultante (così come `executeSQL()`). Tuttavia, il database restituisce questo formato compilato dell'istruzione SQL a `executePreparedSQL()`, che lo conserva in memoria. Questa istruzione SQL compilata è detta **istruzione preparata**.

Nelle successive esecuzioni della stessa query, `executePreparedSQL()` verifica prima se un'istruzione preparata esiste già per la query. Se l'istruzione preparata esiste, `executePreparedSQL()` la invia al database al posto della stringa della query. Le successive esecuzioni di questa query sono più efficienti poiché il database non deve analizzare la stringa e creare l'istruzione preparata.

Con `executePreparedSQL()` è possibile inviare al database i seguenti tipi di query SQL:

- 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 la stessa istruzione SELECT deve essere eseguita più volte, utilizzare `executePreparedSQL()` per creare una versione precompilata dell'istruzione. Per preparare un'istruzione SELECT si deve considerare quanto segue:

- E' possibile utilizzare parametri nell'istruzione SELECT per passare informazioni specifiche per ciascuna esecuzione dell'istruzione preparata. Per un esempio dell'utilizzo dei parametri con un'istruzione preparata, consultare Figura 87.
- Quando si esegue un'istruzione SELECT con `executePreparedSQL()`, si utilizzano sempre gli stessi metodi per accedere ai dati restituiti (Tabella 58 e Tabella 59). Per ulteriori informazioni, consultare "Esecuzione di query statiche che restituiscono dati (SELECT)" a pagina 229.

Esecuzione di query preparate che modificano dati: Se la stessa istruzione INSERT, UPDATE o DELETE deve essere eseguita più volte, utilizzare `executePreparedSQL()` per creare una versione precompilata dell'istruzione. L'istruzione SQL che viene rieseguita *non* deve essere esattamente la stessa per ciascuna esecuzione per poter trarre vantaggio dall'istruzione preparata. E' possibile utilizzare parametri nell'istruzione SQL per fornire dinamicamente informazioni sull'esecuzione di ciascuna istruzione.

Il frammento di codice nella Figura 87 inserisce 50 righe nella tabella `employee`. La prima volta che viene chiamato il metodo `executePreparedSQL()`, invia la versione in formato stringa dell'istruzione INSERT al database, che la analizza, la esegue e restituisce il formato eseguibile: un'istruzione preparata. Le successive 49 volte che viene eseguita l'istruzione INSERT (assumendo che tutte le 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
      {
      // Send the INSERT statement to the database
      connection.executePreparedSQL(
"insert into employee (employee_id, employee_number) values (?, ?)",
          argValues);

      // Increment the argument values
      emp_id++;
      emp_num++
      }

   catch(CwDBSQLException e)
      {
      System.out.println(e.getMessage());
      }
   }
```

Figura 87. Passaggio di valori di argomenti ad un'istruzione preparata

Suggerimenti: L'esecuzione della versione preparata dell'istruzione INSERT normalmente migliora le prestazioni dell'applicazione, anche se aumenta l'utilizzo di memoria dell'applicazione.

Quando si esegue di nuovo un'istruzione SQL che modifica il database, è comunque necessario gestire le transazioni in base al modello di programmazione transazione. Per ulteriori informazioni, consultare "Gestione della transazione" a pagina 240.

Nota: Per semplificare il codice nella Figura 87 *non* includere la gestione transazioni.

Esecuzione di una procedura memorizzata preparata: E' possibile utilizzare il metodo executePreparedSQL() per eseguire una chiamata ad una procedura memorizzata se si verificano *entrambe* le condizioni riportate di seguito:

- La procedura memorizzata *non* contiene parametri OUT.

  Se la procedura memorizzata utilizza parametri OUT, si *deve* utilizzare executeStoredProcedure() per eseguirla.
- La procedura memorizzata viene richiamata più di una volta.

  Il metodo executePreparedSQL() salva l'istruzione preparata per la procedura memorizzata in memoria. Quindi, se la procedura memorizzata viene richiamata una sola volta, l'utilizzo di executePreparedSQL() può richiedere più memoria rispetto al richiamo della procedura memorizzata con executeSQL(), che non salva l'istruzione preparata.

Per ulteriori informazioni, consultare “Esecuzione di procedure memorizzate” a pagina 235.

Esecuzione di procedure memorizzate

Una **procedura memorizzata** è procedura definita dall'utente che contiene istruzioni SQL e logica condizionale. Le procedure memorizzate sono memorizzate nel database insieme a i dati.

Nota: Quando si crea una nuova relazione, Relationship Designer Express crea una procedura memorizzata per la gestione di ciascuna tabella di relazione.

La Tabella 60 mostra i metodi della classe `CwDBConnection` che richiamano una procedura memorizzata.

Tabella 60. Metodi CwDBConnection per il richiamo di una procedura memorizzata

Modalità di richiamo della procedura memorizzata	Metodo CwDBConnection	Utilizzo
Invio al database di un'istruzione CALL per eseguire la procedura memorizzata.	`executeSQL()`	Per richiamare una procedura memorizzata che *non* presenta parametri OUT e viene eseguita *una sola volta*
	`"executePreparedSQL()" a pagina 427`	Per richiamare una procedura memorizzata che *non* presenta parametri OUT e viene eseguita *più di una volta*
Specifica del nome della procedura memorizzata e di un vettore dei relativi parametri per creare la chiamata di procedura, che viene inviata al database per l'esecuzione.	`executeStoredProcedure()`	Per richiamare qualsiasi procedura memorizzata, incluse quelle con parametri OUT

Nota: E' possibile utilizzare i metodi JDBC per eseguire direttamente una procedura memorizzata. Tuttavia, l'interfaccia fornita dalla classe `CwDBConnection` è più semplice e riutilizza le risorse del database, aumentando l'efficienza di esecuzione. E' possibile utilizzare i metodi della classe `CwDBConnection` per eseguire procedure memorizzate.

Una procedura memorizzata può restituire i dati sotto forma di una o più righe. In questo caso, utilizzare i metodi Java (ad esempio `hasMoreRows()` e `nextRow()`) per accedere a queste righe restituite nei risultati della query, nello stesso modo in cui si accede ai dati restituiti da un'istruzione SELECT. Per ulteriori informazioni, consultare “Esecuzione di query statiche che restituiscono dati (SELECT)” a pagina 229.

Come mostrato nella Tabella 60, la scelta del metodo da utilizzare per richiamare una procedura memorizzata dipende da più fattori:

- L'utilizzo da parte della procedura di parametri OUT

  Un parametro OUT è un parametro tramite il quale la procedura memorizzata restituisce un valore al codice chiamante. Se la procedura memorizzata utilizza parametri OUT, si *deve* utilizzare `executeStoredProcedure()` per richiamare la procedura memorizzata.
- Il numero di volte che una procedura memorizzata viene richiamata

  Il metodo `executeStoredProcedure()` salva la versione compilata della procedura memorizzata. Quindi, se si richiama la stessa procedura memorizzata più volte,

(ad esempio, in un loop), l'utilizzo di executeStoredProcedure() può essere più veloce di executeSQL() in quanto il database può riutilizzare la versione precompilata.

Le seguenti sezioni descrivono le modalità di utilizzo dei metodi executeSQL() e executeStoredProcedure() per richiamare una procedura memorizzata.

Richiamo di procedure memorizzate senza parametri OUT: Per richiamare una procedura memorizzata che *non* presenta alcun parametro OUT, è possibile utilizzare i seguenti metodi di CwDBConnection:

- 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. Questa istruzione preparata *non* viene salvata. Quindi, executeSQL() è utile per una procedura memorizzata che deve essere richiamata una sola volta.
- Il metodo "executePreparedSQL()" a pagina 427 invia una chiamata di procedura memorizzata preparata al database.

  Al primo richiamo, questa chiamata di procedura viene inviata al database, che crea l'istruzione preparata e la esegue. Tuttavia, il database restituisce questa istruzione preparata a executePreparedSQL(), che la salva in memoria. Quindi, executePreparedSQL() è utile per una procedura memorizzata che deve essere richiamata più di una volta (ad esempio, in loop).

Per richiamare una procedura memorizzata con uno di questi metodi, specificare come argomento per il metodo una rappresentazione in formato stringa dell'istruzione CALL che include la procedura memorizzata e gli altri argomenti. Nella Figura 88, la chiamata a executeSQL() invia un'istruzione CALL per eseguire la procedura memorizzata setOrderCurrDate().

```
connection.executeSQL("call setOrderCurrDate(345698)");
```

Figura 88. Richiamo di una procedura memorizzata con executeSQL()

Nella Figura 88, 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() perché il suo solo argomento è un parametro IN; quindi, il valore viene *solo* inviato alla procedura memorizzata. Questa procedura memorizzata *non* presenta alcun parametro OUT.

E' possibile utilizzare il modulo di executeSQL() o executePreparedSQL() che accetta un vettore di parametri per passare i valori degli argomenti alla procedura memorizzata. Tuttavia, *non* è possibile utilizzare questi metodi per richiamare una procedura memorizzata che utilizza parametri OUT. Per eseguire una procedura memorizzata di questo tipo, si *deve* utilizzare executeStoredProcedure(). Per ulteriori informazioni, consultare "Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237.

Nota: Utilizzare un blocco anonimo PL/SQL se si prevede di chiamare oggetti memorizzati PL/SQL Oracle tramite ODBC utilizzando il metodo CwDBConnection.executeSQL(). Di seguito viene riportato un esempio di formato ammesso (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, incluse quelle che utilizzano parametri OUT. Questo metodo salva l'istruzione preparata per la chiamata di procedura memorizzata, così come il metodo `executePreparedSQL()`. Quindi, `executeStoredProcedure()` può migliorare le prestazioni di una chiamata di procedura memorizzata che viene eseguita più volte.

Per richiamare una procedura memorizzata con il metodo `executeStoredProcedure()`:

1. Specificare il nome della procedura memorizzata da eseguire come `String`.
2. Creare un vettore di parametri `Vector` per gli oggetti `CwDBStoredProcedureParam`, che fornisce le informazioni sui parametri: il tipo di parametro in/out e il valore di ciascun parametro della procedura memorizzata.

Un **parametro** è un valore che è possibile inviare o ricevere dalla 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 *di solo input*: quindi la procedura memorizzata accetta il suo valore come input, ma *non* utilizza il parametro per restituire un valore al codice chiamante.
- Un parametro OUT è *di solo output*: quindi la procedura memorizzata *non* considera il suo valore come input, ma utilizza il parametro per restituire un valore al codice chiamante.
- Un parametro INOUT è *di input e output*: quindi la procedura memorizzata accetta il suo valore come input ed utilizza anche il parametro per restituire un valore al codice chiamante.

Un oggetto `CwDBStoredProcedureParam` descrive un singolo parametro di una procedura memorizzata. La Tabella 61 mostra le informazioni sul parametro contenute in un oggetto `CwDBStoredProcedureParam` oltre ai metodi per recuperare ed impostare queste informazioni.

Tabella 61. Informazioni di parametro in un oggetto CwDBStoredProcedureParam

Informazioni sul parametro	Metodo CwDBStoredProcedureParam
Valore del parametro	`getValue()`
Tipo di parametro in/out	`getParamType()`

Per passare parametri ad una procedura memorizzata con `executeStoredProcedure()`:

1. Creare un oggetto `CwDBStoredProcedureParam` per contenere 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 è un parametro IN, INOUT o OUT.
   - Il valore del parametro è un tipo di dati Java che contiene il valore da assegnare al parametro. La classe `CwDBStoredProcedureParam` fornisce molte versioni del costruttore per supportare i diversi tipi di dati che possono essere associati al valore del parametro. Per un parametro OUT, questo

valore di parametro può essere un valore 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 ciascun parametro della procedura memorizzata.
3. Creare un oggetto `Vector` con elementi sufficienti a contenere tutti i parametri della procedura memorizzata.
4. Aggiungere l'oggetto inizializzato `CwDBStoredProcedureParam` all'oggetto di parametri `Vector`.

   Utilizzare il metodo `addElement()` o `add()` della classe `Vector` per aggiungere l'oggetto `CwDBStoredProcedureParam`.
5. Dopo aver creato tutti gli oggetti `CwDBStoredProcedureParam` ed averli aggiunti al vettore di parametri `Vector`, passare il vettore di parametri come secondo argomento al metodo `executeStoredProcedure()`.

   Il metodo `executeStoredProcedure()` invia la procedura memorizzata e i relativi parametri al database per l'esecuzione.

Ad esempio, si supponga di avere la procedura memorizzata `get_empno()` definita nel 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()` presenta due parametri:

- Il primo parametro, `emp_id`, è un parametro IN.

  Quindi, è necessario inizializzare l'oggetto associato `CwDBStoredProcedureParam` con un tipo in/out `PARAM_IN`, e con il valore appropriato da inviare alla procedura memorizzata. Dal momento che `emp_id` è stato dichiarato come tipo SQL NUMBER (che contiene un numero intero), il valore del parametro è un `Object` Java che contiene i valori interi: `Integer`.
- Il secondo parametro, `emp_number`, è un parametro OUT.

  Per questo parametro, creare un oggetto `CwDBStoredProcedureParam` *vuoto* da inviare alla procedura memorizzata. Questo oggetto viene inizializzato con un tipo `PARAM_OUT`. Comunque, viene fornito un valore `Integer` fittizio per questo parametro. Dopo che la procedura memorizzata ha terminato l'esecuzione, è possibile ottenere il valore restituito da questo parametro OUT con il metodo getValue().

La Figura 89 la procedura memorizzata `get_empno()` viene eseguita con il metodo `executeStoredProcedure()` per ottenere il numero di impiegato per l'id impiegato 65:

```
CwDBConnection connectn = null;

try
   {
   // Get database connection
   connectn = getDBConnection("CustomerDBPool");

   // Create parameter Vector
   Vector paramData = new Vector(2);

   // Create IN parameter for the employee id and add to parameter
   // vector
   paramData.add(
      new CwDBStoredProcedureParam(PARAM_IN, new Integer(65)));

   // Create dummy argument for OUT parameter and add to parameter
   // vector
   paramData.add(
      new CwDBStoredProcedureParam(PARAM_OUT, new Integer(0));

   // Call the get_empno() stored procedure
   connectn.executeStoredProcedure("get_empno", paramData);

   // Get the result from the OUT parameter
   CwDBStoredProcedureParam outParam =
      (CwDBStoredProcedureParam) paramData.get(1);
   int emp_number = ((Integer) outParam.getValue().Intvalue();
   }
```

Figura 89. Esecuzione della procedura memorizzata get_empno()

Suggerimenti: L'oggetto `Vector` Java è un vettore a base zero. Nel precedente esempio di codice, per accedere al valore del parametro OUT dal vettore di parametri `Vector`, la chiamata `get()` specifica un valore di indice 1, in quanto questo vettore `Vector` è a base zero.

Una procedura memorizzata elabora i suoi parametri come tipi di dati SQL. Dal momento che i tipi di dati SQL e Java *non* sono identici, il metodo `executeStoredProcedure()` deve convertire un valore di parametro tra i due tipi di dati. Per un parametro IN, `executeStoredProcedure()` converte il valore del parametro da un tipo di dati Java al relativo tipo di dati SQL. Per un parametro OUT, `executeStoredProcedure()` converte il valore del parametro da un tipo di dati SQL ad un tipo di dati Java.

Il metodo `executeStoredProcedure()` utilizza internamente il tipo di dati JDBC per contenere il valore del parametro inviato o ricevuto da una procedura memorizzata. JDBC definisce un insieme di identificativi di tipi SQL generici nella classe `java.sql.Types`. Questi tipi rappresentano i tipi SQL più comuni. JDBC inoltre fornisce la mappatura standard dai tipi JDBC ai tipi di dati Java. Ad esempio, un INTEGER JDBC è normalmente mappato ad un tipo `int` Java. Il metodo `executeStoredProcedure()` utilizza le mappature mostrate nella Tabella 62.

Tabella 62. Mappature tra 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
`java.math.BigDecimal`	NUMERIC

Tabella 62. Mappature tra tipi di dati Java e JDBC (Continua)

Tipo di dati Java	Tipo di dati JDBC
`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 costituita da un insieme di operazioni che sono eseguite con una singola unità. Tutte le istruzioni SQL che sono eseguite all'interno di una transazione sono completate positivamente o negativamente con una sola unità. Questa sezione fornisce le seguenti informazioni sulla gestione delle transazioni:

- "Determinazione del modello di programmazione transazione"
- "Specifica dell'ambito della transazione" a pagina 241

Determinazione del modello di programmazione transazione

Il raggruppamento delle fasi di esecuzione delle operazioni del database in transazioni è detto **bracketing transazioni**. Ad ogni connessione è associato uno dei seguenti modelli di programmazione transazione:

- Bracketing transazioni implicite — le operazioni di database fanno parte di una **transazione implicita**, che inizia al momento dell'acquisizione della connessione e termina quando si rilascia la connessione; il bracketing transazione è implicitamente gestito da InterChange Server Express.
- Bracketing transazioni esplicite — le operazioni di database fanno parte di una **transazione esplicita**, per la quale l'inizio e la fine di ciascuna transazione viene determinato da programma.

Al runtime, un oggetto di collaborazione determina quale modello di programmazione transazione utilizzare con ciascuna connessione acquisita. Per impostazione predefinita, un oggetto di collaborazione presume che *tutte* le connessioni acquisite utilizzino il bracketing transazioni implicito come modello di programmazione transazione. E' possibile sostituire il modello di programmazione transazione predefinito con una delle modalità indicate nella Tabella 63.

Tabella 63. Sostituzione del modello di programmazione transazione per una connessione

Modello di programmazione transazione da sostituire	Azione da eseguire
Per specificare un modello di programmazione transazione diverso *per tutte le connessioni* ottenute da un particolare oggetto di collaborazione	Selezionare o deselezionare la casella Transazione database implicita nella finestra di dialogo Proprietà della collaborazione di System Manager. Per ulteriori informazioni, consultare il manuale *System Implementation Guide*.

Tabella 63. Sostituzione del modello di programmazione transazione per una connessione (Continua)

Modello di programmazione transazione da sostituire	Azione da eseguire
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 al metodo `getDBConnection()`. La seguente chiamata `getDBConnection()` specifica bracketing transazioni esplicite per la connessione ottenuta dal pool di connessioni `ConnPool`: `conn = getDBConnection("ConnPool", false);`

E' possibile determinare il modello di programmazione transazione corrente che sarà utilizzato dalle connessioni con il metodo `BaseCollaboration.implicitDBTransactionBracketing()`, che restituisce un valore `boolean` che indica se il modello di programmazione transazione è il bracketing transazioni implicite.

Specifica dell'ambito della transazione

Il modello di programmazione transazione della connessione determina come viene specificato l'ambito della transazione del database. Questa sezione fornisce le seguenti informazioni:

- "Ambito della transazione con bracketing transazioni implicite"
- "Ambito della transazione con bracketing transazioni esplicite" a pagina 243

Ambito della transazione con bracketing transazioni implicite: InterChange Server Express effettua la gestione delle transazioni per tutte le collaborazioni. Tutte le azioni nel processo business della collaborazione sono completate come unità o non completate. Quindi, InterChange Server gestisce il processo business nella sua interezza come una singola transazione implicita. Se un'attività non riesce, l'utente può scegliere come gestire la collaborazione non riuscita mediante il pannello di selezione dei flussi non risolti.

Se una connessione utilizza il bracketing transazioni implicite, InterChange Server Express effettua anche la gestione delle transazioni per le operazioni eseguite su un database esterno, associato ad una connessione del pool di connessioni. Quando una collaborazione esegue operazioni sul database, tali operazioni sono considerate come parte del processo business della collaborazione. InterChange Server Express gestisce queste operazioni di database come una transazione implicita, considerata come transazione secondaria della transazione principale (il processo business della collaborazione). Questa transazione secondaria del database inizia nel momento in cui la collaborazione ottiene la connessione. InterChange Server Express termina la transazione secondaria implicitamente quando viene completata l'esecuzione della collaborazione.

L'esito positivo o negativo di questa transazione secondaria di database dipende dall'esito della transazione principale, secondo quanto riportato di seguito:

- Se la collaborazione ha avuto esito positivo, InterChange Server Express esegue il commit di questa transazione secondaria di database.

- Se la collaborazione ha avuto esito negativo, InterChange Server Express esegue il rollback di questa transazione secondaria di database. Se il rollback non riesce, InterChange Server Express genera l'eccezione `CwDBTransactionException` e registra un errore.

Quando una collaborazione richiama direttamente un'altra collaborazione, la prima collaborazione viene detta principale, l'altra viene detta secondaria. Quando una collaborazione principale richiama una collaborazione secondaria, InterChange Server Express effettua una gestione separata della transazione secondaria. L'esito positivo o negativo di questa collaborazione secondaria è indipendente dall'esito della collaborazione principale. Se la collaborazione secondaria ha esito negativo, la collaborazione principale può decidere come gestire questo esito. Ad esempio, può decidere di terminare anch'essa in errore oppure di correggere o ignorare la situazione e continuare l'esecuzione.

Tuttavia, anche se l'esito di una collaborazione secondaria è indipendente dalla collaborazione principale, lo stesso non è valido per le transazioni di database implicite che possono essere eseguite dalla collaborazione secondaria. Se la collaborazione secondaria esegue operazioni di database attraverso una connessione che utilizza il bracketing transazioni implicite, la collaborazione secondaria eredita la transazione della collaborazione principale. InterChange Server Express gestisce queste operazioni di database come una transazione secondaria della collaborazione principale. L'esito positivo o negativo della collaborazione principale (o di livello principale) determina lo stato finale della transazione per la transazione di database secondaria implicita nella collaborazione secondaria, secondo quanto riportato di seguito:

- Se la collaborazione ha esito positivo, InterChange Server Express esegue il commit della transazione di database secondaria.
- Se la collaborazione ha esito negativo, InterChange Server Express esegue il rollback della transazione di database secondaria.

InterChange Server Express non esegue il commit o rollback della transazione secondaria fino ha quando non viene stabilito l'esito della collaborazione principale.

In base a questo comportamento, se la collaborazione secondaria ha esito negativo e la collaborazione principale sceglie di continuare l'esecuzione, InterChange Server Express eseguirà il commit della transazione di database implicita della collaborazione secondaria.

Nota: InterChange Server Express gestisce le transazioni di database secondarie esplicite eseguite dalla collaborazione secondaria ed ancora attive (ovvero quelle eseguite attraverso una connessione di database che utilizza il bracketing transazioni esplicite ma per le quali non è stato eseguito esplicitamente il commit o rollback nella maschera di collaborazione secondaria) nello stesso modo in cui gestisce le transazioni di database secondarie implicite.

Questo metodo di gestione delle transazioni secondarie fornisce allo sviluppatore di collaborazioni un mezzo per eseguire l'unione transazionale dalla collaborazione secondaria alla principale senza utilizzare esplicitamente la semantica di unione. Se al contrario, per le transazioni di database secondarie viene eseguito il commit o il rollback a livello di collaborazione secondaria, lo sviluppatore non può stabilire

una correlazione tra le transazioni avviate dalla collaborazione secondaria (sia esplicitamente che implicitamente) e la transazione del processo business globale avviato dalla collaborazione principale.

Nota: Le collaborazioni transazionali utilizzano questo stesso modello per le loro transazioni di database principali e secondarie.

Ambito della transazione con bracketing transazioni esplicite: Se la connessione utilizza il bracketing transazioni esplicite, InterChange Server Express si aspetta che la maschera di collaborazione specifichi esplicitamente l'ambito di ciascuna transazione di database. Il bracketing transazioni esplicite è utile se deve essere eseguita un'attività di database che è indipendente dall'esito positivo o negativo della collaborazione. Ad esempio, se si deve eseguire il controllo per stabilire se è stato effettuato l'accesso a determinate tabelle, tale controllo deve essere eseguito qualsiasi sia l'esito dell'accesso alle tabelle. Se le operazioni di controllo del database sono contenute in una transazione esplicita, queste devono essere eseguite qualsiasi sia l'esito della collaborazione.

La Tabella 64 mostra i metodi della classe `CwDBConnection` che forniscono la gestione dei limiti della transazione per le transazioni esplicite.

Tabella 64. Metodi CwDBConnection per la gestione transazioni esplicite

Attività di gestione transazioni	Metodo CwDBConnection
Iniziare una nuova transazione.	`"beginTransaction()" a pagina 425`
Terminare la transazione, eseguendo il commit (salvando) tutte le modifiche al database effettuate durante la transazione.	`"commit()" a pagina 426`
Determinare se una transazione è attualmente attiva.	`inTransaction()`
Terminare la transazione, eseguendo il rollback (ripristinando) tutte le modifiche effettuate durante la transazione.	`rollBack()`

Per specificare l'ambito della specificare per una transazione esplicita, eseguire le seguenti operazioni:

1. Contrassegnare l'inizio della transazione con una chiamata al metodo `beginTransaction()`.
2. Eseguire tutte le istruzioni SQL come una unità di lavoro *compresa tra* questa chiamata a `beginTransaction()` e la fine della transazione.
3. Terminare la transazione in uno dei seguenti modi:
   - Chiamare `commit()` per terminare la transazione con esito positivo. Tutte le modifiche eseguite dalle istruzioni SQL sono *salvate* nel database.
   - Chiamare `rollBack()` per terminare la transazione con esito negativo. Tutte le modifiche eseguite dalle istruzioni SQL sono *ripristinate* nel database.

   L'utente può scegliere quali condizioni determinano l'esito negativo della transazione. Verificare la condizione e chiamare `rollBack()` se si riscontra una condizione di errore. Altrimenti, chiamare `commit()` per terminare la transazione con esito positivo.

Importante: Se *non* si utilizza `beginTransaction()` per specificare l'inizio di una transazione esplicita, il database esegue ogni istruzione SQL come una transazione separata. Se si utilizza `beginTransaction()` ma *non* si specifica il termine della transazione di database con `commit()` o `rollback()` prima che la connessione sia rilasciata, InterChange Server

Express termina implicitamente la transazione in base all'esito della collaborazione. Se la collaborazione ha avuto esito positivo, InterChange Server Express esegue il commit di questa transazione di database. Se la collaborazione *non* ha avuto esito positivo, InterChange Server Express esegue implicitamente il rollback della transazione di database. Indipendentemente dall'esito della collaborazione, InterChange Server Express registra un'avvertenza.

Il seguente frammento di codice aggiorna tre tabelle del database associato con le connessioni presenti 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 errori nella transazione, si verifica un'eccezione `CwDBTransactionException` e il frammento di codice richiama il metodo `rollback()`.

```
CwDBConnection connection = getDBConnection("CustDBConnPool", false);

// Begin a transaction
connection.beginTransaction();

// Update several tables
try
   {
   connection.executeSQL("update table1....");
   connection.executeSQL("update table2....");
   connection.executeSQL("update table3....");

   // Commit the transaction
   connection.commit();
   }

catch (CwDBSQLException e)
   {
   // Roll back the transaction if an executeSQL() call throws
   // an exception
   connection.rollback();
   }

// Release the database connection
connection.release();
```

Per determinare se una transazione è attualmente attiva, utilizzare il metodo `inTransaction()`.

Attenzione: Utilizzare i metodi `beginTransaction()`, `commit()` e `rollback()` *solo* se la connessione utilizza il bracketing transazioni esplicite. Se la connessione utilizza il bracketing transazioni implicite, l'uso di uno di questi metodi determina un'eccezione `CwDBTransactionException`.

Rilascio di una connessione

Dopo che una connessione viene rilasciata, ritorna al relativo pool di connessioni ed è disponibile per essere utilizzata da altri componenti. Il modo in cui viene rilasciata una connessione al database dipende al modello di programmazione transazione. Questa sezione fornisce le seguenti informazioni:

- "Rilascio di una connessione con bracketing transazioni implicite"
- "Rilascio di una connessione con bracketing transazioni esplicite" a pagina 245

Rilascio di una connessione con bracketing transazioni implicite

InterChange Server Express rilascia automaticamente una connessione che utilizza il bracketing transazioni implicite al termine della transazione di database.

InterChange Server Express non termina la transazione di database fino a quando non viene determinato l'esito positivo o negativo della collaborazione; InterChange Server Express rilascia quindi le connessioni quando la collaborazione termina l'esecuzione. Se la collaborazione viene eseguita con esito positivo, InterChange Server Express esegue automaticamente il commit di tutte transazioni di database ancora attive. Se l'esecuzione della collaborazione termina con esito negativo (ad esempio, se viene generata un'eccezione non gestita da un'istruzione `catch`), InterChange Server Express esegue automaticamente il rollback di tutte le transazioni ancora attive.

Rilascio di una connessione con bracketing transazioni esplicite

Per una connessione che utilizza bracketing transazioni esplicite, la connessione termina in uno dei seguenti casi:

- InterChange Server Express rilascia automaticamente una connessione che utilizza il bracketing transazioni esplicite.
- Una connessione viene rilasciata esplicitamente con il metodo `release()` della classe `CwDBConnection`.

E' possibile utilizzare il metodo `CwDBConnection.isActive()` per determinare se una connessione è stata rilasciata. Se la connessione è stata rilasciata, `isActive()` restituisce `false`, come mostrato nel seguente frammento di codice:

```
if (connection.isActive())
   connection.release();
```

Attenzione: *Non* utilizzare il metodo `release()` se una transazione è attualmente attiva. Con il bracketing transazioni implicite, InterChange Server Express non termina la transazione di database fino a quando non viene determinato l'esito positivo o negativo della collaborazione. Pertanto, l'utilizzo di questo metodo per una connessione che utilizza bracketing transazioni implicite genera una eccezione `CwDBTransactionException`. Se non si gestisce in modo esplicito questa eccezione, viene effettuato anche un rollback automatico della transazione attiva. E' possibile utilizzare il metodo `inTransaction()` per determinare se una transazione è attiva. InterChange Server Express rilascia automaticamente una connessione qualsiasi sia il modello di programmazione transazione utilizzato. Nella maggior parte dei casi, non è necessario rilasciare esplicitamente la connessione.

Capitolo 10. Creazione del file di messaggi

Una collaborazione utilizza metodi particolari per la generazione di messaggi. Per generare testi di messaggi visibili all'utente si utilizzano due metodi:

- La collaborazione richiama un metodo per la visualizzazione dei messaggi e include il testo del messaggio come parametro della chiamata.
- La collaborazione richiama il metodo di messaggistica e la chiamata contiene un riferimento ad un file di messaggi esterno che contiene il testo del messaggio.

Normalmente si consiglia di progettare una collaborazione che faccia riferimento ad un file di messaggi invece di generare direttamente il testo. Mantenendo i messaggi in un file di messaggi centralizzato, invece che nelle singole collaborazioni, si facilita la manutenzione, la gestione e l'internazionalizzazione.

Si consiglia di creare un file di messaggi per ciascuna maschera di collaborazione. Process Designer Express fornisce la vista Messaggi per consentire la creazione dei messaggi. Quando InterChange Server Express avvia un oggetto di collaborazione, cerca di caricare in memoria il file di messaggi associato. Se manca il file di messaggi, viene registrata un'avvertenza.

Questo capitolo descrive il funzionamento dei file di messaggi e le relative modalità di creazione e gestione. Sono illustrati i seguenti argomenti:

Operazioni che utilizzano il file di messaggi

Un file di messaggi può contenere il testo dei messaggi utilizzati in diversi tipi di operazioni. Nella Tabella 65 sono riportati i tipi di operazioni che utilizzano un file di messaggi e i metodi della classe BaseCollaboration che eseguono tali operazioni.

Tabella 65. Operazioni che generano messaggi

Operazione	Metodi
Registrazione log	BaseCollaboration.logInfo() BaseCollaboration.logError() BaseCollaboration.logWarning()
Traccia	BaseCollaboration.trace()
Generazione eccezioni	BaseCollaboration.raiseException()

Creazione di un file di messaggi

Per creare un file di messaggi, eseguire le seguenti operazioni:

1. Assicurarsi che Process Designer Express sia aperto.

2. Fare clic su Maschera —> Apri messaggi maschera. Viene visualizzata la finestra Messaggi maschera, come mostrato nella Figura 90.

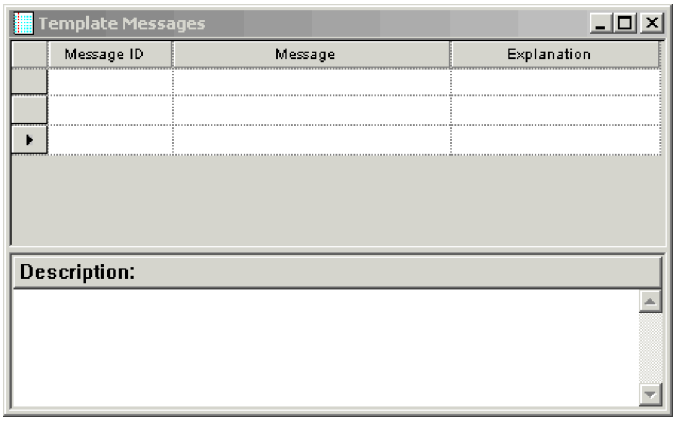


Figura 90. Finestra Messaggi maschera

3. Per ciascun messaggio da creare, eseguire le seguenti operazioni:
   a. Nella colonna ID messaggio, specificare l'identificativo univoco del messaggio.
   b. Nella colonna Messaggio, immettere il testo del messaggio. Questo è il testo che verrà presentato agli utenti quando il messaggio viene visualizzato al runtime. E' possibile includere parametri nel testo del messaggio per consentire il riutilizzo del messaggio. Consultare "Parametri del messaggio" a pagina 250.
   c. Facoltativamente, rendere il messaggio auto-documentante aggiungendo una spiegazione nella colonna Spiegazione. Consultare "Spiegazioni" a pagina 249.
   d. Facoltativamente, aggiungere una descrizione per il messaggio nel riquadro Descrizione. Il testo immesso nel riquadro non sarà visibile all'utente durante il runtime.

File di messaggi: nome e percorso

Il contenuto del file di messaggi della collaborazione viene archiviato come parte della maschera di collaborazione. Quando si compila e distribuisce una collaborazione, Process Designer Express estrae il contenuto del messaggio e crea o aggiorna il file di messaggi da utilizzare al runtime. Dopo la compilazione, il file di messaggi viene posto nella directory \Templates\messages del progetto ICL (Integration Component Library) in System Manager. Dopo la distribuzione, il file di messaggi viene copiato e collocato nella directory *productDir*\collaborations\messages.

Il nome del file di messaggi presenta il seguente formato:

CollaborationName.txt

Ad esempio, quando si compila e distribuisce una maschera di collaborazione denominata SampleHello, Process Designer Express crea un file di messaggi SampleHello.txt e lo pone nella directory collaborations\messages.

Importante: Non effettuare mai direttamente le modifiche ad un file di messaggi della collaborazione. Utilizzare sempre Process Designer Express per modificare i messaggi della maschera.

Il file di messaggi della collaborazione contiene tutte le stringhe di testo utilizzate dalla collaborazione. Queste stringhe includono quelle utilizzate nelle operazioni di registrazione log, gestione delle eccezioni e invio di e-mail.

Nota: Gli standard di InterChange Server Express consigliano di *non* includere i messaggi di traccia nel file di messaggi della collaborazione, in quanto questi non sono normalmente visualizzati agli utenti finali.

Per l'internazionalizzazione di una collaborazione è importante che le stringhe di testo siano isolate nel file di messaggi della collaborazione in modo da consentire la traduzione del file di messaggi. Il nome del file di messaggi della collaborazione tradotto deve includere il nome della locale associata (*CollaborationName_ll_TT*.txt).

Nella riga precedente, *ll* è l'abbreviazione della locale su due caratteri (per convenzione in minuscolo) e *TT* è l'abbreviazione dell'area su due caratteri (per convenzione in maiuscolo). Ad esempio, la versione del file di messaggi della collaborazione SampleHello che contiene i messaggi in Inglese (Stati Uniti) ha il nome SampleHello_en_US.txt

Durante il runtime, l'ambiente di collaborazione runtime individua il file di messaggi appropriato per la locale della collaborazione (ereditato da InterChange Server Express) nella directory collaborations\messages. Ad esempio, se la locale della collaborazione è Inglese (Stati Uniti), l'ambiente di collaborazione runtime recupera i messaggi dal file *CollaborationName*_en_US.txt.

Per ulteriori informazioni sull'utilizzo delle stringhe di testo per l'internazionalizzazione di una collaborazione, consultare "Una collaborazione internazionalizzata" a pagina 65.

Spiegazioni

Utilizzare la colonna Spiegazione nella finestra Messaggi maschera per aggiungere una spiegazione dettagliata di ciascun messaggio, in modo da creare messaggi auto-documentanti. Le spiegazioni sono facoltative, ma possono migliorare l'usabilità della collaborazione.

Ad esempio, si supponga di avere un messaggio con il seguente testo: `Aggiornamento non riuscito. Nell'applicazione di destinazione manca la voce per {1} {2}`. Questo testo del messaggio non fornisce dettagli sufficienti affinché l'utente possa risolvere facilmente l'errore. In questo esempio, l'aggiunta della seguente spiegazione al messaggio può migliorare in modo significativo l'utilità del messaggio per l'utente:

```
Una richiesta di aggiornamento è stata inviata al connettore, che
ha contattato correttamente l'applicazione. Tuttavia l'applicazione
non ha restituito i dati per il valore dell'attributo chiave specificato.
```

Per ulteriori esempi di messaggi auto-documentanti, fare riferimento al file InterchangeSystem.txt installato con InterChange Server Express.

Lettura delle spiegazioni dei messaggi

Le spiegazioni non sono visualizzate insieme ai messaggi durante il runtime. Tuttavia, l'utente può visualizzarle nel file di messaggi della collaborazione.

Se il log di InterChange Server Express viene salvato su file, utilizzare il Log Viewer per leggere le spiegazioni dei messaggi scritti sul file di log. Se il log non

viene salvato su file, le spiegazioni dei messaggi devono essere visualizzate direttamente nel file di messaggi della collaborazione.

Parametri del messaggio

Non è necessario scrivere messaggi separati per tutte le situazioni possibili. Utilizzare i parametri per rappresentare i valori che cambiano al runtime. L'utilizzo dei parametri consente ad ogni messaggio di essere utilizzato in più situazioni e consente di limitare le dimensioni del file di messaggi.

Un paramento viene sempre visualizzato come un numero compreso tra parentesi graffe: {*numero*}. Per ciascun parametro da aggiungere al messaggio, inserire il numero tra parentesi graffe all'interno del testo del messaggio, nel modo seguente:

```
testo messaggio {numero} altro testo messaggio.
```

Il metodo API richiamato per registrare il messaggio deve fornire un valore per ciascun parametro. Ad esempio, consideriamo il seguente messaggio:

```
6
Aggiornamento non riuscito. Nell'applicazione di destinazione manca la voce per {1} {2}
```

Nel frammento di codice che invia questo messaggio, è presente il seguente codice:

```
logWarning(6, " CustomerID" , fromCust.getString("CustomerID"));
```

InterChange Server Express combina i valori dei parametri nella chiamata al metodo logWarning() con il messaggio nel file di log e crea il messaggio. Prima di scrivere il messaggio nel file di log, il server sostituisce i parametri del messaggio con i seguenti valori:

- Il parametro 1 diventa la stringa "CustomerID".
- Il parametro 2 diventa il valore dell'attributo CustomerID nell'oggetto business fromCust.

Il messaggio sarà quindi scritto nel file di log nel modo seguente:

