IBM WebSphere Business Integration
Server Express e Express Plus

Guida allo sviluppo dell'accesso

Versione 44

Nota

Prima di utilizzare queste informazioni e il relativo prodotto, leggere attentamente le informazioni riportate in "Informazioni particolari" a pagina 107.

22 Aprile 2005

Questa edizione della documentazione è valida per IBM WebSphere Business Integration Server Express version 4.4, IBM WebSphere Business Integration Server Express Plus versione 4.4, Toolset Express versione 4.4 e a tutti i rilasci e modifiche successive se non diversamente indicato nelle nuove edizioni.

Per inviare commenti sulla documentazione, inviare un messaggio e-mail a 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 su questo manuale

I prodotti IBM(R) WebSphere Business Integration Server Express e IBM(R) WebSphere Business Integration Server Express Plus sono costituiti dai seguenti componenti: InterChange Server Express, il Toolset Express associato, CollaborationFoundation e e una serie di adattatori di integrazione del software. Gli strumenti di Toolset Express consentono di creare, modificare e gestire i processi di business. E' possibile scegliere tra diversi adattatori preconfigurati per i processi di business che utilizzano le applicazioni. Il modello dei processi standard--CollaborationFoundation--consente di creare processi personalizzati in tempi rapidi.

Tale documento descrive come utilizzare gli API dell'interfaccia di accesso al server IBM per abilitare una capacità del flusso delle chiamate emesse. Un flusso delle chiamate emesse è un flusso avviato da un processo del client di accesso che può creare oggetti business ed eseguire delle collaborazioni.

Se non diversamente indicato, tutte le informazioni riportate in questo manuale sono valide per IBM WebSphere Business Integration Server Express e IBM WebSphere Business Integration Server Express Plus. Il termine "WebSphere Business Integration Server Express" e le sue varianti fanno riferimento ad entrambi i prodotti.

A chi è diretto questo manuale

Questo documento è destinato ai clienti, ai consulenti o ai rivenditori di IBM WebSphere che creano o modificano delle collaborazioni. Prima di iniziare, è necessario comprendere tutti i concetti espressi nella *Guida all'implementazione del sistema*.

Per implementare gli API dell'interfaccia di accesso al server, è necessario conoscere i concetti e la pratica della programmazione standard così come il linguaggio di programmazione Java (TM). Gli API di accesso al server si basano sul linguaggio di programmazione Java.

Prerequisiti per questo documento

Questo manuale presuppone che si stia iniziando con una specifica, un grafico di flusso o una progetto a matita. Non ricopre analisi di processi aziendali, sviluppo di collaborazioni o connettori, oppure la progettazione di oggetti business.

Nota: In questo documento i backslash (\) vengono utilizzati per convenzione per i percorsi delle directory. Per le installazioni su Linux, sostituire i backslash con gli slash (/). Tutti i percorsi dei file sono relativi alla directory del sistema viene installato il prodotto IBM.

Come utilizzare questo manuale

La *Guida allo sviluppo dell'accesso* è strutturata nella seguente maniera:

Parte I: introduzione

	Capitolo 1, “Introduzione all’Interfaccia di accesso al server”, a pagina 3	E’ una panoramica dell’interfaccia di accesso al server.
	Capitolo 2, “Impostazione degli ambienti del client di accesso”, a pagina 13	Indica come installare e impostare l’ambiente di sviluppo e di run-time.
	Capitolo 3, “Configurazione delle collaborazioni per i flussi attivati dalle chiamate”, a pagina 19	Mostra come configurare le collaborazioni per utilizzare con i client di accesso.
	Capitolo 4, “Implementazione di un client di accesso”, a pagina 23	Fornisce una panoramica sulla modalità implementazione di un client di accesso per eseguire una collaborazione.
Parte II: esercitazione		
	Capitolo 5, “Un servlet di esempio con funzioni di gestione dati HTML”, a pagina 31	Mostra un servlet progettato in Java che utilizza gli API.
Parte III: riferimento all’API dell’interfaccia di accesso al server		
	Capitolo 6, “Interfaccia IAccessEngine”, a pagina 53	Contiene la sintassi e i frammenti di codice che mostrano come utilizzare metodi nell’interfaccia `IAccessEngine`.
	Capitolo 7, “Interfaccia IInterchangeAccessSession”, a pagina 57	Contiene la sintassi e i frammenti di codice che mostrano come utilizzare metodi nell’interfaccia `IInterchangeAccessSession`.
	Capitolo 8, “Interfaccia IBusinessObject”, a pagina 67	Contiene la sintassi e i frammenti di codice che mostrano come utilizzare metodi nell’interfaccia `IBusinessObject`.
	Capitolo 10, “Interfaccia IBusinessObjectArray”, a pagina 97	Contiene la sintassi e i frammenti di codice che mostrano come utilizzare metodi nell’interfaccia `IBusinessObjectArray`.
	Capitolo 9, “Eccezioni dell’interfaccia di accesso al server”, a pagina 93	Descrive le eccezioni dell’API dell’interfaccia di accesso al server.

Documentazione correlata

L’intera serie della documentazione descrive le funzioni e i componenti comuni a tutte le installazioni di WebSphere Business Integration Server Express e WebSphere Business Integration Server Express Plus, ed include materiale di riferimento su componenti specifici.

E’ possibile eseguire il download, installare e prendere visione della documentazione al seguente
sito:http://www.ibm.com/websphere/wbiserverexpress/infocenter

Nota: Informazioni importanti relative a questo prodotto potrebbero essere disponibili in Technical Supporto Technotes and Flashes emesso in seguito alla pubblicazione di questo documento. 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

In questo manuale sono utilizzate le seguenti convenzioni:

`font courier`	Indica un valore letterale, come il nome di un comando, il nome di un file, le informazioni che si stanno immettendo oppure informazioni che il sistema stampa su di uno schermo.
Grassetto	Indica un nuovo termine la prima volta che appare.
Corsivo	Indica il nome di una variabile o il nome di un titolo.
`Courier corsivo`	Indica il nome di una variabile all'interno di un testo.
`boxed courier`	Separa un frammento di codice dal resto del testo.
testo blu	Uno schema blu, disponibile quando si visualizza un manuale online, indica un collegamento ipertestuale con riferimenti incrociati. Fare clic all'interno dello schema 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.
`[ ]`	Su una riga della sintassi, le parentesi racchiudono un parametro facoltativo.
`...`	Su una riga della sintassi, i puntini sospensivi indicano una ripetizione del parametro precedente. Ad esempio, `option[,...]` indica che è possibile immettere più opzioni separate da virgole.
dir_prodotto	Rappresenta la directory in cui è installato il prodotto.

Novità in questo rilascio

Questa sezione descrive le nuove funzioni e quelle modificate della *Guida allo sviluppo dell'accesso* per l'ambiente di sviluppo di IBM WebSphere Business Integration Server Express e IBM WebSphere Business Integration Server Express Plus.

Novità nel rilascio 4.4

Per questo rilascio, sono stati apportate alla guida le seguenti modifiche:

- Sotto "Strumenti per lo sviluppo del client di accesso",è stato