```
Aggiornamento non riuscito. Nell'applicazione di destinazione manca la voce per CustomerID 101961
```

Dal momento che il testo del messaggio prende la descrizione della voce mancante e il relativo ID come parametri, invece di includerli come stringhe prefissate, è possibile utilizzare lo stesso messaggio per qualsiasi coppia di attributi mancanti.

Gestione del file

In ambiente di produzione, un amministratore può impostare una procedura per filtrare i messaggi di collaborazione ed utilizzare messaggi e-mail o di cercapersone per comunicare con un responsabile in grado di risolvere i problemi. In questo caso, è importante che i numeri di errore e il relativo significato restino gli stessi dopo il primo rilascio di una maschera di collaborazione. È possibile modificare il testo associato ad un numero di errore, ma non si dovrebbe modificare il significare il significato del testo o riassegnare i numeri di errore.

Se si modifica il significato per un numero di errore, assicurarsi che la modifica sia documentata e comunicarla agli utenti della maschera di collaborazione.

E' possibile modificare un file di messaggi di collaborazione mentre l'oggetto di collaborazione è in esecuzione. Tuttavia, le modifiche saranno effettive solo dopo il successivo avvio dell'oggetto di collaborazione e dopo che il file di messaggi è stato letto nella memoria. Se InterChange Server Express viene interrotto durante

l'esecuzione di una collaborazione, il server legge automaticamente dalla memoria il file di messaggi per tutte le collaborazioni che erano in esecuzione in precedenza.

Parte 3. Blocchi funzione supportati

Capitolo 11. Blocchi funzione di oggetto business

I blocchi funzione nella cartella Generale\APIs\Oggetto Business e nelle relative cartelle secondarie forniscono le funzionalità di base per gestire gli oggetti business. I blocchi funzione nella cartella Generale\APIs\Oggetto Business si basano sui metodi della classe BusObj dell'API di collaborazione. I blocchi funzione nelle cartelle Generale\APIs\Oggetto Business\Vettore e Generale\APIs\Oggetto Business\Costanti corrispondono ai vettori e alle costanti Java della classe BusObj.

Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 66. Riepilogo dei blocchi funzione nella cartella Generale\APIs\Oggetto Business

Blocco funzione	**Pagina**
Copia	256
Duplica	257
Uguale	258
Chiavi uguali	257
Esiste	258
Ottieni valore booleano	259
Ottieni oggetto business	259
Ottieni vettore oggetto business	260
Ottieni tipo di oggetto business	261
Ottieni doppio	262
Ottieni mobile	262
Ottieni numero intero	263
Ottieni locale	264
Ottieni lungo	264
Ottieni testo lungo	265
Ottieni oggetto	266
Ottieni stringa	266
Ottieni verbo	267
E' vuoto	267
E' oggetto business	268
E' chiave	268
E' null	268
E' richiesto	269
Itera secondari	270
Chiave a stringa	270
Nuovo oggetto business	271
Imposta contenuto	272
Imposta valori predefiniti attributo	272
Imposta chiavi	272

Tabella 66. Riepilogo dei blocchi funzione nella cartella Generale\APIs\Oggetto Business (Continua)

Blocco funzione	Pagina
Imposta locale	273
Imposta valore	273
Imposta valore in base alla posizione	274
Imposta valore con Crea	275
Imposta verbo	275
Imposta verbo con Crea	276
Shallow uguale	276
Alla stringa	277
Dati validi	278

Tabella 67. Riepilogo dei blocchi funzione nella cartella Generale\APIs\Oggetto Business\Vettore

Blocco funzione	Pagina
Ottieni BusObj a	261
Nuovo vettore oggetto business	271
Imposta BusObj a	271
Dimensione	277

Tabella 68. Riepilogo dei blocchi funzione nella cartella Generale\APIs\Oggetto Business\Costanti

Blocco funzione	Pagina
Verbo:Crea	278
Verbo:Elimina	278
Verbo:Recupera	279
Verbo:Aggiorna	279

Copia

Copia tutti i valori degli attributi dall'oggetto business di input.

Input

Copia in Un oggetto BusObj che rappresenta l'oggetto di destinazione per l'operazione di copia.

Copia da Un oggetto BusObj che rappresenta l'oggetto business da copiare.

Note

Il blocco funzione Copia copia l'intero oggetto business, inclusi tutti gli oggetti business secondari e i vettori di oggetti business secondari. Questo blocco funzione non imposta un riferimento all'oggetto copiato. Al contrario, clona tutti gli attributi (crea copie separate degli attributi).

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.copy(). Per ulteriori informazioni, consultare “copy()” a pagina 392.

Duplica

Crea un oggetto business esattamente uguale all'originale.

Input

originale L'oggetto business (un oggetto BusObj) da duplicare.

Output

Restituisce l'oggetto business duplicato.

Eccezioni

Il blocco funzione Duplica può impostare il tipo eccezione di un oggetto `CollaborationException` su `ObjectException`.

Note

Questo blocco funzione crea un clone dell'oggetto business e lo restituisce. E' necessario assegnare esplicitamente il valore di ritorno di questo blocco funzione ad una variabile dichiarata di tipo `BusObj`.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.duplicate(). Per ulteriori informazioni, consultare “duplicate()” a pagina 393.

Chiavi uguali

Confronta i valori degli attributi chiave dell'oggetto business corrente con quelli dell'oggetto business di input per determinare se sono uguali.

Input

Oggetto business 1
Il primo oggetto business (un oggetto BusObj) del confronto.

Oggetto business 2
Il secondo oggetto business (un oggetto BusObj) del confronto.

Output

Restituisce `true` se i valori di tutti gli attributi chiave sono gli stessi; restituisce `false` se sono diversi.

Eccezioni

Il blocco funzione Chiavi uguali può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `ObjectException` — Impostato se l'argomento dell'oggetto business non è valido.

Note

Questo blocco funzione esegue un confronto superficiale, ovvero non confronta le chiavi negli oggetti business secondari.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.equalKeys(). Per ulteriori informazioni, consultare “equalKeys()” a pagina 393.

Uguale

Confronta gli attributi di due oggetti business (inclusi gli oggetti business secondari) per determinare se sono uguali.

Input

Oggetto business 1
Il primo oggetto business (un oggetto BusObj) del confronto.

Oggetto business 2
Il secondo oggetto business (un oggetto BusObj) del confronto.

Output

Restituisce `true` se i valori di tutti gli attributi chiave sono gli stessi; altrimenti restituisce `false`.

Eccezioni

Il blocco funzione Uguale può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `ObjectException` — Impostato se l'argomento dell'oggetto business non è valido.

Note

Questo blocco funzione confronta i valori degli attributi di questo oggetto business con quelli dell'oggetto business di input. Se gli oggetti business sono gerarchici, il confronto include tutti gli attributi degli oggetti business secondari.

Nel confronto, un valore null viene considerato equivalente ad un qualsiasi valore con cui viene confrontato e non impedisce di restituire `true`.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.equals(). Per ulteriori informazioni, consultare “equals()” a pagina 394.

Esiste

Verifica l'esistenza di un attributo dell'oggetto business con un nome specificato.

Input

Oggetto business
L'oggetto business (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo la cui esistenza si vuole verificare.

Output

Restituisce `true` , se l'attributo esiste; altrimenti restituisce `false`.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.exists(). Per ulteriori informazioni, consultare "exists()" a pagina 396.

Ottieni valore booleano

Recupera il valore di un singolo attributo da un oggetto business, come booleano.

Input

Oggetto business
L'oggetto business (un oggetto BusObj) nel quale esiste l'attributo.

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Restituisce il valore booleano (`true` o `false`) dell'attributo specificato.

Eccezioni

Il blocco funzione Ottieni valore booleano può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni valore booleano per un attributo che non contiene i dati appropriati.

Note

Il blocco funzione Ottieni valore booleano recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getBoolean(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni oggetto business

Recupera il valore di un singolo attributo da un oggetto business, come oggetto BusObj.

Input

Oggetto business
L'oggetto business (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce il valore dell'attributo specificato, come oggetto BusObj.

Eccezioni

Il blocco funzione Ottieni oggetto business può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni oggetto business per un attributo che non contiene i dati appropriati.

Note

Il blocco funzione Ottieni oggetto business recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getBusObj(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni vettore oggetto business

Recupera il valore di un singolo attributo da un oggetto business, come vettore di oggetti business.

Input

Oggetto business
L'oggetto business (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce il valore dell'attributo specificato, come vettore oggetti business.

Eccezioni

Il blocco funzione Ottieni vettore oggetto business può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni vettore oggetto business per un attributo che non contiene un vettore.

Note

Il blocco funzione Ottieni vettore oggetto business recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getBusObjArray(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni tipo di oggetto business

Recupera il nome della definizione di un oggetto business sul quale è basato l'oggetto business corrente.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Output

Restituisce una stringa (String) che contiene il nome della definizione di oggetto business.

Note

Il tipo di oggetto business è il nome della definizione di oggetto business dalla quale è stato creato l'oggetto business.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getType(). Per ulteriori informazioni, consultare "getType()" a pagina 399.

Ottieni BusObj a

Recupera l'elemento all'indice specificato in un vettore oggetti business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Vettore.

Input

Vettore oggetto business
Un oggetto BusObj[] che rappresenta il vettore oggetti business.

Indice Un numero intero che specifica la posizione dell'indice.

Output

Restituisce l'oggetto business che si trova alla posizione indice specificata.

Ottieni doppio

Recupera il valore di un singolo attributo da un oggetto business, come doppio (double).

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce il valore dell'attributo specificato, come tipo di dati double.

Eccezioni

Il blocco funzione Ottieni doppio può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni doppio per un attributo che non contiene i dati appropriati.

Note

Il blocco funzione Ottieni doppio recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getDouble(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni mobile

Recupera il valore di un singolo attributo da un oggetto business, come mobile (float).

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce il valore dell'attributo specificato, come tipo di dati float.

Eccezioni

Il blocco funzione Ottieni mobile può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni mobile per un attributo String che non contiene cifre numeriche.

Note

Il blocco funzione Ottieni mobile recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getFloat(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni numero intero

Recupera il valore di un singolo attributo da un oggetto business, come numero intero (int).

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce il valore dell'attributo specificato, come numero intero.

Eccezioni

Il blocco funzione Ottieni numero intero può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni numero intero per un attributo String che non contiene cifre numeriche.

Note

Il blocco funzione Ottieni numero intero recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getInt(). Per ulteriori informazioni, consultare “getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()” a pagina 397.

Ottieni locale

Recupera la locale associata ai dati dell'oggetto business.

Input

Oggetto business
L'oggetto business (un oggetto BusObj).

Output

Restituisce un oggetto Locale Java che contiene le informazioni sulla locale dell'oggetto business. Questo oggetto Locale deve essere un'istanza della classe java.util.Locale.

Note

Il blocco funzione Ottieni locale restituisce la locale associata ai dati di un oggetto business. Questa locale è spesso diversa dalla locale di collaborazione in cui viene eseguita la collaborazione.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getLocale(). Per ulteriori informazioni, consultare “getLocale()” a pagina 399.

Ottieni lungo

Recupera il valore di un singolo attributo da un oggetto business, come tipo di dati long.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce il valore dell'attributo specificato, come tipo di dati long.

Eccezioni

Il blocco funzione Ottieni lungo può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni lungo per un attributo String che non contiene cifre numeriche.

Note

Il blocco funzione Ottieni lungo recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getLong(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni testo lungo

Recupera il valore di un singolo attributo da un oggetto business, come testo lungo (long text).

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce il valore dell'attributo specificato, come String.

Eccezioni

Il blocco funzione Ottieni testo lungo può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni testo lungo per un attributo che non contiene i dati appropriati.

Note

Il blocco funzione Ottieni testo lungo recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Questo blocco funzione restituisce un oggetto String in quanto il tipo longtext di InterChange Server Express è un oggetto String privo di dimensione massima.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getLongText(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni oggetto

Recupera il valore di un singolo attributo da un oggetto business, come oggetto.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo o un numero intero che specifica la posizione ordinale di un attributo nell'elenco degli attributi dell'oggetto business.

Output

Restituisce il valore dell'attributo specificato, come oggetto.

Eccezioni

Il blocco funzione Ottieni oggetto può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni oggetto per un attributo che non contiene i dati del tipo appropriato.

Note

Il blocco funzione Ottieni oggetto recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.get(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni stringa

Recupera il valore di un singolo attributo da un oggetto business, come stringa (String).

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da recuperare.

Output

Restituisce una stringa (String) che contiene il valore dell'attributo specificato.

Eccezioni

Il blocco funzione Ottieni stringa può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione utilizza Ottieni stringa per un attributo non esistente.

Note

Il blocco funzione Ottieni stringa recupera un valore di attributo dall'oggetto business corrente e restituisce una copia del valore di attributo. *Non* restituisce un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dal blocco funzione. Ogni volta che si utilizza il blocco funzione, questo restituisce una nuova copia (clone) dell'attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getString(). Per ulteriori informazioni, consultare "getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()" a pagina 397.

Ottieni verbo

Recupera l'istruzione (verb) dell'oggetto business corrente.

Input

Oggetto business

L'oggetto business corrente (un oggetto BusObj).

Output

Restituisce una stringa (String) che contiene il nome dell'istruzione dell'oggetto business.

Note

Questo blocco funzione è utile per gli scenari che gestiscono più tipi di evento in ingresso. Il primo nodo azione in uno scenario richiama Ottieni verbo. Il collegamento di transizione in uscita dal nodo azione verifica quindi il contenuto della stringa restituita, in modo che ogni collegamento di transizione in uscita sia l'inizio di un percorso di esecuzione che gestisce una delle possibili istruzioni.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.getVerb(). Per ulteriori informazioni, consultare "getVerb()" a pagina 400.

E' vuoto

Determina se il valore di un attributo è impostato ad una stringa a lunghezza zero.

Input

Oggetto business

L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da valutare.

Output

Restituisce `true`, se il valore dell'attributo è una stringa a lunghezza zero; altrimenti restituisce `false`.

Note

Una stringa a lunghezza zero può essere paragonata ad una stringa "". E' diversa da null, la cui presenza viene rilevata dal blocco funzione E' null.

Se una collaborazione deve recuperare un valore di attributo e poi eseguire su di esso un'operazione, è possibile richiamare E' vuoto e E' null per verificare che presenti un valore prima di recuperare il valore.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.isBlank(). Per ulteriori informazioni, consultare "isBlank()" a pagina 400.

E' oggetto business

Determina se il valore è un oggetto business.

Input

Valore L'oggetto da valutare.

Output

Restituisce `true`, se il valore è un oggetto business; altrimenti restituisce `false`.

E' chiave

Determina se un attributo dell'oggetto business è definito come attributo chiave.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Restituisce `true`, se il valore è un attributo chiave; altrimenti restituisce `false`.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.isKey(). Per ulteriori informazioni, consultare "isKey()" a pagina 401.

E' null

Determina se il valore di un attributo dell'oggetto business è null.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Restituisce `true` se il valore dell'attributo è null; altrimenti, restituisce `false`.

Note

Null indica nessun valore, in contrasto con un valore stringa a lunghezza zero, che viene rilevato richiamando il blocco funzione E' vuoto. Verificare un oggetto con il blocco funzione E' null prima di utilizzarlo, in quanto se l'oggetto è null, l'operazione potrebbe non riuscire.

Un valore di attributo può essere null in queste situazioni:

- Il valore dell'attributo è stato impostato esplicitamente su null.

  Un valore di attributo può essere impostato su null utilizzando il blocco funzione Imposta valore.
- Il valore dell'attributo non è mai stato impostato.

  Quando una collaborazione utilizza il blocco funzione Nuovo oggetto business per creare un nuovo oggetto business, tutti i valori di attributo sono inizializzati a null. Se il valore dell'attributo non è stato impostato tra il momento della creazione e la chiamata al blocco funzione E' null, il valore è ancora null.
- Il valore null è stato inserito durante la mappatura.

  Quando una collaborazione elabora un oggetto business ricevuto da un connettore, il processo di mappatura potrebbe inserire un valore null. Il processo di mappatura converte l'oggetto business specifico dell'applicazione ricevuto dal connettore nell'oggetto business generico gestito dalla collaborazione. Per ciascun attributo nell'oggetto business generico che non ha un equivalente nell'oggetto specifico di applicazione, la mappa inserisce un valore null.

Suggerimenti: Chiamare sempre E' null prima di eseguire un'operazione su un attributo che rappresenta un oggetto business secondario o un vettore oggetti business secondario, in quanto Java non consente operazioni su oggetti null.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.isNull(). Per ulteriori informazioni, consultare "isNull()" a pagina 401.

E' richiesto

Determina se un attributo dell'oggetto business è definito come attributo obbligatorio.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Restituisce `true`, se l'attributo è obbligatorio; altrimenti restituisce `false`.

Note

Se un attributo è definito come obbligatorio, deve presentare un valore e il valore non deve essere null.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.isRequired(). Per ulteriori informazioni, consultare "isRequired()" a pagina 402.

Itera secondari

Scorre il vettore oggetti business secondario.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Indice corrente
Un numero intero che specifica l'indice corrente.

Elemento corrente
L'elemento corrente (un oggetto BusObj).

Chiave a stringa

Recupera i valori degli attributi di chiave primaria dell'oggetto business e li restituisce come stringa (String).

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Output

Un oggetto String che contiene tutti i valori chiave di un oggetto business, concatenati e ordinati per il valore ordinale degli attributi.

Note

L'output di questo blocco funzione contiene il nome dell'attributo e i relativi valori. Più valori sono valori di attributo della chiave primaria, concatenati e separati da spazi. Ad esempio, se è presente un attributo chiave primaria SS#, l'output potrebbe essere:

```
SS#=100408394
```

Se gli attributi di chiave primaria sono `FirstName` e `LastName`, l'output potrebbe essere:

```
FirstName=Nina LastName=Silk
```

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.keysToString(). Per ulteriori informazioni, consultare “keysToString()” a pagina 403.

Nuovo oggetto business

Crea una nuova istanza dell’oggetto business (BusObj) del tipo specificato.

Input

Tipo Una stringa (String) che specifica il tipo di oggetto business da creare.

Output

Restituisce un nuovo oggetto business del tipo specificato.

Informazioni correlate

Questo blocco funzione si basa sul costruttore Collaboration.BusObj().

Nuovo vettore oggetto business

Crea un nuovo vettore di oggetti business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Vettore.

Input

Dimensione Un numero intero che specifica la dimensione del vettore.

Output

Restituisce un vettore oggetti business della dimensione specificata.

Imposta BusObj a

Imposta l’elemento all’indice specificato in un vettore oggetti business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Vettore.

Input

Vettore oggetto business
Un oggetto BusObj[] che rappresenta il vettore oggetti business.

Indice Un numero intero che specifica la posizione dell’elemento.

Oggetto business
Un oggetto BusObj che rappresenta l’elemento da impostare.

Imposta contenuto

Imposta il contenuto dell'oggetto business corrente per un altro oggetto business. I due oggetti business quindi sono entrambi proprietari del contenuto. Le modifiche effettuate ad un oggetto business sono riportare automaticamente nell'altro oggetto business.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Contenuto L'oggetto business (un oggetto BusObj) il cui contenuto deve essere utilizzato per l'oggetto business corrente.

Eccezioni

Il blocco funzione Imposta contenuto può impostare uno dei seguenti tipi eccezione per un'eccezione `CollaborationException`:

- `AttributeException`— Impostato se si verifica un problema di accesso all'attributo.
- `ObjectException` — Impostato se l'argomento dell'oggetto business non è valido.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.setContent(). Per ulteriori informazioni, consultare le pagine di riferimento a BusObj nel manuale *Guida allo sviluppo delle mappe*.

Imposta valori predefiniti attributo

Imposta tutti gli attributi di un oggetto business al rispettivi valori predefiniti.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Note

Una definizione di oggetto business può includere i valori predefiniti degli attributi. Il blocco funzione imposta i valori degli attributi di questo oggetto business ai valori specificati come predefiniti nella definizione.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.setDefaultAttrValues(). Per ulteriori informazioni, consultare "setDefaultAttrValues()" a pagina 405.

Imposta chiavi

Imposta i valori degli attributi chiave dell'oggetto business corrente ai valori degli attributi chiave di un altro oggetto business.

Input

Dall'oggetto business
L'oggetto business (un oggetto BusObj) di cui utilizzare i valori di attributi chiave.

All'oggetto business
L'oggetto business corrente (un oggetto BusObj) che riceverà i valori di attributi chiave dell'altro oggetto business.

Eccezioni

Il blocco funzione Imposta chiavi può impostare uno dei seguenti tipi eccezione per un'eccezione `CollaborationException`:

- `AttributeException`— Impostato se si verifica un problema di accesso all'attributo.
- `ObjectException` — Impostato se l'argomento dell'oggetto business non è valido.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.setKeys(). Per ulteriori informazioni, consultare "setKeys()" a pagina 405.

Imposta locale

Imposta la locale dell'oggetto business corrente.

Input

Oggetto business
L'oggetto business (un oggetto BusObj) per cui impostare la locale.

Locale L'oggetto Locale Java che contiene le informazioni sulla locale da assegnare all'oggetto business. Questo oggetto Locale deve essere un'istanza della classe java.util.Locale.

Note

Il blocco funzione Imposta locale assegna una locale ai dati associati ad un oggetto business. Questa locale può essere diversa dalla locale di collaborazione in cui viene eseguita la collaborazione.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.setLocale(). Per ulteriori informazioni, consultare "setLocale()" a pagina 405.

Imposta valore

Imposta un attributo dell'oggetto business ad un valore specificato per un tipo di dati particolare.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da impostare.

Valore Il valore dell'attributo. Deve essere il tipo appropriato per l'attributo (boolean, double, float, int, long, Object, String o BusObj).

Eccezioni

Il blocco funzione Imposta valore può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException`— Impostato se si verifica un problema di accesso all'attributo.

Note

Il blocco funzione Imposta valore imposta un valore di attributo nell'oggetto business corrente. Imposta un riferimento all'oggetto quando all'attributo viene assegnato il valore. Il valore dell'attributo *non* viene clonato dall'oggetto business di origine. Di conseguenza, le modifiche al valore nell'oggetto business di origine vengono effettuate anche all'attributo nell'oggetto business che richiama il blocco funzione Imposta valore.

Il blocco funzione Imposta valore fornisce i seguenti moduli:

- Il primo modulo imposta un valore del tipo specificato dal valore di input del blocco funzione. Utilizzare questo modulo per impostare gli attributi con specifici tipi di dati di base o definiti da InterChange Server Express.
- Il secondo modulo imposta il valore di un attributo di *qualsiasi* tipo. E' possibile inviare qualsiasi tipo di dati come valore di attributo in quanto il valore di input è di tipo `Object`. Ad esempio, per impostare un attributo dell'oggetto `BusObj` o `LongText`, utilizzare l'oggetto `BusObj` o `LongText` per il valore di input.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.set(). Per ulteriori informazioni, consultare "set()" a pagina 403.

Imposta valore in base alla posizione

Imposta un attributo dell'oggetto business ad un valore basato sulla posizione dell'attributo.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Posizione Un numero intero che specifica la posizione ordinale di un attributo nell'elenco degli attributi dell'oggetto business.

Valore Il valore dell'attributo. Deve essere del tipo appropriato (Object, String o BusObj).

Eccezioni

Il blocco funzione Imposta valore in base alla posizione può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException`— Impostato se si verifica un problema di accesso all'attributo.

Note

I valori accettabili per questo blocco funzione sono limitati ai tipi di dati Object, String e BusObj.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.set(). Per ulteriori informazioni, consultare “set()” a pagina 403.

Imposta valore con Crea

Imposta un attributo dell'oggetto business ad un valore specificato per un tipo di dati particolare, creando un oggetto per il valore se non esiste già.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo da impostare.

Verbo Una stringa (String) che specifica l'istruzione Create.

Eccezioni

Il blocco funzione Imposta valore con Crea può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException`— Impostato 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 secondario a cardinalità multipla, il `BusObj` viene accodato a `BusObjArray` come ultimo elemento. Tuttavia, se l'attributo di destinazione contiene un `BusObj`, questo oggetto business sostituisce il valore precedente.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.setWithCreate(). Per ulteriori informazioni, consultare “setWithCreate()” a pagina 406.

Imposta verbo

Imposta l'istruzione (verb) di un oggetto business.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Verbo Una stringa (String) che specifica l'istruzione da utilizzare con l'oggetto business.

Note

Questo blocco funzione è utilizzato nella mappatura degli oggetti business. Non utilizzare questo blocco funzione in una maschera di collaborazione, nella quale

deve essere impostata in modo interattivo l'istruzione di un oggetto business, valorizzando le proprietà di una chiamata di servizio.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.setVerb(). Per ulteriori informazioni, consultare "setVerb()" a pagina 406.

Imposta verbo con Crea

Imposta l'istruzione (verb) di un oggetto business secondario, creando l'oggetto business secondario se non esiste già.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Verbo Una stringa (String) che specifica l'istruzione Create.

Eccezioni

Il blocco funzione Imposta verbo con Crea può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException`— Impostato se si verifica un problema di accesso all'attributo.

Note

Se l'attributo specificato dall'attributo di input è di tipo `BusObj` ed è null, viene creata una nuova istanza dell'oggetto business secondario e l'istruzione viene impostata al valore dell'istruzione di input. Se l'istanza di questo oggetto business secondario esiste già, viene soltanto impostata l'istruzione. Se l'oggetto business secondario presenta una cardinalità multipla, l'attributo di input deve specificare l'indice.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.setVerbWithCreate().

Shallow uguale

Confronta i valori di due oggetti business, escludendo gli oggetti business secondari, per determinare se sono uguali.

Input

Oggetto business 1
Il primo oggetto business (un oggetto BusObj) da confrontare.

Oggetto business 2
Il secondo oggetto business (un oggetto BusObj) da confrontare.

Output

Restituisce `true` se i valori di tutti gli attributi sono gli stessi; altrimenti restituisce `false`.

Eccezioni

Il blocco funzione Shallow uguale può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `ObjectException` — Impostato se l'argomento dell'oggetto business non è valido.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.equalsShallow(). Per ulteriori informazioni, consultare "equalsShallow()" a pagina 396.

Dimensione

Recupera la dimensione di questo vettore oggetti business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Vettore.

Input

Vettore oggetto business
Un oggetto BusObj[] che rappresenta il vettore oggetti business.

Output

Restituisce un numero intero che specifica la dimensione del vettore.

Alla stringa

Restituisce i valori di tutti gli attributi di un oggetto business come String.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Output

Un oggetto String costituito da tutti i valori di attributo contenuti in un oggetto business.

Note

La stringa risultante da una chiamata a questo metodo è simile a quella riportata nell'esempio seguente:

```
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
```

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.toString(). Per ulteriori informazioni, consultare "toString()" a pagina 407.

Dati validi

Determina se un valore specificato è un tipo di dati valido per un attributo specificato.

Input

Oggetto business
L'oggetto business corrente (un oggetto BusObj).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Valore Il valore dell'attributo. Può essere di tipo Object, BusObj, BusObjArray, String, long, int, double, float o boolean.

Output

Restituisce `true`, se il valore dell'attributo è un tipo di dati valido; altrimenti restituisce `false`.

Note

Verifica la compatibilità del valore passato con l'attributo di destinazione (come specificato dall'attributo di input). Questi sono i criteri utilizzati:

Per tipi primitivi (String, long, int, double, float, boolean)	deve essere possibile convertire il valore nel tipo di dati dell'attributo
Per un `BusObj`	il valore deve presentare lo stesso tipo dell'attributo di destinazione
Per un `BusObjArray`	il valore deve puntare ad un `BusObj` o `BusObjArray` con lo stesso tipo (definizione di oggetto business) dell'attributo
Per un `Object`	Il valore deve essere di tipo `String`, `BusObj` o `BusObjArray`. Sono quindi applicate le corrispondenti regole di convalida.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObj.validData(). Per ulteriori informazioni, consultare "validData()" a pagina 408.

Verbo:Crea

L'istruzione Crea (Create) dell'oggetto business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Costanti.

Output

Restituisce una stringa (String) che contiene l'istruzione Create.

Verbo:Elimina

L'istruzione Elimina (Delete) dell'oggetto business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Costanti.

Output

Restituisce una stringa (String) che contiene l'istruzione Delete.

Verbo:Recupera

L'istruzione Recupera (Retrieve) dell'oggetto business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Costanti.

Output

Restituisce una stringa (String) che contiene l'istruzione Retrieve.

Verbo:Aggiorna

L'istruzione Aggiorna (Update) dell'oggetto business.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Oggetto Business\Costanti.

Output

Restituisce una stringa (String) che contiene l'istruzione Update.

Capitolo 12. Blocchi funzione di vettore oggetto business

I blocchi funzione nella cartella Generale\APIs\Vettore Oggetto Business forniscono le funzionalità di base per gestire i vettori di oggetti business. Questi blocchi funzione si basano sulla classe BusObjArray dell'API di collaborazione.

Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 69. Riepilogo dei blocchi funzione nella cartella Generale\APIs\Vettore Oggetto Business

Aggiungi elemento

Aggiunge un oggetto business al vettore oggetti business corrente.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Elemento L'oggetto business (specificato come oggetto BusObj) da aggiungere.

Eccezioni

Il blocco funzione Aggiungi elemento può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se l'elemento non è valido.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.addElement(). Per ulteriori informazioni, consultare "addElement()" a pagina 412.

Duplica

Crea un vettore oggetti business esattamente uguale all'originale.

Input

Vettore oggetto business originale
Il vettore oggetti business (specificato come oggetto BusObjArray) da duplicare.

Output

Restituisce il vettore oggetti business duplicato.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.duplicate(). Per ulteriori informazioni, consultare "duplicate()" a pagina 412.

Uguale

Confronta due vettori oggetto business per determinare se sono uguali.

Input

Vettore oggetto business 1
Il primo vettore oggetti business (specificato come oggetto BusObjArray) da confrontare.

Vettore oggetto business 2
Il secondo vettore oggetti business (specificato come oggetto BusObjArray) da confrontare.

Output

Restituisce `true`, se i vettori sono uguali; altrimenti restituisce `false`.

Note

Il confronto tra due vettori oggetto business verifica il numero di elementi e i relativi valori di attributo.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.equals(). Per ulteriori informazioni, consultare "equals()" a pagina 413.

Ottieni elemento

Recupera un singolo oggetto business da un vettore specificando la posizione dell'oggetto nel vettore.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Indice Un numero intero che specifica la posizione dell'indice.

Output

Restituisce l'oggetto business specificato.

Eccezioni

Il blocco funzione Ottieni elemento può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se l'elemento non è valido.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.elementAt(). Per ulteriori informazioni, consultare "elementAt()" a pagina 413.

Ottieni elementi

Recupera il contenuto del vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Output

Restituisce gli oggetti business nel vettore.

Eccezioni

Il blocco funzione Ottieni elementi può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `ObjectException` — Impostato se uno degli elementi non è valido.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.getElements(). Per ulteriori informazioni, consultare "getElements()" a pagina 413.

Ottieni ultimo indice

Recupera l'ultimo indice disponibile da un vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Output

Restituisce il valore dell'ultimo indice, come numero intero.

Note

Anche se il blocco funzione Ottieni ultimo indice esegue una funzione simile al blocco funzione Dimensione, esiste una importante differenza: Dimensione restituisce un valore che è maggiore di 1 rispetto al valore restituito da Ottieni ultimo indice. Questa differenza è dovuta al fatto che l'oggetto BusObjArray utilizzato come input è un vettore con riferimento zero.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.getLastIndex(). Per ulteriori informazioni, consultare "getLastIndex()" a pagina 414.

Valore corretto per Vettore Oggetto Business

Determina se un oggetto è un vettore oggetti business (BusObjArray).

Input

Valore Il nome dell'oggetto da verificare.

Output

Restituisce `true`, se il valore è un vettore oggetti business; altrimenti restituisce `false`.

Valore attributo massimo

Recupera il valore massimo dell'attributo specificato tra tutti gli elementi nel vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Restituisce una stringa (String) che contiene il valore massimo per l'attributo specificato.

Eccezioni

Il blocco funzione Valore attributo massimo può generare le seguenti eccezioni, entrambe sottoclassi di `CollaborationException` :

- `UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.
- `UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Per ognuna di queste eccezioni, il blocco funzione Valore attributo massimo può impostare il tipo di eccezione `AttributeException`.

Note

Il blocco funzione Valore attributo massimo ricerca il valore massimo dell'attributo specificato tra gli oggetti business in questo `BusObjArray`. Ad esempio, se si utilizzano tre oggetti Employee e l'attributo è Salary, di tipo Float, restituirà la stringa che rappresenta il Salary più alto.

Se il valore dell'attributo specificato per un elemento in `BusObjArray` è null, l'elemento viene ignorato. Se il valore dell'attributo specificato è null per tutti gli elementi, viene restituito null.

Quando il tipo dell'attributo è String, Valore attributo massimo restituisce il valore dell'attributo che rappresenta la stringa lessicale più lunga.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.max(). Per ulteriori informazioni, consultare "max()" a pagina 414.

Massimo Vettore Oggetto Business

Restituisce gli oggetti business con il massimo valore per l'attributo specificato, come vettore oggetti business (oggetto BusObjArray).

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Un elenco di oggetti business sotto forma di oggetto BusObjArray.

Eccezioni

Il blocco funzione Massimo Vettore Oggetto Business può generare le seguenti eccezioni, entrambe sottoclassi di `CollaborationException` :

- `UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.
- `UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Per ognuna di queste eccezioni, il blocco funzione Massimo Vettore Oggetto Business può impostare il tipo di eccezione `AttributeException`.

Note

Il blocco funzione Massimo Vettore Oggetto Business individua uno o più oggetti business con il valore massimo per l'attributo specificato, e restituisce questi oggetti business in un oggetto `BusObjArray`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il blocco funzione determina il valore maggiore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore massimo di Salary, il blocco funzione restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il blocco funzione restituisce la stringa lessicale più lunga.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.maxBusObjArray(). Per ulteriori informazioni, consultare "maxBusObjArray()" a pagina 415.

Oggetti Business massimi

Restituisce gli oggetti business con il valore massimo per l'attributo specificato, come vettore di oggetti BusObj.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Un elenco di oggetti business sotto forma di oggetto BusObj[].

Eccezioni

Il blocco funzione Oggetti Business massimi può generare le seguenti eccezioni, entrambe sottoclassi di `CollaborationException` :

- `UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.
- `UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Per ognuna di queste eccezioni, il blocco funzione Oggetti Business massimi può impostare il tipo di eccezione `AttributeException`.

Note

Il blocco funzione Oggetti Business massimi individua uno o più oggetti business con il valore massimo per l'attributo specificato, e restituisce questi oggetti business come vettore di oggetti `BusObj`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il blocco funzione determina il valore maggiore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore massimo di Salary, il blocco funzione restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il blocco funzione restituisce la stringa lessicale più lunga.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.maxBusObjs(). Per ulteriori informazioni, consultare “maxBusObjs()” a pagina 416.

Valore attributo minimo

Recupera il valore minimo dell'attributo specificato tra tutti gli elementi nel vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Restituisce una stringa (String) che contiene il valore minimo per l'attributo specificato.

Eccezioni

Il blocco funzione Valore attributo minimo può generare le seguenti eccezioni, entrambe sottoclassi di `CollaborationException` :

- `UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.
- `UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Per ognuna di queste eccezioni, il blocco funzione Valore attributo minimo può impostare il tipo di eccezione `AttributeException`.

Note

Il blocco funzione Valore attributo minimo ricerca il valore minimo dell'attributo specificato tra gli oggetti business in questo vettore oggetti business.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il blocco funzione determina il valore minore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore minimo di Salary, il blocco funzione restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il blocco funzione restituisce la stringa lessicale più corta.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.min(). Per ulteriori informazioni, consultare “min()” a pagina 417.

Vettore Oggetto Business minimo

Restituisce gli oggetti business con il minimo valore per l'attributo specificato, come vettore oggetti business (oggetto BusObjArray).

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Un elenco di oggetti business sotto forma di oggetto BusObjArray.

Eccezioni

Il blocco funzione Vettore Oggetto Business minimo può generare le seguenti eccezioni, entrambe sottoclassi di `CollaborationException` :

- `UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.
- `UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Per ognuna di queste eccezioni, il blocco funzione Vettore Oggetto Business minimo può impostare il tipo di eccezione `AttributeException`.

Note

Il blocco funzione Vettore Oggetto Business minimo individua uno o più oggetti business con il valore minimo per l'attributo specificato, e restituisce questi oggetti business in un oggetto `BusObjArray`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il blocco funzione determina il valore minore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore minimo di Salary, il blocco funzione restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il blocco funzione restituisce la stringa lessicale più corta.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.minBusObjArray(). Per ulteriori informazioni, consultare "minBusObjArray()" a pagina 418.

Oggetti Business minimi

Restituisce gli oggetti business con il valore minimo per l'attributo specificato, come vettore di oggetti BusObj.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto BusObjArray).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Un elenco di oggetti business sotto forma di oggetto BusObj[].

Eccezioni

Il blocco funzione Oggetti Business minimi può generare le seguenti eccezioni, entrambe sottoclassi di `CollaborationException` :

- `UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.
- `UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Per ognuna di queste eccezioni, il blocco funzione Oggetti Business minimi può impostare il tipo di eccezione `AttributeException`.

Note

Il blocco funzione Oggetti Business minimi individua uno o più oggetti business con il valore minimo per l'attributo specificato, e restituisce questi oggetti business come vettore di oggetti `BusObj`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il blocco funzione determina il valore minore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore minimo di Salary, il blocco funzione restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il blocco funzione restituisce la stringa lessicale più corta.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.minBusObjs(). Per ulteriori informazioni, consultare "minBusObjs()" a pagina 419.

Rimuovi tutti gli elementi

Rimuove tutti gli elementi dal vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (BusObjArray) da cui verranno rimossi gli elementi.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.removeAllElements(). Per ulteriori informazioni, consultare “removeAllElements()” a pagina 420.

Rimuovi Elemento

Rimuove un elemento oggetto business da un vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (BusObjArray) da cui verrà rimosso l'elemento.

Elemento L'elemento oggetto business (BusObj) da rimuovere dal vettore.

Eccezioni

Il blocco funzione Rimuovi Elemento può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se l'elemento non è valido.

Note

Dopo aver eliminato un elemento dal vettore, il vettore viene ridimensionato, modificando gli indici degli elementi esistenti.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.removeElement(). Per ulteriori informazioni, consultare “removeElement()” a pagina 420.

Rimuovi elemento a

Rimuove un elemento oggetto business da una particolare posizione nel vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (BusObjArray) da cui verrà rimosso l'elemento.

Indice La posizione di indice (specificata come numero intero) dell'elemento da rimuovere.

Eccezioni

Il blocco funzione Rimuovi elemento a può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se l'elemento non è valido.

Note

Dopo aver rimosso un elemento dal vettore, il vettore viene ridimensionato, modificando eventualmente gli indici degli elementi esistenti.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.removeElementAt(). Per ulteriori informazioni, consultare “removeElementAt()” a pagina 421.

Imposta elemento a

Imposta il valore di un oggetto business in un vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto di tipo BusObjArray) in cui verrà impostato il valore dell'elemento.

Elemento L'elemento oggetto business (specificato come oggetto di tipo BusObj) per il quale verrà impostato il valore.

Indice La posizione di indice (specificata come numero intero) dell'elemento oggetto business da impostare.

Eccezioni

Il blocco funzione Imposta elemento a può impostare il seguente tipo di eccezione per un'eccezione `CollaborationException`:

- `AttributeException` — Impostato se l'elemento non è valido.

Note

Questo blocco funzione imposta i valori dell'oggetto business in una posizione di vettore specificata sui valori di un oggetto business di input.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.setElementAt(). Per ulteriori informazioni, consultare “setElementAt()” a pagina 421.

Dimensione

Determina il numero di elementi in questo vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto di tipo BusObjArray) per il quale determinare la dimensione.

Output

Restituisce un numero intero che specifica il numero di elementi nel vettore.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.size(). Per ulteriori informazioni, consultare “size()” a pagina 422.

Somma

Aggiunge i valori dell'attributo specificato per tutti gli oggetti business in questo vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto di tipo BusObjArray).

Attributo Una stringa (String) che specifica il nome dell'attributo.

Output

Restituisce la somma (come tipo di dati double) dell'attributo specificato dall'elenco di oggetti business.

Eccezioni

Il blocco funzione Somma può generare le seguenti eccezioni, entrambe sottoclassi di `CollaborationException` :

- `UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.
- `UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Per ognuna di queste eccezioni, il blocco funzione Somma può impostare il tipo di eccezione `AttributeException`.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.sum(). Per ulteriori informazioni, consultare "sum()" a pagina 422.

Cambia

Inverte la posizione di due oggetti business nel vettore oggetti business.

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto di tipo BusObjArray).

Indice 1 Un numero intero che specifica la posizione del primo oggetto business.

Indice 2 Un numero intero che specifica la posizione del secondo oggetto business.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.swap(). Per ulteriori informazioni, consultare "swap()" a pagina 422.

Alla stringa

Recupera i valori del vettore oggetti business e li restituisce come singola stringa (String).

Input

Vettore oggetto business
Il vettore oggetti business (specificato come oggetto di tipo BusObjArray).

Output

Restituisce una stringa (String) che contiene tutti i valori nel vettore oggetti business.

Informazioni correlate

Questo blocco funzione si basa sul metodo BusObjArray.toString(). Per ulteriori informazioni, consultare “toString()” a pagina 423.

Capitolo 13. Blocchi funzione di maschera di collaborazione

I blocchi funzione della maschera di collaborazione forniscono le funzionalità di base per gestire gli oggetti di collaborazione. Questi blocchi funzione sono organizzati nelle seguenti cartelle:

- Generale\APIs\Maschera di collaborazione — Utilizzati per gestire gli oggetti di collaborazione.
- Generale\APIs\Maschera di collaborazione\Eccezione — Utilizzati per creare i nuovi oggetti eccezione all'interno di una maschera di collaborazione.
- Generale\APIs\Maschera di collaborazione\Eccezione\Costanti — Utilizzati per rappresentare tipi di eccezione specifici all'interno dell'oggetto eccezione di collaborazione.

Nelle seguenti sezioni sono riportate ulteriori informazioni sui singoli blocchi funzione della maschera di collaborazione.

Tabella 70. Riepilogo dei blocchi funzione di maschera di collaborazione

Percorso	Blocco funzione	Pagina
Generale\APIs\Maschera di collaborazione	Ottieni locale	296
	Ottieni messaggio	297
	Ottieni messaggio con parametro	297
	Ottieni nome	298
	Ottieni proprietà	298
	Ottieni vettore proprietà	298
	Bracketing DB implicito	299
	E' abilitata la funzione di traccia?	299
	La proprietà è esistente	300
	Invia Email	306
Generale\APIs\Maschera di collaborazione\Eccezione	Aumenta l'eccezione collaborazione	301
	Aumenta l'eccezione collaborazione 1	302
	Aumenta l'eccezione collaborazione 2	303
	Aumenta l'eccezione collaborazione 2	303
	Aumenta l'eccezione collaborazione 4	304
	Aumenta l'eccezione collaborazione 5	305
	Aumenta l'eccezione collaborazione con i parametri	305

Tabella 70. Riepilogo dei blocchi funzione di maschera di collaborazione (Continua)

Percorso	Blocco funzione	Pagina
Generale\APIs\Maschera di collaborazione\Eccezione\Costanti	AnyException	296
	AttributeException	296
	JavaException	300
	ObjectException	300
	OperationException	300
	ServiceCallException	307
	SystemException	307
	TransactionException	307

AnyException

Una costante che rappresenta un qualsiasi tipo di eccezione.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″AnyException″.

AttributeException

Una costante che rappresenta un problema di accesso all'attributo (ad esempio, se una collaborazione utilizza il blocco funzione Ottieni doppio per un attributo basato su String oppure utilizza il blocco funzione Ottieni stringa su un attributo non esistente).

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″AttributeException″.

Ottieni locale

Recupera la locale di collaborazione per l'oggetto di collaborazione corrente.

Input

Collaborazione

L'oggetto di collaborazione corrente.

Output

Restituisce un oggetto Locale Java che contiene i codici di lingua e paese della locale di collaborazione. Questo oggetto Locale deve essere un'istanza della classe java.util.Locale.

Note

Il blocco funzione Ottieni locale restituisce la locale del flusso corrente. Questa locale del flusso è la locale associata all'oggetto business di attivazione dell'oggetto di collaborazione.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.getLocale(). Per ulteriori informazioni, consultare "getLocale()" a pagina 374.

Ottieni messaggio

Recupera un messaggio dal file messaggi di collaborazione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

ID Un numero intero che specifica il numero di un messaggio nel file di messaggi di collaborazione. Il file di messaggi è indicizzato per numero di messaggio.

Output

Restituisce un oggetto String che contiene il testo del messaggio specificato dall'ID di input.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.getMessage(). Per ulteriori informazioni, consultare "getMessage()" a pagina 374.

Ottieni messaggio con parametro

Recupera un messaggio dal file messaggi di collaborazione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

ID Un numero intero che specifica il numero di un messaggio nel file di messaggi di collaborazione. Il file di messaggi è indicizzato per numero di messaggio.

Parametri Un vettore di valori di parametri del messaggio. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio. All'interno del messaggio (nel file di messaggi di collaborazione), i parametri del messaggio sono indicati da numeri interi racchiusi tra parentesi graffe; ad esempio, {1}.

Output

Restituisce un oggetto String che contiene il testo del messaggio identificato dall'ID e dai parametri di input.

Note

Il blocco funzione Ottieni messaggio con parametro riceve un numero messaggio e un vettore di valori di parametri del messaggio. Recupera il messaggio associato dal file di messaggi di collaborazione, sostituisce i relativi parametri del messaggio con gli oggetti nel vettore dei parametri e restituisce il messaggio risultante come oggetto `String`.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.getMessage(). Per ulteriori informazioni, consultare "getMessage()" a pagina 374.

Ottieni nome

Recupera il nome di questo oggetto di collaborazione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Output

Restituisce una stringa (String) che contiene il nome dell'oggetto di collaborazione corrente.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.getName(). Per ulteriori informazioni, consultare "getName()" a pagina 375.

Ottieni proprietà

Recupera il valore di una proprietà di configurazione collaborazione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Nome proprietà
Una stringa (String) che specifica la proprietà di configurazione collaborazione da rilevare.

Output

Restituisce una stringa (String) che contiene il valore della proprietà di configurazione collaborazione specificata.

Ottieni vettore proprietà

Recupera il valore di una proprietà di configurazione collaborazione a più elementi.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Nome proprietà
Una stringa (String) che specifica la proprietà di configurazione collaborazione da rilevare.

Output

Restituisce un vettore di oggetti String; ciascun oggetto String nel vettore contiene il valore di un elemento della proprietà di configurazione collaborazione.

Bracketing DB implicito

Recupera il modello di programmazione transazione utilizzato dall'oggetto di collaborazione per le connessioni ottenute.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Output

Restituisce un valore booleano per indicare il modello di programmazione transazione da utilizzare per tutte le connessioni al database.

- Un valore `true` indica che tutte le connessioni utilizzano il bracketing transazioni *implicite*.
- Un valore `false` indica che tutte le connessioni utilizzano il bracketing transazioni *esplicite*.

Note

Il blocco funzione Bracketing DB implicito restituisce un valore booleano che indica quale modello di programmazione transazione l'oggetto di collaborazione assume sia utilizzato da *tutte* le connessioni ottenute. Questo blocco funzione è utile, prima di ottenere una connessione, per verificare se il modello di programmazione transazione è appropriato per la connessione.

Nota: Il modello di programmazione transazione può essere sovrascritto per una specifica connessione mediante il blocco funzione Ottieni connessione database (disponibile nella cartella Generale\APIs\Connessione Database).

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.implicitDBTransactionBracketing(). Per ulteriori informazioni, consultare "implicitDBTransactionBracketing()" a pagina 376.

E' abilitata la funzione di traccia?

Confronta il livello di traccia specificato con il livello di traccia corrente della collaborazione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Livello traccia Un numero intero che specifica il livello di traccia da confrontare con il livello di traccia corrente.

Output

Restituisce `true` se il livello di traccia di sistema corrente viene impostato al livello di traccia specificato; restituisce `false` se i due livelli di traccia non sono uguali.

Note

Il blocco funzione E' abilitata la funzione di traccia è utile per determinare se un messaggio di traccia deve essere o non essere registrato. Dal momento che la traccia può peggiorare le prestazioni, questo blocco funzione risulta utile nella fase di sviluppo di un progetto.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.isTraceEnabled(). Per ulteriori informazioni, consultare "isTraceEnabled()" a pagina 377.

JavaException

Una costante che rappresenta un problema con il codice Java nella logica business della maschera di collaborazione.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″JavaException″.

ObjectException

Una costante che rappresenta un errore causato dal passaggio di un oggetto business non valido ad un blocco funzione oppure dall'accesso ad un oggetto null.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″ObjectException″.

OperationException

Una costante che rappresenta un errore causato da una chiamata di servizio configurata in modo non corretto che non può essere inviata.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″OperationException″.

La proprietà è esistente

Determina se una proprietà di configurazione collaborazione esiste.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Nome proprietà
Una stringa (String) che specifica il nome della proprietà di configurazione collaborazione da rilevare.

Output

Restituisce `True`, se la proprietà di configurazione collaborazione esiste; altrimenti restituisce `False`.

Aumenta l'eccezione collaborazione

Prepara un'eccezione di collaborazione per innalzarla al successivo livello superiore di esecuzione. Questo blocco funzione crea un nuovo oggetto eccezione con il tipo eccezione specificato e una stringa di messaggio. Utilizzare questo modulo per passare un messaggio di eccezione memorizzato come stringa.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Tipo eccezione
Una stringa (String) che specifica il tipo di eccezione.

messageNum Un numero intero che specifica il numero del messaggio associato con l'oggetto eccezione.

Note

Il blocco funzione Aumenta l'eccezione collaborazione prepara un'eccezione di collaborazione da innalzare al successivo livello superiore di esecuzione. Quando l'ambiente runtime di collaborazione esegue il blocco funzione Aumenta l'eccezione collaborazione, modifica l'esecuzione della collaborazione allo stato Eccezione, quindi procede con la logica del diagramma di attività. La modalità di risposta del diagramma di attività all'eccezione innalzata dipende dal nodo di termine del percorso di esecuzione così come riportato di seguito:

- Se il percorso di esecuzione termina in Fine completamento, il controllo passa al successivo livello superiore di esecuzione.

  Se il successivo nodo di questo diagramma principale è un nodo decisionale, l'ambiente runtime di collaborazione controlla i rami di esecuzione del nodo decisionale che gestiscono l'eccezione innalzata. Questo diagramma principale può accedere all'eccezione mediante la variabile di sistema `currentException`.
- Se il percorso di esecuzione termina in Errore su completamento, l'ambiente runtime di collaborazione termina la collaborazione, genera una voce nel log di collaborazione e crea un flusso non risolto.

  L'ambiente runtime di collaborazione associa al flusso non risolto un testo di eccezione contenuto nell'eccezione. Se l'eccezione non contiene un testo di eccezione, l'ambiente runtime di collaborazione utilizza il messaggio predefinito:

  ```
  Scenario failed.
  ```

E' meglio aumentare il livello di un'eccezione in modo esplicito quando si verifica l'eccezione, invece di terminare semplicemente in errore. Quando il codice innalza esplicitamente un'eccezione all'ambiente runtime di collaborazione, l'amministratore può utilizzare il Flow Manager per visualizzare il testo dell'eccezione come parte del flusso non risolto. Per ulteriori informazioni, consultare "Innalzamento dell'eccezione" a pagina 187.

Sono presenti più blocchi funzione Aumenta l'eccezione collaborazione, ognuno svolge un'attività leggermente diversa. I blocchi funzione Aumenta l'eccezione collaborazione 1, Aumenta l'eccezione collaborazione 2, Aumenta l'eccezione collaborazione 3 Aumenta l'eccezione collaborazione 4 e Aumenta l'eccezione collaborazione 5 consentono di specificare fino a cinque valore di parametro del messaggio per il testo del messaggio di eccezione. Il blocco funzione Aumenta l'eccezione collaborazione con i parametri consente di specificare un vettore di valori di parametri del messaggio.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.sendEmail(). Per ulteriori informazioni, consultare "sendEmail()" a pagina 387.

Aumenta l'eccezione collaborazione 1

Prepara un'eccezione di collaborazione per innalzarla al successivo livello superiore di esecuzione. Questo blocco funzione crea un nuovo oggetto eccezione con il tipo eccezione specificato e un messaggio di eccezione ottenuto dal file di messaggi di collaborazione. Il messaggio viene identificato in base al numero nel file di messaggi. Questo blocco funzione fornisce la possibilità di passare un singolo valore di parametro del messaggio per il testo del messaggio.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Tipo eccezione
Una stringa (String) che specifica il tipo di eccezione.

messageNum Un numero intero che specifica il numero del messaggio associato con l'oggetto eccezione.

Parametro 1 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Informazioni correlate

Per ulteriori informazioni sull'utilizzo dei blocchi funzione Aumenta l'eccezione collaborazione, consultare "Aumenta l'eccezione collaborazione" a pagina 301.

Questo blocco funzione si basa sul metodo BaseCollaboration.sendEmail(). Per ulteriori informazioni, consultare "sendEmail()" a pagina 387.

Aumenta l'eccezione collaborazione 2

Prepara un'eccezione di collaborazione per innalzarla al successivo livello superiore di esecuzione. Questo blocco funzione crea un nuovo oggetto eccezione con il tipo eccezione specificato e un messaggio di eccezione ottenuto dal file di messaggi di collaborazione. Il messaggio viene identificato in base al numero nel file di messaggi. Questo blocco funzione fornisce la possibilità di passare due valori di parametro del messaggio per il testo del messaggio.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Tipo eccezione
Una stringa (String) che specifica il tipo di eccezione.

messageNum Un numero intero che specifica il numero del messaggio associato con l'oggetto eccezione.

Parametro 1 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 2 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Informazioni correlate

Per ulteriori informazioni sull'utilizzo dei blocchi funzione Aumenta l'eccezione collaborazione, consultare "Aumenta l'eccezione collaborazione" a pagina 301.

Questo blocco funzione si basa sul metodo BaseCollaboration.sendEmail(). Per ulteriori informazioni, consultare "sendEmail()" a pagina 387.

Aumenta l'eccezione collaborazione 2

Prepara un'eccezione di collaborazione per innalzarla al successivo livello superiore di esecuzione. Questo blocco funzione crea un nuovo oggetto eccezione con il tipo eccezione specificato e un messaggio di eccezione ottenuto dal file di messaggi di collaborazione. Il messaggio viene identificato in base al numero nel file di messaggi. Questo blocco funzione fornisce la possibilità di passare tre valori di parametro del messaggio per il testo del messaggio.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Tipo eccezione
Una stringa (String) che specifica il tipo di eccezione.

messageNum Un numero intero che specifica il numero del messaggio associato con l'oggetto eccezione.

Parametro 1 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 2 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 3 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Informazioni correlate

Per ulteriori informazioni sull'utilizzo dei blocchi funzione Aumenta l'eccezione collaborazione, consultare "Aumenta l'eccezione collaborazione" a pagina 301.

Questo blocco funzione si basa sul metodo BaseCollaboration.sendEmail(). Per ulteriori informazioni, consultare "sendEmail()" a pagina 387.

Aumenta l'eccezione collaborazione 4

Prepara un'eccezione di collaborazione per innalzarla al successivo livello superiore di esecuzione. Questo blocco funzione crea un nuovo oggetto eccezione con il tipo eccezione specificato e un messaggio di eccezione ottenuto dal file di messaggi di collaborazione. Il messaggio viene identificato in base al numero nel file di messaggi. Questo blocco funzione fornisce la possibilità di passare quattro valori di parametro del messaggio per il testo del messaggio.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Tipo eccezione
Una stringa (String) che specifica il tipo di eccezione.

messageNum Un numero intero che specifica il numero del messaggio associato con l'oggetto eccezione.

Parametro 1 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 2 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 3 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 4 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Informazioni correlate

Per ulteriori informazioni sull'utilizzo dei blocchi funzione Aumenta l'eccezione collaborazione, consultare "Aumenta l'eccezione collaborazione" a pagina 301.

Questo blocco funzione si basa sul metodo BaseCollaboration.sendEmail(). Per ulteriori informazioni, consultare "sendEmail()" a pagina 387.

Aumenta l'eccezione collaborazione 5

Prepara un'eccezione di collaborazione per innalzarla al successivo livello superiore di esecuzione. Questo blocco funzione crea un nuovo oggetto eccezione con il tipo eccezione specificato e un messaggio di eccezione ottenuto dal file di messaggi di collaborazione. Il messaggio viene identificato in base al numero nel file di messaggi. Questo blocco funzione fornisce la possibilità di passare cinque valori di parametro del messaggio per il testo del messaggio.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Tipo eccezione
Una stringa (String) che specifica il tipo di eccezione.

messageNum Un numero intero che specifica il numero del messaggio associato con l'oggetto eccezione.

Parametro 1 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 2 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 3 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 4 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Parametro 5 Una stringa (String) che specifica il valore di un singolo parametro del messaggio.

Informazioni correlate

Per ulteriori informazioni sull'utilizzo dei blocchi funzione Aumenta l'eccezione collaborazione, consultare "Aumenta l'eccezione collaborazione" a pagina 301.

Questo blocco funzione si basa sul metodo BaseCollaboration.sendEmail(). Per ulteriori informazioni, consultare "sendEmail()" a pagina 387.

Aumenta l'eccezione collaborazione con i parametri

Prepara un'eccezione di collaborazione per innalzarla al successivo livello superiore di esecuzione. Questo blocco funzione fornisce un'altro modo per creare un nuovo oggetto eccezione che contiene un messaggio specificato nel file di messaggi. Tutti i valori di parametro sono collocati in un vettore di `Objects`.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Tipo eccezione
Una stringa (String) che specifica il tipo di eccezione.

messageNum Un numero intero che specifica il numero del messaggio associato con l'oggetto eccezione.

Parametri Un vettore di valori di parametri del messaggio. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio. All'interno del messaggio (nel file di messaggi di collaborazione), i parametri del messaggio sono indicati da numeri interi racchiusi tra parentesi graffe; ad esempio, {1}.

Note

Questo blocco funzione è utile per innalzare un oggetto di eccezione che:

- E' stato gestito precedentemente dalla collaborazione. Ad esempio, uno scenario potrebbe ricevere un'eccezione, assegnarla ad una variabile ed eseguire altre attività.
- Presenta più di cinque parametri di messaggio. Mentre gli altri blocchi funzione Aumenta l'eccezione collaborazione non possono gestire più di cinque parametri, il vettore di parametri può contenere qualsiasi numero di parametri.

Informazioni correlate

Per ulteriori informazioni sull'utilizzo dei blocchi funzione Aumenta l'eccezione collaborazione, consultare "Aumenta l'eccezione collaborazione" a pagina 301.

Questo blocco funzione si basa sul metodo BaseCollaboration.raiseException(). Per ulteriori informazioni, consultare "raiseException()" a pagina 382.

Invia Email

Invia un messaggio e-mail in modo asincrono.

Input

Collaborazione
L'oggetto di collaborazione corrente.

Messaggio Una stringa (String) che contiene il testo del messaggio e-mail.

Oggetto La riga oggetto del messaggio e-mail.

Destinatari Un vettore (Vector) che contiene gli indirizzi e-mail dei destinatari del messaggio. Questo vettore contiene oggetti String.

Note

Il blocco funzione Invia Email può inviare un messaggio e-mail ai destinatari specificati nel vettore Destinatari, se:

- E' stato specificato un indirizzo e-mail nel campo **Indirizzo e-mail per notifica** nella finestra di dialogo Proprietà oggetto di collaborazione.
- La collaborazione e-mail e l'adattatore e-mail sono in esecuzione. (La collaborazione e-mail viene istanziata e configurata automaticamente quando si avvia InterChange Server Express e non richiede immissioni da parte dell'utente). Se l'adattatore e-mail non è in esecuzione, Invia Email *non* determina l'arresto dell'esecuzione della collaborazione.

Nota: Il blocco funzione Errore log invia automaticamente un messaggio di errore ad un destinatario e-mail (presupponendo che la collaborazione e l'adattatore e-mail siano in esecuzione). Il blocco funzione Invia Email consente di inviare esplicitamente un messaggio e-mail.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.sendEmail(). Per ulteriori informazioni, consultare "sendEmail()" a pagina 387.

ServiceCallException

Una costante che rappresenta un errore causato da una chiamata di servizio non riuscita (ad esempio, se un adattatore o un'applicazione non sono disponibili).

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″ServiceCallException″.

SystemException

Una costante che rappresenta un errore interno nel sistema InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″SystemException″.

TransactionException

Una costante che rappresenta un errore relativo al comportamento di transazione di una collaborazione transazionale (ad esempio, un rollback non riuscito o la mancata applicazione della compensazione da parte della collaborazione).

Nota: Questo blocco funzione si trova nella cartella Generale\APIs\Maschera di collaborazione\Eccezione\Costanti.

Output

Restituisce una stringa (String) che contiene il valore ″TransactionException″.

Capitolo 14. Blocchi funzione di connessione database

I blocchi funzione nella cartella Generale/APIs/Connessione Database forniscono le funzionalità di base per gestire le connessioni ai database e l'esecuzione di query SQL nel database. Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 71. Riepilogo dei blocchi funzione di connessione database

Inizia transazione

Inizia una transazione esplicita per la connessione corrente.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Eccezioni

Il blocco funzione Inizia transazione può generare l'eccezione `CwDBConnectionException` se si verifica un errore di database.

Note

Il blocco funzione Inizia transazione contrassegna l'inizio di una nuova transazione esplicita nella connessione corrente. L'insieme dei blocchi funzione Inizia transazione, Convalida e Esegui Rollback consente di gestire i limiti delle transazioni per una transazione esplicita. Questa transazione contiene query SQL,

che comprendono le istruzioni SQL INSERT, DELETE o UPDATE, e una procedura memorizzata che include una delle istruzioni SQL.

Importante
Utilizzare Inizia transazione solo se la connessione utilizza il bracketing transazioni esplicite. Se la connessione utilizza il bracketing transazioni implicite, l'utilizzo di Inizia transazione genera un'eccezione `CwDBTransactionException`.

Prima dell'inizio di una transazione esplicita, è necessario creare un oggetto `CwDBConnection` con il blocco funzione Ottieni connessione database. Assicurarsi che questa connessione utilizzi il bracketing transazioni esplicite.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.beginTransaction(). Per ulteriori informazioni, consultare "beginTransaction()" a pagina 425.

Convalida

Esegue il commit della transazione attiva associata con la connessione corrente.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Eccezioni

Il blocco funzione Convalida può generare l'eccezione `CwDBConnectionException` se si verifica un errore di database.

Note

Il blocco funzione Convalida termina la transazione attiva eseguendo il commit delle modifiche al database associato con la connessione corrente. L'insieme dei blocchi funzione Inizia transazione, Convalida e Esegui Rollback consente di gestire i limiti delle transazioni per una transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE o UPDATE, e una procedura memorizzata che include una delle istruzioni SQL.

Importante
Utilizzare Convalida solo se la connessione utilizza il bracketing transazioni esplicite. Se la connessione utilizza il bracketing transazioni implicite, l'utilizzo di Convalida genera un'eccezione `CwDBTransactionException`. Se un transazione esplicita non termina con Convalida (o Esegui rollback) prima di rilasciare la connessione, InterChange Server Express termina implicitamente la transazione in base all'esito della collaborazione. Se la collaborazione ha avuto esito positivo, ICS esegue il commit di questa transazione di database. Se la collaborazione *non* ha avuto esito positivo, ICS esegue implicitamente il rollback della transazione di database. Indipendentemente dall'esito della collaborazione, ICS registra un'avvertenza.

Prima dell'inizio di una transazione esplicita, è necessario creare un oggetto `CwDBConnection` con il blocco funzione Ottieni connessione database. Assicurarsi che questa connessione utilizzi il bracketing transazioni esplicite.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.commit(). Per ulteriori informazioni, fare riferimento a "commit()" a pagina 426.

Esegui SQL preparata

Esegue una query SQL preparata specificando la relativa sintassi.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Query Un rappresentazione in formato String della query SQL da eseguire nel database.

Eccezioni

Il blocco funzione Esegui SQL preparata può generare l'eccezione `CwDBSQLException` se si verifica un errore di database.

Note

Le istruzioni SQL che possono essere eseguite (se si dispone delle autorizzazioni necessarie sul database) sono le seguenti:

- L' istruzione SELECT per richiedere dati da una o più tabelle di database.

  Utilizzare i blocchi funzione Più righe presenti e Ottieni riga successiva per accedere ai dati recuperati.
- Le istruzioni SQL che modificano i dati nel database
  - INSERT
  - DELETE
  - UPDATE

  Il blocco funzione Esegui SQL preparata invia la stringa *query* specificata come istruzione SQL preparata al database associato con la connessione corrente. La prima volta che viene eseguita, questa query viene inviata come stringa al database, che la compila in formato eseguibile (istruzione preparata), esegue l'istruzione SQL e restituisce l'istruzione preparata al blocco funzione. Il blocco funzione salva l'istruzione preparata in memoria. Utilizzare Esegui SQL preparata per le istruzioni SQL che devono essere eseguite più volte.

  Nota: Il blocco funzione Esegui SQL *non* salva l'istruzione preparata ed è quindi utile per le query che sono eseguite una sola volta.

> **Importante**
> Prima di eseguire una query con Esegui SQL preparata, è necessario ottenere una connessione al database desiderato generando un oggetto `CwDBConnection` con il blocco funzione Ottieni connessione database.

Se la connessione utilizza il bracketing transazioni esplicite, è necessario avviare esplicitamente ogni transazione con Inizia transazione e terminarla con Convalida o Esegui rollback.

- L'istruzione CALL per eseguire una procedura memorizzata preparata, con la limitazione che questa procedura memorizzata *non può* utilizzare alcun parametro OUT.

  Per eseguire procedure memorizzate con parametri OUT, utilizzare il blocco funzione Esegui procedura memorizzata. Per ulteriori informazioni, consultare "Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.executePreparedSQL(). Per ulteriori informazioni, consultare "executePreparedSQL()" a pagina 427.

Esegui SQL preparata con parametro

Esegue una query SQL preparata specificando la relativa sintassi e i parametri.

Input

Connessione database
: Un oggetto CwDBConnection che rappresenta la connessione al database.

Query
: Un rappresentazione in formato String della query SQL da eseguire nel database.

Parametri
: Un oggetto Vector di argomenti da passare ai parametri della query SQL.

Eccezioni

Il blocco funzione Esegui SQL preparata con parametro può generare l'eccezione `CwDBSQLException` se si verifica un errore di database.

Note

Fare riferimento alla sezione Note per il blocco funzione Esegui SQL preparata in questo capitolo.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.executePreparedSQL(). Per ulteriori informazioni, consultare "executePreparedSQL()" a pagina 427.

Esegui SQL

Esegue una query SQL statica specificando la relativa sintassi.

Input

Connessione database
: Un oggetto CwDBConnection che rappresenta la connessione al database.

Query Un rappresentazione in formato String della query SQL da eseguire nel database.

Eccezioni

Il blocco funzione Esegui SQL può generare l'eccezione `CwDBSQLException` se si verifica un errore di database.

Note

Il blocco funzione Esegui SQL invia la stringa *query* specificata come istruzione SQL statica al database associato con la connessione corrente. Questa query viene inviata come stringa al database, che la compila in formato eseguibile ed esegue l'istruzione SQL senza salvarla nel formato eseguibile. Utilizzare Esegui SQL per le istruzioni SQL che devono essere eseguite una sola volta.

Nota: I blocchi funzione Esegui SQL preparata e Esegui SQL preparata con parametro salvano il formato eseguibile (istruzione preparata) e sono quindi utili per le query che sono eseguite più volte.

Importante

Prima di eseguire una query con Esegui SQL, è necessario ottenere una connessione al database desiderato generando un oggetto `CwDBConnection` con il blocco funzione Ottieni connessione database.

Le istruzioni SQL che possono essere eseguite (se si dispone delle autorizzazioni necessarie sul database) sono le seguenti:

- L' istruzione SELECT per richiedere dati da una o più tabelle di database.

  Utilizzare i blocchi funzione Più righe presenti e Ottieni riga successiva per accedere ai dati recuperati.
- Le istruzioni SQL che modificano i dati nel database
  - INSERT
  - DELETE
  - UPDATE

  Se la connessione utilizza il bracketing transazioni esplicite, è necessario avviare esplicitamente ogni transazione con il blocco funzione Inizia transazione e terminarla con il blocco funzione Convalida o Esegui rollback.
- L'istruzione CALL per eseguire una procedura memorizzata preparata con la limitazione che questa procedura memorizzata *non può* utilizzare alcun parametro OUT.

  Per eseguire procedure memorizzate con parametri OUT, utilizzare il blocco funzione Esegui procedura memorizzata. Per ulteriori informazioni, consultare "Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.executeSQL(). Per ulteriori informazioni, consultare "executeSQL()" a pagina 429.

Esegui SQL con parametro

Esegue una query SQL statica specificando la relativa sintassi e i parametri.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Query Un rappresentazione in formato String della query SQL da eseguire nel database.

Parametri Un oggetto Vector di argomenti da passare ai parametri della query SQL.

Eccezioni

Il blocco funzione Esegui SQL con parametro può generare l'eccezione `CwDBSQLException` se si verifica un errore di database.

Note

Fare riferimento alla sezione Note per il blocco funzione Esegui SQL in questo capitolo.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.executeSQL(). Per ulteriori informazioni, consultare "executeSQL()" a pagina 429.

Esegui procedura memorizzata

Esegue una procedura memorizzata SQL specificando il nome e un vettore di parametri.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Procedura memorizzata
Un rappresentazione in formato String della procedura memorizzata da eseguire nel database.

Parametri Un oggetto Vector di argomenti da passare ai parametri della query SQL.

Eccezioni

Il blocco funzione Esegui procedura memorizzata può generare l'eccezione `CwDBSQLException` se si verifica un errore di database.

Note

Il blocco funzione Esegui procedura memorizzata invia una chiamata alla *`storedProcedure`* specificata del database associato con la connessione corrente. Questo blocco funzione invia la chiamata alla procedura memorizzata come istruzione SQL preparata; la prima volta che viene eseguita, questa procedura memorizzata viene inviata come stringa al database, che la compila in formato eseguibile (istruzione preparata), esegue l'istruzione SQL e restituisce l'istruzione preparata al blocco funzione Esegui procedura memorizzata. Il blocco funzione salva l'istruzione preparata in memoria.

Importante
Prima di eseguire una procedura memorizzata con questo blocco funzione, è necessario creare un oggetto `CwDBConnection` con il blocco funzione Ottieni connessione database.

Per gestire i dati restituiti dalla procedura memorizzata, utilizzare i blocchi funzione Più righe presenti e Ottieni riga successiva.

E' possibile anche utilizzare il blocco funzione Esegui SQL, Esegui SQL con parametro, Esegui SQL preparata o Esegui SQL preparata con parametro per eseguire una procedura memorizzata se la procedura memorizzata *non* contiene parametri OUT. Se la procedura memorizzata utilizza parametri OUT, si *deve* utilizzare il blocco funzione Esegui procedura memorizzata per eseguirla.

A differenza dei blocchi funzione Esegui SQL e Esegui SQL preparata, non è necessario passare l'istruzione SQL completa per eseguire la procedura memorizzata. Con Esegui procedura memorizzata, è necessario passare solo il nome della procedura memorizzata e un vettore di parametri (`Vector`) di oggetti `CwDBStoredProcedureParam`. Il procedura memorizzata Esegui procedura memorizzata può determinare il numero di parametri dal vettore *`storedProcParameters`* e generare l'istruzione di chiamata per la procedura memorizzata.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.executeStoredProcedure(). Per ulteriori informazioni, consultare "executeStoredProcedure()" a pagina 430.

Ottieni connessione database

Stabilisce una connessione ad un database.

Input

Nome pool connessione
Una stringa (String) che specifica il nome di un pool di connessioni valido.

Output

Restituisce un oggetto CwDBConnection.

Eccezioni

Il blocco funzione Ottieni connessione database può generare l'eccezione `CwDBConnectionFactoryException` se si verifica un errore nel tentativo di stabilire la connessione al database.

Note

Il blocco funzione Ottieni connessione database ottiene una connessione dal pool di connessioni specificato dal Nome pool di connessioni di input. Questa connessione fornisce un meccanismo per eseguire query ed aggiornamenti sul database associato con la connessione. Tutte le connessioni di un particolare pool di

connessioni sono associate allo stesso database. Il blocco funzione restituisce un oggetto `CwDBConnection` mediante il quale è possibile eseguire query e gestire transazioni.

Per impostazione predefinita, tutte le connessioni utilizzano bracketing transazioni implicite in base al proprio modello di programmazione transazione. Per specificare un modello di programmazione transazione *per una particolare connessione*, utilizzare il blocco funzione Ottieni connessione database con transazione.

La connessione viene rilasciata quando l'oggetto di collaborazione termina l'esecuzione. La connessione può essere chiusa in modo esplicito con il blocco funzione Rilascio. E' possibile determinare se una connessione è stata rilasciata mediante il blocco funzione E' attiva. Per ulteriori informazioni, consultare "Rilascio di una connessione" a pagina 244.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.getDBConnection(). Per ulteriori informazioni, consultare "getDBConnection()" a pagina 372.

Ottieni connessione database con transazione

Stabilisce una connessione ad un database utilizzando un modello di programmazione transazione specifico.

Input

Nome pool connessione
Una stringa (String) che specifica il nome di un pool di connessioni valido.

Transazione implicita
Un valore booleano per indicare il modello di programmazione transazione da utilizzare per il database associato con la connessione. I valori validi sono:

`true` Il database utilizza bracketing transazioni implicite

`false` Il database utilizza bracketing transazioni esplicite

Output

Restituisce un oggetto CwDBConnection.

Eccezioni

Il blocco funzione Ottieni connessione database con transazione può generare l'eccezione `CwDBConnectionFactoryException` se si verifica un errore nel tentativo di stabilire la connessione al database.

Note

Il blocco funzione Ottieni connessione database ottiene una connessione dal pool di connessioni specificato dal Nome pool di connessioni di input. Questa connessione fornisce un meccanismo per eseguire query ed aggiornamenti sul database associato con la connessione. Tutte le connessioni di un particolare pool di connessioni sono associate allo stesso database. Il blocco funzione restituisce un oggetto `CwDBConnection` mediante il quale è possibile eseguire query e gestire transazioni.

La connessione viene rilasciata quando l'oggetto di collaborazione termina l'esecuzione. La connessione può essere chiusa in modo esplicito con il blocco funzione Rilascio. E' possibile determinare se una connessione è stata rilasciata mediante il blocco funzione E' attiva. Per ulteriori informazioni, consultare "Rilascio di una connessione" a pagina 244.

Informazioni correlate

Questo blocco funzione si basa sul metodo BaseCollaboration.getDBConnection(). Per ulteriori informazioni, consultare "getDBConnection()" a pagina 372.

Ottieni riga successiva

Ottiene la riga successiva dai risultati della query.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Output

Restituisce una riga di dati dai risultati della query associati con la connessione corrente. Utilizzare questo metodo per recuperare i risultati da una query che restituisce dati. Tali query comprendono istruzioni SELECT e procedure memorizzate.

Eccezioni

Il blocco funzione Ottieni riga successiva può generare l'eccezione `CwDBSQLException` se si verifica un errore di database.

Note

Il blocco funzione Ottieni riga successiva restituisce una riga di dati dai risultati della query associati con la connessione corrente. Utilizzare questo blocco funzione per recuperare i risultati da una query che restituisce dati. Tali query comprendono istruzioni SELECT e procedure memorizzate. Alla connessione può essere associata una sola query alla volta. Pertanto, se si esegue un'altra query prima che Ottieni riga successiva abbia restituito l'ultima riga di dati, i risultati della query iniziale sono persi.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.nextRow(). Per ulteriori informazioni, fare riferimento a "nextRow()" a pagina 434.

Ottieni conteggio di aggiornamento

Determina il numero di righe interessate dall'ultima operazione di scrittura nel database.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Output

Restituisce un numero intero che rappresenta il numero di righe interessate dall'ultima operazione di scrittura.

Eccezioni

Il blocco funzione Ottieni conteggio di aggiornamento può generare l'eccezione `CwDBConnectionException` se si verifica un errore di database.

Note

Il blocco funzione Ottieni conteggio di aggiornamento indica quante righe sono state modificate dall'ultima operazione di aggiornamento nel database associato alla connessione corrente. Questo metodo è utile dopo aver inviato un'istruzione UPDATE o INSERT al database, se si desidera determinare il numero di righe interessate da questa istruzione SQL.

Importante

Prima di utilizzare questo metodo, è necessario creare un oggetto `CwDBConnection` con il blocco funzione Ottieni connessione database ed inviare una query che aggiorni il database con uno dei seguenti blocchi funzione:

- Esegui SQL
- Esegui SQL con parametro
- Esegui SQL preparata
- Esegui SQL preparata con parametro

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.getUpdateCount(). Per ulteriori informazioni, consultare "getUpdateCount()" a pagina 431.

Più righe presenti

Determina se nei risultati della query corrente sono presenti altre righe da elaborare.

Input

Connessione database

Un oggetto CwDBConnection che rappresenta la connessione al database.

Output

Restituisce `true` se ci sono altre righe da elaborare nei risultati della query; altrimenti restituisce `false`.

Eccezioni

Il blocco funzione Più righe presenti può generare l'eccezione `CwDBSQLException` se si verifica un errore di database.

Note

Il blocco funzione Più righe presenti determina se nei risultati della query associati con la connessione corrente sono presenti altre righe da elaborare. Utilizzare questo blocco funzione per recuperare i risultati da una query che restituisce dati. Tali query comprendono istruzioni SELECT e procedure memorizzate. Alla connessione può essere associata una sola query alla volta. Pertanto, se si esegue un'altra query prima che Più righe presenti abbia restituito `false`, i dati della query iniziale sono persi.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.hasMoreRows(). Per ulteriori informazioni, consultare "hasMoreRows()" a pagina 432.

In transazione

Determina se una transazione è attiva nella connessione database corrente.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Output

Restituisce `true` se una transazione è attiva; altrimenti restituisce `false`.

Eccezioni

Il blocco funzione In transazione può generare l'eccezione `CwDBConnectionException` se si verifica un errore di database.

Note

Il blocco funzione In transazione restituisce un valore booleano che indica se la connessione corrente presenta una transazione attiva; ovvero, una transazione è stata avviata e non è terminata.

Importante
Prima dell'inizio di una transazione, è necessario creare un oggetto `CwDBConnection` con il blocco funzione Ottieni connessione database.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.inTransaction(). Per ulteriori informazioni, consultare "inTransaction()" a pagina 433.

E' attiva

Determina se la connessione corrente è attiva.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Output

Restituisce `true` se la connessione corrente è attiva; altrimenti, restituisce `false`.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.isActive(). Per ulteriori informazioni, consultare "isActive()" a pagina 433.

Rilascio

Rilascia la connessione corrente e la restituisce al pool di connessioni.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Eccezioni

Il blocco funzione Rilascio può generare l'eccezione `CwDBConnectionException` se si verifica un errore di database.

Note

Il blocco funzione Rilascio rilascia esplicitamente la connessione corrente dall'oggetto di collaborazione. Una volta rilasciata, la connessione ritorna al relativo pool di connessioni, dove è disponibile per gli altri componenti (mappe o collaborazioni) che richiedono una connessione al database associato. Se non si rilascia esplicitamente una connessione, l'oggetto di collaborazione la rilascia implicitamente alla fine dell'esecuzione della collaborazione corrente. Pertanto, *non* è possibile salvare una connessione in una variabile statica e riutilizzarla.

Importante
Non utilizzare Rilascio se una transazione è attualmente attiva. Con il bracketing transazioni implicite, ICS non chiude la transazione di database fino a quando non viene determinato l'esito positivo o negativo della collaborazione. Pertanto, l'utilizzo di questo blocco funzione per una connessione che utilizza bracketing transazioni implicite genera una eccezione `CwDBTransactionException`. Se non si gestisce in modo esplicito questa eccezione, viene effettuato anche un rollback automatico della transazione attiva. E' possibile utilizzare il blocco funzione In transazione per determinare se una transazione è attiva.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.release(). Per ulteriori informazioni, consultare "release()" a pagina 434.

Esegui Rollback

Esegue il rollback della transazione attiva associata con la connessione corrente.

Input

Connessione database
Un oggetto CwDBConnection che rappresenta la connessione al database.

Eccezioni

Il blocco funzione Esegui Rollback può generare l'eccezione `CwDBTransactionException`.

Note

Il blocco funzione Esegui Rollback termina la transazione attiva eseguendo il rollback delle modifiche al database associato con la connessione corrente. L'insieme dei blocchi funzione Inizia transazione, Convalida e Esegui Rollback consente di gestire i limiti delle transazioni per una transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE o UPDATE, e una procedura memorizzata che include una delle istruzioni SQL. Se il rollback non riesce, Esegui Rollback genera un'eccezione `CwDBTransactionException` e registra un errore.

Importante
Utilizzare Esegui Rollback solo se la connessione utilizza il bracketing transazioni esplicite. Se la connessione utilizza il bracketing transazioni implicite, l'utilizzo di Esegui Rollback genera un'eccezione `CwDBTransactionException`. Se un transazione esplicita non termina con il blocco funzione Esegui Rollback (o Convalida) prima di rilasciare la connessione, InterChange Server Express termina implicitamente la transazione in base all'esito della collaborazione. Se la collaborazione ha avuto esito positivo, InterChange Server Express esegue il commit di questa transazione di database. Se la collaborazione *non* ha avuto esito positivo, InterChange Server Express esegue implicitamente il rollback della transazione di database. Indipendentemente dall'esito della collaborazione, InterChange Server Express registra un'avvertenza.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBConnection.rollBack(). Per ulteriori informazioni, consultare "rollBack()" a pagina 435.

Capitolo 15. Blocchi funzione di procedura memorizzata database

I blocchi funzione di procedura memorizzata database forniscono le funzionalità di base per gestire i parametri di procedura memorizzata. Questi blocchi funzione si trovano nella cartella \Generale\APIs\Parametro di procedura memorizzata DB.

Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 72. Riepilogo dei blocchi funzione di procedura memorizzata database

Blocco funzione	Pagina
Ottieni tipo di parametro	323
Ottieni valore parametro	324
Nuovo parametro di procedura memorizzata DB	324

Ottieni tipo di parametro

Recupera il tipo in/out del parametro di procedura memorizzata corrente.

Input

CwDBStoredProcedureParam
Il parametro di procedura memorizzata (un oggetto CwDBStoredProcedureParam) il cui tipo in/out si vuole recuperare.

Output

Restituisce il tipo in/out del parametro di procedura memorizzata associato come costante di numero intero.

Note

Il blocco funzione Ottieni tipo di parametro restituisce il tipo in/out del parametro di procedura memorizzata corrente. Il tipo di parametro in/out indica come il parametro viene utilizzato dalla procedura memorizzata. La classe `CwDBStoredProcedureParam` rappresenta ciascun tipo in/out come costante, come mostrato nella Tabella 73.

Tabella 73. Tipi di parametro in/out (blocco funzione Ottieni tipo di parametro)

Tipo di parametro in/out	Descrizione	Costante tipo in/out
parametro IN	Un parametro IN è *di solo input*; quindi la procedura memorizzata accetta il suo valore come input, ma *non* utilizza il parametro per restituire un valore.	`PARAM_IN`
parametro OUT	Un parametro OUT è *di solo output*; quindi la procedura memorizzata *non* considera il suo valore come input, ma utilizza il parametro per restituire un valore.	`PARAM_OUT`

Tabella 73. Tipi di parametro in/out (blocco funzione Ottieni tipo di parametro) (Continua)

Tipo di parametro in/out	Descrizione	Costante tipo in/out
parametro INOUT	Un parametro INOUT è *di input e output*; quindi la procedura memorizzata accetta il suo valore come input ed utilizza anche il parametro per restituire un valore.	`PARAM_INOUT`

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBStoredProcedureParam.getParamType(). Per ulteriori informazioni, consultare "getParamType()" a pagina 438.

Ottieni valore parametro

Recupera il valore del parametro di procedura memorizzata corrente.

Input

CwDBStoredProcedureParam
Il parametro di procedura memorizzata (un oggetto CwDBStoredProcedureParam) il cui valore si vuole recuperare.

Output

Restituisce il valore del parametro di procedura memorizzata associato come oggetto Java.

Note

Il blocco funzione Ottieni valore parametro restituisce il valore del parametro come oggetto Java (ad esempio `Integer`, `Double` o `String`). Se il valore restituito ad un parametro OUT è `NULL` JDBC, Ottieni valore parametro restituisce la costante `null`.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwDBStoredProcedureParam.getValue(). Per ulteriori informazioni, consultare "getValue()" a pagina 439.

Nuovo parametro di procedura memorizzata DB

Crea una nuova istanza CwDBStoredProcedureParam contenente le informazioni sull'argomento per il parametro di una procedura memorizzata.

Input

Tipo di parametro
Il tipo in/out del parametro della procedura memorizzata associato.

Valore parametro
Il valore di argomento da inviare alla procedura memorizzata. Questo valore è uno dei tipi di dati Java riportati di seguito:

- String
- int

- Integer
- Long
- double
- Double
- float
- Float
- BigDecimal
- boolean
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp
- java.sql.Blob
- java.sql.Clob
- byte[]
- Array
- Struct

Output

Restituisce un nuovo oggetto CwDBStoredProcedureParam per contenere le informazioni per un argomento nella dichiarazione della procedura memorizzata.

Note

Il blocco funzione Nuovo parametro di procedura memorizzata DB crea un'istanza `CwDBStoredProcedureParam` per descrivere un parametro per la procedura memorizzata. Le informazioni di parametro sono le seguenti:

- Tipo in/out del parametro

  Il primo valore di input del blocco funzione inizializza questo tipo di parametro in/out. Per un elenco dei tipi di parametro in/out validi, consultare la Tabella 73 a pagina 323.

- Valore del parametro

  Il secondo valore di input del blocco funzione inizializza questo valore di parametro. La classe `CwDBStoredProcedureParam` fornisce un modulo del costruttore per ciascuno dei tipi di dati supportati per il valore di parametro. Per un elenco delle associazioni tra tipi di dati Java e tipi di dati JDBC per i parametri di procedura memorizzata, consultare la Tabella 62 a pagina 239.

Viene fornito un `Vector` Java di parametri di procedura memorizzata al blocco funzione Esegui procedura memorizzata, che crea una chiamata alla procedura memorizzata da un nome di procedura memorizzata e da un vettore di parametri, ed invia la chiamata al database associato con la connessione corrente.

Informazioni correlate

Questo blocco funzione si basa sul costruttore CwDBStoredProcedureParam. Per ulteriori informazioni, consultare "CwDBStoredProcedureParam()" a pagina 437.

Capitolo 16. Blocchi funzione di eccezione

I blocchi funzione nella cartella Generale/APIs/Eccezione collaborazione forniscono le funzionalità di base per gestire le eccezioni. Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 74. Riepilogo dei blocchi funzione di eccezione

Cattura Eccezione collaborazione

Cattura un'eccezione generata nell'attività corrente o nelle relative attività secondarie.

Input

Eccezione collaborazione
L'eccezione di collaborazione (oggetto CollaborationException) che verrà catturata dal blocco funzione.

Note

Per definire un'attività secondaria, fare doppio clic sul blocco funzione Cattura Eccezione collaborazione nell'area di disegno.

Ottieni messaggio

Recupera il testo del messaggio dall'oggetto eccezione.

Input

Eccezione collaborazione
L'eccezione di collaborazione (un oggetto CollaborationException).

Output

Restituisce una stringa (String) che contiene il testo del messaggio dall'oggetto eccezione.

Note

Il blocco funzione Ottieni messaggio è utile per estrarre il testo dell'eccezione dalla variabile di sistema `currentException`. Questo testo di eccezione può essere incluso in una chiamata ad uno dei blocchi funzione Aumenta l'eccezione collaborazione per assicurarsi che la causa dell'eccezione sia innalzata al successivo livello superiore di esecuzione.

Nota: E' possibile utilizzare il blocco funzione In una stringa per recuperare il tipo e il testo dell'eccezione dall'eccezione corrente come stringa formattata.

Informazioni correlate

Questo blocco funzione si basa sul metodo collaborationException.getMessage(). Per ulteriori informazioni, consultare "getMessage()" a pagina 449.

Ottieni numero del messaggio

Recupera il numero di messaggio per il messaggio associato con l'oggetto eccezione.

Input

Eccezione collaborazione
L'eccezione di collaborazione (un oggetto CollaborationException).

Output

Il numero del messaggio, come numero intero (int), associato con il messaggio di eccezione corrente. Se il messaggio di eccezione non proviene da un file di messaggi, questo blocco funzione restituisce zero (0).

Note

Il blocco funzione Ottieni numero del messaggio è utile per ottenere il numero di messaggio associato con un messaggio di eccezione. E' possibile passare questo numero di messaggio ad uno dei blocchi funzione di Aumenta l'eccezione collaborazione, oppure al blocco funzione Errore log.

Informazioni correlate

Questo blocco funzione si basa sul metodo collaborationException.getMsgNumber(). Per ulteriori informazioni, consultare "getMsgNumber()" a pagina 450.

Ottieni tipo secondario

Recupera il tipo secondario di eccezione dall'oggetto eccezione.

Input

Eccezione collaborazione
L'eccezione di collaborazione (un oggetto CollaborationException).

Output

Restituisce una stringa (String) che contiene il tipo secondario di eccezione per l'eccezione corrente. Per ulteriori informazioni sui tipi secondari di eccezione validi, consultare la sezione Note.

Note

Per eccezioni i cui tipi non identificano adeguatamente la causa dell'eccezione, il tipo secondario dell'eccezione può fornire ulteriori informazioni. I seguenti tipi di eccezione utilizzano comunemente i tipi secondari di eccezione:

- JavaException

L'ambiente runtime di collaborazione cattura le eccezioni Java e le include in un'eccezione di collaborazione con un tipo associato di eccezione Java. Una collaborazione può utilizzare il blocco funzione Ottieni tipo secondario sull'eccezione di collaborazione per recuperare il tipo originale di eccezione Java (il nome classe dell'eccezione Java catturata). Tuttavia, normalmente questo non è necessario.

- ServiceCallException

  Il tipo eccezione ServiceCallException si verifica se una chiamata di servizio non riesce. Per sviluppare un meccanismo di collaborazione più robusto, è possibile utilizzare il tipo secondario dell'eccezione per determinare la causa dell'errore della chiamata di servizio. I tipi secondari di eccezione validi sono:

`AppTimeOut`	Un connettore non è riuscito a completare la comunicazione con l'applicazione.
`AppLogOnFailure`	Un connettore non è riuscito ad accedere all'applicazione.
`AppRetrieveByContentFailed`	Un'operazione Retrieve per valori non chiave, eseguita dall'applicazione, non è riuscita ad individuare una corrispondenza.
`AppMultipleHits`	Un'applicazione ha individuato e recuperato più di una entità in risposta ad una richiesta Retrieve.
`AppBusObjDoesNotExist`	Un'operazione Retrieve è stata eseguita nell'applicazione, ma l'entità rappresentata dall'oggetto business non esiste nel database dell'applicazione.
`AppRequestNotYetSent`	Nel caso di un agente connettore parallelo, la richiesta è stata accodata nel master agente ma non è stata inoltrata all'applicazione; pertanto, è possibile inviare nuovamente la richiesta. Per ulteriori informazioni, consultare "Richieste di chiamate di servizio non inviate" a pagina 194.
`ServiceCallTransportException`	Si è verificato un errore nel trasporto e non è possibile determinare con certezza se la richiesta ha raggiunto l'applicazione. Per ulteriori informazioni, consultare "Gestione delle eccezioni runtime correlate al trasporto" a pagina 192.
`AppUnknown`	Qualsiasi tipo di errore non compreso nei precedenti tipi secondari. Se è presente questo tipo secondario di eccezione, l'operazione dell'applicazione richiesta nella chiamata di servizio potrebbe essere terminata o non terminata.

Importante

I tipi secondari di eccezione `AppTimeOut`, `AppLogOnFailure`, `AppRetrieveByContent`, `AppMultipleHits` e `AppUnknown` corrispondono a valori di stato in uscita, che l'adattatore può restituire per indicare la causa dell'errore. Gli adattatori più vecchi potrebbero non supportare tutti i corrispondenti valori di stato in uscita. Assicurarsi che sia eseguita una verifica di tutti gli adattatori collegati alla collaborazione con lo strumento Test Connector per determinare i valori di stato in uscita effettivamente restituiti.

Informazioni correlate

Questo blocco funzione si basa sul metodo collaborationException.getSubType(). Per ulteriori informazioni, consultare "getSubType()" a pagina 450.

Ottieni tipo

Recupera il tipo secondario di eccezione di collaborazione dall'oggetto eccezione. Il tipo di eccezione è una stringa (String) che identifica la causa dell'eccezione.

Input

Eccezione collaborazione
L'eccezione di collaborazione (un oggetto CollaborationException).

Output

Restituisce una stringa (String) che contiene il tipo di eccezione per l'eccezione corrente. Confrontare il valore di questa stringa con una delle seguenti variabili statiche di tipo eccezione:

`AnyException`	Qualsiasi tipo di eccezione. Se sono presenti due collegamenti di eccezione, uno che verifica il tipo di eccezione specifico ed uno che verifica `AnyException`, il collegamento che verifica il tipo di eccezione specifico viene controllato per primo. Se l'eccezione corrente non corrisponde all'eccezione specifica, successivamente viene elaborato il collegamento che verifica `AnyException`.
`AttributeException`	Problema di accesso all'attributo. Ad esempio, la collaborazione ha chiamato `getDouble()` per un attributo String o `getString()` per un attributo non esistente.
`JavaException`	Problema con il codice Java nella logica di collaborazione.
`ObjectException`	L'oggetto business passato ad un metodo non era valido o è stato effettuato l'accesso ad un oggetto null.
`OperationException`	La chiamata di servizio è stata impostata in modo non corretto e non ha potuto essere inviata.
`ServiceCallException`	Chiamata di servizio non riuscita. Ad esempio, un connettore o un'applicazione non sono disponibili.
`SystemException`	Errore interno InterChange Server Express.
`TransactionException`	Errore relativo al comportamento di transazione per una collaborazione transazionale. Ad esempio, il rollback non è riuscito o la collaborazione non ha potuto applicare alcuna compensazione.

Note

Il blocco funzione Ottieni tipo recupera il tipo di eccezione per l'eccezione corrente. Il tipo di eccezione è una `String` che identifica la causa dell'eccezione.

Informazioni correlate

Questo blocco funzione si basa sul metodo collaborationException.getType(). Per ulteriori informazioni, consultare "getType()" a pagina 452.

In una stringa

Formatta in una stringa (String) le informazioni sull'eccezione, che includono il tipo e il testo dell'eccezione.

Input

Eccezione collaborazione
L'eccezione di collaborazione (un oggetto CollaborationException).

Output

Restituisce una stringa (String) che contiene il tipo e il testo dell'eccezione.

Note

Il blocco funzione In una stringa formatta le informazioni di eccezione per l'eccezione corrente come indicato di seguito:

```
exceptionType: messageText
```

Nella riga precedente, *exceptionType* è il tipo di eccezione dell'oggetto eccezione e *messageText* è il relativo testo dell'eccezione.

Nota: E' possibile utilizzare il blocco funzione In una stringa per recuperare solo il testo dell'eccezione dall'eccezione corrente.

Informazioni correlate

Questo blocco funzione si basa sul metodo collaborationException.toString(). Per ulteriori informazioni, consultare "toString()" a pagina 453.

Capitolo 17. Blocchi funzione di esecuzione

I blocchi funzione nella cartella Generale\APIs\Contesto esecuzione forniscono le funzionalità per il contesto di esecuzione. Operano in un contesto di esecuzione globale, che rappresenta un contenitore delle informazioni di contesto accessibili all'utente associate con un flusso fornito. Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 75. Riepilogo dei blocchi funzione del contesto di esecuzione

Blocco funzione	Pagina
Ottieni contesto	333
MAPCONTEXT	333
Nuovo contesto di esecuzione	333
Imposta contesto	334

Ottieni contesto

Recupera il contesto di esecuzione specificato dal contesto di esecuzione globale.

Input

Contesto di esecuzione
Il contesto di esecuzione globale (un oggetto CxExecutionContext).

Nome contesto
Un oggetto String che contiene il nome di un contesto di esecuzione da ottenere dal contesto di esecuzione globale.

Output

Restituisce una istanza del contesto di esecuzione specificato.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwExecutionContext.getContext(). Per ulteriori informazioni, consultare "getContext()" a pagina 442.

MAPCONTEXT

Una costante String utilizzata per indicare che il contesto di esecuzione è specifico della mappa.

Output

Restituisce la stringa MAPCONTEXT.

Nuovo contesto di esecuzione

Crea una nuova istanza di un contesto di esecuzione globale.

Output

Restituisce una nuova istanza del contesto di esecuzione globale.

Note

Il blocco funzione Nuovo contesto di esecuzione restituisce un contesto di esecuzione globale, necessario per contenere il contesto di esecuzione di mappa prima di richiamare una mappa dalla collaborazione.

Informazioni correlate

Questo blocco funzione si basa sul costruttore CwExecutionContext(). Per ulteriori informazioni, consultare "CxExecutionContext()" a pagina 441.

Imposta contesto

Imposta un particolare contesto di esecuzione in modo che sia parte del contesto di esecuzione globale.

Input

Contesto di esecuzione
Il contesto di esecuzione globale (un oggetto CxExecutionContext).

Nome contesto
Un oggetto String che contiene il nome di un contesto di esecuzione da ottenere dal contesto di esecuzione globale.

Contesto Un oggetto che contiene le informazioni per il contesto di esecuzione. Per i contesti di esecuzione di mappa, questo oggetto è di tipo MapExeContext.

Informazioni correlate

Questo blocco funzione si basa sul metodo CwExecutionContext.setContext(). Per ulteriori informazioni, consultare "setContext()" a pagina 442.

Capitolo 18. Blocchi funzione di data

I blocchi funzione nella cartella Generale\Data e nelle relative cartelle secondarie \Formati forniscono le funzionalità per gestire le date.

Tabella 76. Riepilogo dei blocchi funzione di data

Cartella	Blocco funzione	Pagina
Generale\Data	Aggiungi giorno	335
	Aggiungi mese	335
	Aggiungi anno	336
	Data successiva	336
	Data precedente	336
	Data uguale a	337
	Modifica formato	337
	Ottieni giorno	337
	Ottieni mese	338
	Ottieni anno	338
	Ottieni giorno mese anno	338
Generale\Data\Formati	gg-MM-aaaa	339
	ggMMaaaa	339
	ggMMaaaa HH:mm:ss	339

Aggiungi giorno

Aggiunge giorni alla data originale (come specificato dal valore di input Dalla data).

Input

Dalla data Un oggetto String che rappresenta la data originale.

Formato data Un oggetto String che rappresenta il formato della data.

Giorno da aggiungere
Un numero intero che specifica il numero di giorni da aggiungere alla data originale.

Output

Restituisce un oggetto String che contiene la data aggiornata.

Aggiungi mese

Aggiunge mesi alla data originale.

Input

Dalla data Un oggetto String che rappresenta la data originale.

Formato data Un oggetto String che rappresenta il formato della data.

Mese da aggiungere
Un numero intero che specifica il numero di mesi da aggiungere alla data originale.

Output

Restituisce un oggetto String che contiene la data aggiornata.

Aggiungi anno

Aggiunge anni alla data originale.

Input

Dalla data Un oggetto String che rappresenta la data originale.

Formato data Un oggetto String che rappresenta il formato della data.

Anno da aggiungere
Un numero intero che specifica il numero di anni da aggiungere alla data originale.

Output

Un oggetto String che contiene la data aggiornata.

Data successiva

Confronta due date e determina se la Data 1 è successiva alla Data 2.

Input

Data 1 Un oggetto String che rappresenta la prima data da confrontare.

Formato data 1
Un oggetto String che rappresenta il formato della Data 1.

Data 2 Un oggetto String che rappresenta la seconda data da confrontare.

Formato data 2
Un oggetto String che rappresenta il formato della Data 2.

Output

Restituisce `True` se la Data 1 è successiva alla Data 2; altrimenti restituisce `False`.

Data precedente

Confronta due date e determina se la Data 1 è precedente alla Data 2.

Input

Data 1 Un oggetto String che rappresenta la prima data da confrontare.

Formato data 1
Un oggetto String che rappresenta il formato della Data 1.

Data 2 Un oggetto String che rappresenta la seconda data da confrontare.

Formato data 2
Un oggetto String che rappresenta il formato della Data 2.

Output

Restituisce `True` se la Data 1 è precedente alla Data 2; altrimenti restituisce `False`.

Data uguale a

Confronta due date e determina se sono uguali.

Input

Data 1 Un oggetto String che rappresenta la prima data da confrontare.

Formato data 1
Un oggetto String che rappresenta il formato della Data 1.

Data 2 Un oggetto String che rappresenta la seconda data da confrontare.

Formato data 2
Un oggetto String che rappresenta il formato della Data 2.

Output

Restituisce `True`, se le date sono uguali; altrimenti restituisce `False`.

Modifica formato

Modifica un formato di data.

Input

Data Un oggetto String che rappresenta la data a cui si desidera applicare un nuovo formato.

Formato di input
Un oggetto String che rappresenta il formato originale della data.

Formato di output
Un oggetto String che rappresenta il nuovo formato della data.

Output

Restituisce un oggetto String che contiene la data riformattata.

Ottieni giorno

Restituisce il giorno del mese come numero in base all'espressione di data.

Input

Data Un oggetto String che rappresenta la data.

Formato Un oggetto String che rappresenta il formato della data.

Output

Restituisce un numero intero che specifica il giorno del mese.

Ottieni mese

Restituisce il mese dell'anno come numero in base all'espressione di data.

Input

Data Un oggetto String che rappresenta la data.

Formato Un oggetto String che rappresenta il formato della data.

Output

Restituisce un numero intero che specifica il valore numerico del mese.

Ottieni anno

Restituisce l'anno come numero in base all'espressione di data.

Input

Data Un oggetto String che rappresenta la data.

Formato Un oggetto String che rappresenta il formato della data.

Output

Restituisce un numero intero che specifica l'anno.

Ottieni giorno mese anno

Estrae gli elementi giorno, mese e anno da una data di input.

Input

Data Un oggetto String che rappresenta la data.

Formato Un oggetto String che rappresenta il formato della data.

Output

Restituisce tre numeri interi: uno per l'anno, uno per il mese e uno per il giorno.

Ora

Recupera la data odierna.

Input

Formato Un oggetto String che rappresenta il formato da utilizzare per la data.

Output

Restituisce un oggetto String che contiene la data odierna, formattata in base al valore fornito come formato di input.

gg-MM-aaaa

Rappresenta un formato data gg-MM-aaaa (ad esempio, 25-11-2003).

Nota: Questo blocco funzione si trova nella cartella Generale\Data\Formati.

Output

Restituisce un oggetto String che contiene la data formattata come gg-MM-aaaa.

Note

Questo blocco funzione non formatta realmente una data e non può essere utilizzato come blocco funzione indipendente. Deve essere utilizzato insieme ad uno o più blocchi funzione nella cartella Generale\Data (ad esempio, con il blocco funzione Modifica formato o Aggiungi giorno).

ggMMaaaa

Rappresenta un formato data ggMMaaaa (ad esempio, 25112003).

Nota: Questo blocco funzione si trova nella cartella Generale\Data\Formati.

Output

Restituisce un oggetto String che contiene la data formattata come ggMMaaaa.

Note

Questo blocco funzione non formatta realmente una data e non può essere utilizzato come blocco funzione indipendente. Deve essere utilizzato insieme ad uno o più blocchi funzione nella cartella Generale\Data (ad esempio, con il blocco funzione Modifica formato o Aggiungi giorno).

ggMMaaaa HH:mm:ss

Rappresenta un formato data ggMMaaaa HH:mm:ss (ad esempio, 25112003 12:36:40).

Nota: Questo blocco funzione si trova nella cartella Generale\Data\Formati.

Output

Restituisce un oggetto String che contiene la data formattata come ggMMaaaa HH:mm:ss.

Note

Questo blocco funzione non formatta realmente una data e non può essere utilizzato come blocco funzione indipendente. Deve essere utilizzato insieme ad uno o più blocchi funzione nella cartella Generale\Data (ad esempio, con il blocco funzione Modifica formato o Aggiungi giorno).

Capitolo 19. Blocchi funzione di registro e traccia

I blocchi funzione nella cartella Generale\Registro e Traccia e nelle relative cartelle secondarie forniscono le funzionalità per gestire i messaggi di errore, informativi, di avviso e di traccia.

Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 77. Riepilogo dei blocchi funzione di registro e traccia

Cartella	Blocco funzione	Pagina
Generale\Registro e Traccia	Errore log	341
	ID errore log	341
	Informazioni log	343
	ID informativo log	343
	Avviso log	344
	ID di avviso log	344
	Traccia	346
Generale\Registro e Traccia\Errore log	ID errore log 1	342
	ID errore log 2	342
	ID errore log 3	342
Generale\Registro e Traccia\Informazioni log	ID informativo log 1	343
	ID informativo log 2	343
	ID informativo log 3	344
Generale\Registro e Traccia\Avviso log	ID di avviso log 1	345
	ID di avviso log 2	345
	ID di avviso log 3	345
Generale\Registro e Traccia\Traccia	ID traccia 1	346
	ID traccia 2	346
	ID traccia 3	347
	Traccia a livello	347

Errore log

Invia il messaggio di errore specificato al file di log di InterChange Server Express.

Input

Messaggio Il messaggio da inviare al file di log. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID errore log

Invia il messaggio di errore associato all'ID specificato al file di log di InterChange Server Express.

Input

ID L'ID del messaggio di errore da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID errore log 1

Utilizza il parametro specificato per formattare il messaggio di errore associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Errore log.

Input

ID L'ID del messaggio di errore da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro Il parametro utilizzato per formattare il messaggio di errore. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID errore log 2

Utilizza i due parametri specificati per formattare il messaggio di errore associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Errore log.

Input

ID L'ID del messaggio di errore da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio di errore. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio di errore. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID errore log 3

Utilizza i tre parametri specificati per formattare il messaggio di errore associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Errore log.

Input

ID L'ID del messaggio di errore da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio di errore. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio di errore. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 3 Il terzo parametro utilizzato per formattare il messaggio di errore. Questo input può essere di tipo String, byte, short, int, long, float o double.

Informazioni log

Invia il messaggio informativo specificato al file di log di InterChange Server Express.

Input

Messaggio Il messaggio da inviare al file di log. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID informativo log

Invia il messaggio informativo associato all'ID specificato al file di log di InterChange Server Express.

Input

ID L'ID del messaggio informativo da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID informativo log 1

Utilizza il parametro specificato per formattare il messaggio informativo associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Informazioni log.

Input

ID L'ID del messaggio informativo da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro Il parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID informativo log 2

Utilizza i due parametri specificati per formattare il messaggio informativo associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Informazioni log.

Input

ID L'ID del messaggio informativo da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID informativo log 3

Utilizza i tre parametri specificati per formattare il messaggio informativo associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Informazioni log.

Input

ID L'ID del messaggio informativo da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 3 Il terzo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Avviso log

Invia il messaggio di avviso specificato al file di log di InterChange Server Express.

Input

Messaggio Il messaggio da inviare al file di log. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID di avviso log

Invia il messaggio di avviso associato all'ID specificato al file di log di InterChange Server Express.

Input

ID L'ID del messaggio di avviso da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID di avviso log 1

Utilizza il parametro specificato per formattare il messaggio di avviso associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Avviso log.

Input

ID L'ID del messaggio di avviso da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro Il parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID di avviso log 2

Utilizza i due parametri specificati per formattare il messaggio di avviso associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Avviso log.

Input

ID L'ID del messaggio di avviso da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID di avviso log 3

Utilizza i tre parametri specificati per formattare il messaggio di avviso associato con l'ID e poi invia il messaggio al file di log InterChange Server Express.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Avviso log.

Input

ID L'ID del messaggio di avviso da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 3 Il terzo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Traccia

Invia il messaggio di traccia specificato al file di log di InterChange Server Express.

Input

Messaggio Il messaggio da inviare al file di log. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID traccia 1

Utilizza il parametro specificato per formattare il messaggio di traccia associato con l'ID. Utilizza il livello specificato per determinare se il messaggio di traccia viene visualizzato; se la traccia della collaborazione è impostata al livello specificato o ad un livello superiore, il messaggio di traccia viene visualizzato.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Traccia.

Input

ID L'ID del messaggio di traccia da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Livello Il livello di traccia minimo per il quale viene visualizzato il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro Il parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID traccia 2

Utilizza i due parametri specificati per formattare il messaggio di traccia associato con l'ID. Utilizza il livello specificato per determinare se il messaggio di traccia viene visualizzato; se la traccia della collaborazione è impostata al livello specificato o ad un livello superiore, il messaggio di traccia viene visualizzato.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Traccia.

Input

ID L'ID del messaggio di traccia da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Livello Il livello di traccia minimo per il quale viene visualizzato il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

ID traccia 3

Utilizza i tre parametri specificati per formattare il messaggio di traccia associato con l'ID. Utilizza il livello specificato per determinare se il messaggio di traccia viene visualizzato; se la traccia della collaborazione è impostata al livello specificato o ad un livello superiore, il messaggio di traccia viene visualizzato.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Traccia.

Input

ID L'ID del messaggio di traccia da registrare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Livello Il livello di traccia minimo per il quale viene visualizzato il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 1 Il primo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 2 Il secondo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Parametro 3 Il terzo parametro utilizzato per formattare il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Traccia a livello

Visualizza il messaggio di traccia se la traccia della collaborazione è impostata al livello specificato o ad un livello superiore.

Nota: Questo blocco funzione si trova nella cartella Generale\Registro e Traccia\Traccia.

Input

Messaggio Il messaggio da visualizzare. Questo input può essere di tipo String, byte, short, int, long, float o double.

Livello Il livello di traccia minimo per il quale viene visualizzato il messaggio. Questo input può essere di tipo String, byte, short, int, long, float o double.

Capitolo 20. Blocchi funzione di stringa

I blocchi funzione nella cartella Generale\Stringa forniscono le funzionalità per gestire gli oggetti String. Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 78. Riepilogo dei blocchi funzione di stringa

Aggiungi testo

Accoda il valore di input di Nella stringa 2 al valore di input di Nella stringa 1.

Input

Nella stringa 1
Un oggetto String.

Nella stringa 2
L'oggetto String da accodare alla stringa specificata in Nella stringa 1.

Output

Un oggetto String che contiene la stringa di Nella stringa 1 a cui è accodata la stringa di Nella stringa 2. Ad esempio, se il valore di Nella stringa 1 è ″Hello world.″ e il valore di Nella stringa 2 è ″How are you?″, l'output di questo blocco funzione è la stringa ″Hello world. How are you?″.

Se

Restituisce il primo valore se la condizione è true, il secondo valore se la condizione è false.

Input

Condizione La condizione utilizzata per determinare l'output. Questo input può essere di tipo boolean o Boolean.

Valore 1 Un oggetto String.

Valore 2 Un oggetto String.

Output

Restituisce l'oggetto String associato con Valore 1 o Valore 2, in base alla condizione verificata.

E' vuoto

Restituisce il secondo valore se il primo valore è vuoto.

Input

Valore 1 Un oggetto String.

Valore 2 Un oggetto String.

Output

Restituisce l'oggetto String associato con Valore 2 se Valore 1 è vuoto. Altrimenti, non restituisce niente.

E' NULL

Restituisce il secondo valore se il primo valore è null.

Input

Valore 1 Un oggetto String.

Valore 2 Un oggetto String.

Output

Restituisce l'oggetto String associato con Valore 2 se Valore 1 è null. Altrimenti, non restituisce niente.

Compila a sinistra

Restituisce un oggetto String della lunghezza specificata; riempie la parte a sinistra con il valore indicato.

Input

Stringa Un oggetto String.

Compila stringa
Un oggetto String.

Lunghezza Un numero intero che specifica la lunghezza dell'oggetto String da restituire.

Output

Restituisce un oggetto String riempito.

Stringa sinistra

Restituisce la parte a sinistra di una stringa per il numero di posizioni specificato.

Input

Stringa L'oggetto String da recuperare.

Lunghezza Un numero intero che specifica il numero di posizioni nell'oggetto String da restituire.

Output

Restituisce un oggetto String che contiene la parte a sinistra dell'oggetto String originale.

Minuscolo

Modifica in minuscolo tutti i caratteri di un oggetto String.

Input

Da stringa L'oggetto String originale.

Output

Restituisce l'oggetto String con tutti i caratteri in minuscolo.

Oggetto a Stringa

Recupera una rappresentazione in formato stringa di un oggetto.

Input

Oggetto L'oggetto da recuperare.

Output

Restituisce l'oggetto specificato come oggetto String.

Ripeti

Restituisce un oggetto String che contiene un'espressione di carattere specificata che viene ripetuta un numero di volte specificato.

Input

Stringa ripetitiva
L'oggetto String ripetitivo da recuperare.

Ripeti conteggio
Un numero intero che specifica il numero di volte che deve essere ripetuta la stringa di caratteri specificata prima di recuperare la stringa.

Output

Restituisce la stringa di caratteri ripetuti.

Sostituisci

Sostituisce parte di una stringa di caratteri con una nuova stringa di caratteri.

Input

Stringa Un oggetto String.

Stringa obsoleta
Un oggetto String che contiene il sottoinsieme di stringa particolare da sostituire.

Stringa nuova Un oggetto String che contiene la nuova stringa di caratteri da utilizzare come sostituzione.

Output

Restituisce un oggetto String che contiene la stringa di caratteri aggiornata.

Compila a destra

Restituisce un oggetto String della lunghezza specificata; riempie la parte a destra con il valore indicato.

Input

Stringa Un oggetto String.

Compila stringa
L'oggetto String utilizzato per riempire la parte destra.

Lunghezza Un numero intero che specifica la lunghezza dell'oggetto String da restituire.

Output

Restituisce l'oggetto String specificato, riempito con il valore specificato.

Stringa destra

Restituisce la parte a destra di una stringa per il numero di posizioni specificato.

Input

Stringa L'oggetto String da restituire.

Lunghezza Un numero intero che specifica il numero di posizioni nella stringa di caratteri da restituire.

Output

Restituisce un oggetto String con la parte a destra della stringa specificata, per il numero di posizioni specificato.

Stringa secondaria in base alla posizione

Restituisce una parte dell'oggetto String in base ai parametri di inizio e fine.

Input

Stringa L'oggetto String.

Posizione di inizio
Un numero intero che specifica l'inizio della parte della stringa da restituire.

Posizione finale
Un numero intero che specifica la fine della parte della stringa da restituire.

Output

Restituisce un oggetto String che contiene la parte di stringa specificata.

Stringa secondaria in base al valore

Restituisce una parte di un oggetto String in base ai valori di inizio e fine specificati. Notare che la parte di stringa non include i valori di inizio e fine, ma include quanto compreso tra tali valori.

Input

Stringa L'oggetto String.

Valore di inizio
Un numero intero che specifica il valore di inizio della parte di stringa.

Valore di fine Un numero intero che specifica il valore di fine della parte di stringa.

Output

Restituisce un oggetto String che contiene la parte di stringa specificata.

Testo uguale

Confronta le stringhe di caratteri di due oggetti String per determinare se sono uguali.

Input

Nella stringa 1
Il primo oggetto String del confronto.

Nella stringa 2
Il secondo oggetto String del confronto.

Output

Restituisce `True`, se il contenuto dei due oggetti String è uguale; altrimenti restituisce `False`.

Testo uguale ignora maiuscolo/minuscolo

Confronta le stringhe di caratteri di due oggetti String in modo lessicale (ignorando la distinzione tra maiuscolo e minuscolo) per determinare se sono uguali.

Input

Nella stringa 1
Il primo oggetto String del confronto.

Nella stringa 2
Il secondo oggetto String del confronto.

Output

Restituisce `True`, se il contenuto dei due oggetti String è uguale; altrimenti restituisce `False`.

Lunghezza testo

Individua il numero totale di caratteri in un oggetto String.

Input

Stringa L'oggetto String.

Output

Restituisce la lunghezza dell'oggetto String come uno dei seguenti tipi di dati: byte, short, int, long, float o double.

Taglia a sinistra

Elimina il numero specificato di caratteri dalla parte sinistra dell'oggetto String.

Input

Stringa L'oggetto String da cui eliminare caratteri.

Taglia lunghezza
Un numero intero che specifica il numero di caratteri da eliminare.

Output

Restituisce un oggetto String privo dei caratteri eliminati.

Taglia a destra

Elimina il numero specificato di caratteri dalla parte destra dell'oggetto String.

Input

Stringa L'oggetto String da cui eliminare caratteri.

Taglia lunghezza
Un numero intero che specifica il numero di caratteri da eliminare.

Output

Restituisce un oggetto String privo dei caratteri eliminati.

Taglia testo

Elimina gli spazi prima e dopo i caratteri in un oggetto String.

Input

In stringa L'oggetto String da cui eliminare caratteri.

Output

Restituisce un oggetto String privo dei caratteri eliminati.

Maiuscolo

Modifica in maiuscolo tutti i caratteri di un oggetto String.

Input

Da stringa L'oggetto String originale i cui caratteri si desidera convertire in maiuscolo.

Output

Restituisce l'oggetto String con tutti i caratteri convertiti in maiuscolo.

Capitolo 21. Blocchi funzione di utilità

I blocchi funzione nella cartella Generale\Utilità e nelle relative cartelle secondarie forniscono le funzionalità per gestire oggetti Vector, eccezioni nelle attività e nelle attività secondarie e problemi di locale. Nelle seguenti sezioni sono descritti in dettaglio i singoli blocchi funzione.

Tabella 79. Riepilogo dei blocchi funzione di utilità

Cartella	Blocco funzione	Pagina
Generale\Utilità	Errore	358
	Tipo di errore	358
	Condizione	358
	Loop	361
	Sposta attributo in secondario	361
	Errore	362
	Tipo di errore	362
Generale\Programmi di utilità\Locale	Ottieni paese	359
	Ottieni lingua	360
	Nuova locale	361
	Nuova locale con lingua	362
Generale\Programmi di utilità\Locale\Costanti	Inglese	358
	Francese	359
	Tedesco	359
	Italiano	360
	Giapponese	360
	Coreano	361
	Cinese semplificato	363
	Cinese tradizionale	363
Generale\Utilità\Vettore	Aggiungi elemento	357
	Ottieni elemento	359
	Itera vettore	360
	Nuovo vettore	362
	Dimensione	363
	In vettore	363

Aggiungi elemento

Aggiunge l'elemento specificato alla fine del vettore, incrementando di uno la relativa dimensione.

Nota: Questo blocco funzione si trova nella cartella Generale\Utilità\Vettore.

Input

Vettore Un oggetto java.util.Vector.

Output

Restituisce l'elemento.

Errore

Cattura tutte le eccezioni generate nell'attività corrente o nelle relative attività secondarie. (E' necessario fare doppio clic sull'icona del blocco funzione nell'area di disegno per definire l'attività secondaria).

Input

Nome errore Una stringa (String) che specifica il nome dell'errore.

Messaggio di errore
Una stringa (String) che specifica il contenuto del messaggio di errore.

Tipo di errore

Cattura il tipo di eccezione specificato nell'attività corrente o nelle relative attività secondarie. (E' necessario fare doppio clic sull'icona del blocco funzione nell'area di disegno per definire l'attività secondaria).

Input

Tipo di errore Una stringa (String) che specifica il tipo di eccezione da catturare.

Messaggio di errore
Una stringa (String) che specifica il contenuto del messaggio di errore.

Condizione

Se la condizione specificata esiste, esegue l'attività secondaria definita in ″Azione vero″. Altrimenti, esegue l'attività secondaria definita in ″Azione falso″. (E' necessario fare doppio clic sull'icona del blocco funzione nell'area di disegno per definire l'attività secondaria).

Input

Condizione Un valore booleano che specifica la condizione da verificare.

Inglese

Una costante che rappresenta la locale Inglese.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Un oggetto java.util.Locale inglese.

Francese

Una costante che rappresenta la locale Francese.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Un oggetto java.util.Locale francese.

Tedesco

Una costante che rappresenta la locale Tedesco.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Un oggetto java.util.Locale tedesco.

Ottieni paese

Determina il codice paese/regione per la locale corrente.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Input

Locale Un oggetto java.util.Locale che rappresenta la locale corrente.

Output

Restituisce un oggetto String che contiene il codice paese/regione per la locale specificata. Generalmente è una stringa vuota oppure un codice di due lettere in maiuscolo ISO 3166.

Note

Questo blocco funzione si basa sul metodo java.util.Locale.getCountry().

Ottieni elemento

Ottiene l'elemento all'indice specificato nell'oggetto Vector.

Nota: Questo blocco funzione si trova nella cartella Generale\Utilità\Vettore.

Input

Vettore Un oggetto java.util.Vector.

Indice Un numero intero che specifica la posizione dell'indice.

Output

Restituisce l'elemento che si trova alla posizione indice specificata.

Ottieni lingua

Determina il codice lingua per la locale corrente.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Input

Locale Un oggetto java.util.Locale che rappresenta la locale corrente.

Output

Restituisce un oggetto String che contiene il codice lingua per la locale. Può essere una stringa vuota o un codice in minuscolo ISO 639.

Note

Questo blocco funzione si basa sul metodo java.util.Locale.getLanguage().

Italiano

Una costante che rappresenta la locale Italiano.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Un oggetto java.util.Locale italiano.

Itera vettore

Scorre un oggetto Vector.

Nota: Questo blocco funzione si trova nella cartella Generale\Utilità\Vettore.

Input

Vettore Un oggetto java.util.Vector.

Indice corrente
Un numero intero che specifica la posizione dell'indice.

Elemento corrente
Un oggetto che specifica l'elemento.

Giapponese

Una costante che rappresenta la locale Giapponese.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Un oggetto java.util.Locale giapponese.

Coreano

Una costante che rappresenta la locale Coreano.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Un oggetto java.util.Locale coreano.

Loop

Ripete l'attività secondaria fino a quando la condizione specificata è falsa. (E' necessario fare doppio clic sull'icona del blocco funzione nell'area di disegno per definire l'attività secondaria).

Input

Condizione Un valore booleano che specifica la condizione da verificare.

Sposta attributo in secondario

Sposta il valore di un attributo in un altro attributo.

Input

Principale di origine
L'oggetto business (un oggetto BusObj) che contiene l'attributo di oggetto business secondario da spostare.

Attributo BO secondario di origine
Una stringa (String) che identifica il nome dell'oggetto business secondario che contiene l'attributo il cui valore si desidera spostare.

Attributo da Una stringa (String) che identifica il nome dell'attributo da spostare.

Principale di destinazione
L'oggetto business (un oggetto BusObj) nel quale spostare il valore dell'attributo originale.

Attributo BO secondario di destinazione
Una stringa (String) che identifica il nome dell'oggetto business secondario che contiene l'attributo il cui valore si desidera sostituire.

Attributo a Una stringa (String) che specifica il nome dell'attributo il cui valore si desidera sostituire con il valore di **Attributo da**.

Nuova locale

Crea una nuova locale in base alla lingua e paese specificato.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Input

Lingua Una stringa (String) che specifica la lingua corrente della locale.

Paese Una stringa (String) che specifica il paese corrente della locale.

Output

Restituisce un oggetto java.util.Locale.

Note

Questo blocco funzione si basa sul metodo util.Locale().

Nuova locale con lingua

Crea una nuova locale da un codice lingua.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Input

Lingua Un oggetto String che specifica la lingua della locale.

Output

Restituisce un nuovo oggetto java.util.Locale.

Note

Questo blocco funzione si basa sul metodo util.Locale().

Nuovo vettore

Crea un nuovo oggetto Vector.

Nota: Questo blocco funzione si trova nella cartella Generale\Utilità\Vettore.

Output

Restituisce un nuovo oggetto java.util.Vector.

Errore

Genera una nuova eccezione Java con il messaggio specificato.

Input

Messaggio Un oggetto String che contiene il messaggio per l'eccezione Java.

Tipo di errore

Genera l'eccezione Java specificata con il messaggio specificato.

Input

Tipo di errore Un oggetto String che identifica il tipo di eccezione Java da generare.

Messaggio Un oggetto String che contiene il messaggio per l'eccezione Java.

Cinese semplificato

Una costante che rappresenta la locale Cinese semplificato.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Restituisce un oggetto java.util.Locale cinese semplificato.

Dimensione

Determina il numero di elementi in un oggetto Vector.

Nota: Questo blocco funzione si trova nella cartella Generale\Utilità\Vettore.

Input

Vettore Un oggetto java.util.Vector.

Output

Restituisce un numero intero che specifica il numero di elementi nell'oggetto Vector.

In vettore

Ottiene la rappresentazione di vettore che contiene tutti gli elementi nell'oggetto Vector corrente.

Nota: Questo blocco funzione si trova nella cartella Generale\Utilità\Vettore.

Input

Vettore Un oggetto java.util.Vector.

Output

Restituisce tutti gli elementi nell'oggetto Vector come vettore di tipo Object[].

Cinese tradizionale

Una costante che rappresenta la locale Cinese tradizionale.

Nota: Questo blocco funzione si trova nella cartella Generale\Programmi di utilità\Locale\Costanti.

Output

Un oggetto java.util.Locale cinese tradizionale.

Parte 4. Riferimenti API di collaborazione

Capitolo 22. Classe BaseCollaboration

I metodi documentati in questo capitolo agiscono su oggetti di collaborazione. Questi sono definiti nella classe `BaseCollaboration` basta su InterChange Server Express. La classe `BaseCollaboration` è la classe di base per tutte le collaborazioni. Tutte le collaborazioni create sono sottoclassi di `BaseCollaboration`, tutte ereditano questi metodi.

La Tabella 80 riepiloga i metodi della classe `BaseCollaboration`.

Tabella 80. Riepilogo dei metodi BaseCollaboration

Metodo	Descrizione	Pagina
`dropFailedEvent()`	Rimuove un evento non riuscito da una collaborazione	368
`dynamicSend()`	Invia una richiesta di servizio ad un consumer della richiesta che viene determinato dinamicamente al runtime.	368
`existsConfigProperty()`	Verifica l'esistenza di una proprietà di configurazione collaborazione.	370
`getConfigProperty()`	Recupera il valore di una proprietà di configurazione collaborazione.	370
`getConfigPropertyArray()`	Recupera il valore di una proprietà di configurazione collaborazione a più elementi.	370
`getDBConnection()`	Stabilisce una connessione con un database e restituisce un oggetto `CwDBConnection`.	372
`getLocale()`	Recupera la locale della collaborazione.	375
`getMessage()`	Recupera un messaggio, identificato dal relativo numero di messaggio, dal file di messaggi di collaborazione.	374
`getName()`	Recupera il nome di questo oggetto di collaborazione.	375
`implicitDBTransactionBracketing()`	Recupera il modello di programmazione transazione utilizzato dall'oggetto di collaborazione per le connessioni ottenute.	376
`isCallerInRole()`	Determina il ruolo specifico di un client Web.	376
`isTraceEnabled()`	Confronta il livello di traccia specificato con il livello di traccia corrente della collaborazione.	377
`logError(), logInfo(), logWarning()`	Invia un messaggio di errore, informativo o di avvertenza al file di log.	377
queryFailedEvents()	Genera un elenco di eventi non riusciti che corrispondono ai criteri della query.	380
`raiseException()`	Innalza un'eccezione di collaborazione.	382
`resubmitFailedEvent()`	Invia nuovamente un evento dalla coda eventi non riusciti salvata per essere elaborato dalla collaborazione.	385
`saveFailedEvent()`	Salva un evento non riuscito nella coda di eventi non riusciti.	386
`sendEmail()`	Invia un messaggio e-mail in modo asincrono.	387
`trace()`	Genera un messaggio di traccia.	388

dropFailedEvent()

Questo metodo rimuove un evento non riuscito da una coda eventi di collaborazione. Utilizzarlo quando il tentativo di inviare nuovamente un evento non riuscito ha esito negativo.

Sintassi

```
void dropFailedEvent (String ownerName, String connectorname, int index)
```

Parametri

connectorName Il nome del connettore a cui appartiene l'evento.

ownerName La collaborazione a cui appartiene l'evento.

index L'indice dell'evento non riuscito. Richiamare questo indice utilizzando il metodo queryFailedEvents.

Esempi

Il seguente esempio ottiene un vettore di eventi non riusciti e poi li cancella dalla coda.

```
// Set up to query the failed events of a specific type for the collaboration.
EventQueryDef queryOption = new EventQueryDef();
// set query condition of busobj name
queryOption.nameBusObj = "TestBO";
// set query condition of connector name
queryOption.nameConnector = "SourceConnector";
// define query condition array
EventQueryDef[] queryOptions = new EventQueryDef[1];
queryOptions[0] = queryOption;
// query failed events with status of
// STATUS_DELIVERY_POSSIBLE_DUPLICATE, no time limitation
FailedEventInfo[]resultArray = queryFailedEvents(queryOptions,
           "", "", FailedEventInfo.STATUS_DELIVERY_POSSIBLE_DUPLICATE);

// get the query result
for (int i = 0; i < resultArray.length; i++) {
 String busObjName = resultArray[i].nameBusObj;
 String nameConnector = resultArray[i].nameConnector;
 int wipIndex = resultArray[i].wipIndex;
// Drop the failed events with status STATUS_DELIVERY_POSSIBLE_DUPLICATE
// from the queue.
 dropFailedEvent (busObjName, nameConnector, wipIndex);
 }
```

Consultare anche

Per informazioni sull'interrogazione degli eventi non riusciti, consultare "queryFailedEvents()" a pagina 380.

dynamicSend()

Invia una richiesta di servizio ad un consumer della richiesta che viene determinato dinamicamente al runtime.

Sintassi

```
protected void dynamicSend(String consumerType, String consumerName,
   string consumerPort, BusObj busObj, String verb)
   throws CollaborationException
```

Parametri

consumerType
ConsumerType è il tipo di destinazione per ricevere l'oggetto business. Il valore può essere **BaseCollaboration.CONSUMER_CONNECTOR** o **BaseCollaboration.CONSUMER_COLLABORATION**.

consumerName
ConsumerName è il nome del connettore o della collaborazione.

consumerPort
Per una collaborazione, *consumerPort* è il nome della porta di collaborazione. Se un connettore utilizza **dynamicSend()**, impostare questo parametro a null.

busObj *BusObj* è il nome dell'oggetto business utilizzato in una richiesta di servizio.

verb *Verb* è il nome dell'istruzione associata con la richiesta di servizio.

Valori di ritorno

Nessuno.

Esempi

I seguenti esempi illustrano le modalità di estrazione delle informazioni da un oggetto business e l'invio di una chiamata di servizio dinamica con formato corretto.

```
BusObj processingBusObj = new BusObj(triggeringBusObj.getType());
processingBusObj.copy(triggeringBusObj);

/* Extract the dynamic service call parameters from the input BO */
String verb = processingBusObj.getVerb();
String consumerName = processingBusObj.getString("ConsumerName");

/* Consumer type should be a collaboration or a connector only */
String consumerType = processingBusObj.getString("ConsumerType");
String consumerPort = null;

if ((consumerType != null ) &&
(!consumerType.equalsIgnoreCase(BaseCollaboration.CONSUMER_CONNECTOR) )&&
(!consumerType.equalsIgnoreCase(BaseCollaboration.CONSUMER_COLLABORATION)))
{
    // raise an exception here
    raiseException(AnyException, "Invalid consumer type encountered");
}
if
 (consumerType.equalsIgnoreCase(BaseCollaboration.CONSUMER_COLLABORATION))
 consumerPort = processingBusObj.getString("ConsumerPort");

trace(1, "Making Dynamic Service call with parameters: " +
     "ConsumerName = " + consumerName + " ConsumerType = " +
      consumerType + " ConsumerPort = " + consumerPort +" Verb = "
      + verb);

//make a dynamic call
dynamicSend(consumerType, consumerName, consumerPort, processingBusObj, verb);
```

Note

Quando si aggiungono chiamate di servizio dinamiche alle collaborazioni, valutare le seguenti considerazioni:

- Non esistono requisiti per associare una porta al momento della progettazione. Le informazioni necessarie per elaborare l'oggetto business sono parte della chiamata del servizio.
- La collaborazione può interagire con molti connettori fornendo una maggiore flessibilità in un ambiente di elaborazione dinamica.
- Le collaborazioni gestiscono le richieste in modo sincrono.

existsConfigProperty()

Verifica l'esistenza di una proprietà di configurazione collaborazione.

Sintassi

```
boolean existsConfigProperty(String propertyName)
```

Parametri

propertyName Il nome della proprietà definita nella maschera di collaborazione.

Valori di ritorno

Restituisce `true` se esiste una proprietà; restituisce `false` se non esiste.

Esempi

Il seguente esempio verifica l'esistenza della proprietà `VALIDATE_CUSTOMER`.

```
boolean validatePropExists =
   existsConfigProperty("VALIDATE_CUSTOMER");
```

getConfigProperty()

Recupera il valore di una proprietà di configurazione collaborazione.

Sintassi

```
String getConfigProperty(String propertyName)
```

Parametri

propertyName Il nome della proprietà definita nella maschera di collaborazione.

Valori di ritorno

Restituisce il valore della proprietà di configurazione. Se la proprietà non esiste, restituisce una stringa vuota ("").

Esempi

Il seguente esempio ottiene il valore della proprietà `VALIDATE_CUSTOMER` e lo assegna alla variabile *validateProp*.

```
String validateProp = getConfigProperty("VALIDATE_CUSTOMER");
```

getConfigPropertyArray()

Recupera il valore di una proprietà di configurazione collaborazione a più elementi.

Sintassi

```
String[] getConfigPropertyArray(String propertyName)
```

Parametri

propertyName Il nome della proprietà definita nella maschera di collaborazione.

Valori di ritorno

Un vettore di valori di proprietà.

Note

Questo metodo recupera il valore di una proprietà di configurazione a più elementi. Una proprietà di configurazione a più elementi è costituita da numerosi valori, separati da punti e virgola.

Se la proprietà non esiste, il vettore è vuoto. Se la proprietà presenta un singolo elemento, il vettore contiene un solo elemento.

È possibile utilizzare una proprietà di configurazione a più elementi per ricevere l'input da un utente. La collaborazione può quindi utilizzare questa proprietà a più elementi per generare una richiesta Retrieve, in assenza di valori dell'attributo chiave di un oggetto business. L'utente può specificare gli attributi da utilizzare per recuperare l'oggetto business specificando i valori per gli elementi della proprietà di configurazione.

Esempi

Il seguente esempio recupera un elenco di proprietà associate con nome ATTR_LIST.

```
String[] list = getConfigPropertyArray("ATTR_LIST");
```

---

getCurrentLoopIndex()

Recupera il valore della variabile indice quando un iteratore viene configurato come loop.

Sintassi

```
int getCurrentLoopIndex()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il valore della variabile indice del loop. Restituisce zero (0) se è al di fuori del loop.

Esempi

Il seguente esempio ottiene l'indice del loop corrente:

```
int currentIndex = getCurrentLoopIndex();
```

getDBConnection()

Stabilisce una connessione ad un database e restituisce un oggetto CwDBConnection.

Sintassi

```
CwDBConnection getDBConnection(String connectionPoolName)
CwDBConnection getDBConnection(String connectionPoolName,
   boolean implicitTransaction)
```

Parametri

connectionPoolName
: Il nome del pool di connessioni. Il metodo si connette al database la cui connessione si trova nel pool di connessioni specificato.

implicitTransaction
: Un valore booleano per indicare il modello di programmazione transazione da utilizzare per il database associato con la connessione. I valori validi sono:

  `true` Il database utilizza bracketing transazioni implicite

  `false` Il database utilizza bracketing transazioni esplicite

Valori di ritorno

Restituisce un oggetto CwDBConnection.

Eccezioni

CwDBConnectionFactoryException – Indica se si verifica un errore nel tentativo di stabilire la connessione al database.

Note

Il metodo getDBConnection() ottiene una connessione dal pool di connessioni specificato da *connectionPoolName*. Questa connessione fornisce un meccanismo per eseguire query ed aggiornamenti sul database associato con la connessione. Tutte le connessioni di un particolare pool di connessioni sono associate allo stesso database. Il metodo restituisce un oggetto CwDBConnection mediante il quale è possibile eseguire query e gestire transazioni. Per ulteriori informazioni, consultare i metodi nella classe CwDBConnection.

Per impostazione predefinita, tutte le connessioni utilizzano bracketing transazioni implicite in base al proprio modello di programmazione transazione. Per specificare un modello di programmazione transazione *per una particolare connessione*, fornire un valore booleano per indicare il modello di programmazione transazione come argomento facoltativo *implicitTransaction* al metodo getDBConnection(). La seguente chiamata getDBConnection() specifica bracketing transazioni esplicite per la connessione ottenuta dal pool di connessioni ConnPool:

```
conn = getDBConnection("ConnPool",false);
```

La connessione viene rilasciata quando l'oggetto di collaborazione termina l'esecuzione. La connessione può essere chiusa in modo esplicito con il metodo release(). E' possibile determinare se una connessione è stata rilasciata mediante il metodo isActive(). Per ulteriori informazioni, consultare "Rilascio di una connessione" a pagina 244.

Esempi

Il seguente esempio stabilisce una connessione al database associato con le connessioni nel pool di connessioni CustConnPool. Quindi utilizza una transazione implicita per inserire e aggiornare righe in una tabella del database.

```
CwDBConnection connection = getDBConnection("CustConnPool");

// Insert a row
connection.executeSQL("insert...");

// Update rows...
connection.executeSQL("update...");
```

Dal momento che la precedente chiamata a getDBConnection() *non* include il secondo argomento facoltativo, questa connessione utilizza il bracketing transazioni implicite così come il modello di programmazione transazione (a meno che il modello di programmazione transazione sia sovrascritto nella finestra di dialogo Proprietà collaborazione di System Manager). Pertanto, non sono specificati limiti di transazione espliciti con beginTransaction(), commit() e rollback(). In effetti, il tentativo di richiamare uno di questi metodi di transazione con il bracketing transazioni implicite genera una eccezione CwDBTransactionException.

Nota: E' possibile verificare il modello di programmazione transazione corrente mediante il metodo implicitDBTransactionBracketing().

Il seguente esempio stabilisce anche una connessione al database associato con le connessioni nel pool di connessioni CustConnPool. Tuttavia, specifica l'utilizzo di bracketing transazioni esplicite per la connessione. Quindi, utilizza una transazione esplicita per contenere gli inserimenti e aggiornamenti sulle righe delle tabelle 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 precedente chiamata a getDBConnection() include l'argomento facoltativo *implicitTransaction* per impostare il modello di programmazione transazione a bracketing transazioni esplicite. Questo esempio utilizza le chiamate di transazione esplicite per indicare i limiti della transazione. Se questi metodi di transazione sono omessi, InterChange Server Express gestisce la transazione come se fosse una transazione implicita.

Consultare anche

Capitolo 25, "Classe CwDBConnection", isActive(), release()

getLocale()

Recupera la locale di collaborazione per l'oggetto di collaborazione corrente.

Sintassi

```
java.util.Locale getLocale()
```

Parametri

Nessuno.

Valori di ritorno

Un oggetto Locale Java che contiene i codici di lingua e paese della locale di collaborazione. Questo oggetto Locale deve essere un'istanza della classe `java.util.Locale`.

Note

Il metodo `getLocale()` restituisce la locale del flusso corrente. Questa locale del flusso è la locale associata all'oggetto business di attivazione dell'oggetto di collaborazione.

Esempi

Il seguente esempio ottiene la locale della collaborazione, recupera i codici di paese e lingua dall'oggetto Locale e poi riporta i valori in un messaggio di traccia:

```
Locale collaborationLocale = getLocale();
String collaborationCountry = collaborationLocale.getCountry();
String collaborationLanguage = collaborationLocale.getLanguage();

trace(3, "THE COUNTRY CODE FOR THE COLLABORATION IS " +
    collaborationCountry +
   ", AND THE LANGUAGE CODE FOR THE COLLABORATION IS " +
    collaborationLanguage +
   ".");
```

getMessage()

Recupera un messaggio dal file messaggi di collaborazione.

Sintassi

```
public String getMessage(int messageNum)
public String getMessage(int messageNum, Object[] paramArray)
```

Parametri

messageNum

Il numero di un messaggio nel file di messaggi di collaborazione, che è indicizzato per numero messaggio. Per informazioni sull'impostazione di un file di testo dei messaggi, consultare la sezione Capitolo 10, "Creazione del file di messaggi", a pagina 247.

paramArray

Un vettore di valori di parametri del messaggio. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio. All'interno del messaggio (nel file di messaggi di collaborazione), i parametri del messaggio sono indicati da numeri interi racchiusi tra parentesi graffe; ad esempio, {1}.

Valori di ritorno

Un oggetto `String` che contiene il testo del messaggio identificato da *messageNum*.

Note

Il metodo `getMessage()` fornisce due moduli:

- Il primo modulo riceve un numero messaggio e recupera il messaggio associato al file di messaggi di collaborazione come oggetto `String`.
- Il secondo modulo riceve un numero messaggio e un vettore di valori di parametri del messaggio. Recupera il messaggio associato dal file di messaggi di collaborazione, sostituisce i relativi parametri del messaggio con gli oggetti nel vettore dei parametri e restituisce il messaggio risultante come oggetto `String`.

Per informazioni sui file di messaggi e sui parametri dei messaggi, consultare Capitolo 10, “Creazione del file di messaggi”, a pagina 247.

Esempi

Si supponga che il file di messaggi di collaborazione definisca i due messaggi seguenti con i numeri di messaggio 8 e 9:

```
8
Error occurred during JDBC URL conversion. Reason:{1}
[EXPL]
An error, indicated by the reason, occurred during the
conversion of a JDBC URL string.
9
Invalid login encountered in command-line arguments. A valid
login must contain a login name and a password.
[EXPL]
A password has been specified but a user name has not. If no
login name is specified, the default login name "crossworlds" is assumed.
```

La seguente chiamata a `getMessage()` ottiene il testo associato al messaggio 9:

```
String invalidLogin = getMessage(9);
```

La seguente chiamata a `getMessage()` ottiene il testo associato al messaggio 8 e include una valore del parametro Reason del messaggio:

```
String reason = "Invalid database table.";
Object[] paramList = new Object[1];
paramList[0] = reason;
badConversion = getMessage(8, paramList);
```

Il messaggio ottenuto dalla precedente chiamata a `getMessage()` sarà:

```
Error occurred during JDBC URL conversion. Reason:Invalid database table.
```

---

getName()

Recupera il nome di questo oggetto di collaborazione.

Sintassi

```
String getName()
```

Esempi

Il seguente esempio ottiene il nome dell’oggetto di collaborazione corrente e registra un messaggio informativo:

```
String collabName = getname();
logInfo(collabName + " is starting");
```

implicitDBTransactionBracketing()

Recupera il modello di programmazione transazione utilizzato dall'oggetto di collaborazione per le connessioni ottenute.

Sintassi

```
boolean implicitDBTransactionBracketing()
```

Parametri

Nessuno.

Valori di ritorno

Un valore `boolean` per indicare il modello di programmazione transazione da utilizzare per tutte le connessioni al database

Note

Il metodo `implicitDBTransactionBracketing()` restituisce un valore booleano che indica quale modello di programmazione transazione l'oggetto di collaborazione assume sia utilizzato da *tutte* le connessioni ottenute:

- Un valore `true` indica che tutte le connessioni utilizzano il bracketing transazioni *implicite*.
- Un valore `false` indica che tutte le connessioni utilizzano il bracketing transazioni *esplicite*.

Questo metodo è utile, prima di ottenere una connessione, per verificare se il modello di programmazione transazione è appropriato per la connessione.

Nota: Il modello di programmazione transazione può essere sovrascritto per una specifica connessione mediante il metodo `getDBConnection()`.

Esempi

Il seguente esempio assicura che l'oggetto di collaborazione utilizzi bracketing transazioni esplicite per il database associato alla connessione `conn`:

```
if (implicitDBTransactionBracketing())
   CwDBConnection conn = getDBConnection("ConnPool", false);
```

Consultare anche

"Gestione della transazione" a pagina 240

`getDBConnection()`

isCallerInRole()

Restituisce il ruolo specifico di un client Web che richiede un servizio. Questo consente alla collaborazione di personalizzare il processo business basato sull'identità del client.

Sintassi

```
boolean isCallerInRole(String Role)
```

Parametri

Role Il ruolo specifico che si sta verificando.

Restituzioni

Restituisce `true` se il client appartiene al *Role* specificato; altrimenti restituisce `false`.

Note

E' necessario attivare il controllo accessi basato sul ruolo (RBAC) nella scheda Sicurezza di System Manager affinché la collaborazione possa accedere ai ruoli. Per ulteriori dettagli, consultare il manuale *System Administration Guide*.

isTraceEnabled()

Confronta il livello di traccia specificato con il livello di traccia corrente della collaborazione.

Sintassi

```
public Boolean isTraceEnabled(int traceLevel)
```

Parametri

traceLevel Il livello di traccia da confrontare con il livello di traccia corrente.

Valori di ritorno

Restituisce `true` se il livello di traccia di sistema corrente viene impostato al livello di traccia specificato; restituisce `false` se i due livelli di traccia non sono uguali.

Note

Il metodo `isTraceEnabled()` è utile per determinare se un messaggio di traccia deve essere o non essere registrato. Dal momento che la traccia può peggiorare le prestazioni, questo metodo risulta utile nella fase di sviluppo di un progetto.

Esempi

```
if ( isTraceEnabled(3) )
   {
   trace("Print this level-3 trace message");
   }
```

logError(), logInfo(), logWarning()

Scrive un messaggio di errore, informativo o di avvertenza nella destinazione del log.

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 da registrare.

messageNum Il numero di un messaggio nel file di messaggi di collaborazione, che è indicizzato per numero messaggio. Per informazioni sull'impostazione di un file di testo dei messaggi, consultare la sezione Capitolo 10, "Creazione del file di messaggi", a pagina 247.

param Il valore di un singolo parametro del messaggio. Possono essere presenti fino a cinque parametri di messaggio, separati da virgole. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio.

paramArray Un vettore di valori di parametri del messaggio. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio.

Note

I metodi `logError()`, `logWarning()` e `logInfo()` inviano un messaggio alla destinazione del log della collaborazione. Per impostazione predefinita, la destinazione del log è il file `InterchangeSystem.log`. La destinazione del log può essere modificata immettendo un valore per il parametro `LOG_FILE` nel file di configurazione InterChange Server Express, `InterchangeSystem.cfg`. Il valore del parametro può essere un nome file oppure `STDOUT`, che determina la scrittura del log nella finestra comandi di InterChange Server Express.

Possono essere impostati altri tre parametri di configurazione di sistema relativi alla registrazione log. Tutti i parametri si trovano nel file di configurazione di InterChange Server Express, `InterchangeSystem.cfg`.

- Impostare la dimensione massima del file di log con il parametro `MAX_LOG_FILE_SIZE`. Poiché la dimensione predefinita del file è illimitata, si dovrebbe sempre impostare un valore massimo.
- Impostare da uno a cinque file di log di archivio per il parametro `NUMBER_OF_ARCHIVE_LOGS`. Il valore predefinito è cinque, se il parametro non è impostato.
- Impostare il parametro `MIRROR_LOG_TO_STDOUT` se si desidera che i messaggi di errore siano visualizzati su STDOUT nello stesso momento in cui vengono scritti nel file di log.

Per una guida all'utilizzo del metodo che registra i messaggi di errore, avvertenza o informativi, consultare "Registrazione dei messaggi" a pagina 207. Il testo del

messaggio che viene visualizzato nel file di log dell'utente viene prefissato dall'indicazione Informazione, Avvertenza o Errore, in base al metodo utilizzato per registrare il messaggio.

Ognuno dei metodi di registrazione presenta diversi moduli:

- Il primo modulo include tutto il testo necessario a generare un messaggio. Invia questo messaggio alla destinazione del log.
- Il secondo modulo ottiene un messaggio che *non gestisce* parametri dal file di messaggi di collaborazione ed invia tale messaggio alla destinazione del log.
- Il terzo ottiene un messaggio che *gestisce* parametri dal file di messaggi di collaborazione. Inoltre fornisce un elenco di valori dei parametri del messaggio.
- Anche il quarto modulo ottiene un messaggio che gestisce parametri dal file di messaggi di collaborazione. Tuttavia, fornisce i valori dei parametri del messaggio in un vettore di valori di parametro.

Tutti i moduli del metodo che ricevono un parametro *messageNum* richiedono l'utilizzo di un file di messaggi indicizzato per numero messaggio. Per informazioni sull'impostazione di un file di testo dei messaggi, consultare la sezione Capitolo 10, "Creazione del file di messaggi", a pagina 247.

Oltre all'invio di un messaggio alla destinazione del log, il metodo `logError()` invia anche il messaggio di errore ad un destinatario di posta elettronica se:

- E' stato specificato un indirizzo e-mail nel campo `Indirizzo e-mail per notifica` nella finestra di dialogo Proprietà oggetto di collaborazione.
- La collaborazione e-mail e il connettore e-mail sono in esecuzione. (La collaborazione e-mail viene istanziata e configurata automaticamente quando si avvia InterChange Server Express e non richiede immissioni da parte dell'utente).

Nota:

1. CollaborationFoundation fornisce la proprietà di configurazione `SEND_EMAIL` per consentire di specificare per numero messaggio quali messaggi genereranno una notifica e-mail. Per ulteriori informazioni, consultare "Maschera CollaborationFoundation" a pagina 30.
2. Il metodo `logError()` invia automaticamente un messaggio di errore ad un destinatario di posta elettronica (assumendo che la collaborazione e-mail e l'adattatore e-mail siano in esecuzione). Il metodo `sendEmail()` consente di inviare esplicitamente un messaggio e-mail.

Esempi

Il seguente esempio registra un messaggio di errore, utilizzando `getString()` per ottenere un valore di attributo nel messaggio.

```
logError("Incorrect customer: CustomerID: "
   + fromCustomerBusObj.getString("CustomerID"));
```

Il seguente esempio registra un messaggio di errore il cui testo è contenuto nel file di messaggi di collaborazione. Il messaggio, che è il numero 10 nel file di messaggi, riceve due parametri: il cognome del cliente (attributo `LName`) e il nome del cliente (attributo `FName`).

```
logError(10, customer.get("LName"), customer.get("FName");
```

Il seguente esempio registra un messaggio di errore, utilizzando un vettore di parametri. A scopo dimostrativo, l'esempio utilizza un vettore con due soli

parametri. L'esempio dichiara il vettore `args`, con due elementi, l'ID cliente e il nome cliente. Il metodo `logError()` registra quindi un errore, utilizzando il numero messaggio 12 e i valori del vettore `args`.

```
Object[] args =
   {
      fromCustomerBusObj.getString("CustomerID"),
      fromCustomerBusObj.getString("CustomerName");
   }
logError(12, args);
```

queryFailedEvents()

Questo metodo interroga il database degli eventi non riusciti per individuare gli eventi che rispettano i criteri di selezione.

Sintassi

```
queryFailedEvents (EventQueryDef[] queryOptions, String startDateTime,
      String endDateTime, int eventStatus, int maxrows)
```

Parametri

queryOptions Un vettore di criteri di selezione con il seguente formato:

- *nameConnector* Nome del connettore di interesse.
- *nameCollaboration* Nome della collaborazione di interesse.
- *nameBusObj* Nome dell'oggetto business di interesse.
- *verb* Nome dell'istruzione di interesse. Se null, sono selezionate tutte le istruzioni.

startDateTime La prima data e ora in cui si è verificato l'evento. Il formato è MESE GG AAAA HH24:MI:SS.

endDateTime L'ultima data e ora in cui si è verificato l'evento. Il formato è MESE GG AAAA HH24:MI:SS.

Nota: HH24 indica il formato ora su 24 ore.

eventStatus Un numero intero che indica lo stato dell'evento, che può essere uno dei seguenti:

- STATUS_ANY_UNRESOLVED
- STATUS_FAILED
- STATUS_DEFERRED_RECOVERY
- STATUS_SERVICECALL_IN_TRANSIT
- STATUS_DELIVER_POSSIBLE_DUPLICATE
- STATUS_SERVICECALL_WAITING
- STATUS_USER_GENERATED

maxrows Maxrows limita il numero di elementi restituiti dal metodo. Se questo parametro è omesso o impostato a zero, non ci sono limiti al numero di elementi restituiti.

I risultati restituiti dipendono da *eventStatus* e dalla specifica del periodo. La Tabella 81 illustra i risultati dell'impostazione di diversi parametri di periodo.

Tabella 81. Risultati di queryFailedEvents in base ai parametri di periodo

startDateTime	endDateTime	Risultati
Non null	Non null	Tutti gli eventi che hanno riscontrato l'*eventStatus* tra *startDateTime* e *endDateTime*
Null	Non null	Tutti gli eventi che hanno riscontrato l'*eventStatus* prima di *endDateTime*
Non null	Null	Tutti gli eventi che hanno riscontrato l'*eventStatus* dopo *startDateTime*
Non null	Non null	Tutti gli eventi che hanno riscontrato l'*eventStatus* indipendentemente dal periodo di tempo.

Restituzioni

Un vettore di eventi non riusciti, **FailedEventInfo**, in cui ogni elemento presenta la seguente struttura:

nameOwner *NameOwner* è il nome della collaborazione a cui appartiene questo evento.

nameConnector *NameConnector* è il nome del connettore a cui appartiene questo evento.

nameBusObj *NameBusObj* è il nome dell'oggetto business utilizzato in questo evento.

nameVerb *NameVerb* è l'istruzione in esecuzione durante questo evento.

strTime *StrTime* è l'orario dell'evento.

strMessage *strMessage* è un messaggio associato all'evento non riuscito.

wipIndex *wipIndex* è un numero intero che indica la posizione dell'evento.

EventKeyAttrDef[]
Un vettore che è costituito da *nkeys* elementi composti da coppie di attributi di eventi. Il vettore presenta la seguente struttura:

keyName
KeyName è il nome attributo.

keyValue
KeyValue è il valore dell'attributo.

nkeys *NKeys* è il numero di attributi chiave nel vettore **EventKeyAttrDef**.

eventStatus *EventStatus* è un numero intero che indica lo stato dell'evento, che può essere uno dei seguenti:

STATUS_ANY_UNRESOLVED
STATUS_FAILED
STATUS_DEFERRED_RECOVERY
STATUS_SERVICECALL_IN_TRANSIT
STATUS_DELIVER_POSSIBLE_DUPLICATE
STATUS_SERVICECALL_WAITING
STATUS_USER_GENERATED

expirationTime *ExpirationTime* è l'orario in cui scade l'evento per un LLBP (Long-Lived Business Process).

scenarioName *ScenarioName* è il nome dello scenario al momento dell'evento.

scenarioState *ScenarioState* è lo stato di completamento dello scenario in esecuzione.

Esempi

Il seguente esempio interroga tutti gli eventi che corrispondono allo stato STATUS_USER_GENERATED.

```
// Set up to query the failed events of a specific type for the collaboration.
EventQueryDef queryOption = new EventQueryDef();
// set query condition of busobj name
queryOption.nameBusObj = "TestBO";
// set query condition of connector name
queryOption.nameConnector = "SourceConnector";
// define query condition array
EventQueryDef[] queryOptions = new EventQueryDef[1];
queryOptions[0] = queryOption;
// query failed events with status of user_generated, no time
// limitation
FailedEventInfo[] resultArray = queryFailedEvents(queryOptions,
            "", "", FailedEventInfo.STATUS_USER_GENERATED);

// get the query result
for (int i = 0; i < resultArray.length; i++) {
 String busObjName = resultArray[i].nameBusObj;
 logInfo(busObjName + " failed");
 }
```

Consultare anche

- “dropFailedEvent()” a pagina 368
- “resubmitFailedEvent()” a pagina 385
- “saveFailedEvent()” a pagina 386

---

raiseException()

Prepara un’eccezione di collaborazione per l’innalzamento al successivo livello superiore di esecuzione.

Sintassi

```
void raiseException(String exceptionType, String message)
void raiseException(String exceptionType, int messageNum,
      String parameter[,...])
void raiseException (String exceptionType, int messageNum,
      Object[] paramArray)
void raiseException(CollaborationException exceptionObject)
```

Parametri

exceptionType Il tipo eccezione per l’eccezione da innalzare. Specificare questo tipo di eccezione come una delle seguenti variabili statiche di tipo eccezione, che identifica la causa dell’eccezione di collaborazione:

`AnyException`	Qualsiasi tipo di eccezione.
`AttributeException`	Problema di accesso all’attributo. Ad esempio, la collaborazione ha chiamato `getDouble()` per un attributo String o `getString()` per un attributo non esistente.

`JavaException`	Problema con il codice Java nella logica di collaborazione.
`ObjectException`	L'oggetto business passato ad un metodo non era valido o è stato effettuato l'accesso ad un oggetto null.
`OperationException`	La chiamata di servizio è stata impostata in modo non corretto e non ha potuto essere inviata.
`ServiceCallException`	Chiamata di servizio non riuscita. Ad esempio, un connettore o un'applicazione non sono disponibili.
`SystemException`	Qualsiasi errore interno nel sistema di InterChange Server Express.
`TransactionException`	Errore relativo al comportamento di transazione per una collaborazione transazionale. Ad esempio, il rollback non è riuscito o la collaborazione non ha potuto applicare alcuna compensazione.

message	Una stringa di testo che contiene il messaggio di eccezione.
messageNum	Il numero di un messaggio nel file di messaggi di collaborazione, che è indicizzato per numero messaggio. Per informazioni sull'impostazione di un file di testo dei messaggi, consultare la sezione Capitolo 10, "Creazione del file di messaggi", a pagina 247.
parameter	Il valore di un singolo parametro del messaggio. Possono essere presenti fino a cinque parametri di messaggio, separati da virgole. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio.
paramArray	Un vettore di valori di parametri del messaggio. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio.
exceptionObject	Il nome di una variabile oggetto di eccezione `CollaborationException`.

Le seguenti spiegazioni ed esempi di codice devono essere fornite nell'API di collaborazione:

Note

Il metodo `raiseException()` prepara un'eccezione di collaborazione da innalzare al successivo livello superiore di esecuzione. Quando l'ambiente runtime di collaborazione esegue la chiamata `raiseException()`, modifica l'esecuzione della collaborazione allo stato Eccezione, quindi procede con la logica del diagramma di attività. La modalità di risposta del diagramma di attività all'eccezione innalzata dipende dal nodo di termine del percorso di esecuzione così come riportato di seguito:

- Se il percorso di esecuzione termina in Fine completamento, il controllo passa al successivo livello superiore di esecuzione.

  Se il successivo nodo di questo diagramma principale è un nodo decisionale, l'ambiente runtime di collaborazione controlla i rami di esecuzione del nodo

decisionale che gestiscono l'eccezione innalzata. Questo diagramma principale può accedere all'eccezione mediante la variabile di sistema `currentException`.

- Se il percorso di esecuzione termina in Errore su completamento, l'ambiente runtime di collaborazione termina la collaborazione, genera una voce nel log di collaborazione e crea un flusso non risolto.

  L'ambiente runtime di collaborazione associa al flusso non risolto un testo di eccezione contenuto nell'eccezione. Se l'eccezione non contiene un testo di eccezione, l'ambiente runtime di collaborazione utilizza il messaggio predefinito:

  ```
  Scenario failed.
  ```

E' meglio aumentare il livello di un'eccezione in modo esplicito quando si verifica l'eccezione, invece di terminare semplicemente in errore. Quando il codice innalza esplicitamente un'eccezione all'ambiente runtime di collaborazione, l'amministratore può utilizzare il Flow Manager per visualizzare il testo dell'eccezione come parte del flusso non risolto. Per ulteriori informazioni, consultare "Innalzamento dell'eccezione" a pagina 187.

Il metodo `raiseException()` fornisce numerosi moduli:

- Il *primo* modulo crea un nuovo oggetto eccezione con il tipo eccezione specificato e una stringa di messaggio. Utilizzare questo modulo per passare un messaggio di eccezione memorizzato come stringa. E' anche possibile inviare questo messaggio stringa ad uno dei metodi di log per registrarlo.
- Il *secondo* modulo crea un nuovo oggetto eccezione con il tipo eccezione specificato e un messaggio di eccezione ottenuto dal file di messaggi di collaborazione. Il messaggio viene identificato in base al numero nel file di messaggi. Questo modulo di chiamata al metodo consente di passare fino a cinque valori di parametri del messaggio per il testo del messaggio. Separare questi parametri di messaggio con una virgola. Nel testo del messaggio (all'interno del file di messaggi), i parametri sono specificati da un numero tra parentesi graffe, ad esempio `{1}`. Il metodo `raiseException()` deve fornire un valore per ogni parametro di messaggio nel messaggio.
- Il *terzo* modulo fornisce un'altro modo per creare un nuovo oggetto eccezione che contiene un messaggio specificato nel file di messaggi. Il vettore `Objects` dei parametri del messaggio si comporta in modo simile all'elenco di parametri `String` nel secondo modulo di questo metodo. Tuttavia, mentre ciascun valore di parametro del messaggio nell'elenco `String` viene specificato separatamente, questo modulo dispone tutti i valori di parametro nel vettore `Objects`.

  Questo modulo è utile per innalzare un oggetto di eccezione che:

  – E' stato gestito precedentemente dalla collaborazione. Ad esempio, uno scenario potrebbe ricevere un'eccezione, assegnarla ad una variabile ed eseguire altre attività.

  – Presenta più di cinque parametri di messaggio. Mentre l'elenco `String` non può contenere più di cinque parametri, il vettore di parametri può contenere qualsiasi numero di parametri.
- Il *quarto* modulo *non* crea un'eccezione. Invece, innalza solo l'oggetto di eccezione specificato (oggetto `CollaborationException`) che viene fornito come argomento.

Nota: Tutti i moduli del metodo che ricevono un parametro *messageNum* richiedono l'utilizzo di un file di messaggi indicizzato per numero messaggio. Per informazioni sull'impostazione di un file di testo dei messaggi, consultare la sezione Capitolo 10, "Creazione del file di messaggi", a pagina 247.

Esempi

Questa sezione fornisce esempi di ognuno dei moduli del metodo `raiseException()`:

1. Il seguente esempio utilizza il primo modulo per innalzare un'eccezione di tipo `ServiceCallException`. Il testo viene passato direttamente nella chiamata al metodo.

   ```
   raiseException(ServiceCallException, "Attempt to validate
   Customer failed.");
   ```

2. Il successivo esempio utilizza il secondo modulo per innalzare un'eccezione di tipo `OperationException`, i cui messaggi sono riportati nel file di messaggi come segue:

   ```
   23
   Customer update failed for CustomerID={1} CustomerName={2}
   ```

   Questa chiamata `raiseException()` recupera il messaggio 23 e recupera i valori dei due parametri del messaggio (ID e nome cliente) dalla variabile `fromCustomer` per generare il messaggio di eccezione:

   ```
   raiseException(OperationException, 23,
      fromCustomer.getString("CustomerID"),
      fromCustomer.getString("CustomerName"));
   ```

3. Il seguente esempio utilizza il terzo modulo per inviare i valori dei parametri del messaggio come vettore `Objects`.

   Ad esempio, si supponga che il file di messaggi contenga il seguente testo del messaggio:

   ```
   2000
   Collaboration Message: BOName: {1} with Verb: {2} encountered
   an undefined error.
   ```

   Il codice seguente crea un vettore di parametri `Objects`, carica in esso i valori e richiama il metodo `raiseException()`:

   ```
   Object[] myParamArray = new Object[2];

   myParamArray[0] = triggeringBusObj.getType();
   myParamArray[1] = triggeringBusObj.getVerb();

   raiseException(AnyException, 2000, myParamArray);
   ```

4. L'ultimo esempio utilizza il quarto modulo del metodo per innalzare l'eccezione precedentemente gestita. La variabile definita dal sistema `currentException` è un oggetto eccezione che contiene l'eccezione.

   ```
   raiseException(currentException);
   ```

resubmitFailedEvent()

Questo metodo invia nuovamente l'evento ad una collaborazione per l'elaborazione.

Sintassi

```
resubmitFailedEvent (String ownerName, String connectorName, int index,
     int option, int flowTraceLevel)
```

Parametri

ownerName *OwnerName* è il nome della collaborazione a cui si invia nuovamente l'evento non riuscito.

connectorName *ConnectorName* è il nome del connettore a cui si invia nuovamente l'evento non riuscito.

index *Index* è l'indice dell'evento non riuscito.

option *Option* specifica come inviare nuovamente l'evento non riuscito. *Option* può assumere uno dei seguenti valori:

FailedEventInfo.RESUBMIT_ORIGINAL_BUSOBJ
Il metodo invia di nuovo l'oggetto business originale non riuscito.

FailedEventInfo.RESUBMIT_NEW_BUSOBJ
Il metodo richiede un nuovo oggetto business dal connettore e lo invia per l'elaborazione.

flowtracelevel *flowtracelevel* è un numero intero che indica il livello di traccia per il flusso non riuscito inviato nuovamente.

Esempi

Il seguente esempio invia nuovamente gli oggetti business con lo stato evento STATUS_SERVICECALL_IN_TRANSIT per essere rielaborati.

```
// Set up to query the failed events of a specific type for the collaboration.
EventQueryDef queryOption = new EventQueryDef();
// set query condition of busobj name
queryOption.nameBusObj = "TestBO";
// set query condition of connector name
queryOption.nameConnector = "SourceConnector";
// define query condition array
EventQueryDef[] queryOptions = new EventQueryDef[1];
queryOptions[0] = queryOption;
// query failed events with status of STATUS_SERVICECALL_IN_TRANSIT,
// no time limitation
FailedEventInfo[]resultArray = queryFailedEvents(queryOptions,
     "", "", FailedEventInfo.STATUS_SERVICECALL_IN_TRANSIT);
int newStatus = FailedEventInfo.RESUBMIT_ORIGINAL_BUSOBJ;
int myFlowTraceLevel = 3;

// get the query result
for (int i = 0; i < resultArray.length; i++) {
 String busObjName = resultArray[i].nameBusObj;
 String nameConnector = resultArray[i].nameConnector;
 int wipIndex = resultArray[i].wipIndex;

// Resubmit the failed events with status STATUS_SERVICECALL_IN_TRANSIT
// from queue.
 resubmitFailedEvent (busObjName, nameConnector, wipIndex, newStatus,
 myFlowTraceLevel);
 }
```

Consultare anche

"queryFailedEvents()" a pagina 380

"saveFailedEvent()"

---

saveFailedEvent()

Questo metodo salva un evento non riuscito e ne imposta lo stato a STATUS_USER_GENERATED.

Sintassi

```
saveFailedEvent (BusObj busObj)
```

Parametri

busObj — *BusObj* è l'oggetto business da salvare. L'oggetto business salvato presenta un struttura identica all'evento originale non riuscito e tale struttura deve essere definita prima di richiamare il metodo. L'eventStatus dell'evento non riuscito diventa STATUS_USER_GENERATED.

Esempi

```
// Synchronize product information in a batch using wrapper business
// object ProductSet.
// Create an empty parent bo
BusObj parentBo = new BusObj("ProductSet");

// Set the values which are the same as the tringgeringBusObj
parentBo.set("ProductSetId", triggeringBusObj.get("ProductSetId"));
parentBo.set("CreateDate", triggeringBusObj.get("CreateDate"));

// Set Verb to the parent bo
parentBo.setVerb(triggeringBusObj.getVerb());

// Add the failed business object to parent bo. Here iterProduct
// is the child BO which gets failed during iteration process.
parentBo.set("Products", iterProduct);

// Call collaboration API to save the failed event
saveFailedEvent(parentBo);
```

Note

- L'oggetto business salvato deve essere supportato dalla collaborazione.
- In un gruppo di collaborazione, il salvataggio degli eventi non riusciti può essere eseguito solo al vertice del gruppo di collaborazione.
- Questo metodo non supporta Access Framework.

Consultare anche

"queryFailedEvents()" a pagina 380

sendEmail()

Invia un messaggio e-mail in modo asincrono.

Sintassi

```
void sendEmail(String message, String subject, Vector recipients)
```

Parametri

message — Il testo del messaggio e-mail.

subject — La riga oggetto del messaggio e-mail

recipients — Un `Vector` che contiene gli indirizzi e-mail dei destinatari del messaggio. Questo vettore contiene oggetti `String`; più oggetti sono separati da virgole.

Note

Il metodo `sendEmail()` può inviare un messaggio e-mail ai destinatari specificati nel vettore *recipients*, se:

- E' stato specificato un indirizzo e-mail nel campo `Indirizzo e-mail per notifica` nella finestra di dialogo Proprietà oggetto di collaborazione.
- La collaborazione e-mail e l'adattatore e-mail sono in esecuzione. (La collaborazione e-mail viene istanziata e configurata automaticamente quando si avvia InterChange Server Express e non richiede immissioni da parte dell'utente). Se l'adattatore e-mail non è in esecuzione, `sendEmail()` *non* determina l'arresto dell'esecuzione della collaborazione.

Nota: Il metodo `logError()` invia automaticamente un messaggio di errore ad un destinatario e-mail (presupponendo che la collaborazione e l'adattatore e-mail siano in esecuzione). Il metodo `sendEmail()` consente di inviare esplicitamente un messaggio e-mail.

Esempi

```
// Initialize the Vector for the list of email addresses
Vector emailList = new Vector();

// Add as many email addresses as Strings to the Vector
emailList.add("dbadmin@us.ibm.com, netadmin@us.ibm.com,
     cwadmin@us.ibm.com");

// Initialize the message and subject as Strings
String message = "This is the body of the email";
String subject = "This is the subject of the email";

// Make the call to sendEmail()
sendEmail(message, subject, emailList);
```

---

trace()

Scrive un messaggio di traccia nella destinazione del log.

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 utilizzato per determinare quali messaggi di traccia sono emessi. Il metodo scrive il messaggio di traccia quando il livello di traccia per l'oggetto di collaborazione è maggiore o uguale a questo valore *traceLevel*. I livelli di traccia di questa collaborazione devono essere definiti e documentati in modo che l'amministratore sappia quale livello utilizzare per l'oggetto di collaborazione.

traceMsg — Il testo del messaggio di traccia scritto nel file di traccia.

messageNum — Il numero di un messaggio nel file di messaggi di collaborazione, che è indicizzato per numero messaggio. Per informazioni

sull'impostazione di un file di testo dei messaggi, consultare la sezione Capitolo 10, "Creazione del file di messaggi", a pagina 247.

param Il valore di un singolo parametro del messaggio. Possono essere presenti fino a cinque parametri di messaggio, separati da virgole. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio.

paramArray Un vettore di valori di parametri del messaggio. Ognuno viene risolto sequenzialmente con un parametro nel testo del messaggio.

Note

Il metodo `trace()` invia un messaggio di traccia alla destinazione del log della collaborazione. Per impostazione predefinita, la destinazione del log è il file `InterchangeSystem.log`. La destinazione del log può essere modificata immettendo un valore per il parametro `LOG_FILE` nel file di configurazione InterChange Server Express, `InterchangeSystem.cfg` . Il valore del parametro può essere un nome file oppure `STDOUT`, che determina la scrittura del log nella finestra comandi di InterChange Server Express.

Possono essere impostati altri tre parametri di configurazione di sistema relativi alla registrazione di traccia. Tutti i parametri si trovano nel file di configurazione, `InterchangeSystem.cfg`:

- Impostare la dimensione massima del file di traccia con il parametro `MAX_TRACE_FILE_SIZE`. Poiché la dimensione predefinita del file è illimitata, si dovrebbe sempre impostare un valore massimo.
- Impostare da uno a cinque file di traccia di archivio per il parametro `NUMBER_OF_ARCHIVE_TRACES`. Il valore predefinito è cinque, se il parametro non è impostato.
- Impostare il parametro `MIRROR_TRACE_TO_STDOUT` se si desidera che i messaggi di errore siano visualizzati su STDOUT nello stesso momento in cui vengono scritti nel file di traccia. Il valore predefinito è false; i messaggi non sono scritti contemporaneamente in STDOUT.

Il metodo `trace()` fornisce numerosi moduli:

- Il primo modulo riceve solo un messaggio stringa che viene visualizzato quando la traccia è impostata sul livello 1 o superiore.
- Il secondo modulo riceve un livello di traccia e un messaggio stringa che viene visualizzato quando la funzione di traccia è impostata sul livello specificato o su un livello superiore.
- Il terzo modulo riceve un livello di traccia e un numero che rappresenta un messaggio nel file di messaggi di collaborazione. L'intero testo del messaggio viene visualizzato nel file di messaggi e stampato senza parametri, quando la funzione di traccia è impostata sul livello specificato o su un livello superiore.
- Il quarto modulo riceve un livello di traccia, un numero che rappresenta un messaggio nel file di messaggi di collaborazione e uno o più parametri da utilizzare nel messaggio. E' possibile inviare fino a cinque parametri da utilizzare con il messaggio, separando i valori con una virgola.
- Il quinto modulo riceve un livello di traccia, un numero che rappresenta un messaggio nel file di messaggi di collaborazione e un vettore di valori di parametri.

Un oggetto di collaborazione può essere configurato per produrre una traccia generata dal sistema o una traccia generata dalla collaborazione. Il metodo trace()

genera un messaggio che viene stampato dall'oggetto di collaborazione, se è stato configurato per stampare una traccia generata dalla collaborazione. Per una guida all'utilizzo della traccia, fare riferimento a "Aggiunta di messaggi di traccia" a pagina 209.

Esempi

Il seguente esempio utilizza il secondo modulo del metodo per generare un messaggio di traccia a livello 2 con il testo del messaggio fornito:

```
trace (2, "Starting to trace at Level 2");
```

Il seguente esempio utilizza il quarto modulo del metodo per scrivere il messaggio 201 del file di messaggi di collaborazione, se il livello di traccia dell'oggetto è 2 o superiore. Il messaggio presenta due parametri, nome e anno, i cui valori sono passati dalla chiamata del metodo.

```
trace(2, 201, "DAVID", "1961");
```

Capitolo 23. Classe BusObj

I metodi documentati in questo capitolo agiscono su oggetti della classe BusObj. Questi oggetti rappresentano oggetti business di InterChange Server Express.

Nota: La classe BusObj viene utilizzata per lo sviluppo e la mappatura della collaborazione; verificare nella sezione Note i problemi relativi all'utilizzo di ciascun metodo.

Nella Tabella 82 sono riportati i metodi della classe BusObj.

Tabella 82. Riepilogo dei metodi BusObj

Metodo	Descrizione	Pagina
copy()	Copia tutti i valori degli attributi dall'oggetto business di input a questo oggetto business.	392
duplicate()	Crea un oggetto business (oggetto BusObj) esattamente uguale a questo.	393
equalKeys()	Confronta i valori degli attributi chiave di questo oggetto business con quelli dell'oggetto business di input.	393
equals()	Confronta i valori degli attributi di questo oggetto business con quelli dell'oggetto business di input, includendo gli oggetti business secondari.	394
equalsShallow()	Confronta i valori degli attributi di questo oggetto business con quelli dell'oggetto business di input, escludendo dal confronto gli oggetti business secondari.	396
exists()	Verifica l'esistenza di un attributo dell'oggetto business con un nome specificato.	396
getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()	Recupera il valore di un singolo attributo da un oggetto business.	397
getLocale()	Recupera la locale dei dati dell'oggetto business.	399
getType()	Recupera il nome della definizione di un oggetto business sul quale è basato questo oggetto business.	399
getVerb()	Recupera l'istruzione di questo oggetto business.	400
isBlank()	Individua se il valore di un attributo è impostato ad una stringa a lunghezza zero.	400
isKey()	Individua se un attributo dell'oggetto business è definito come attributo chiave.	401
isNull()	Individua se il valore di un attributo dell'oggetto business è null.	401

Tabella 82. Riepilogo dei metodi BusObj (Continua)

Metodo	Descrizione	Pagina
`isRequired()`	Individua se un attributo dell'oggetto business è definito come attributo obbligatorio.	402
`keysToString()`	Recupera i valori degli attributi di chiave primaria dell'oggetto business come stringa.	403
`set()`	Imposta un attributo dell'oggetto business ad un valore specificato per un tipo di dati particolare.	403
`setDefaultAttrValues()`	Imposta tutti gli attributi ai valori predefiniti.	405
`setKeys()`	Imposta i valori degli attributi chiave di questo oggetto business ai valori degli attributi chiave di un altro oggetto business.	405
`setVerb()`	Imposta l'istruzione di un oggetto business.	406
`setWithCreate()`	Imposta un attributo dell'oggetto business al valore del tipo di dati specificato.	406
`toString()`	Restituisce i valori di tutti gli attributi di un oggetto business come stringa.	407
`validData()`	Verifica se un valore specificato è un tipo di dati valido per un attributo specificato.	408

copy()

Copia tutti i valori degli attributi dall'oggetto business di input a questo oggetto business.

Sintassi

```
void copy(BusObj inputBusObj)
```

Parametri

inputBusObj Il nome dell'oggetto business i cui valori di attributi sono copiati nell'oggetto business corrente.

Note

Il metodo `copy()` copia l'intero oggetto business, inclusi tutti gli oggetti business secondari e i vettori di oggetti business secondari. Questo metodo non imposta un riferimento all'oggetto copiato. Al contrario, clona tutti gli attributi, ovvero crea copie separate degli attributi.

Esempi

Il seguente esempio copia i valori contenuti in `sourceCustomer` a `destCustomer`.

```
destCustomer.copy(sourceCustomer);
```

Il seguente esempio crea tre oggetti business (`myBusObj`, `myBusObj2` e `mysettingBusObj`) ed imposta l'attributo `attr1` di `myBusObj` con il valore di `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 di codice, `myBusObj.attr1` e `myBusObj2.attr1` sono *entrambi* impostati sull'oggetto business `mySettingBusObj`. Tuttavia, se `mySettingBusObj` viene modificato, `myBusObj.attr1` cambia mentre `myBusObj2.attr1` non viene modificato. Dal momento che gli attributi di `myBusObj2` erano stati impostati con `copy()`, i loro valori erano stati clonati. Pertanto, il valore di `attr1` in `myBusObj2` è ancora il valore originale di `mySettingBusObj.attr1` *prima* della modifica.

---

duplicate()

Crea un oggetto business (oggetto `BusObj`) esattamente uguale a questo.

Sintassi

```
BusObj duplicate()
```

Valori di ritorno

L'oggetto business duplicato.

Eccezioni

`CollaborationException` — Il metodo `duplicate()` può impostare il seguente tipo eccezione per questa eccezione: `ObjectException`.

Note

Questo metodo crea un clone dell'oggetto business e lo restituisce. E' necessario assegnare esplicitamente il valore di ritorno di questa chiamata al metodo ad una variabile dichiarata di tipo `BusObj`.

Esempi

Il seguente esempio duplica `sourceCustomer` per creare `destCustomer`.

```
BusObj destCustomer = sourceCustomer.duplicate();
```

---

equalKeys()

Confronta i valori degli attributi chiave di questo oggetto business con quelli dell'oggetto business di input.

Sintassi

```
boolean equalKeys(BusObjinputBusObj)
```

Parametri

inputBusObj Un oggetto business da confrontare con questo oggetto business.

Valori di ritorno

Restituisce `true` se i valori di tutti gli attributi chiave sono gli stessi; restituisce `false` se sono diversi.

Eccezioni

`CollaborationException` — Il metodo `equalKeys()` può impostare il seguente tipo eccezione per questa eccezione:

- `ObjectException` – Impostato se l'argomento dell'oggetto business non è valido.

Consultare anche

`equals()`, `equalsShallow()`

Note

Questo metodo esegue un confronto superficiale, ovvero non confronta le chiavi negli oggetti business secondari.

Esempi

Il seguente esempio confronta i valori chiave di `order2` con quelli di `order1`.

```
boolean areEqual = order1.equalKeys(order2);
```

---

equals()

Confronta i valori degli attributi di questo oggetto business con quelli dell'oggetto business di input, includendo gli oggetti business secondari .

Sintassi

```
-boolean equals(Object inputBusObj)
```

Parametri

inputBusObj Un oggetto business da confrontare con questo oggetto business.

Valori di ritorno

Restituisce `true` se i valori di tutti gli attributi chiave sono gli stessi; altrimenti restituisce `false`.

Eccezioni

`CollaborationException` — Il metodo `equals()` può impostare il seguente tipo eccezione per questa eccezione:

- `ObjectException` — Impostato se l'argomento dell'oggetto business non è valido.

Note

Questo metodo confronta i valori degli attributi di questo oggetto business con quelli dell'oggetto business di input. Se gli oggetti business sono gerarchici, il confronto include tutti gli attributi degli oggetti business secondari.

1. Passando l'oggetto business come `Object` si è certi che questo metodo `equals()` sovrascriva il metodo `Object.equals()`.
2. Nel confronto, un valore null nell'oggetto di origine viene considerato equivalente ad un qualsiasi valore con cui viene confrontato e restituisce `true`. I risultati di questo metodo possono non essere su base simmetrica nell'ordine di

confronto. Un confronto A.equals(B) può non essere uguale a B.equals(A). Il secondo esempio mostra i risultati di diversi confronti degli stessi oggetti.

3. L'ordine degli oggetti secondari influenza i risultati di questo metodo. Se tutte le altre condizioni sono rispettate, una variazione dell'ordine restituisce un valore false.
4. L'istruzione non ha effetti sul risultato del confronto.

Consultare anche

`equalKeys()`, `equalsShallow()`

Esempi

Il seguente esempio confronta tutti gli attributi di `order2` a tutti gli attributi di `order1` ed assegna il risultato del confronto alla variabile `areEqual`. Il confronto include gli attributi degli oggetti business secondari, se presenti.

```
boolean areEqual = order1.equals(order2);
```

Il successivo esempio confronta `triggeringBusObj` con `sourceBusObj` e mostra l'effetto dell'ordine di confronto sui risultati.

```
logInfo("***Comparing triggering BO to empty Source BO");
if (triggeringBusObj.equals(SourceBusObj)) {
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - TRUE");
} else
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - FALSE");
if (SourceBusObj.equals(triggeringBusObj)) {
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - TRUE");
} else
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - FALSE");

logInfo("*** Copying Source BO verb from triggering BO");
SourceBusObj.copy(triggeringBusObj);
if (triggeringBusObj.equals(SourceBusObj)) {
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - TRUE");
} else
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - FALSE");
if (SourceBusObj.equals(triggeringBusObj)) {
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - TRUE");
} else
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - FALSE");

logInfo("*** Setting Source BO verb to Update");
SourceBusObj.setVerb("Update");
if (triggeringBusObj.equals(SourceBusObj)) {
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - TRUE");
} else
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - FALSE");
if (SourceBusObj.equals(triggeringBusObj)) {
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - TRUE");
} else
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - FALSE");

logInfo("*** Swapping order of Source BO children");
BusObjArray TempArray = SourceBusObj.getBusObjArray("Attr3");
TempArray.swap(0,1);
if (triggeringBusObj.equals(SourceBusObj)) {
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - TRUE");
} else
  logInfo("**** triggeringBusObj.equals(SourceBusObj) - FALSE");
if (SourceBusObj.equals(triggeringBusObj)) {
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - TRUE");
} else
  logInfo("**** SourceBusObj.equals(triggeringBusObj) - FALSE");
```

Il precedente codice ha i seguenti risultati:

```
[Mesg: Info ***Comparing triggering BO to empty Source BO]
[Mesg: Info **** triggeringBusObj.equals(SourceBusObj) - FALSE]
[Mesg: Info **** SourceBusObj.equals(triggeringBusObj) - TRUE]
[Mesg: Info *** Copying Source BO from triggering BO]
[Mesg: Info **** triggeringBusObj.equals(SourceBusObj) - TRUE]
[Mesg: Info **** SourceBusObj.equals(triggeringBusObj) - TRUE]
[Mesg: Info *** Setting Source BO verb to Update]
[Mesg: Info **** triggeringBusObj.equals(SourceBusObj) - TRUE]
[Mesg: Info **** SourceBusObj.equals(triggeringBusObj) - TRUE]
[Mesg: Info *** Swapping order of Source BO children]
[Mesg: Info **** triggeringBusObj.equals(SourceBusObj) - FALSE]
[Mesg: Info **** SourceBusObj.equals(triggeringBusObj) - FALSE]
```

equalsShallow()

Confronta i valori degli attributi di questo oggetto business con quelli dell'oggetto business di input, escludendo dal confronto gli oggetti business secondari.

Sintassi

```
boolean equalsShallow(BusObj inputBusObj)
```

Parametri

inputBusObj Un oggetto business da confrontare con questo oggetto business.

Valori di ritorno

Restituisce `true` se i valori di tutti gli attributi chiave sono gli stessi; altrimenti restituisce `false`.

Eccezioni

`CollaborationException` — Il metodo `equalsShallow()` può impostare il seguente tipo eccezione per questa eccezione:

- `ObjectException` – Impostato se l'argomento dell'oggetto business non è valido.

Consultare anche

`equalKeys()`, `equals()`

Esempi

Il seguente esempio confronta gli attributi di `order2` con gli attributi di `order1`, escludendo gli attributi degli oggetti business secondari, se presenti.

```
boolean areEqual = order1.equalsShallow(order2);
```

exists()

Verifica l'esistenza di un attributo dell'oggetto business con un nome specificato.

Sintassi

```
boolean exists(String attribute)
```

Parametri

attribute Il nome di un attributo

Valori di ritorno

Restituisce `true` se l'attributo esiste; restituisce `false` se non esiste.

Esempi

Il seguente esempio verifica se l'oggetto business `order` presenta un attributo denominato `Notes`.

```
boolean notesAreHere = order.exists("Notes");
```

---

getBoolean(), getDouble(), getFloat(), getInt(), getLong(), get(), getBusObj(), getBusObjArray(), getLongText(), getString()

Recupera il valore di un singolo attributo da 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 specifica la posizione ordinale di un attributo nell'elenco degli attributi dell'oggetto business.

Valori di ritorno

Il valore dell'attributo specificato.

Eccezioni

`CollaborationException` — Questi metodi `get` possono impostare il seguente tipo eccezione per questa eccezione:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo. Ad esempio, questa eccezione può essere generata se la collaborazione richiama `getDouble()` per un attributo String che non è costituito da cifre o richiama `getString()` per un attributo non esistente.

Note

Questi metodi "get" recuperano un valore di attributo dall'oggetto business corrente. Restituiscono una copia del valore dell'attributo. *Non* restituiscono un riferimento all'oggetto per questo attributo nell'oggetto business di origine. Pertanto, una variazione del valore dell'attributo nell'oggetto business di origine *non* viene eseguita sul valore restituito dallo specifico metodo get. Ogni volta che si richiama uno di questi metodi get, questo restituisce una nuova copia (clone) dell'attributo.

Questi metodi get forniscono i seguenti moduli:

- Il primo modulo restituisce un valore del tipo specificato nel nome del metodo. Ad esempio, getBoolean() restituisce un valore booleano, getBusObj() restituisce un valore BusObj, getDouble() restituisce un valore double, ecc. Tuttavia, getLongText() restituisce un oggetto String in quanto il tipo longtext di InterChange Server Express è un oggetto String privo di dimensione massima. Utilizzare questi moduli per recuperare gli attributi con specifici tipi di dati di base o definiti da InterChange Server Express.

  Questi metodi offrono la possibilità di accedere ad un valore attributo specificando il *nome* dell'attributo.
- Il secondo modulo, get(), recupera il valore di un attributo di *qualsiasi* tipo. E' possibile eseguire il casting del valore restituito al valore del tipo attributo appropriato.

  Questo metodo offre la possibilità di accedere ad un valore attributo specificando *uno tra* il *nome* dell'attributo e la *posizione* dell'indice dell'attributo nell'elenco degli attributi dell'oggetto business.

Esempi

Il seguente esempio illustra le modalità con le quali `get()` restituisce una copia (clone) del valore dell'attributo invece di un 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* viene modificato in quanto la chiamata `get()` ha restituito una copia dell'attributo `attr1`.

Il seguente esempio utilizza `getBusObj()` per recuperare un oggetto business secondario che contiene un indirizzo del cliente dall'oggetto business `customer` e lo assegna alla variabile `address`.

```
BusObj address = customer.getBusObj("Address");
```

Il seguente esempio utilizza `getString()` per recuperare il valore dell'attributo `CustomerName`. La variabile dell'oggetto business è `sourceCustomer`.

```
String customerName = sourceCustomer.getString("CustomerName");
```

Il seguente esempio utilizza `getInt()` per recuperare i valori `Quantity` da due oggetti business le cui variabili sono `item1` e `item2`. L'esempio calcola poi la somma delle quantità.

```
int sumQuantity = item1.getInt("Quantity") + item2.getInt("Quantity");
```

Il seguente esempio recupera l'attributo `Item` dalla variabile dell'oggetto business `order`. L'attributo `Item` è un vettore oggetti business.

```
BusObjArray items = order.getBusObjArray("Item");
```

Il seguente esempio riceve il valore dell'attributo `CustID` dall'oggetto business di origine ed imposta il valore `Customer` nell'oggetto business di destinazione in modo corrispondente.

```
destination.set("Customer", source.get("CustID"));
```

Il seguente esempio accede ad un valore di attributo che utilizza la posizione ordinale dell'attributo nell'elenco di attributi:

```
for(i=0; i<maxAttrCount; i++)
{
     String strValue = (String)myBusObj.get(i);
     ...
```

getLocale()

Recupera la locale associata ai dati dell'oggetto business.

Sintassi

```
java.util.Locale getLocale()
```

Parametri

Nessuno.

Valori di ritorno

Un oggetto Locale Java che contiene le informazioni sulla locale dell'oggetto business. Questo oggetto Locale deve essere un'istanza della classe `java.util.Locale`.

Note

Il metodo `getLocale()` restituisce la locale associata ai dati di un oggetto business. Questa locale è spesso diversa dalla locale di collaborazione in cui viene eseguita la collaborazione.

Consultare anche

getLocale() (classe `BaseCollaboration`), setLocale()

getType()

Recupera il nome della definizione di un oggetto business sul quale è basato questo oggetto business.

Sintassi

```
String getType()
```

Valori di ritorno

Il nome di una definizione di oggetto business.

Note

Il tipo di oggetto business, per questo metodo, è il nome della definizione di oggetto business dalla quale è stato creato l'oggetto business.

Esempi

Il seguente esempio recupera il tipo di un oggetto business denominato `sourceShipTo`.

```
String typeName = sourceShipTo.getType();
```

Il seguente esempio copia un evento di attivazione in un nuovo oggetto business del tipo appropriato.

```
BusObj source = new BusObj(triggeringBusObj.getType());
```

getVerb()

Recupera l'istruzione di questo oggetto business.

Sintassi

```
String getVerb()
```

Valori di ritorno

Il nome di un'istruzione, ad esempio Create, Retrieve, Update o Delete.

Note

Questo metodo è utile per gli scenari che gestiscono più tipi di evento in ingresso. Il primo nodo azione in uno scenario richiama `getVerb()`. Il collegamento di transizione in uscita dal nodo azione verifica quindi il contenuto della stringa restituita, in modo che ogni collegamento di transizione in uscita sia l'inizio di un percorso di esecuzione che gestisce una delle possibili istruzioni.

Esempi

Il seguente esempio ottiene l'istruzione da un oggetto business denominato `orderEvent` e lo assegna ad una variabile denominata `orderVerb`.

```
String orderVerb = orderEvent.getVerb();
```

isBlank()

Individua se il valore di un attributo è impostato ad una stringa a lunghezza zero.

Sintassi

```
boolean isBlank(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori di ritorno

Restituisce `true` , se il valore dell'attributo è una stringa a lunghezza zero; altrimenti restituisce `false`.

Note

Una stringa a lunghezza zero può essere paragonata ad una stringa "". E' diversa da null, la cui presenza viene rilevata dal metodo `isNull()`.

Se una collaborazione deve recuperare un valore di attributo e poi eseguire su di esso un'operazione, è possibile richiamare `isBlank()` e `isNull()` per verificare che presenti un valore prima di recuperare il valore.

Esempi

Il seguente esempio verifica se l'attributo `Material` dell'oggetto business `sourcePaperClip` è una stringa a lunghezza zero.

```
boolean key = sourcePaperClip.isBlank("Material");
```

isKey()

Individua se un attributo dell'oggetto business è definito come attributo chiave.

Sintassi

```
boolean isKey(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori di ritorno

Restituisce `true` se l'attributo è un attributo chiave; restituisce `false` se non è un attributo chiave.

Esempi

Il seguente esempio determina se l'attributo `CustID` dell'oggetto business `customer` è un attributo chiave.

```
boolean keyId = (customer.isKey("CustID"));
```

isNull()

Individua se il valore di un attributo dell'oggetto business è null.

Sintassi

```
boolean isNull(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori di ritorno

Restituisce `true` se il valore dell'attributo è null; restituisce `false` se non è null.

Note

Null indica nessun valore, in contrasto con un valore stringa a lunghezza zero, che viene rilevato richiamando `isBlank()`. Verificare un oggetto con `isNull()` prima di utilizzarlo, in quanto se l'oggetto è null, l'operazione potrebbe non riuscire.

Un valore di attributo può essere null in queste situazioni:

- Il valore dell'attributo è stato impostato esplicitamente su null.

  Un valore di attributo può essere impostato su null utilizzando il metodo `set()`.
- Il valore dell'attributo non è mai stato impostato.

  Quando una collaborazione utilizza il metodo `new()` per creare un nuovo oggetto business, tutti i valori di attributo sono inizializzati a null. Se il valore dell'attributo non è stato impostato tra il momento della creazione e la chiamata `isNull()`, il valore è ancora null.
- Il valore null è stato inserito durante la mappatura.

  Quando una collaborazione elabora un oggetto business ricevuto da un connettore, il processo di mappatura potrebbe inserire un valore null. Il processo

di mappatura converte l'oggetto business specifico dell'applicazione ricevuto dal connettore nell'oggetto business generico gestito dalla collaborazione. Per ciascun attributo nell'oggetto business generico che non ha un equivalente nell'oggetto specifico di applicazione, la mappa inserisce un valore null.

Suggerimenti: Chiamare sempre `isNull()` prima di eseguire un'operazione su un attributo che rappresenta un oggetto business secondario o un vettore oggetti business secondario, in quanto Java non consente operazioni su oggetti null.

Esempi

Il seguente esempio verifica se l'attributo `Material` dell'oggetto business `sourcePaperClip` presenta un valore null.

```
boolean key = sourcePaperClip.isNull("Material");
```

Il seguente esempio verifica se l'attributo `CustAddr` dell'oggetto business `contract1` è null prima di recuperarlo. Il recupero dell'attributo procede solo se la verifica `isNull()` è false, dimostrando che l'attributo non è null.

```
if (! contract1.isNull("CustAddr"))
   {
   BusObj customerAddress = contract1.getBusObj("CustAddr");
   // do something with the "customerAddress" business object
   }
```

isRequired()

Individua se un attributo dell'oggetto business è definito come attributo obbligatorio.

Sintassi

```
boolean isRequired(String attribute)
```

Parametri

attribute Il nome di un attributo.

Valori di ritorno

Restituisce `true` se il valore dell'attributo è obbligatorio; restituisce `false` se non è obbligatorio.

Note

Se un attributo è definito come obbligatorio, deve presentare un valore e il valore non deve essere null.

Esempi

Il seguente esempio registra un'avvertenza se un attributo obbligatorio presenta un valore null.

```
if ( (customer.isRequired("Address"))
      && (customerBusObj.isNull("Address")) )
   {
   logWarning(12, "Address is required and cannot be null.");
   }
```

```
else
   {
   // do something else
   }
```

keysToString()

Recupera i valori degli attributi di chiave primaria dell'oggetto business come stringa.

Sintassi

```
String keysToString()
```

Valori di ritorno

Un oggetto String che contiene tutti i valori chiave di un oggetto business, concatenati e ordinati per il valore ordinale degli attributi.

Note

L'output di questo metodo contiene il nome dell'attributo e i relativi valori. Più valori sono valori di attributo della chiave primaria, concatenati e separati da spazi. Ad esempio, se è presente un attributo chiave primaria SS#, l'output potrebbe essere:

```
SS#=100408394
```

Se gli attributi di chiave primaria sono `FirstName` e `LastName`, l'output potrebbe essere:

```
FirstName=Nina LastName=Silk
```

Esempi

Il seguente esempio restituisce i valori degli attributi chiave dell'oggetto business rappresentato dal nome variabile `fromOrder`.

```
String keyValues = fromOrder.keysToString();
```

set()

Imposta un attributo dell'oggetto business ad un valore specificato per un tipo di dati particolare.

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 specifica la posizione ordinale di un attributo nell'elenco degli attributi dell'oggetto business.

value Un valore di attributo.

Eccezioni

`CollaborationException` — Il metodo `set()` può impostare il seguente tipo eccezione per questa eccezione:

- `AttributeException` — Impostato se si verifica un problema di accesso all'attributo.

Note

Questi metodi set() impostano un valore di attributo nell'oggetto business corrente. Questi metodi impostano un riferimento all'oggetto quando all'attributo viene assegnato il valore. Il valore dell'attributo *non* viene clonato dall'oggetto business di origine. Di conseguenza, le modifiche al *`valore`* nell'oggetto business di origine vengono effettuate anche all'attributo nell'oggetto business che richiama set().

Il metodo set() fornisce i seguenti moduli:

- Il primo modulo imposta un valore del tipo specificato dal secondo tipo di parametro del metodo. 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, ecc. Utilizzare questo modulo per impostare gli attributi con specifici tipi di dati di base o definiti da InterChange Server Express.

  Questi metodi offrono la possibilità di accedere ad un valore attributo specificando il *nome* dell'attributo.
- Il secondo modulo imposta il valore di un attributo di *qualsiasi* tipo. E' possibile inviare qualsiasi tipo di dati come valore di attributo in quanto il parametro valore attributo è di tipo Object. Ad esempio, per impostare un attributo dell'oggetto `BusObj` o `LongText`, utilizzare questo modulo del metodo e passare l'oggetto `BusObj` o `LongText` come valore di attributo.

  Questo modulo del metodo set() offre la possibilità di accedere ad un valore attributo specificando *uno tra* il *nome* dell'attributo e la *posizione* dell'indice dell'attributo nell'elenco degli attributi dell'oggetto business.

Esempi

Il seguente esempio imposta l'attributo `LName` in `toCustomer` sul valore `Smith`.

```
toCustomer.set("LName", "Smith");
```

Il seguente esempio illustra come `set()` assegna un riferimento all'oggetto invece di clonare il valore:

```
BusObj BusObj myBusObj = new BusObj();
BusObj mySettingBusObj = new BusObj();
myBusObj.set("attr1", mySettingBusObj);
```

Dopo l'esecuzione di questo frammento di codice, l'attributo `attr1` di `myBusObj` viene impostato sull'oggetto business `mySettingBusObj`. Se `mySettingBusObj` viene modificato, `myBusObj.attr1` viene modificato nello stesso modo, in quanto `set()` crea un riferimento all'oggetto `mySettingBusObj` quando imposta l'attributo `attr1`; *non* crea una copia statica di `mySettingBusObj`.

Il seguente esempio imposta un valore di attributo che utilizza la posizione ordinale dell'attributo nell'elenco di attributi:

```
for(i=0; i<maxAttrCount; i++)
{
     myBusObj.set(i, strValue);
     ...
```

setDefaultAttrValues()

Imposta tutti gli attributi ai valori predefiniti.

Sintassi

```
void setDefaultAttrValues()
```

Note

Una definizione di oggetto business può includere i valori predefiniti degli attributi. Il metodo imposta i valori degli attributi di questo oggetto business ai valori specificati come predefiniti nella definizione.

Esempi

Il seguente esempio imposta i valori dell'oggetto business `PaperClip` ai rispettivi valori 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 eccezione per questa eccezione:

- `AttributeException` – Impostato se si verifica un problema di accesso all'attributo.
- `ObjectException` – Impostato se l'argomento dell'oggetto business non è valido.

Esempi

Il seguente esempio imposta i valori chiave dell'oggetto business `helpdeskCustomer` ai valori chiave 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 Locale Java che contiene le informazioni sulla locale da assegnare all'oggetto business. Questo oggetto Locale deve essere un'istanza della classe `java.util.Locale`.

Valori di ritorno

Nessuno.

Note

Il metodo `setLocale()` assegna una locale ai dati associati ad un oggetto business. Questa locale può essere diversa dalla locale di collaborazione in cui viene eseguita la collaborazione.

Consultare anche

getLocale()

---

setVerb()

Imposta l'istruzione di un oggetto business.

Sintassi

```
void setVerb(String verb)
```

Parametri

verb L'istruzione dell'oggetto business.

Note

Questo metodo è utilizzato nella mappatura degli oggetti business.

Non utilizzare questo metodo nella maschera di collaborazione, nella quale deve essere impostata in modo interattivo l'istruzione di un oggetto business, valorizzando le proprietà di una chiamata di servizio.

Esempi

Il seguente esempio imposta l'istruzione Delete nell'oggetto business `contactAddress`.

```
contactAddress.setVerb("Delete");
```

---

setWithCreate()

Imposta un attributo dell'oggetto business al valore del tipo di dati specificato.

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.
busObj	L'oggetto business da inserire nell'attributo di destinazione.
busObjArray	Il vettore dell'oggetto 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 il seguente tipo eccezione per questa eccezione:

- `AttributeException` — Impostato 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 secondario a cardinalità multipla, `BusObj` a `BusObjArray` come ultimo elemento. Tuttavia, se l'attributo di destinazione contiene un `BusObj`, questo oggetto business sostituisce il valore precedente.

Esempi

Il seguente esempio imposta un attributo denominato `ChildAttrAttr` sul valore 5. L'attributo si trova in un oggetto business contenuto nell'attributo `ChildAttr` di `myBO`. Se l'oggetto business `childAttr` non esiste al momento della chiamata, questa chiamata al metodo crea l'oggetto.

```
myBO.setWithCreate("childAttr.childAttrAttr", "5");
```

---

toString()

Restituisce i valori di tutti gli attributi di un oggetto business come stringa.

Sintassi

```
String toString()
```

Valori di ritorno

Un oggetto String costituito da tutti i valori di attributo contenuti in un oggetto business.

Note

La stringa risultante da una chiamata a questo metodo è simile a quella riportata nell'esempio seguente:

```
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

Il seguente esempio restituisce una stringa che contiene i valori di attributo della variabile dell'oggetto business `fromOrder`.

```
String values = fromOrder.toString();
```

validData()

Verifica se un valore specificato è un tipo di dati valido per un attributo specificato.

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

Verifica la compatibilità del valore passato con l'attributo di destinazione (come specificato da *attributeName*). Questi sono i criteri utilizzati:

Per tipi primitivi (String, long, int, double, float, boolean)	deve essere possibile convertire il valore nel tipo di dati dell'attributo
Per un `BusObj`	il valore deve presentare lo stesso tipo dell'attributo di destinazione
Per un `BusObjArray`	il valore deve puntare ad un `BusObj` o `BusObjArray` con lo stesso tipo (definizione di oggetto business) dell'attributo
Per un `Object`	Il valore deve essere di tipo `String`, `BusObj` o `BusObjArray`. Sono quindi applicate le corrispondenti regole di convalida.

Metodi obsoleti

Alcuni metodi nella classe `BusObj` erano supportati nelle versioni precedenti ma non sono più supportati. L'utilizzo di questi **metodi obsoleti** non genera un errore. Tuttavia, si consiglia di evitarne l'uso e di migrare il codice esistente ai nuovi metodi. I metodi obsoleti potrebbero essere eliminati in un futuro rilascio.

Nella Tabella 83 sono riportati i metodi obsoleti per la classe `BusObj` e i metodi sostitutivi. Se in precedenza non è mai stato utilizzato Process Designer Express, ignorare questa sezione.

Tabella 83. Metodi obsoleti, classe BusObj

Metodo precedente	Metodo sostitutivo
`getCount()`	`BusObjArray.size()`
`getKeys()`	`keysToString()`
`getValues()`	`toString()`
`not`	operatore Java standard NOT, `"!"`
`set(BusObj inputBusObj)`	`copy()`
Tutti i metodi che ricevono un oggetto business secondario o un secondario di oggetto business secondario come argomento di input	Ottenere un handle per l'oggetto business secondario o il vettore oggetti business ed utilizzare i metodi della classe `BusObj` o `BusObjArray`

Il metodo `setVerb()`, che era stato precedentemente indicato come obsoleto, è stato ora ripristinato per essere utilizzato nella mappatura. Non utilizzarlo all'interno di una collaborazione.

Capitolo 24. Classe BusObjArray

I metodi documentati in questo capitolo agiscono su oggetti della classe BusObjArray. Questi sono definiti nella classe BusObjArray definita da InterChange Server Express. La classe BusObjArray incapsula un vettore di oggetti business. In un oggetto business gerarchico, un attributo è un riferimento ad un vettore di oggetti business secondari quando la sua cardinalità è uguale a n. Le operazioni sulla classe BusObjArray possono restituire un oggetto BusObjArray o un vettore di oggetti business.

Nella Tabella 84 sono riportati i metodi documentati in questo capitolo.

Tabella 84. Riepilogo dei metodi BusObjArray

Metodo	Descrizione	Pagina
addElement()	Aggiunge un oggetto business a questo vettore oggetti business.	412
duplicate()	Crea un vettore oggetti business (oggetto BusObjArray) esattamente uguale a questo.	412
elementAt()	Recupera un singolo oggetto business specificando la sua posizione in questo vettore oggetti business.	413
equals()	Confronta un altro vettore oggetti business con questo vettore.	413
getElements()	Recupera il contenuto di questo vettore oggetti business.	413
getLastIndex()	Recupera l'ultimo indice disponibile da un vettore oggetti business.	414
max()	Recupera il valore massimo dell'attributo specificato tra tutti gli elementi in questo vettore oggetti business.	414
maxBusObjArray()	Restituisce gli oggetti business con il massimo valore per l'attributo specificato, come vettore oggetti business (oggetto BusObjArray).	415
maxBusObjs()	Restituisce gli oggetti business con il valore massimo per l'attributo specificato, come vettore di oggetti BusObj.	416
min()	Recupera il valore minimo per l'attributo specificato tra gli oggetti business di questo vettore.	417
minBusObjArray()	Restituisce gli oggetti business con il valore minimo per l'attributo specificato, come oggetto BusObjArray.	418
minBusObjs()	Restituisce gli oggetti business con il valore minimo per l'attributo specificato, come vettore di oggetti BusObj.	419
removeAllElements()	Rimuove tutti gli elementi da questo vettore oggetti business.	420
removeElement()	Elimina un elemento da un vettore oggetti business.	420
removeElementAt()	Rimuove un elemento ad una particolare posizione in questo vettore oggetti business.	421
setElementAt()	Imposta il valore di un oggetto business in un vettore oggetti business.	421

Tabella 84. Riepilogo dei metodi BusObjArray (Continua)

Metodo	Descrizione	Pagina
`size()`	Restituisce il numero di elementi in questo vettore oggetti business.	422
`sum()`	Aggiunge i valori dell'attributo specificato per tutti gli oggetti business in questo vettore oggetti business.	422
`swap()`	Inverte la posizione di due oggetti in un vettore oggetti business. Considerare che il primo elemento nel vettore è zero (0), il secondo è 1, il terzo è 2, ecc.	422
`toString()`	Recupera i valori in questo vettore oggetti business e li restituisce in un'unica stringa.	423

addElement()

Aggiunge un oggetto business a questo vettore oggetti business.

Sintassi

```
void addElement(BusObj element)
```

Parametri

element Un oggetto business da aggiungere a questo vettore.

Eccezioni

`CollaborationException` — Il metodo `addElement()` può impostare il seguente tipo eccezione per questa eccezione:

- `AttributeException` – Impostato se l'elemento non è valido.

Esempi

Il seguente esempio utilizza il metodo `getBusObArray()` per recuperare un vettore oggetti business denominato `itemList` dall'oggetto business `order`. Il vettore è assegnato ad `items`, quindi un nuovo oggetto business viene aggiunto a `items`.

```
BusObjArray items = order.getBusObjArray("itemList");
items.addElement(new BusObj("oneItem"));
```

duplicate()

Crea un vettore oggetti business (oggetto `BusObjArray`) esattamente uguale a questo.

Sintassi

```
BusObjArray duplicate()
```

Valori di ritorno

Un vettore oggetti business.

Esempi

Il seguente esempio duplica il vettore `items`, creando `newItems`.

```
BusObjArray newItems = items.duplicate();
```

elementAt()

Recupera un singolo oggetto business specificando la sua posizione in questo vettore oggetti business.

Sintassi

```
BusObj elementAt(int index)
```

Parametri

index Un elemento di vettore da recuperare. Il primo elemento nel vettore è zero (0), il secondo è 1, il terzo è 2, ecc.

Eccezioni

`CollaborationException` — Il metodo `elementAt()` può impostare il seguente tipo eccezione per questa eccezione:

- `AttributeException` – Impostato se l'elemento non è valido.

Esempi

Il seguente esempio recupera l'undicesimo oggetto business nel vettore `items` e lo assegna alla variabile `Item`.

```
BusObj Item = items.elementAt(10);
```

equals()

Confronta un altro vettore oggetti business con questo vettore.

Sintassi

```
boolean equals(BusObjArray inputBusObjArray)
```

Parametri

inputBusObjArray
Un vettore oggetti business da confrontare con questo vettore oggetti business.

Note

Il confronto tra due vettori oggetto business verifica il numero di elementi e i relativi valori di attributo.

Esempi

Il seguente esempio utilizza `equals()` per impostare un loop condizionale, la cui parte interna non viene mostrata.

```
if (items.equals(newItems))
   {
...
   }
```

getElements()

Recupera il contenuto di questo vettore oggetti business.

Sintassi

```
BusObj[] getElements()
```

Eccezioni

`CollaborationException` — Il metodo `getElements()` può impostare il seguente tipo eccezione per questa eccezione:

- `ObjectException` — Impostato se uno degli elementi non è valido.

Esempi

Il seguente esempio stampa gli elementi del vettore items.

```
BusObj[] elements = items.getElements();
for (i=0, i<size; i++)
   {
   trace(1, elements[i].toString());
   }
```

getLastIndex()

Recupera l'ultimo indice disponibile da un vettore oggetti business.

Sintassi

```
int getLastIndex()
```

Restituzioni

L'ultimo indice relativo all'ultimo elemento in questo `BusObjArray`.

Note

Notare che in precedenza, a questo scopo veniva utilizzato il metodo `size()`. Quindi l'utente potrebbe utilizzare `size()` su un vettore oggetti business per recuperare l'ultimo indice disponibile in un `BusObjArray`. Sfortunatamente, questo approccio conduce a dati non corretti se `BusObjArray` contiene un intervallo di numerazione.

Come tutti i vettori Java, `BusObjArray` è un vettore con riferimento zero. Questo significa che il metodo `size()` restituirà un numero maggiore di uno rispetto al metodo `getLastIndex()`.

Esempi

Il seguente esempio recupera l'ultimo indice nel vettore oggetti business.

```
int lastElementIndex = items.getLastIndex();
```

max()

Recupera il valore massimo dell'attributo specificato tra tutti gli elementi in questo vettore oggetti business.

Sintassi

```
String max(String attr)
```

Parametri

attr Una variabile che fa riferimento a un attributo dell'oggetto business. L'attributo deve presentare uno dei tipi seguenti: String, LongText, int, float e double.

Restituzioni

Il valore massimo dell'attributo specificato come stringa, o null se il valore di questo attributo è null per tutti gli elementi di questo `BusObjArray`.

Eccezioni

`UnknownAttributeException` — Quando l'attributo specificato non è un attributo valido negli oggetti business passati.

`UnsupportedAttributeTypeException` — Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Tutte le eccezioni riportate sono sottoclassi di `CollaborationException`. Il metodo `max()` può impostare il seguente tipo eccezione per queste eccezioni: `AttributeException`.

Note

Il metodo `max()` ricerca il valore massimo dell'attributo specificato tra gli oggetti business in questo `BusObjArray`. Ad esempio, se si utilizzano tre oggetti Employee e l'attributo è "Salary", di tipo "Float", restituirà la stringa che rappresenta il Salary più alto.

Se il valore dell'attributo specificato per un elemento in `BusObjArray` è null, l'elemento viene ignorato. Se il valore dell'attributo specificato è null per tutti gli elementi, viene restituito null.

Quando il tipo dell'attributo è String, `max()` restituisce il valore dell'attributo che rappresenta la stringa lessicale più lunga.

Esempi

```
String maxSalary = items.max("Salary");
```

---

maxBusObjArray()

Restituisce gli oggetti business con il massimo valore per l'attributo specificato, come vettore oggetti business (oggetto `BusObjArray`).

Sintassi

```
BusObjArray maxBusObjArray(String attr)
```

Parametri

attr Una variabile String, LongText, int, float o double che fa riferimento a un attributo di un oggetto business nel vettore oggetti business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObjArray` o `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è un attributo valido negli oggetti business passati.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Tutte le eccezioni riportate sono sottoclassi di `CollaborationException`. Il metodo `maxBusObjArray()` può impostare il seguente tipo eccezione per queste eccezioni: `AttributeException`.

Note

Il metodo `maxBusObjArray()` individua uno o più oggetti business con il valore massimo per l'attributo specificato, e restituisce questi oggetti business in un oggetto `BusObjArray`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il metodo determina il valore maggiore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore massimo di Salary, il metodo restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il metodo restituisce la stringa lessicale più lunga.

Esempi

```
BusObjArray boarrayWithMaxSalary = items.maxBusObjArray("Salary");
```

---

maxBusObjs()

Restituisce gli oggetti business con il massimo valore per l'attributo specificato, come vettore di oggetti `BusObj`.

Sintassi

```
BusObj[] maxBusObjs(String attr)
```

Parametri

attr Una variabile String, LongText, int, float o double che fa riferimento a un attributo di un oggetto business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObj[]` o `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è un attributo valido negli oggetti business passati.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Tutte le eccezioni riportate sono sottoclassi di `CollaborationException`. Il metodo `maxBusObjs()` può impostare il seguente tipo eccezione per queste eccezioni: `AttributeException`.

Note

Il metodo `maxBusObjs()` individua uno o più oggetti business con il valore massimo per l'attributo specificato, e restituisce questi oggetti business come vettore di oggetti `BusObj`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il metodo determina il valore maggiore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore massimo di Salary, il metodo restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il metodo restituisce la stringa lessicale più lunga.

Esempi

```
BusObj[] bosWithMaxSalary = items.maxBusObjs("Salary");
```

---

min()

Recupera il valore minimo dell'attributo specificato tra gli oggetti business di questo vettore.

Sintassi

```
String min(String attr)
```

Parametri

attr Una variabile String, LongText, int, float o double che fa riferimento a un attributo di un oggetto business.

Restituzioni

Il valore minimo dell'attributo specificato come stringa, o null se il valore di questo attributo è null per tutti gli elementi di questo `BusObjArray`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è un attributo valido negli oggetti business passati.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Tutte le eccezioni riportate sono sottoclassi di `CollaborationException`. Il metodo `min()` può impostare il seguente tipo eccezione per queste eccezioni: `AttributeException`.

Note

Il metodo `min()` ricerca il valore minimo dell'attributo specificato tra gli oggetti business in questo vettore oggetti business.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il metodo determina il valore minore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore minimo di Salary, il metodo restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il metodo restituisce la stringa lessicale più corta.

Esempi

```
String minSalary = items.min("Salary");
```

minBusObjArray()

Restituisce gli oggetti business con il minimo valore per l'attributo specificato, come oggetto `BusObjArray`.

Sintassi

```
BusObjArray minBusObjArray(String attr)
```

Parametri

attr Una variabile `String`, `LongText`, `int`, `float` o `double` che fa riferimento a un attributo di un oggetto business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObjArray` o `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è un attributo valido negli oggetti business passati.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Tutte le eccezioni riportate sono sottoclassi di `CollaborationException`. Il metodo `minBusObjArray()` può impostare il seguente tipo eccezione per queste eccezioni: `AttributeException`.

Note

Il metodo `minBusObjArray()` individua uno o più oggetti business con il valore minimo per l'attributo specificato, e restituisce questi oggetti business in un oggetto `BusObjArray`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il metodo determina il valore minore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore minimo di Salary, il metodo restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il metodo restituisce la stringa lessicale più corta.

Esempi

```
BusObjArray boarrayWithMinSalary = items.minBusObjArray("Salary");
```

---

minBusObjs()

Restituisce gli oggetti business con il minimo valore per l'attributo specificato, come vettore di oggetti `BusObj`.

Sintassi

```
BusObj[] minBusObjs(String attr)
```

Parametri

attr Una variabile String, LongText, int, float o double che fa riferimento a un attributo di un oggetto business.

Restituzioni

Un elenco di oggetti business sotto forma di `BusObj[]` o `null`.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è un attributo valido negli oggetti business passati.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Tutte le eccezioni riportate sono sottoclassi di `CollaborationException`. Il metodo `minBusObjs()` può impostare il seguente tipo eccezione per queste eccezioni: `AttributeException`.

Note

Il metodo `minBusObjs()` individua uno o più oggetti business con il valore minimo per l'attributo specificato, e restituisce questi oggetti business come vettore di oggetti `BusObj`.

Ad esempio, si supponga che questo sia un vettore oggetti business che contiene oggetti business Employee e che l'argomento di input sia l'attributo Salary, di tipo Float. Il metodo determina il valore minore per Salary in tutti gli oggetti business Employee e restituisce l'oggetto business che contiene tale valore. Se più oggetti business presentano lo stesso valore minimo di Salary, il metodo restituisce tutti questi oggetti business.

Un oggetto business viene ignorato se l'attributo specificato contiene null. Se il valore è null in tutti gli oggetti business del vettore, viene restituito null.

Quando il tipo dell'attributo è String, il metodo restituisce la stringa lessicale più corta.

Esempi

```
BusObj[] bosWithMinSalary = items.minBusObjs("Salary");
```

removeAllElements()

Rimuove tutti gli elementi da questo vettore oggetti business.

Sintassi

```
void removeAllElements()
```

Esempi

Il seguente esempio rimuove tutti gli elementi del vettore `items`.

```
items.removeAllElements();
```

removeElement()

Elimina un elemento da un vettore oggetti business.

Sintassi

```
void removeElement(BusObj element)
```

Parametri

elementReference	Una variabile che fa riferimento a un elemento del vettore.

Eccezioni

`CollaborationException` — Il metodo `removeElement()` può impostare il seguente tipo eccezione per questa eccezione:

- `AttributeException` – Impostato se l'elemento non è valido.

Note

Dopo aver eliminato un elemento dal vettore, il vettore viene ridimensionato, modificando gli indici degli elementi esistenti.

Esempi

Il seguente esempio elimina l'elemento `Child1` dal vettore oggetti business `items`.

```
items.removeElement(Child1);
```

removeElementAt()

Rimuove un elemento ad una particolare posizione in questo vettore oggetti business.

Sintassi

```
void removeElementAt(int index)
```

Note

Dopo aver rimosso un elemento dal vettore, il vettore viene ridimensionato, modificando eventualmente gli indici degli elementi esistenti.

Parametri

index Un numero intero che rappresenta la posizione dell'elemento nel vettore. Il primo elemento nel vettore è alla posizione zero (0), il secondo elemento è alla posizione 1, il terzo è alla posizione 2, ecc.

Eccezioni

`CollaborationException` — Il metodo `removeElementAt()` può impostare il seguente tipo eccezione per questa eccezione:

- `AttributeException` – Impostato se l'elemento non è valido.

Esempi

Il seguente esempio elimina il sesto oggetto business nel vettore `items`.

```
items.removeElementAt(5);
```

setElementAt()

Imposta il valore di un oggetto business in un vettore oggetti business.

Sintassi

```
void setElementAt (int index, BusObj element)
```

Parametri

index Un numero intero che rappresenta la posizione nel vettore. Il primo elemento nel vettore è zero (0), il secondo è 1, il terzo è 2, ecc.

inputBusObj L'oggetto business che contiene i valori a cui impostare l'elemento del vettore.

Eccezioni

`CollaborationException` — Il metodo `setElementAt()` può impostare il seguente tipo eccezione per questa eccezione:

- `AttributeException` – Impostato se l'elemento non è valido.

Note

Questo metodo imposta i valori dell'oggetto business in una posizione di vettore specificata sui valori di un oggetto business di input.

Esempi

Il seguente esempio crea un nuovo oggetto business di tipo Item e lo aggiunge al vettore `items`, come quarto elemento.

```
items.setElementAt(3, new BusObj("Item"));
```

size()

Restituisce il numero di elementi in questo vettore oggetti business.

Sintassi

```
int size()
```

Esempi

Il seguente esempio restituisce il numero di elementi nel vettore `items`.

```
int size = items.size();
```

sum()

Aggiunge i valori dell'attributo specificato per tutti gli oggetti business in questo vettore oggetti business.

Sintassi

```
double sum(String attrName)
```

Parametri

attr Una variabile che fa riferimento a un attributo dell'oggetto business. L'attributo deve essere di tipo int, float o double.

Restituzioni

La somma dell'attributo specificato dall'elenco di oggetti business.

Eccezioni

`UnknownAttributeException` – Quando l'attributo specificato non è un attributo valido negli oggetti business passati.

`UnsupportedAttributeTypeException` – Quando il tipo dell'attributo specificato non è uno dei tipi di attributo supportati elencati nella sezione Note.

Tutte le eccezioni riportate sono sottoclassi di `CollaborationException`. Il metodo `sum()` può impostare il seguente tipo eccezione per queste eccezioni: `AttributeException`.

Esempi

```
double sumSalary = items.sum("Salary");
```

swap()

Inverte la posizione di due oggetti business in un vettore oggetti business. Considerare che il primo elemento nel vettore è zero (0), il secondo è 1, il terzo è 2, ecc.

Sintassi

```
void swap(int index1, int index2)
```

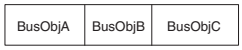
Parametri

index1 La posizione di un elemento del vettore da scambiare.

index2 La posizione dell'altro elemento del vettore da scambiare.

Esempi

Il seguente esempio utilizza `swap()` per invertire le posizioni di BusObjA e BusObjC nel vettore seguente:




```
swap(0,2);
```

Il risultato della chiamata a `swap()` è il vettore seguente:

toString()

Recupera i valori in questo vettore oggetti business e li restituisce in un'unica stringa.

Sintassi

```
String toString()
```

Esempi

Il seguente esempio utilizza `toString()` per recuperare il contenuto del vettore oggetti business `items` e poi utilizza `logInfo()` per scrivere il contenuto nel file di log.

```
logInfo(items.toString());
```

Capitolo 25. Classe CwDBConnection

La classe `CwDBConnection` fornisce i metodi per l'esecuzione delle query SQL in un database. Le query sono eseguite attraverso una connessione ottenuta da un pool di connessioni. Per creare un'istanza di questa classe, è necessario richiamare `getDBConnection()` nella classe `BaseCollaboration`. Tutte le collaborazioni sono derivate o sottoclassi di `BaseCollaboration`, quindi hanno accesso a `getDBConnection()`.

La Tabella 85 riepiloga i metodi della classe `CwDBConnection`.

Tabella 85. Riepilogo dei metodi CwDBConnection

Metodo	Descrizione	Pagina
`beginTransaction()`	Inizia una transazione esplicita per la connessione corrente.	425
`commit()`	Esegue il commit della transazione attiva associata con la connessione corrente.	426
`executeSQL()`	Esegue una query SQL statica specificando la relativa sintassi e un vettore di parametri facoltativo.	429
`executePreparedSQL()`	Esegue una query SQL preparata specificando la relativa sintassi e un vettore di parametri facoltativo.	427
`executeStoredProcedure()`	Esegue una procedura memorizzata SQL specificando il nome e un vettore di parametri.	430
`getUpdateCount()`	Restituisce il numero di righe interessate dall'ultima operazione di scrittura nel database.	431
`hasMoreRows()`	Determina se nei risultati della query sono presenti altre righe da elaborare.	432
`inTransaction()`	Determina se una transazione è attiva nella connessione corrente.	433
`isActive()`	Determina se la connessione corrente è attiva.	433
`nextRow()`	Recupera la riga successiva dai risultati della query.	434
`release()`	Rilascia la connessione corrente, restituendola al relativo pool di connessioni.	434
`rollBack()`	Esegue il rollback della transazione attiva associata con la connessione corrente.	435

beginTransaction()

Inizia una transazione esplicita per la connessione corrente.

Sintassi

```
void beginTransaction()
```

Parametri

Nessuno.

Valori di ritorno

Nessuno.

Eccezioni

`CwDBConnectionException` — Se si verifica un errore di database.

Note

Il metodo `beginTransaction()` contrassegna l'inizio di una nuova transazione esplicita nella connessione corrente. I metodi `beginTransaction()`, `commit()` e `rollBack()` contribuiscono a fornire la gestione dei limiti della transazione per una transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE o UPDATE, e una procedura memorizzata che include una delle istruzioni SQL.

Importante: Utilizzare `beginTransaction()` solo se la connessione utilizza il bracketing transazioni esplicite. Se la connessione utilizza il bracketing transazioni implicite, l'utilizzo di `beginTransaction()` genera un'eccezione `CwDBTransactionException`.

Prima dell'inizio di una transazione esplicita, è necessario creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` dalla classe `BaseCollaboration`. Assicurarsi che questa connessione utilizzi il bracketing transazioni esplicite

Esempi

Il seguente esempio utilizza una transazione per eseguire una query per l'inserimento di righe in una tabella del database associato con le connessioni 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 anche

"Gestione della transazione" a pagina 240

`"commit()"`, `getDBConnection()`, `inTransaction()`, `rollBack()`

---

commit()

Esegue il commit della transazione attiva associata con la connessione corrente.

Sintassi

```
void commit()
```

Parametri

Nessuno.

Valori di ritorno

Nessuno.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore di database.

Note

Il metodo `commit()` termina la transazione attiva eseguendo il commit delle modifiche al database associato con la connessione corrente. I metodi `beginTransaction()`, `commit()` e `rollBack()` contribuiscono a fornire la gestione dei limiti della transazione per una transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE o UPDATE, e una procedura memorizzata che include una delle istruzioni SQL.

Importante: Utilizzare `commit()` solo se la connessione utilizza il bracketing transazioni esplicite. Se la connessione utilizza il bracketing transazioni implicite, l'utilizzo di `commit()` genera un'eccezione `CwDBTransactionException`. Se un transazione esplicita non termina con `commit()` (o `rollback()`) prima di rilasciare la connessione, InterChange Server Express termina implicitamente la transazione in base all'esito della collaborazione. Se la collaborazione ha avuto esito positivo, InterChange Server Express esegue il commit di questa transazione di database. Se la collaborazione *non* ha avuto esito positivo, InterChange Server Express esegue implicitamente il rollback della transazione di database. Indipendentemente dall'esito della collaborazione, InterChange Server Express registra un'avvertenza.

Prima dell'inizio di una transazione esplicita, è necessario creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` dalla classe `BaseCollaboration`. Assicurarsi che questa connessione utilizzi il bracketing transazioni esplicite

Esempi

Per un esempio di commit di una transazione, consultare l'esempio di `"beginTransaction()" a pagina 425`.

Consultare anche

"Gestione della transazione" a pagina 240

`"beginTransaction()" a pagina 425`, `getDBConnection()`, `inTransaction()`, `rollBack()`

---

executePreparedSQL()

Esegue una query SQL preparata specificando la relativa sintassi e un vettore di parametri facoltativo.

Sintassi

```
void executePreparedSQL(String query)
void executePreparedSQL(String query, Vector queryParameters)
```

Parametri

query Un rappresentazione in formato stringa della query SQL da eseguire nel database.

queryParameters
Un oggetto `Vector` di argomenti da passare ai parametri della query SQL.

Valori di ritorno

Nessuno.

Eccezioni

`CwDBSQLException` – Se si verifica un errore di database.

Note

Il metodo `executePreparedSQL()` invia la stringa *query* specificata come istruzione SQL preparata al database associato con la connessione corrente. La prima volta che viene eseguita, questa query viene inviata come stringa al database, che la compila in formato eseguibile (istruzione preparata), esegue l'istruzione SQL e restituisce l'istruzione preparata a `executePreparedSQL()`. Il metodo `executePreparedSQL()` salva l'istruzione preparata in memoria. Utilizzare `executePreparedSQL()` per le istruzioni SQL che devono essere eseguite più volte. Il metodo `executeSQL()` *non* salva l'istruzione preparata ed è quindi utile per le query che sono eseguite una sola volta.

Importante: Prima di eseguire una query con `executePreparedSQL()`, è necessario ottenere una connessione al database desiderato generando un oggetto `CwDBConnection` con il metodo `getDBConnection()` dalla classe `BaseDLM`.

Le istruzioni SQL che possono essere eseguite (se si dispone delle autorizzazioni necessarie sul database) sono le seguenti:

- L'istruzione SELECT per richiedere dati da una o più tabelle di database.

  Utilizzare i metodi `hasMoreRows()` e `nextRow()` per accedere ai dati recuperati.
- Le istruzioni SQL che modificano i dati del database
  - INSERT
  - DELETE
  - UPDATE

  Se la connessione utilizza il bracketing transazioni esplicite, è necessario avviare esplicitamente ogni transazione con `beginTransaction()` e terminarla con `commit()` o `rollback()`.
- L'istruzione CALL per eseguire una procedura memorizzata preparata con la limitazione che questa procedura memorizzata *non può* utilizzare alcun parametro OUT.

  Per eseguire procedure memorizzate con parametri OUT, utilizzare il metodo `executeStoredProcedure()`. Per ulteriori informazioni, consultare "Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237.

Consultare anche

"Esecuzione di query preparate" a pagina 233

"beginTransaction()" a pagina 425, "commit()" a pagina 426, executeSQL(), executeStoredProcedure(), getDBConnection(), hasMoreRows(), nextRow(), rollBack()

---

executeSQL()

Esegue una query SQL statica specificando la relativa sintassi e un vettore di parametri facoltativo.

Sintassi

```
void executeSQL(String query)
void executeSQL(String query, Vector queryParameters)
```

Parametri

query Un rappresentazione in formato stringa della query SQL da eseguire nel database.

queryParameters
Un oggetto Vector di argomenti da passare ai parametri della query SQL.

Valori di ritorno

Nessuno.

Eccezioni

CwDBSQLException – Se si verifica un errore di database.

Note

Il metodo executeSQL() invia la stringa *query* specificata come istruzione SQL statica al database associato con la connessione corrente. Questa query viene inviata come stringa al database, che la compila in formato eseguibile ed esegue l'istruzione SQL senza salvarla nel formato eseguibile. Utilizzare executeSQL() per le istruzioni SQL che devono essere eseguite una sola volta. Il metodo executePreparedSQL() salva il formato eseguibile (istruzione preparata) ed è quindi utile per le query che sono eseguite più volte.

Importante: Prima di eseguire una query con executeSQL(), è necessario ottenere una connessione al database desiderato generando un oggetto CwDBConnection con il metodo getDBConnection() dalla classe BaseDLM.

Le istruzioni SQL che possono essere eseguite (se si dispone delle autorizzazioni necessarie sul database) sono le seguenti:

- L' istruzione SELECT per richiedere dati da una o più tabelle di database.
  Utilizzare i metodi hasMoreRows() e nextRow() per accedere ai dati recuperati.
- Le istruzioni SQL che modificano i dati del database
  - INSERT
  - DELETE
  - UPDATE

  Se la connessione utilizza il bracketing transazioni esplicite, è necessario avviare esplicitamente ogni transazione con beginTransaction() e terminarla con commit() o rollback().

- L' istruzione CALL per eseguire una procedura memorizzata preparata con la limitazione che questa procedura memorizzata *non può* utilizzare alcun parametro OUT.

  Per eseguire procedure memorizzate con parametri OUT, utilizzare il metodo `executeStoredProcedure()`. Per ulteriori informazioni, consultare "Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237.

Esempi

Il seguente esempio esegue una query per inserire righe in un database le cui connessioni risiedono 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 un esempio più completo del codice per selezionare i dati da una tabella di database, consultare "Esecuzione di query statiche che restituiscono dati (SELECT)" a pagina 229.

Consultare anche

"Esecuzione di query statiche" a pagina 229

"executePreparedSQL()" a pagina 427, executeStoredProcedure(), getDBConnection(), hasMoreRows(), nextRow()

---

executeStoredProcedure()

Esegue una procedura memorizzata SQL specificando il nome e un 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 `Vector` di parametri da passare alla procedura memorizzata. Ciascun parametro è un'istanza della classe `CwDBStoredProcedureParam`. Per ulteriori informazioni su come passare parametri mediante un vettore, consultare "Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237.

Valori di ritorno

Nessuno.

Eccezioni

`CwDBSQLException` – Se si verifica un errore di database.

Note

Il metodo `executeStoredProcedure()` invia una chiamata alla *`storedProcedure`* specificata del database associato con la connessione corrente. Questo metodo invia la chiamata alla procedura memorizzata come istruzione SQL preparata; la prima volta che viene eseguita, questa procedura memorizzata viene inviata come stringa al database, che la compila in formato eseguibile (istruzione preparata), esegue l'istruzione SQL e restituisce l'istruzione preparata a `executeStoredProcedure()`. Il metodo `executeStoredProcedure()` salva l'istruzione preparata in memoria.

Importante: Prima di eseguire una procedura memorizzata con `executeStoredProcedure()`, è necessario creare un oggetto `CwDBConnection` con il metodo `getDBConnection()`dalla classe `BaseDLM`.

Per gestire i dati restituiti dalla procedura memorizzata, utilizzare i metodi `hasMoreRows()` e `nextRow()`.

E' possibile anche utilizzare il metodo `executeSQL()` o `executePreparedSQL()` per eseguire una procedura memorizzata quando la procedura memorizzata *non* contiene parametri OUT. Se la procedura memorizzata utilizza parametri OUT, si *deve* utilizzare `executeStoredProcedure()` per eseguirla. A differenza di `executeSQL()` o `executePreparedSQL()`, non è necessario passare l'istruzione SQL completa per eseguire la procedura memorizzata. Con `executeStoredProcedure()`, è necessario passare solo il nome della procedura memorizzata e un vettore di parametri (`Vector`) di oggetti `CwDBStoredProcedureParam`. Il metodo `executeStoredProcedure()` può determinare il numero di parametri dal vettore *`storedProcParameters`* e generare l'istruzione di chiamata per la procedura memorizzata.

Consultare anche

"Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237

`"executePreparedSQL()" a pagina 427`, `executeSQL()`, `getDBConnection()`, `hasMoreRows()`, `nextRow()`

---

getUpdateCount()

Restituisce il numero di righe interessate dall'ultima operazione di scrittura nel database.

Sintassi

```
int getUpdateCount()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce un `int` che rappresenta il numero di righe interessate dall'ultima operazione di scrittura.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore di database.

Note

Il metodo `getUpdateCount()` indica quante righe sono state modificate dall'ultima operazione di aggiornamento nel database associato alla connessione corrente. Questo metodo è utile dopo aver inviato un' istruzione UPDATE o INSERT al database, se si desidera determinare il numero di righe interessate da questa istruzione SQL.

Importante: Prima di utilizzare questo metodo, è necessario creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` dalla classe `BaseDLM` ed inviare una query che aggiorna il database con il metodo `executeSQL()` o il metodo `executePreparedSQL()` dalla classe `CwDBConnection`.

Consultare anche

`"executePreparedSQL()" a pagina 427`, `executeSQL()`, `getDBConnection()`

---

hasMoreRows()

Determina se nei risultati della query sono presenti altre righe da elaborare.

Sintassi

```
boolean hasMoreRows()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce `true` se sono presenti altre righe.

Eccezioni

`CwDBSQLException` – Se si verifica un errore di database.

Note

Il metodo `hasMoreRows()` determina se nei risultati della query associati con la connessione corrente sono presenti altre righe da elaborare. Utilizzare questo metodo per recuperare i risultati da una query che restituisce dati. Tali query comprendono istruzioni SELECT e procedure memorizzate. Alla connessione può essere associata una sola query alla volta. Pertanto, se si esegue un'altra query prima che `hasMoreRows()` abbia restituito `false`, i dati della query iniziale sono persi.

Consultare anche

"Esecuzione di query statiche che restituiscono dati (SELECT)" a pagina 229

"executePreparedSQL()" a pagina 427, executeSQL(), nextRow()

inTransaction()

Determina se una transazione è attiva nella connessione corrente.

Sintassi

```
boolean inTransaction()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce `true` se una transazione è attualmente attiva nella connessione corrente; altrimenti restituisce `false`.

Eccezioni

`CwDBConnectionException` – Se si verifica un errore di database.

Note

Il metodo `inTransaction()` restituisce un valore booleano che indica se la connessione corrente presenta una transazione attiva; ovvero, una transazione è stata avviata e non è terminata.

Importante: Prima dell'inizio di una transazione, è necessario creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` dalla classe `BaseDLM`.

Consultare anche

"Gestione della transazione" a pagina 240

"beginTransaction()" a pagina 425, "commit()" a pagina 426, getDBConnection(), rollBack()

isActive()

Determina se la connessione corrente è attiva.

Sintassi

```
boolean isActive()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce `true` se la connessione corrente è attiva; restituisce `false` se la connessione è stata rilasciata.

Eccezioni

Nessuno.

Consultare anche

`getDBConnection()`, `release()`

nextRow()

Recupera la riga successiva dai risultati della query.

Sintassi

```
Vector nextRow()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce la riga successiva dei risultati della query come un oggetto `Vector`.

Eccezioni

`CwDBSQLException` – Se si verifica un errore di database.

Note

Il metodo `nextRow()` restituisce una riga di dati dai risultati della query associati con la connessione corrente. Utilizzare questo metodo per recuperare i risultati da una query che restituisce dati. Tali query comprendono istruzioni SELECT e procedure memorizzate. Alla connessione può essere associata una sola query alla volta. Pertanto, se si esegue un'altra query prima che `nextRow()` abbia restituito l'ultima riga di dati, i risultati della query iniziale sono persi.

Consultare anche

"Esecuzione di query statiche che restituiscono dati (SELECT)" a pagina 229

`hasMoreRows()`, `"executePreparedSQL()" a pagina 427`, `executeSQL()`, `executeStoredProcedure()`

release()

Rilascia la connessione, restituendola al relativo pool di connessioni.

Sintassi

```
void release()
```

Parametri

Nessuno.

Valori di ritorno

Nessuno.

Eccezioni

`CwDBConnectionException`

Note

Il metodo `release()` rilascia esplicitamente la connessione corrente dall'oggetto di collaborazione. Una volta rilasciata, la connessione ritorna al relativo pool di connessioni, dove è disponibile per gli altri componenti (mappe o collaborazioni) che richiedono una connessione al database associato. Se non si rilascia esplicitamente una connessione, l'oggetto di collaborazione la rilascia implicitamente alla fine dell'esecuzione della collaborazione corrente. Pertanto, *non* è possibile salvare una connessione in una variabile statica e riutilizzarla.

Attenzione: *Non* utilizzare il metodo `release()` se una transazione è attualmente attiva. Con il bracketing transazioni implicite, InterChange Server Express non termina la transazione di database fino a quando non viene determinato l'esito positivo o negativo della collaborazione. Pertanto, l'utilizzo di questo metodo per una connessione che utilizza bracketing transazioni implicite genera una eccezione `CwDBTransactionException`. Se non si gestisce in modo esplicito questa eccezione, viene effettuato anche un rollback automatico della transazione attiva. E' possibile utilizzare il metodo `inTransaction()` per determinare se una transazione è attiva.

Consultare anche

"Rilascio di una connessione" a pagina 244

`getDBConnection()`, `inTransaction()`, `isActive()`

---

rollBack()

Esegue il rollback della transazione attiva associata con la connessione corrente.

Sintassi

```
void rollBack()
```

Parametri

Nessuno.

Valori di ritorno

Nessuno.

Eccezioni

`CwDBTransactionException` – Se

Note

Il metodo `rollback()` termina la transazione attiva eseguendo il rollback delle modifiche al database associato con la connessione corrente. I metodi `beginTransaction()`, `commit()` e `rollBack()` contribuiscono a fornire la gestione dei limiti della transazione per una transazione esplicita. Questa transazione contiene query SQL, che comprendono le istruzioni SQL INSERT, DELETE o UPDATE, e

una procedura memorizzata che include una delle istruzioni SQL. Se il rollback non riesce, `rollback()` genera un'eccezione `CwDBTransactionException` e registra un errore.

Importante: Utilizzare `rollback()` solo se la connessione utilizza il bracketing transazioni esplicite. Se la connessione utilizza il bracketing transazioni implicite, l'utilizzo di `rollback()` genera un'eccezione `CwDBTransactionException`. Se un transazione esplicita non termina con `rollback()` (o `commit()`) prima di rilasciare la connessione, InterChange Server Express termina implicitamente la transazione in base all'esito della collaborazione. Se la collaborazione ha avuto esito positivo, InterChange Server Express esegue il commit di questa transazione di database. Se la collaborazione *non* ha avuto esito positivo, InterChange Server Express esegue implicitamente il rollback della transazione di database. Indipendentemente dall'esito della collaborazione, InterChange Server Express registra un'avvertenza.

Prima dell'inizio di una transazione esplicita, è necessario creare un oggetto `CwDBConnection` con il metodo `getDBConnection()` dalla classe `BaseCollaboration`. Assicurarsi che questa connessione utilizzi il bracketing transazioni esplicite

Esempi

Per un esempio della gestione di una transazione con `rollback()`, consultare l'esempio in "Ambito della transazione con bracketing transazioni esplicite" a pagina 243.

Consultare anche

"Gestione della transazione" a pagina 240

`"beginTransaction()" a pagina 425`, `"commit()" a pagina 426`, `getDBConnection()`, `inTransaction()`

Capitolo 26. Classe CwDBStoredProcedureParam

Un oggetto CwDBStoredProcedureParam descrive un singolo parametro per una procedura memorizzata. La Tabella 86 riepiloga i metodi nella classe CwDBStoredProcedureParam.

Tabella 86. Riepilogo dei metodi CwDBStoredProcedureParam

Metodo	Descrizione	Pagina
CwDBStoredProcedureParam()	Crea una nuova istanza di CwDBStoredProcedureParam contenente le informazioni di argomento per il parametro di una procedura memorizzata.	437
getParamType()	Recupera il tipo in/out del parametro di procedura memorizzata corrente come costante di numero intero.	438
getValue()	Recupera il valore del parametro di procedura memorizzata corrente.	439

CwDBStoredProcedureParam()

Crea una nuova istanza di CwDBStoredProcedureParam contenente le informazioni di 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 in/out del parametro della procedura memorizzata associato.

paramValue Il valore di argomento da inviare alla procedura memorizzata. Questo valore è uno dei tipi di dati Java riportati di seguito

Valori di ritorno

Restituisce un nuovo oggetto `CwDBStoredProcedureParam` per contenere le informazioni per un argomento nella dichiarazione della procedura memorizzata.

Eccezioni

Nessuna.

Note

Il costruttore `CwDBStoredProcedureParam()` crea un'istanza `CwDBStoredProcedureParam` per descrivere un parametro per la procedura memorizzata. Le informazioni di parametro sono le seguenti:

- Tipo in/out del parametro

  Il primo argomento del costruttore inizializza questo tipo di parametro in/out. Per un elenco dei tipi di parametro in/out validi, consultare la Tabella 87 a pagina 439.
- Valore del parametro

  Il secondo argomento del costruttore inizializza questo valore di parametro. La classe `CwDBStoredProcedureParam` fornisce un modulo del costruttore per ciascuno dei tipi di dati supportati per il valore di parametro. Per un elenco delle associazioni tra tipi di dati Java e tipi di dati JDBC per i parametri di procedura memorizzata, consultare la Tabella 62 a pagina 239.

Viene fornito un `Vector` Java di parametri di procedura memorizzata al metodo `executeStoredProcedure()`, che crea una chiamata alla procedura memorizzata da un nome di procedura memorizzata e da un vettore di parametri, ed invia la chiamata al database associato con la connessione corrente.

Consultare anche

"Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237

`executeStoredProcedure()`

---

getParamType()

Recupera il tipo in/out del parametro di procedura memorizzata corrente come costante di numero intero.

Sintassi

```
int getParamType()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il tipo in/out del parametro `CwDBStoredProcedureParam` associato.

Eccezioni

Nessuna.

Note

Il metodo `getType()` restituisce il tipo in/out del parametro di procedura memorizzata corrente. Il tipo di parametro in/out indica come il parametro viene utilizzato dalla procedura memorizzata. La classe `CwDBStoredProcedureParam` rappresenta ciascun tipo in/out come costante, come mostrato nella Tabella 87.

Tabella 87. Tipi di parametro In/Out

Tipo di parametro In/Out	Descrizione	Costante di tipo In/Out
parametro IN	Un parametro IN è *di solo input*; quindi la procedura memorizzata accetta il suo valore come input, ma *non* utilizza il parametro per restituire un valore.	`PARAM_IN`
parametro OUT	Un parametro OUT è *di solo output*; quindi la procedura memorizzata *non* considera il suo valore come input, ma utilizza il parametro per restituire un valore.	`PARAM_OUT`
parametro INOUT	Un parametro INOUT è *di input e output*; quindi la procedura memorizzata accetta il suo valore come input ed utilizza anche il parametro per restituire un valore.	`PARAM_INOUT`

Consultare anche

"Richiamo di procedure memorizzate con executeStoredProcedure()" a pagina 237

`CwDBStoredProcedureParam()`, `getValue()`

getValue()

Recupera il valore del parametro di procedura memorizzata corrente.

Sintassi

```
Object getValue()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce il valore del parametro `CwDBStoredProcedureParam` associato come `Object` Java.

Eccezioni

Nessuna.

Note

Il metodo `getParamValue()` restituisce il valore del parametro come `Object` Java (ad esempio, `Integer`, `Double` o `String`). Se il valore restituito ad un parametro OUT è `NULL` JDBC, `getParamValue()` restituisce la costante `null`.

Consultare anche

`CwDBStoredProcedureParam()`, `getParamType()`

Capitolo 27. Classe CxExecutionContext

I metodi documentati in questo capitolo agiscono sul contesto di esecuzione, che è un contenitore delle informazioni di contesto accessibili all'utente associate con un flusso fornito. Questo è rappresentato dalla classe definita da InterChange Server Express, `CxExecutionContext`. Attualmente, solo le informazioni di esecuzioni specifiche di mappa sono disponibili come contesto di esecuzione di mappa. Nel codice di una mappa, Map Designer Express dichiara automaticamente una variabile speciale per accedere al contesto di esecuzione di mappa, `cwExecCtx`. Tuttavia, quando si richiama una mappa all'interno di una collaborazione, è necessario istanziare il proprio contesto di esecuzione globale ed il contesto di esecuzione di mappa. Per ulteriori informazioni, consultare "Richiamo di una mappa nativa" a pagina 214.

Nota: Per ulteriori informazioni sul contesto di esecuzione di mappa, consultare il manuale *Guida allo sviluppo delle mappe*.

La Tabella 88 riepiloga i metodi della classe `CxExecutionContext`.

Tabella 88. Riepilogo dei metodi CxExecutionContext

Metodo	Descrizione	Pagina
`CxExecutionContext()`	Crea una nuova istanza di un contesto di esecuzione globale.	441
`getContext()`	Recupera il contesto di esecuzione specificato dal contesto di esecuzione globale.	442
`setContext()`	Imposta un particolare contesto di esecuzione in modo che sia parte del contesto di esecuzione globale.	442

Costanti statiche

La classe `CxExecutionContext` definisce le costanti statiche che sono descritte nella Tabella 89.

Tabella 89. 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, necessario per contenere il contesto di esecuzione di mappa prima di richiamare una mappa dalla collaborazione.

Consultare anche

"Richiamo di una mappa nativa" a pagina 214

---

getContext()

Recupera il contesto di esecuzione specificato dal contesto di esecuzione globale.

Sintassi

```
Object getContext(String contextName)
```

Parametri

contextName Il nome del contesto di esecuzione ottenuto dal contesto di esecuzione globale.

Valori di ritorno

Restituisce una istanza del contesto di esecuzione specificato.

Esempi

Il seguente esempio ottiene un contesto di esecuzione di mappa da un contesto di esecuzione globale.

```
(MapExeContext) mapExeContext = globalExeContext.getContext(
   CxExecutionContext.MAPCONTEXT);
```

Consultare anche

"Richiamo di una mappa nativa" a pagina 214

---

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 il solo valore valido.

context Un oggetto che contiene le informazioni per il contesto di esecuzione. Per i contesti di esecuzione di mappa, questo `Object` è di tipo `MapExeContext`.

Valori di ritorno

Nessuno

Esempi

Il seguente esempio salva un contesto di esecuzione di mappa in un contesto di esecuzione globale:

```
globalExeContext.setContext(CxExecutionContext.MAPCONTEXT,
   mapExeContext);
```

Consultare anche

"Richiamo di una mappa nativa" a pagina 214

Capitolo 28. Classe CxBiDiEngine

La classe CxBidiEngine fornisce metodi per il trasferimento di oggetti business e stringhe da un formato bidirezionale all'altro.

La Tabella 90 riepiloga i metodi nella classe CxBidiEngine.

Tabella 90. Riepilogo dei metodi CwBidiEngine

Metodo	Descrizione	Pagina
`BiDiBOTransformation()`	Trasforma gli oggetti business di tipo BusinessObject da un formato bidirezionale all'altro.	445
`BiDiBusObjTransformation()`	Trasforma gli oggetti business di tipo BusObj da un formato bidirezionale ad un altro formato.	446
`BiDiStringTransformation()`	Trasforma le stringhe da un formato bidirezionale all'altro.	447

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 input. Consultare la Tabella 91 a pagina 446 per verificare i valori validi di questa stringa. Se questo parametro è null, il metodo imposta il valore predefinito sul formato standard bidirezionale di Windows.

formatOut Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare Tabella 91 a pagina 446 per i valori validi della stringa. Se questo parametro è null, il metodo imposta il valore predefinito sul formato standard bidirezionale di Windows.

replace Un valore che indica se l'oggetto business di input 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 null.

Eccezioni

Nessuno.

Esempi

Consultare l'esempio in "BiDiStringTransformation()" a pagina 447.

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 input. Consultare la Tabella 91 per verificare i valori validi di questa stringa. Se questo parametro è null, il metodo imposta il valore predefinito sul formato standard bidirezionale di Windows.

formatOut Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare la Tabella 91 per verificare i valori validi di questa stringa. Se questo parametro è null, il metodo imposta il valore predefinito sul formato standard bidirezionale di Windows.

replace Un valore che indica se l'oggetto business di input deve essere sostituito. Il valore valido è `true` o `false`.

Tabella 91. 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	Testo 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 null.

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 input. Consultare la Tabella 92 per verificare i valori validi di questa stringa. Se questo parametro è null, il metodo imposta il valore predefinito sul formato standard bidirezionale di Windows.

formatOut Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare la Tabella 92 per verificare i valori validi di questa stringa. Se questo parametro è null, il metodo imposta il valore predefinito sul formato standard bidirezionale di Windows.

Tabella 92. 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	Testo 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 29. Classe CollaborationException

I metodi documentati in questo capitolo agiscono su oggetti della classe `CollaborationException`. Questi oggetti rappresentano eccezioni di collaborazione. Le eccezioni possono verificarsi durante l'esecuzione di un oggetto di collaborazione. Lo scenario può catturare e gestire queste eccezioni. Una collaborazione può gestire due categorie di eccezioni:

- Eccezioni del processo business

  Le eccezioni del del processo business sono generate dal codice che utilizza i metodi API di collaborazione. Ad esempio, un'eccezione del processo business si può verificare quando lo scenario imposta il valore di un attributo dell'oggetto business, invia una richiesta ad un connettore, ecc.
- Eccezioni Java native

  Le eccezioni Java dal codice utente che utilizza i metodi Java nativi. L'ambiente runtime di collaborazione cattura e gestisce le eccezioni Java generate nel codice.

Nella Tabella 93 sono riportati i metodi descritti in questo capitolo.

Tabella 93. Riepilogo dei metodi CollaborationException

Metodo	Descrizione	Pagina
`getMessage()`	Recupera il testo del messaggio dall'eccezione corrente.	449
`getMsgNumber()`	Recupera il numero di messaggio del testo associato con l'eccezione corrente.	450
`getSubType()`	Recupera il tipo secondario di una eccezione.	450
`getType()`	Recupera il tipo di eccezione di collaborazione.	452
`toString()`	Scrive le informazioni di eccezione in una stringa.	453

getMessage()

Recupera il testo del messaggio dall'oggetto eccezione.

Sintassi

```
String getMessage()
```

Valori di ritorno

Una `String` che contiene il messaggio associato con l'oggetto eccezione.

Note

Il metodo getMessage() è utile per estrarre il testo dell'eccezione dalla variabile di sistema `currentException`. Questo testo di eccezione può essere incluso in una chiamata a raiseException() per assicurarsi che la causa dell'eccezione sia innalzata al successivo livello superiore di esecuzione.

Nota: E' possibile utilizzare il metodo toString() per recuperare il tipo e il testo dell'eccezione dall'eccezione corrente come stringa formattata.

getMsgNumber()

Recupera il numero di messaggio per il messaggio associato con l'oggetto eccezione.

Sintassi

```
int getMsgNumber()
```

Valori di ritorno

Il numero del messaggio (`int`) associato con il messaggio di eccezione corrente. Se il messaggio di eccezione non proviene da un file di messaggi, questo metodo restituisce zero (0).

Note

Il metodo getMsgNumber() è utile per ottenere il numero di messaggio associato con un messaggio di eccezione. Questo numero di messaggio può essere passato ad una chiamata a raiseException() o logError().

getSubType()

Recupera il tipo secondario di eccezione dall'oggetto eccezione.

Sintassi

```
String getSubType()
```

Valori di ritorno

Una `String` che contiene il tipo secondario dell'eccezione per l'eccezione corrente. Per ulteriori informazioni sui tipi secondari di eccezione validi, consultare la sezione Note.

Note

Il metodo getSubType() recupera il tipo secondario dell'eccezione per l'eccezione corrente. Per eccezioni i cui tipi non identificano adeguatamente la causa dell'eccezione, il tipo secondario dell'eccezione può fornire ulteriori informazioni. I seguenti tipi di eccezione utilizzano comunemente i tipi secondari di eccezione:

- `JavaException`

  L'ambiente runtime di collaborazione cattura le eccezioni Java e le include in un'eccezione di collaborazione con un tipo associato di eccezione Java. Una collaborazione può utilizzare getSubType() sull'eccezione di collaborazione per recuperare il tipo originale di eccezione Java (il nome classe dell'eccezione Java catturata). Tuttavia, normalmente questo non è necessario.
- `ServiceCallException`

  Il tipo eccezione `ServiceCallException` si verifica se una chiamata di servizio non riesce. Per sviluppare un meccanismo di collaborazione più robusto, è possibile utilizzare il tipo secondario dell'eccezione per determinare la causa dell'errore della chiamata di servizio. I tipi secondari di eccezione validi sono:

`AppTimeOut`	Un connettore non è riuscito a completare la comunicazione con l'applicazione.
`AppLogOnFailure`	Un connettore non è riuscito ad accedere all'applicazione.

AppRetrieveByContentFailed	Un'operazione Retrieve per valori non chiave, eseguita dall'applicazione, non è riuscita ad individuare una corrispondenza.
AppMultipleHits	Un'applicazione ha individuato e recuperato più di una entità in risposta ad una richiesta Retrieve.
AppBusObjDoesNotExist	Un'operazione Retrieve è stata eseguita nell'applicazione, ma l'entità rappresentata dall'oggetto business non esiste nel database dell'applicazione.
AppRequestNotYetSent	Nel caso di un agente connettore parallelo, la richiesta è stata accodata nel master agente ma non è stata inoltrata all'applicazione; pertanto, è possibile inviare nuovamente la richiesta. Per ulteriori informazioni, consultare "Richieste di chiamate di servizio non inviate" a pagina 194.
ServiceCallTransportException	Si è verificato un errore nel trasporto e non è possibile determinare con certezza se la richiesta ha raggiunto l'applicazione. Per ulteriori informazioni, consultare "Gestione delle eccezioni runtime correlate al trasporto" a pagina 192.
AppUnknown	Qualsiasi tipo di errore non compreso nei precedenti tipi secondari. Se è presente questo tipo secondario di eccezione, l'operazione dell'applicazione richiesta nella chiamata di servizio potrebbe essere terminata o non terminata.

Per ulteriori informazioni, consultare "Gestione di eccezioni di chiamate di servizio particolari" a pagina 191.

Importante: I tipi secondari di eccezione AppTimeOut, AppLogOnFailure, AppRetrieveByContent, AppMultipleHits e AppUnknown corrispondono a valori di stato in uscita, che l'adattatore può restituire per indicare la causa dell'errore. Gli adattatori più vecchi potrebbero non supportare tutti i corrispondenti valori di stato in uscita. Assicurarsi che sia eseguita una verifica di tutti gli adattatori collegati alla collaborazione con lo strumento Test Connector per determinare i valori di stato in uscita effettivamente restituiti.

Esempi

Questa sezione fornisce esempi di recupero dei tipi secondari di eccezione per i seguenti tipi di eccezione:

- JavaException

  Il codice del seguente esempio recupera un'eccezione Java generata da una funzione matematica.

```
//
// If the preceding division operation threw an exception,
// set the result to 0.
//
if (currentException.getType().equals("JavaException"))
   {
   String subType = currentException.getSubType();
   logWarning("Caught exception " + subType +
      ".  Setting result to 0.");
   result = 0;
   }
```

- ServiceCallException

Per esempi della gestione dei due tipi secondari di ServiceCallException, consultare le sezioni specificate:

- Per `AppRequestNotYetSent`, consultare "Richieste di chiamate di servizio non inviate" a pagina 194.
- Per `ServiceCallTransportException` , consultare "Gestione delle eccezioni runtime correlate al trasporto" a pagina 192.

getType()

Recupera il tipo di eccezione di collaborazione dall'oggetto di eccezione.

Sintassi

```
String getType()
```

Valori di ritorno

Una `String` che contiene il tipo di eccezione per l'eccezione corrente. Confrontare il valore di questa stringa con una delle seguenti variabili statiche di tipo eccezione:

`AnyException`	Qualsiasi tipo di eccezione. Se sono presenti due collegamenti di eccezione, uno che verifica il tipo di eccezione specifico ed uno che verifica `AnyException`, il collegamento che verifica il tipo di eccezione specifico viene controllato per primo. Se l'eccezione corrente non corrisponde all'eccezione specifica, successivamente viene elaborato il collegamento che verifica `AnyException`.
`AttributeException`	Problema di accesso all'attributo. Ad esempio, la collaborazione ha chiamato `getDouble()` per un attributo String o `getString()` per un attributo non esistente.
`JavaException`	Problema con il codice Java nella logica di collaborazione.
`ObjectException`	L'oggetto business passato ad un metodo non era valido o è stato effettuato l'accesso ad un oggetto null.
`OperationException`	La chiamata di servizio è stata impostata in modo non corretto e non ha potuto essere inviata.
`ServiceCallException`	Chiamata di servizio non riuscita. Ad esempio, un connettore o un'applicazione non sono disponibili.
`SystemException`	Errore interno InterChange Server Express.
`TransactionException`	Errore relativo al comportamento di transazione per una collaborazione transazionale. Ad esempio, il rollback non è riuscito o la collaborazione non ha potuto applicare alcuna compensazione.

Note

Il metodo getType() recupera il tipo di eccezione per l'eccezione corrente. Il tipo di eccezione è una `String` che identifica la causa dell'eccezione.

Esempi

Il seguente esempio recupera il tipo di eccezione di collaborazione ed lo utilizza in una chiamata al metodo raiseException().

```
String problem currentException.getType();
raiseException(problem, 1234);
```

toString()

Formatta le informazioni di eccezione, inclusi il tipo di eccezione e il testo, in una stringa.

Sintassi

```
String toString()
```

Parametri

exception La variabile che contiene un oggetto eccezione.

Note

Il metodo toString() formatta le informazioni di eccezione per l'eccezione corrente come indicato di seguito:

exceptionType: *messageText*

Nella riga precedente, *exceptionType* è il tipo di eccezione dell'oggetto eccezione e *messageText* è il relativo testo dell'eccezione.

Nota: E' possibile utilizzare il metodo getMessage() per recuperare solo il testo dell'eccezione dall'eccezione corrente.

Esempi

Il seguente esempio scrive le informazioni sull'eccezione corrente nella variabile String newmessage:

```
String newmessage = currentException.toString();
```

Metodi obsoleti

Nella Tabella 94 sono riportati i metodi obsoleti disponibili nelle precedenti versioni di Process Designer Express. Se in precedenza non è mai stato utilizzato Process Designer Express, ignorare questa sezione. Per una spiegazione dei metodi obsoleti, consultare "Metodi obsoleti" a pagina 408.

Tabella 94. Metodi obsoleti, classe exception

Metodo precedente	Metodo sostitutivo
getText()	toString()

Capitolo 30. Pacchetto EventManagement

Le classi illustrate in questo capitolo creano istanze di oggetti nel pacchetto EventManagement. Questi oggetti sono vettori utilizzati dai metodi della classe BaseCollaboration.

La Tabella 95 elenca le classi del pacchetto EventManagement descritte in questo capitolo.

Tabella 95. Classi del pacchetto EventManagement

Classe	Descrizione	Pagina
EventKeyAttrDef()	Crea una nuova istanza della classe EventKeyAttrDef.	455
EventQueryDef()	Crea una nuova istanza della classe EventQueryDef.	457
FailedEventInfo()	Crea una nuova istanza della classe FailedEventInfo.	459

EventKeyAttrDef()

EventKeyAttrDef()

Crea una nuova istanza del vettore EventKeyAttrDef.

Sintassi

```
EventKeyAttrDef()
```

Parametri

Nessuno.

Valori di ritorno

Restituisce un vettore con la seguente struttura:

keyName *KeyName* è il nome attributo.

keyValue *KeyValue* è il valore dell'attributo.

Esempi

```
myEventKeys new EventKeyAttrDef();
```

Consultare anche

"FailedEventInfo()" a pagina 459
"queryFailedEvents()" a pagina 380

EventQueryDef()

Costanti statiche

La classe EventQueryDef() definisce le costanti statiche mostrate nella Tabella 96.

Tabella 96. Costanti statiche nella classe EventQueryDef()

Costante	**Significato**
STATUS_ANY_UNRESOLVED	Gli eventi con un qualsiasi stato non risolto.
STATUS_FAILED	Gli eventi con uno stato non riuscito.
STATUS_DEFERRED_RECOVERY	Gli eventi con uno stato ripristino rinviato.
STATUS_SERVICECALL_IN_TRANSIT	Gli eventi con chiamate di servizio in corso.
STATUS_DELIVER_POSSIBLE_DUPLICATE	Gli eventi che possono essere duplicati.
STATUS_SERVICECALL_WAITING	Gli eventi in attesa del completamento di una chiamata di servizio.
STATUS_USER_GENERATED	Gli eventi salvati con il metodo saveFailedEvent().

EventQueryDef()

Crea un'istanza di un vettore della classe EventQueryDef.

Sintassi:

```
EventQueryDef()
```

Parametri: Nessuno.

Esempi:

```
myQueryOptions new EventQueryDef();
```

Valori di ritorno: Un vettore con la struttura:

nameConnector Nome del connettore di interesse.

nameCollaboration
Nome della collaborazione di interesse.

nameBusObj Nome dell'oggetto business di interesse.

verb Nome dell'istruzione di interesse. Se null, sono selezionate tutte le istruzioni.

Consultare anche:

"queryFailedEvents()" a pagina 380

FailedEventInfo()

Costanti statiche

La classe FailedEventInfo() definisce le costanti statiche mostrate nella Tabella 97.

Tabella 97. Costanti statiche della classe FailedEventInfo().

Costante	Significato
RESUBMIT_ORIGINAL_BUSOBJ	Invia nuovamente l'evento non riuscito con l'oggetto business originale.
RESUBMIT_NEW_BUSOBJ	Invia nuovamente l'evento non riuscito con il nuovo oggetto business.

FailedEventInfo()

Crea un'istanza vettore di FailedEventInfo.

Sintassi

```
FailedEventInfo()
```

Parametri

Nessuno

Esempi

```
myFailedEventInfo new FailedEventInfo();
```

Valori di ritorno

Restituisce un vettore con la seguente struttura:

nameOwner *NameOwner* è il nome della collaborazione a cui appartiene questo evento.

nameConnector *NameConnector* è il nome del connettore a cui appartiene questo evento.

nameBusObj *NameBusObj* è il nome dell'oggetto business utilizzato in questo evento.

nameVerb *NameVerb* è l'istruzione in esecuzione durante questo evento.

strTime *StrTime* è l'orario dell'evento.

strMessage *strMessage* è un messaggio associato all'evento non riuscito.

wipIndex *wipIndex* è un numero intero che indica la posizione dell'evento.

EventKeyAttrDef[]
Un vettore come definito in "EventKeyAttrDef()" a pagina 456.

nkeys *NKeys* è il numero di attributi chiave nel vettore **EventKeyAttrDef[]**.

eventStatus *EventStatus* è un numero intero che indica lo stato dell'evento, che può essere una delle costanti statiche definite dalla classe EventQueryDef() in "Costanti statiche" a pagina 457.

expirationTime *ExpirationTime* è l'orario in cui si è verificato l'evento.

scenarioName *ScenarioName* è il nome dello scenario al momento dell'evento.

scenarioState *ScenarioState* è lo stato di completamento dello scenario in esecuzione.

Consultare anche

"EventKeyAttrDef()" a pagina 455
"EventKeyAttrDef()" a pagina 455
"queryFailedEvents()" a pagina 380

Capitolo 31. Classe Filter

I metodi documentati in questo capitolo agiscono su oggetti della classe Filter. Il filtro dei dati è un'attività comune nel processo di collaborazione; questi metodi filtrano i dati contenuti negli attributi specificati di un oggetto business. I risultati possono essere utilizzati per determinare se la collaborazione deve sincronizzare un oggetto business con dati specifici.

Nella Tabella 98 sono riportati i metodi descritti in questo capitolo.

Tabella 98. Riepilogo dei metodi Filter

Metodo	Descrizione	Pagina
Filter()	Crea una nuova istanza della classe Filter.	462
filterExcludes()	Determina se il valore di attributo fornito è uguale al valore o ai valori di esclusione.	463
filterIncludes()	Determina se il valore di attributo fornito è uguale al valore o ai valori di inclusione.	464
recurseFilter()	Determina se il valore di attributo fornito è uguale al valore o ai valori di inclusione o di esclusione.	465
recursePreReqs()	Individua in modo ricorsivo la posizione del vettore di un tipo di oggetto business specificato in un vettore fornito di oggetti business univoci.	466

Filter()

Crea una nuova istanza di Filter.

Sintassi

```
Filter(BaseCollaboration baseCollab)
```

Parametri

baseCollab
Specifica l'istanza di collaborazione corrente.

Valori di ritorno

Restituisce una nuova istanza di oggetto Filter.

filterExcludes()

Determina se il valore dell'attributo specificato è uguale a quello dei valori di esclusione.

Sintassi

```
boolean filterExcludes(String FilterAttributeValue,
                       String ExcludeValues)
```

Parametri

FilterAttributeValue
Il valore dell'attributo da filtrare.

ExcludesValues
I valori utilizzati dalla collaborazione come filtro per prevenire la sincronizzazione dell'oggetto business.

Valori di ritorno

Restituisce `False` se il valore di *FilterAttributeValue* corrisponde ad uno dei valori elencati in *ExcludesValues*. Altrimenti, il metodo restituisce `True`.

filterIncludes()

Determina se il valore dell'attributo specificato è uguale a quello dei valori di inclusione.

Sintassi

```
boolean filterIncludes(String FilterAttributeValue, String IncludeValues)
```

Parametri

FilterAttributeValue
Il valore dell'attributo da filtrare.

IncludesValues
I valori utilizzati dalla collaborazione come filtro per consentire la sincronizzazione dell'oggetto business.

Valori di ritorno

Restituisce `True` se il valore di *FilterAttributeValue* corrisponde ad uno dei valori elencati in *IncludesValues*. Altrimenti, il metodo restituisce `False`.

recurseFilter()

Determina se il valore dell'attributo specificato è uguale a quello dei valori di inclusione o esclusione.

Sintassi

```
boolean recurseFilter(BusObj busObj,
                      String filterAttribute,
                      boolean stopOnFail,
                      String includeValues,
                      String excludeValues)
```

Parametri

busObj L'istanza dell'oggetto business su cui eseguire il filtro.

filterAttribute
Il nome dell'attributo dell'oggetto business utilizzato quando si confrontano i valori specificati da *includeValues* e *excludeValues*. La collaborazione confronta il valore dell'attributo di filtro con i valori di inclusione o esclusione specificati per prevenire o abilitare la sincronizzazione dell'oggetto business.

stopOnFail
Specifica come gestire il valore dell'attributo *filterAttribute* se non sono soddisfatti i criteri di filtro.

includesValues
I valori utilizzati dalla collaborazione come filtro per consentire la sincronizzazione dell'oggetto business.

excludesValues
I valori utilizzati dalla collaborazione come filtro per prevenire la sincronizzazione dell'oggetto business.

Valori di ritorno

Restituisce `True` se *filterAttribute* contiene un valore specificato in *includesValues* o un valore non specificato in *excludesValues*. Altrimenti, il metodo restituisce `False`.

Eccezioni

CollaborationException — Questa eccezione viene generata se il valore dell'attributo *filterAttribute* non viene specificato come valore incluso ma come escluso, e se il parametro *stopOnFail* è impostato su `True`.

recursePreReqs()

Individua in modo ricorsivo la posizione del vettore di un tipo di oggetto business specificato in un vettore fornito di oggetti business univoci.

Sintassi

```
int recursePreReqs(String Type, Vector busObjs)
```

Parametri

Type Il tipo di oggetto business da ricercare nel vettore *busObj*.

busObjs
Il vettore di oggetti business dei tipi di oggetti business univoci.

Valori di ritorno

Restituisce la posizione nel vettore *busObjs* che contiene l'oggetto business specificato da *Type*.

Eccezioni

CollaborationException — Questa eccezione viene generata se la proprietà di configurazione della collaborazione PREQ_*Type* è mancante.

Capitolo 32. Classe Globals

I metodi documentati in questo capitolo agiscono su oggetti della classe Globals.

La classe Globals gestisce una tabella hash globale per supportare l'elaborazione di eventi asincroni. Nelle precedenti versioni di InterChange Server Express, una collaborazione comunicava con altre collaborazioni e connettori esclusivamente mediante chiamate di servizio sincrone. Con l'introduzione della classe Globals, le collaborazioni possono ora inviare o ricevere un comunicazione da un'altra collaborazione o connettore, attendere il tempo specificato per la risposta appropriata, e poi continuare ad elaborare l'evento nello stesso thread.

Nella Tabella 99 sono riportati i metodi forniti nella classe Globals.

Tabella 99. Riepilogo dei metodi della classe Globals

Metodo	Descrizione	Pagina
Globals()	Crea una nuova istanza della classe Globals.	468
callMap()	Fornisce un wrapper per l'API DtpMapService.runMap in modo da rendere più facile il richiamo di una mappa da una collaborazione.	469

Globals()

Crea una nuova istanza della classe Globals.

Sintassi

```
Globals(BaseCollaboration baseCollab)
```

Parametri

baseCollab
Specifica l'istanza di collaborazione corrente.

Valori di ritorno

Una nuova istanza di oggetto Globals.

callMap()

Fornisce un wrapper per l'API DtpMapService.runMap in modo da rendere più facile il richiamo di una mappa da una collaborazione.

Sintassi

```
BusObj callMap(String mapName, BusObj srcBusObj)
```

Parametri

mapName
Specifica il nome della mappa da eseguire.

srcBusObj
Specifica l'oggetto business di origine per la mappa.

Valori di ritorno

Restituisce l'oggetto business di destinazione per la mappa specificata da *mapName*.

Eccezioni

CollaborationException — Generata se si verifica un errore durante il tentativo di eseguire la mappa *mapName*.

Capitolo 33. Classe SmartCollabService

I metodi documentati in questo capitolo agiscono su oggetti della classe SmartCollabService. Questa classe fornisce un insieme di metodi per semplificare la suddivisione, l'unione e l'aggregazione di attributi di vettore in un oggetto business.

Nella Tabella 100 sono riportati i metodi forniti nella classe SmartCollabService.

Tabella 100. Riepilogo dei metodi SmartCollabService

Metodo	Descrizione	Pagina
SmartCollabService()	Crea una nuova istanza della classe SmartCollabService.	471
doAgg()	Aggrega attributi di contenitore simili in un unico attributo di contenitore, in base a criteri specificati dall'utente e attributi.	472
doMergeHash()	Riceve una raccolta di oggetti business e li raggruppa in un nuovo oggetto business principale specificato dal livello di divisione. Gli oggetti business sono raggruppati in base al contenuto simile specificato nell'attributo o negli attributi chiave.	472
doRecursiveAgg()	Aggrega in modo ricorsivo attributi di contenitore gerarchicamente simili in un unico attributo di contenitore, in base a criteri specificati dall'utente e attributi.	473
doRecursiveSplit()	Recupera gli oggetti business del contenitore da un particolare livello di una gerarchia di oggetti business, e facoltativamente li restituisce nell'oggetto business di livello superiore.	473
getKeyValues()	Calcola il valore chiave per l'oggetto business da utilizzare in una tabella hash, in base ai valori separati da virgola specificati nell'attributo o negli attributi chiave.	474
merge()	Unisce una raccolta di oggetti business in un unico oggetto business di livello superiore.	475
split()	Suddivide un oggetto business in oggetti business di contenitore, come specificato dal livello di divisione.	475

SmartCollabService()

Crea una nuova istanza di SmartCollabService.

Sintassi

```
SmartCollabService()
SmartCollabService(com.crossworlds.BaseCollaboration baseCollab)
```

Parametri

baseCollab
Specifica l'istanza di collaborazione corrente.

Valori di ritorno

Restituisce una nuova istanza di oggetto SmartCollabService.

doAgg()

Aggrega attributi di contenitore simili in un unico attributo di contenitore in base ai criteri specificati dall'utente e all'elenco di attributi da aggregare.

Sintassi

```
BusObj doAgg(BusObj inBusObj,
          String Level,
          String KeyAttr,
          String Attr)
```

Parametri

inBusObj
Specifica l'oggetto business da aggregare.

Level Specifica il livello di aggregazione, che comprende il tipo di oggetto business (la relativa definizione di oggetto business) e il nome dell'attributo di vettore da aggregare. I nomi sono delimitati da punti (.).

KeyAttr
Specifica gli attributi chiave dell'oggetto business utilizzato per l'aggregazione.

Attr Specifica gli attributi da aggregare. Più nomi sono delimitati da virgole (,).

Valori di ritorno

Restituisce l'oggetto business aggregato.

Eccezioni

CollaborationException — Generata se si verifica un errore durante l'aggregazione degli attributi dell'oggetto business.

doMergeHash()

Raggruppa una raccolta di oggetti business in un nuovo oggetto business principale, come specificato dal livello di divisione. Gli oggetti business sono raggruppati in base al contenuto simile nell'attributo o negli attributi chiave.

Sintassi

```
java.util.Vector doMergeHash(java.util.Vector BusObj,
          String Level,
          String KeyAttr)
```

Parametri

BusObj
Specifica un vettore che contiene la raccolta di oggetti business da unire.

Level Specifica il livello di unione, che comprende il tipo di oggetto business principale (la relativa definizione di oggetto business) e l'attributo destinato a contenere l'oggetto business secondario. I nomi sono delimitati da punti (.).

KeyAttr
Specifica gli attributi dell'oggetto business utilizzati come criteri di unione. I valori sono delimitati da virgole (,).

Valori di ritorno

Restituisce il vettore degli oggetti business uniti.

Eccezioni

CollaborationException — Generata se si verifica un errore durante l'unione degli oggetti business.

---

doRecursiveAgg()

Aggrega in modo ricorsivo attributi di vettore gerarchicamente simili in un unico attributo di contenitore in base ai criteri specificati dall'utente e all'elenco di attributi da aggregare.

Sintassi

```
BusObj doRecursiveAgg(BusObj inBusObj,
          String Level,
          String KeyAttr,
          String Attr)
```

Parametri

inBusObj
Specifica l'oggetto business da aggregare.

Level Specifica il livello di aggregazione, che comprende il tipo di oggetto business (la relativa definizione di oggetto business) e il nome dell'attributo di vettore da aggregare. I nomi sono delimitati da punti (.).

KeyAttr
Specifica gli attributi chiave dell'oggetto business utilizzato per l'aggregazione.

Attr Specifica gli attributi da aggregare. Più nomi sono delimitati da virgole (,).

Valori di ritorno

Restituisce l'oggetto business aggregato.

Eccezioni

CollaborationException — Generata se si verifica un errore durante l'aggregazione degli attributi dell'oggetto business.

---

doRecursiveSplit()

Recupera gli oggetti business del contenitore da un livello specificato nella gerarchia di oggetti business.

Sintassi

```
java.util.Vector doRecursiveSplit(BusObj inBusObj,
          String Level)
java.util.Vector doRecursiveSplit(BusObj inBusObj,
          String Level,
          boolean KeepParents)
```

Parametri

inBusObj
Specifica l'oggetto business di livello superiore i cui attributi di vettore sono suddivisi dal metodo.

Level Specifica il percorso dell'attributo di vettore in base al quale suddividere l'oggetto business. I valori sono delimitati da punti (.).

KeepParents
Specifica se l'oggetto business suddiviso viene restituito in un oggetto business principale o come oggetto indipendente. Impostare il parametro su `True` per restituire l'oggetto business suddiviso nell'oggetto business principale.

Valori di ritorno

Restituisce un vettore di oggetti business.

Eccezioni

CollaborationException — Generata se si verifica un errore durante la suddivisione degli oggetti business.

---

getKeyValues()

Calcola il valore chiave di un oggetto business da utilizzare in una tabella hash.

Sintassi

```
String getKeyValues(BusObj inBusObj,
               String KeyAttr)
```

Parametri

inBusObj
Specifica l'oggetto business per il quale viene calcolato il valore chiave.

KeyAttr
Specifica il nome dell'attributo su cui agisce il metodo. Più valori sono delimitati da virgole (,).

Valori di ritorno

Restituisce il valore chiave da utilizzare in una tabella hash Java.

Eccezioni

CollaborationException — Generata se si verifica un errore durante il calcolo del valore chiave degli oggetti business.

merge()

Unisce una raccolta di oggetti business in un unico oggetto business di livello superiore.

Sintassi

```
BusObj merge(java.util.Vector BusObjs,
          String Level)
BusObj merge(java.util.Vector BusObjs,
          String Attr,
          BusObj mergeBusObj)
```

Parametri

BusObjs
Specifica la raccolta di oggetti business secondari da unire.

Livello Specifica il livello di unione, che comprende il tipo di oggetto business e il nome del relativo attributo di vettore i cui oggetti business secondari devono essere uniti. I nomi sono delimitati da punti (.).

Attr Specifica il nome dell'attributo di vettore nell'oggetto business *mergeBusObj* nel quale devono essere uniti gli oggetti business secondari.

mergeBusObj
Specifica l'oggetto business di livello superiore che conterrà la raccolta unita di oggetti business secondari.

Valori di ritorno

Restituisce un oggetto business di livello superiore (nuovo o specificato) che contiene la raccolta unita di oggetti business secondari.

Eccezioni

CollaborationException — Generata se si verifica un errore durante l'unione degli oggetti business.

split()

Suddivide un oggetto business nel numero di oggetti business di contenitore specificato dal livello di divisione.

Sintassi

```
Vector split(BusObj inBusObj,
          String Attr)
```

Parametri

inBusObj
Specifica l'oggetto business di livello principale su cui agisce il metodo split().

Attr Specifica il nome dell'attributo di vettore in base al quale suddividere l'oggetto business.

Valori di ritorno

Restituisce un vettore di oggetti business, un oggetto business per ciascun oggetto business secondario nell'attributo di vettore dell'oggetto business principale.

Eccezioni

CollaborationException — Generata se si verifica un errore durante la suddivisione degli oggetti business.

Appendice. Informazioni sulle collaborazioni standard

Dal momento che molte delle collaborazioni sviluppate da IBM ereditano i processi business dalla maschera CollaborationFoundation, questa appendice illustra i processi business comuni a tutte le collaborazioni. La documentazione utente relativa alla collaborazione fa riferimento a questo documento per una spiegazione di tali processi. Per informazioni su CollaborationFoundation, consultare "Maschera CollaborationFoundation" a pagina 30.

Questa appendice descrive le linee guida utili per la realizzazione di collaborazioni riusabili con comportamento affidabile. Sono illustrati i seguenti argomenti:

Processi standard per le maschere di collaborazione

Questa sezione descrive il flusso dei processi business di una collaborazione InterChange Server Express tipica. I processi standard sono costituiti da:

- Processo di recupero
- Processo USE_RETRIEVE
- Processo di filtro
- Processo ADDITIONAL_RETRIEVE
- Processo e-mail per la gestione degli errori

Il flusso dei processi business può variare in base all'istruzione di attivazione ed ai valori delle proprietà di configurazione dell'oggetto di collaborazione.

Processo di recupero

Una collaborazione può ricevere un oggetto business di attivazione che utilizza l'istruzione Retrieve. Si assume che l'oggetto business di un'applicazione di origine utilizzi tale istruzione quando i suoi attributi contengono valori chiave con riferimenti incrociati. La mappa dell'oggetto business e il connettore dell'applicazione di destinazione utilizzano la chiave primaria per restituire l'oggetto business dall'applicazione di destinazione.

La Figura 91 a pagina 478 illustra un processo di collaborazione per il recupero di oggetti business quando l'istruzione di attivazione è Retrieve.

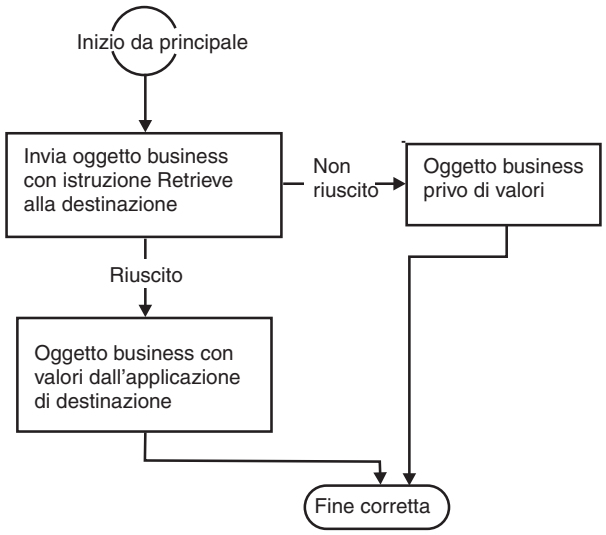


Figura 91. Processo di recupero

Processo Use_Retrieve

La Figura 92 a pagina 479 illustra il processo Use_Retrieve di una collaborazione quando la proprietà di configurazione USE_RETRIEVE assume il valore true. Per informazioni sulla proprietà di configurazione, consultare la relativa voce nella Tabella 102 a pagina 484

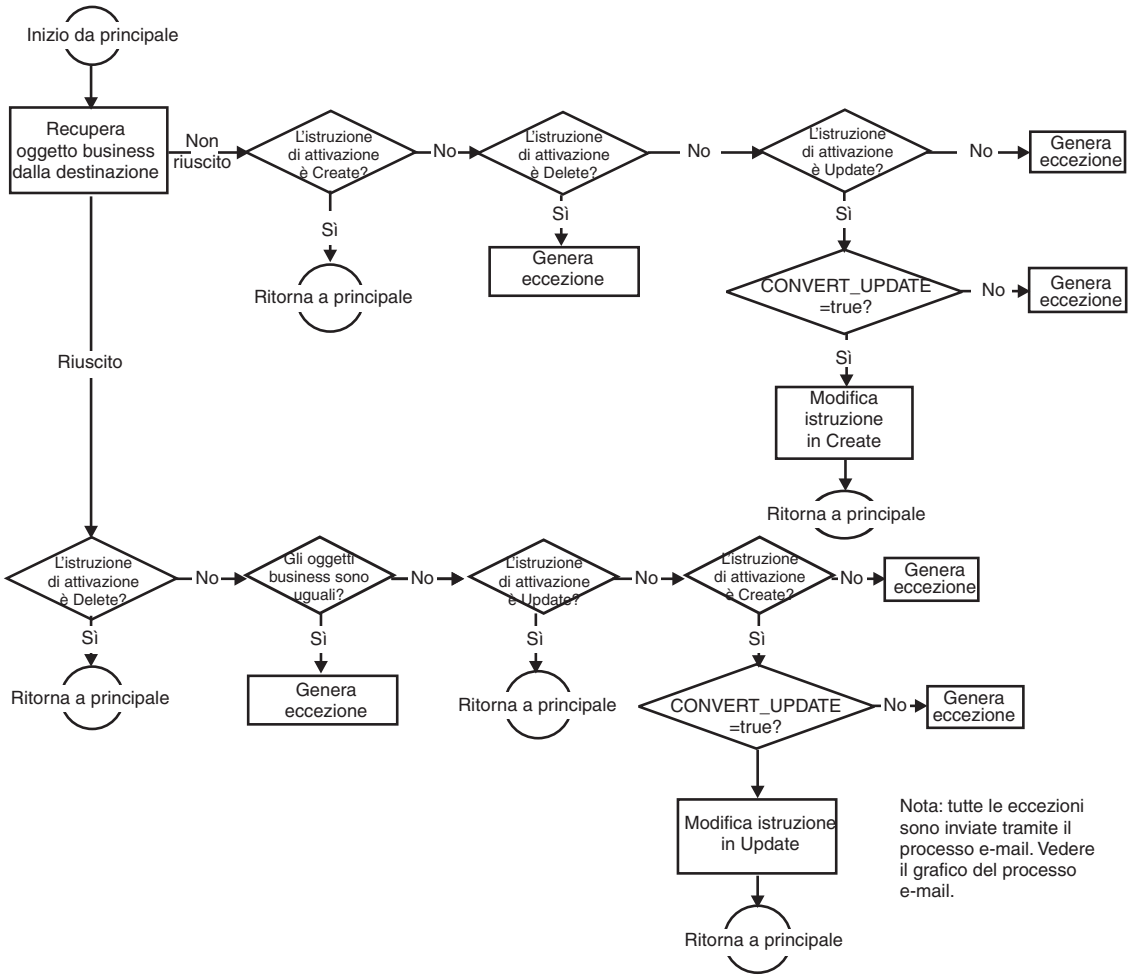


Figura 92. Processo Use_Retrieve

La Tabella 101 a pagina 480 illustra la relazione tra l'impostazione della proprietà USE_RETRIEVE e della proprietà CONVERT_CREATE quando l'istruzione di attivazione è Create. La relazione è simile a quella tra le proprietà USE_RETRIEVE e CONVERT_UPDATE.

Tabella 101. Relazione tra le proprietà USE_RETRIEVE e CONVERT_CREATE

Valore di USE_RETRIEVE	**Valore di CONVERT_CREATE**	**Azione**
true	true	Recupera l'oggetto business dall'applicazione di destinazione. Se l'applicazione di destinazione non restituisce un oggetto business, invia l'oggetto business con l'istruzione Create all'applicazione di destinazione. Se l'applicazione di destinazione restituisce un oggetto business, verifica se gli oggetti business di origine e di destinazione sono identici: • Se sono identici, genera un'eccezione che indica che gli oggetti business sono identici ed arresta l'elaborazione. • Se non sono identici, converte l'istruzione Create in Update ed invia l'oggetto business alla destinazione.
false	true	Invia l'oggetto business con l'istruzione Create all'applicazione di destinazione. • Se l'applicazione di destinazione non riesce a creare l'oggetto business, converte l'istruzione Create in Update ed invia di nuovo l'oggetto business. • Se l'applicazione di destinazione riesce a creare l'oggetto business, l'elaborazione continua.

Tabella 101. Relazione tra le proprietà USE_RETRIEVE e CONVERT_CREATE (Continua)

Valore di USE_RETRIEVE	Valore di CONVERT_CREATE	Azione
false	false	Invia l'oggetto business con l'istruzione Create all'applicazione di destinazione. • Se l'applicazione di destinazione non riesce a creare l'oggetto business, genera un'eccezione ed arresta l'elaborazione. • Se l'applicazione di destinazione riesce a creare l'oggetto business, l'elaborazione continua.
true	false	Recupera l'oggetto business dall'applicazione di destinazione. Se l'applicazione di destinazione non restituisce un oggetto business, invia l'oggetto business con l'istruzione Create all'applicazione di destinazione. Se l'applicazione di destinazione restituisce un oggetto business, determina se gli oggetti business di origine e di destinazione sono identici. • Se sono identici, genera un'eccezione che indica che il valore della proprietà CONVERT_CREATE è false ed arresta l'elaborazione. • Se sono identici, genera un'eccezione ed arresta l'elaborazione.

Processo di filtro

La Figura 93 a pagina 482 illustra un processo di collaborazione per filtrare gli oggetti business in base ai valori degli attributi specificati.

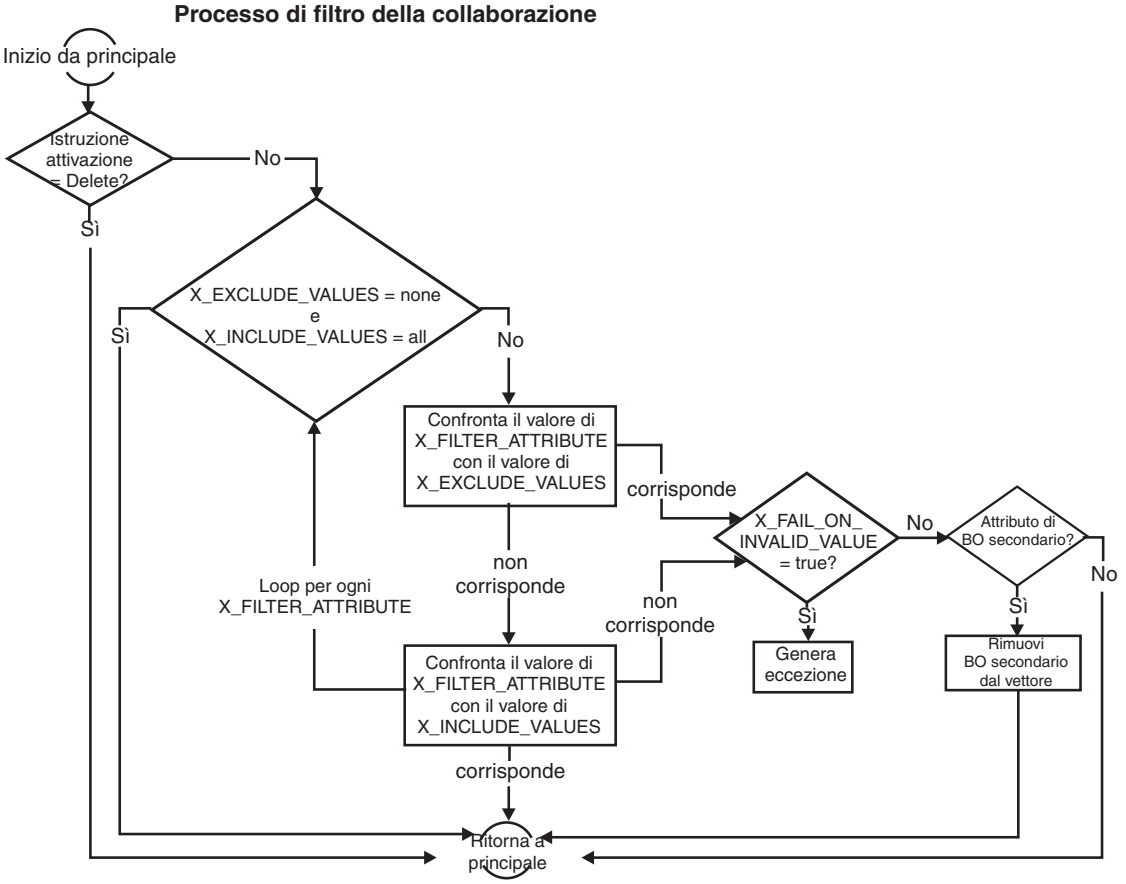


Figura 93. Processo di filtro

Il diagramma sopra riportato fa riferimento al filtro delle proprietà di configurazione con "X_" invece di "1_" poiché è possibile aggiungere diverse proprietà di configurazione di filtro alla collaborazione allo scopo di applicare un filtro su più attributi. Per ulteriori informazioni sull'aggiunta di proprietà di filtro, consultare "Filtraggio dei dati nell'oggetto business di destinazione" a pagina 32 e la Tabella 102 a pagina 484.

Per comprendere meglio il comportamento della collaborazione quando 1_FAIL_ON_INVALID_VALUE presenta il valore false, si supponga che un oggetto business a livello principale denominato BusObjA contenga un vettore di oggetti business secondari denominato BusObjB. Si supponga inoltre che ogni oggetto business secondario di BusObjB contenga un vettore di oggetti business secondari denominato BusObjC. Per filtrare in base ad un attributo denominato Type nell'oggetto business BusObjB, impostare il valore di 1_FILTER_ATTRIBUTE a BusObjA.BusObjB.Type. Per escludere tutti gli oggetti business il cui attributo Type contiene il valore "Non-standard", impostare il valore della proprietà 1_EXCLUDE_VALUES a "Non-standard".

Ora si supponga che uno degli oggetti business nel vettore di BusObjB presenti il valore "Non-standard" nel proprio attributo Type e che 1_FAIL_ON_INVALID_VALUE sia false. La collaborazione elimina questo oggetto business e i relativi oggetti business secondari dall'elaborazione.

Nota: Se l'attributo che non corrisponde al filtro si trova nell'oggetto business di livello principale, la collaborazione continua l'elaborazione per l'intero oggetto business.

Processo Additional_Retrieve

Se la proprietà di configurazione ADDITIONAL_RETRIEVE assume il valore true, la collaborazione recupera l'oggetto business dall'applicazione di destinazione dopo aver sincronizzato i dati correttamente. Questa proprietà è utile quando l'applicazione di origine richiede che l'applicazione di destinazione restituisca un oggetto business con valori completi, ma il connettore dell'applicazione di destinazione non restituisce un oggetto business completo dopo la creazione o l'aggiornamento dei dati. Per ulteriori informazioni sulla proprietà di configurazione ADDITIONAL_RETRIEVE, consultare "Proprietà standard per le maschere di collaborazione" a pagina 484.

La Figura 94 illustra il processo Additional_Retrieve di una collaborazione.

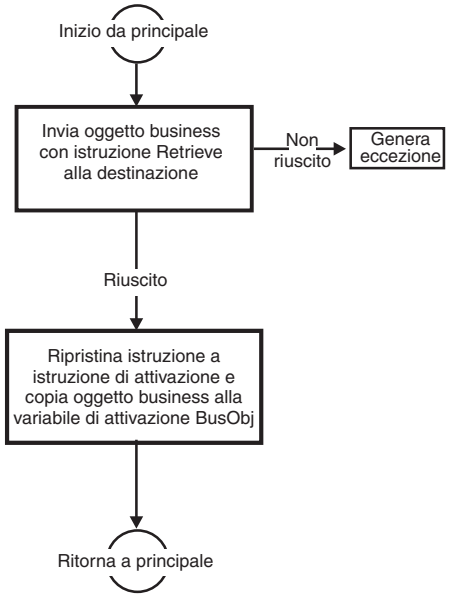


Figura 94. Processo Additional_Retrieve

Processo e-mail per la gestione degli errori

L'impostazione delle proprietà SEND_EMAIL e INFORMATIONAL_EXCEPTIONS di una collaborazione determina uno dei seguenti comportamenti della collaborazione:

- Invio di e-mail ad un indirizzo specificato quando si verifica un errore
- Chiusura con errore quando si verifica una particolare eccezione

La Figura 95 a pagina 484 illustra il processo di invio di e-mail di una collaborazione e la determinazione delle eccezioni che possono causare la fine con errore di una collaborazione.

Fine corretta

Figura 95. Processo e-mail per la gestione degli errori

Proprietà standard per le maschere di collaborazione

Nella Tabella 102 sono riportate le proprietà di configurazione standard per le maschere di collaborazione. Queste proprietà si basano sulla maschera CollaborationFoundation.

Tabella 102. Proprietà standard per le maschere di collaborazione

Proprietà di configurazione standard	Descrizione
1_EXCLUDE_VALUES	Specifica i valori utilizzati dalla collaborazione come filtro per prevenire la sincronizzazione dell'oggetto business di attivazione.
1_FAIL_ON_INVALID_VALUE	Individua il comportamento della collaborazione nel caso in cui il valore dell'attributo specificato in 1_FILTER_ATTRIBUTE non corrisponde ai criteri di filtro.
1_FILTER_ATTRIBUTE	Specifica l'attributo di oggetto business utilizzato dalla collaborazione per confrontare i valori specificati nelle proprietà 1_EXCLUDE_VALUES e 1_INCLUDE_VALUES.
1_INCLUDE_VALUES	Specifica i valori utilizzati dalla collaborazione come filtro per consentire la sincronizzazione dell'oggetto business di attivazione.
ADDITIONAL_RETRIEVE	Indica se la collaborazione recupera l'oggetto business dall'applicazione di destinazione dopo aver sincronizzato i dati correttamente.
CONVERT_CREATE	Individua il comportamento della collaborazione nel caso in cui, a fronte di una richiesta Create, l'oggetto business di attivazione creato nell'applicazione di origine esiste già nell'applicazione di destinazione.
CONVERT_UPDATE	Individua il comportamento della collaborazione nel caso in cui, a fronte di una richiesta Update, l'oggetto business di attivazione aggiornato nell'applicazione di origine non esiste nell'applicazione di destinazione.
INFORMATIONAL_EXCEPTIONS	Specifica il comportamento della collaborazione in caso di eccezione.
SEND_EMAIL	Indica se la collaborazione invia e-mail quando riceve un'eccezione.

Tabella 102. Proprietà standard per le maschere di collaborazione (Continua)

Proprietà di configurazione standard	Descrizione
USE_RETRIEVE	Indica se la collaborazione recupera l'oggetto business di attivazione dall'applicazione di destinazione prima di sincronizzare i dati.

1_EXCLUDE_VALUES

La proprietà di configurazione della collaborazione 1_EXCLUDE_VALUES specifica i valori utilizzati dalla collaborazione come filtro per prevenire la sincronizzazione dell'oggetto business di attivazione. La collaborazione esclude un oggetto business dalla sincronizzazione se il valore dell'attributo specificato da 1_FILTER_ATTRIBUTE corrisponde ad uno dei valori elencati in questa proprietà.

I valori ammessi per 1_EXCLUDE_VALUES sono i seguenti:

- Un elenco delimitato da virgole di valori da escludere (ad esempio, ″in-stock, at-customer″).
- Il valore `none`, che indica che nessun oggetto deve essere escluso dall'elaborazione.
- Nessun valore, utile per escludere un attributo il cui valore è null (CxIgnore) o una stringa vuota (CxBlank).

Il valore predefinito è `none`.

E' possibile eliminare questa proprietà dall'elenco delle proprietà di configurazione disponibili per la maschera di collaborazione. Eliminando la proprietà 1_EXCLUDE_VALUES si indica che la proprietà non viene utilizzata nel processo di filtro.

Notare che la collaborazione non effettua la distinzione maiuscolo/minuscolo dei valori specificati, né considera la presenza di spazi aggiuntivi nell'elenco dei valori.

Specificare i valori per questa proprietà se l'elenco dei valori di esclusione è *più breve* dell'elenco dei valori di inclusione. In caso contrario, specificare i valori per 1_INCLUDE_VALUES. Se si specificano i valori in 1_INCLUDE_VALUES, impostare il valore di 1_EXCLUDE_VALUES su none.

1_FAIL_ON_INVALID_VALUE

La proprietà di configurazione della collaborazione 1_FAIL_ON_INVALID_VALUE individua il comportamento della collaborazione nel caso in cui il valore dell'attributo specificato in 1_FILTER_ATTRIBUTE non corrisponde ai criteri di filtro. Impostare la proprietà 1_FAIL_ON_INVALID_VALUE ad uno dei seguenti valori:

- true — Impostare su true se si vuole che la collaborazione generi un'eccezione ed arresti l'elaborazione se il valore dell'attributo specificato in 1_FILTER_ATTRIBUTE non corrisponde ai criteri di filtro specificati in 1_EXCLUDE_VALUES o 1_INCLUDE_VALUES.
- false — Impostare su false se si vuole che la collaborazione esegua uno dei seguenti processi quando il valore dell'attributo specificato non soddisfa i criteri di filtro:
  - Se l'attributo specificato in 1_FILTER_ATTRIBUTE fa parte dell'oggetto business principale, la collaborazione continua l'elaborazione.

– Se l'attributo specificato in 1_FILTER_ATTRIBUTE fa parte di un oggetto business secondario, la collaborazione rimuove l'oggetto business secondario che contiene l'attributo e tutti i suoi oggetti secondari; la collaborazione continua l'elaborazione.

Il valore predefinito è true. Ai valori non si applica la distinzione maiuscolo/minuscolo.

E' possibile eliminare questa proprietà dall'elenco delle proprietà di configurazione disponibili per la maschera di collaborazione. Eliminando la proprietà si indica che questa proprietà non verrà utilizzata nel processo di filtro.

1_FILTER_ATTRIBUTE

La proprietà di configurazione della collaborazione 1_FILTER_ATTRIBUTE specifica l'attributo di oggetto business utilizzato dalla collaborazione per confrontare i valori specificati nelle proprietà 1_EXCLUDE_VALUES e 1_INCLUDE_VALUES. La collaborazione confronta il valore dell'attributo specificato con i valori specificati per l'esclusione o inclusione, per prevenire o consentire la sincronizzazione degli oggetti business con i valori specifici. Ad esempio, si supponga che un oggetto business a livello principale denominato BusObjA contenga un oggetto business secondario denominato BusObjB. Per filtrare in base ad un attributo denominato Type nell'oggetto business BusObjB, impostare il valore di 1_FILTER_ATTRIBUTE a BusObjA.BusObjB.Type.

Nota: La collaborazione valuta questa proprietà solo se 1_EXCLUDE_VALUES presenta un valore diverso da none o 1_INCLUDE_VALUES presenta un valore diverso da all.

I valori validi per questa proprietà di configurazione della collaborazione sono i seguenti:

- Il nome esatto dell'oggetto business e il relativo attributo, come definiti in InterChange Server Express (ad esempio, InstallShipment.ShipmentType). Per determinare il nome dell'attributo, eseguire una delle seguenti operazioni:
  - Visualizzare l'oggetto business in System Manager.
  - Visualizzare il file del repository per l'oggetto business, che presenta il nome repository\BO_*BusinessObjectName*.txt

  Nota: In questo documento, le barre rovesciate (\) vengono utilizzare come convenzione per i percorsi di directory. Per installazioni Linux, sostituire le barre rovesciate con barre (/). Tutti i nomi di percorso sono relativi alla directory di sistema in cui è installato il prodotto.
- *BusinessObjectName*.ObjectEventId, che è l'attributo che contiene un identificativo interno univoco per ciascun oggetto business. Questo è il valore predefinito per 1_FILTER_ATTRIBUTE.

La maschera di collaborazione è progettata per filtrare qualsiasi numero di attributi di oggetti business. Per aggiungere ulteriori filtri, aggiungere un insieme di proprietà di configurazione (*X*_FILTER_ATTRIBUTE, *X*_FAIL_ON_INVALID_VALUE ed uno tra *X*_EXCLUDE_VALUES e *X*_INCLUDE_VALUES) alla collaborazione, dove *X* è un numero intero compreso tra 2 e N. Ulteriori proprietà devono essere aggiunte con numerazione consecutiva da 2 a N e devono avere esattamente i nomi sopra indicati.

1_INCLUDE_VALUES

La proprietà di configurazione della collaborazione 1_INCLUDE_VALUES specifica i valori utilizzati dalla collaborazione come filtro per consentire la sincronizzazione dell'oggetto business di attivazione. La collaborazione sincronizza gli oggetti business con i valori specificati solo se il valore dell'attributo specificato da 1_FILTER_ATTRIBUTE corrisponde ad uno dei valori elencati in questa proprietà.

I valori validi per la proprietà 1_INCLUDE_VALUES sono i seguenti:

- Un elenco delimitato da virgole di valori da includere (ad esempio, "in-stock, at-customer").
- Il valore `all`, che indica che tutti gli oggetti devono essere inclusi nell'elaborazione.

Il valore predefinito è all. Notare che ai valori non si applica la distinzione maiuscolo/minuscolo. Inoltre, la collaborazione ignora eventuali spazi presenti nell'elenco dei valori.

Se questa proprietà non deve essere utilizzata nel processo di filtro, è possibile eliminarla dall'elenco delle proprietà di configurazione disponibili per la collaborazione.

Specificare i valori per questa proprietà se l'elenco dei valori di inclusione è *più breve* dell'elenco dei valori di esclusione. In caso contrario, specificare i valori per 1_EXCLUDE_VALUES. Se si specificano i valori in 1_EXCLUDE_VALUES, impostare il valore di 1_INCLUDE_VALUES su all.

ADDITIONAL_RETRIEVE

La proprietà di configurazione della collaborazione ADDITIONAL_RETRIEVE indica se la collaborazione recupera l'oggetto business dall'applicazione di destinazione dopo aver sincronizzato i dati correttamente. Impostare la proprietà ADDITIONAL_RETRIEVE nel modo seguente:

- Impostare su true, se si vuole che la collaborazione recuperi l'oggetto business dall'applicazione di destinazione dopo aver sincronizzato i dati correttamente. Questa impostazione è utile quando l'applicazione di origine richiede che l'applicazione di destinazione restituisca un oggetto business con valori completi, ma il connettore dell'applicazione di destinazione non restituisce un oggetto business completo dopo la creazione o l'aggiornamento dei dati.
  - Se la collaborazione recupera correttamente l'oggetto business con i valori completi, lo salva nella variabile triggeringBusObj per restituirlo all'applicazione di origine. Se la collaborazione è collegata ad un connettore di origine che effettua una richiesta sincrona, i valori dell'oggetto business di attivazione sono restituiti al connettore di origine quando la collaborazione completa l'elaborazione. I valori sono restituiti come se i valori dell'oggetto business fossero stati passati per riferimento.

    Nota: Inoltre, la collaborazione reimposta l'istruzione della variabile triggeringBusObj da Retrieve all'istruzione originale. La reimpostazione dell'istruzione consente alla collaborazione di utilizzare l'istruzione di attivazione originale se richiesto dal processo secondario. Per ulteriori informazioni sul processo Additional_Retrieve, consultare "Processo Additional_Retrieve" a pagina 483.
  - Se la collaborazione non riesce a recuperare l'oggetto business con i valori completi, genera un'eccezione. La gestione delle eccezioni viene determinata dall'impostazione della proprietà INFORMATIONAL_EXCEPTIONS.

- Impostare su false, se si vuole che la collaborazione non recuperi l'oggetto business dall'applicazione di destinazione dopo aver sincronizzato i dati correttamente.

Il valore predefinito è false.

CONVERT_CREATE

La proprietà di configurazione della collaborazione CONVERT_CREATE individua il comportamento della collaborazione nel caso in cui, a fronte di una richiesta Create, l'oggetto business di attivazione creato nell'applicazione di origine esiste già nell'applicazione di destinazione. Impostare la proprietà CONVERT_CREATE nel modo seguente:

- Impostare su true se si vuole che la collaborazione converta una richiesta Create in una richiesta Update se l'oggetto business di attivazione creato nell'applicazione di origine esiste già nella destinazione.
- Impostare su false, se si vuole che la collaborazione generi un'eccezione. La gestione delle eccezioni viene determinata dall'impostazione della proprietà INFORMATIONAL_EXCEPTIONS.

Il valore predefinito è false.

Le azioni intraprese dalla collaborazione in base a CONVERT_CREATE dipendono dal valore della proprietà di configurazione della collaborazione USE_RETRIEVE.

- Se USE_RETRIEVE assume il valore true, la collaborazione valuta la proprietà CONVERT_CREATE solo se l'istruzione di attivazione è Create e la collaborazione è riuscita a recuperare l'oggetto business.
- Se USE_RETRIEVE assume il valore false, la collaborazione valuta la proprietà CONVERT_CREATE solo dopo che il tentativo di creare l'oggetto business di attivazione nella destinazione non è riuscito.

Per una tabella delle relazioni tra le proprietà di configurazione USE_RETRIEVE e CONVERT_CREATE, consultare "Processo Use_Retrieve" a pagina 478.

CONVERT_UPDATE

La proprietà di configurazione della collaborazione CONVERT_UPDATE individua il comportamento della collaborazione nel caso in cui, a fronte di una richiesta Update, l'oggetto business di attivazione aggiornato nell'applicazione di origine non esiste nell'applicazione di destinazione. Impostare la proprietà CONVERT_UPDATE nel modo seguente:

- Impostare su true se si vuole che la collaborazione converta una richiesta Update in una richiesta Create se l'oggetto business di attivazione aggiornato nell'applicazione di origine non esiste nella destinazione.
- Impostare su false, se si vuole che la collaborazione generi un'eccezione. La gestione delle eccezioni viene determinata dall'impostazione della proprietà INFORMATIONAL_EXCEPTIONS.

Il valore predefinito è false.

Le azioni intraprese dalla collaborazione in base a CONVERT_UPDATE dipendono dal valore della proprietà di configurazione della collaborazione USE_RETRIEVE:

- Se USE_RETRIEVE assume il valore true, la collaborazione valuta la proprietà CONVERT_UPDATE solo se l'istruzione di attivazione è Update e l'operazione di recupero accerta che l'oggetto business non esiste nella destinazione.

- Se USE_RETRIEVE assume il valore false, la collaborazione valuta la proprietà CONVERT_UPDATE solo dopo che il tentativo di aggiornare l'oggetto business di attivazione nella destinazione non è riuscito.

INFORMATIONAL_EXCEPTIONS

La proprietà di configurazione della collaborazione INFORMATIONAL_EXCEPTIONS indica il comportamento della collaborazione in caso di eccezione. La proprietà può assumere i seguenti valori:

- `All` — Quando si verifica un'eccezione, la collaborazione invia l'eccezione alla traccia e termina correttamente.
- `None` — Quando si verifica un'eccezione, la collaborazione genera l'eccezione e termina con un errore.
- Un elenco separato da virgole di numeri di messaggi — Quando si verifica una delle eccezioni riportate in elenco, la collaborazione invia l'eccezione alla traccia e termina correttamente. I numeri dei messaggi di eccezione corrispondono a quelli del file di messaggi della collaborazione (collaborations\messages*CollaborationName*.txt.).

Il valore predefinito è `1000, 2000, 2005, 2010, 2015, 2020, 3000, 3010, 3020`.

SEND_EMAIL

La proprietà di configurazione della collaborazione SEND_EMAIL indica se la collaborazione invia e-mail quando riceve un'eccezione. Impostare la proprietà SEND_EMAIL nel modo seguente:

- `All` — Quando si verifica un'eccezione, la collaborazione invia e-mail.
- `None` — Quando si verifica un'eccezione, la collaborazione non invia e-mail.
- Un elenco separato da virgole di numeri di messaggi — Quando si verifica una delle eccezioni riportate in elenco, la collaborazione invia e-mail. I numeri dei messaggi di eccezione corrispondono a quelli del file di messaggi della collaborazione (collaborations\messages*CollaborationName*.txt.).
- Un intervallo di numeri di messaggi — Quando si verifica una delle eccezioni comprese nell'intervallo specificato, la collaborazione invia e-mail.

Il valore predefinito è none.

Nota: Specificare l'indirizzo nel campo **Indirizzo e-mail per notifica** nella finestra di dialogo Proprietà oggetto di collaborazione in System Manager. Se per una collaborazione viene specificato un indirizzo e-mail, la collaborazione richiama la Collaborazione e-mail quando si verifica un errore. La Collaborazione e-mail può essere installata e configurata come parte di InterChange Server Express.

USE_RETRIEVE

La proprietà di configurazione della collaborazione USE_RETRIEVE indica se la collaborazione recupera l'oggetto business di attivazione dall'applicazione di destinazione prima di sincronizzare i dati. Impostare la proprietà USE_RETRIEVE nel modo seguente:

- Impostare su true, se si vuole che la collaborazione recuperi l'oggetto business di attivazione dall'applicazione di destinazione prima di sincronizzare i dati. Questa impostazione è utile quando:
  - La collaborazione è attivata da un'altra collaborazione e potrebbe essere richiesta la compensazione se una fase delle collaborazioni di gruppo non

riesce. Obbligare la collaborazione a recuperare i valori prima di modificarli, consente alla collaborazione di ripristinare i valori originali nei processi di aggiornamento o eliminazione.

  – La collaborazione è attivata da una collaborazione wrapper, che invia sempre l'oggetto business di attivazione con l'istruzione Create. Impostando questa proprietà su true impedisce alla collaborazione di terminare in errore se l'oggetto business esiste già nella destinazione. Quando il valore di USE_RETRIEVE è true, la collaborazione confronta i valori nelle applicazioni di origine e di destinazione prima di eseguire le operazioni di creazione o aggiornamento. Se gli oggetti sono identici, la collaborazione termina correttamente.

- Impostare su false, se si vuole che la collaborazione inizi l'elaborazione senza recuperare l'oggetto business di attivazione dall'applicazione di destinazione. Questa impostazione è utile quando non è importante confrontare i valori negli oggetti business di origine e di destinazione.

Il valore predefinito è false.

Nota: Per ulteriori informazioni sull'utilizzo di questa proprietà, consultare "CONVERT_CREATE" a pagina 488 e "CONVERT_UPDATE" a pagina 488.

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 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.

Glossario

azione. Un simbolo del diagramma di attività che indica una singola fase del processo business. Un nodo azione contiene un frammento di codice.

diagramma di attività. Definisce il flusso di controllo per uno scenario di collaborazione. Un diagramma di attività è costituito da un insieme di simboli che specifica le azioni richieste dal processo business e la logica che determina l'ordine di esecuzione delle azioni.

attributo. Un elemento di dati in un oggetto business.

BaseCollaboration. Una classe definita da InterChange Server Express dalla quale derivano le altre classi di collaborazione. La classe BaseCollaboration contiene i metodi per la gestione di una collaborazione.

binding . L'operazione di collegamento di un oggetto di collaborazione agli oggetti che possono fornire o ricevere oggetti business. Gli oggetti ai quali una collaborazione si collega possono essere connettori o altri oggetti di collaborazione.

interruzione. Un simbolo collocato nel diagramma di attività di un iteratore per forzare la chiusura anticipata dell'iterazione.

oggetto business. Un insieme di dati che rappresenta un'entità business, insieme all'istruzione che indica un'azione sui dati.

definizione dell'oggetto business. Una descrizione del formato e dei dati contenuti nell'oggetto business. Una definizione dell'oggetto business contiene il nome, la versione, un insieme di istruzioni supportate e un insieme ordinato di attributi.

analisi oggetto business. Effettua il monitoraggio e la creazione di prospetti dei valori degli attributi di un oggetto business specificato durante il runtime. Le analisi degli oggetti business si possono trovare su un qualsiasi collegamento di transizione (ad eccezione di un collegamento di transizione in ingresso di un nodo decisione e di un collegamento di chiamata servizio).

BusObj. Una classe definita da InterChange Server Express che rappresenta un oggetto business.

BusObjArray. Una classe definita da InterChange Server Express che rappresenta un vettore di oggetti business. BusObjArray viene utilizzato per un attributo di oggetto business il cui valore è un riferimento ad un vettore di oggetti business secondari.

frammento di codice. La specifica di un'azione attraverso una serie di istruzioni di codice, che utilizzano l'API di collaborazione o altro codice Java.

collaborazione. La logica business che descrive un generico processo business distribuito. Una collaborazione interagisce con le applicazioni individuali, collegando eventi e dati di queste applicazioni ed estendendone le funzionalità.

gruppo di collaborazione. Un insieme eseguibile di oggetti di collaborazione, formato collegando gli oggetti di collaborazione.

oggetto di collaborazione. Un oggetto creato da una maschera di collaborazione. Un oggetto di collaborazione è eseguibile quando è configurato e collegato ad applicazioni, rappresentate da connettori, o ad altre collaborazioni.

proprietà di configurazione collaborazione. Un'informazione configurabile relativa ad un oggetto InterChange Server Express. Una maschera di collaborazione presenta proprietà standard e proprietà specifiche della collaborazione. Uno sviluppatore di collaborazioni crea proprietà specifiche della collaborazione per consentire ad un amministratore di specificare alcuni aspetti del comportamento di runtime dell'oggetto di collaborazione.

maschera di collaborazione. La logica e la struttura di una collaborazione. Una maschera di collaborazione fornisce la definizione di una collaborazione e consente di creare un'istanza di un oggetto di collaborazione; una maschera di collaborazione non è mai direttamente eseguibile.

CollaborationException. Un oggetto eccezione definito da InterChange Server Express.

compensazione. L'azione intrapresa da una collaborazione durante il rollback di una transazione per annullare una chiamata di servizio precedente eseguita.

flusso di controllo. Il flusso della logica del processo business. All'interno delle collaborazioni, un diagramma di attività definisce il flusso di controllo per un particolare scenario, specificando le azioni richieste per il processo business. I nodi decisione e gli iteratori sono utilizzati nel diagramma di attività per specificare ulteriormente l'ordine di esecuzione dei nodi azioni.

attributo di correlazione. Identifica una conversazione tra due processi business quando una collaborazione viene utilizzata come LLBP (Long-Lived Business Process). Gli attributi di correlazione sono inizializzati da un nodo avvio o da una chiamata di servizio in uscita; possono quindi essere utilizzati dai partecipanti alla conversazione per effettuare chiamate esterne o per ricevere un evento corrispondente da origini esterne.

currentException. Una variabile definita da InterChange Server Express che contiene il valore della eccezione che precede. L'ambito della `currentException` viene innalzato nell'azione, attività secondaria o iteratore che precede.

nodo decisione. Un nodo che gestisce una sezione di decisioni in uno scenario. I nodi decisione sono utilizzati quando ci sono più possibili uscite a fronte di una azione, un diagramma secondario o un nodo iteratore. Ogni sezione di un nodo decisione presenta una condizione, ed il flusso di controllo passa alla sezione le cui condizioni sono considerate verificate.

dichiarazione. Il nome e il tipo della variabile da utilizzare. Il compilatore richiede una dichiarazione per ogni variabile utilizzata.

isolamento evento. Assicura che più collaborazioni non elaboreranno contemporaneamente eventi relativi agli stessi dati di oggetto business.

sequenza eventi. InterChange Server Express assicura che una collaborazione elaborerà più eventi relativi allo stesso oggetto business uno alla volta, nello stesso ordine con il quale gli eventi sono ricevuti.

eccezione. Un oggetto utilizzato per passare un errore di runtime ad altre entità che possono gestire l'errore. In un diagramma di attività, un'eccezione viene catturata in un collegamento di transizione di eccezione.

evento non riuscito. Questo termine è stato cambiato. Vedere flusso non risolto.

oggetto business con valori completi. Un oggetto business che presenta valori di dati per più attributi rispetto ai semplici attributi di chiave primaria.

dichiarazione di importazione. Una dichiarazione Java che include una classe o un pacchetto nella classe di collaborazione.

iteratore. Un simbolo del diagramma di attività che racchiude un riferimento ad un diagramma nidificato che implementa un'operazione di loop, e il diagramma che contiene il comportamento di loop. Un iteratore può eseguire il loop di tutti gli attributi in un oggetto business o di tutti gli elementi di un vettore oggetti business.

valori chiave. I valori degli attributi che generalmente comprendono l'identificativo univoco di un oggetto business o l'entità di applicazione associata.

LLBP (Long-Lived Business Process). Un metodo di configurazione e distribuzione di una collaborazione per abilitare la comunicazione asincrona tra processi business. In un processo LLBP (Long-Lived Business Process), il contesto del flusso di eventi persiste per tutta la durata di una chiamata di servizio.

livello di transazione minimo. Il livello di transazione, impostato da uno sviluppatore di maschere di collaborazione, che indica il livello dei servizi di transazione richiesti per l'esecuzione di oggetti di collaborazione creati dalla maschera.

pacchetto. Un gruppo di classi Java correlate. Una maschera di collaborazione può far parte di un pacchetto e importare altri pacchetti.

porta. L'interfaccia tra una collaborazione e gli altri oggetti nel sistema InterChange Server Express. Un oggetto di collaborazione si collega ad un connettore o ad un altro oggetto di collaborazione attraverso una porta.

oggetto business con valori di riferimento. Un oggetto business che contiene solo i suoi attributi chiave. Non contiene valori per attributi non chiave.

scenario. Il codice che gestisce uno o più eventi in ingresso. Gli scenari possono essere utilizzati per suddividere la logica di collaborazione.

struttura scenario. L'insieme di scenari, visualizzati in modo gerarchico, che include scenari composti, diagrammi secondari e iteratori.

variabile di scenario. Una variabile il cui ambito si estende a tutte le parti di tutti i diagrammi in uno scenario.

chiamata di servizio. Un simbolo del diagramma di attività che rappresenta una richiesta per un oggetto InterChange Server Express al di fuori della collaborazione, ad esempio un connettore o un'altra collaborazione.

diagramma secondario. Un simbolo del diagramma di attività che rappresenta un altro diagramma di attività nidificato e lo stesso diagramma nidificato.

variabile di maschera. Una variabile il cui ambito si estende a tutti gli scenari in una maschera di collaborazione.

vista struttura maschera. La visualizzazione della struttura che mostra le definizioni di maschera, la struttura dello scenario e il file di messaggi della maschera di collaborazione. La visualizzazione della struttura della maschera è facoltativa.

collaborazione transazionale. Una collaborazione che segue il modello di database transazionale e fornisce la congruenza dei dati per i processi business. Una collaborazione transazionale è in grado di eseguire il rollback quando un errore di runtime provoca un errore nell'oggetto di collaborazione. In una collaborazione transazionale, le chiamate di servizio presentano una compensazione definita.

collegamento di transizione. Un simbolo del diagramma di attività che indica il flusso di controllo tra gli altri simboli di un diagramma di attività.

istruzione transazionale. Un istruzione dell'oggetto business che indica una modifica dei dati, ad esempio Create, Update o Delete. Retrieve non è un'istruzione transazionale perché non modifica i dati.

evento di attivazione. L'oggetto business che un connettore invia ad una collaborazione di sottoscrizione quando si verifica un evento di applicazione.

triggeringBusObj. La variabile dichiarata dal Designer che contiene un evento di attivazione dello scenario quando lo scenario inizia l'esecuzione.

UID. Un identificativo univoco per ciascun simbolo nei diagrammi di attività di uno scenario.

flusso non risolto. L'oggetto business la cui ricezione determina l'esecuzione di uno scenario terminato con esito negativo da parte della collaborazione. Un flusso non risolto può essere un flusso non riuscito (un flusso non riuscito a causa di problemi di applicazione o di logica), un flusso rinviato (un flusso il cui ripristino è stato rinviato), un flusso in transito (un flusso creato quando il server termina in modo anomalo durante la trasmissione di una chiamata di servizio in una collaborazione configurata per il mantenimento della chiamata di servizio in stato di transito) o un possibile flusso duplicato (un flusso che può essere già stato ricevuto dalla collaborazione).

Indice analitico

Caratteri speciali

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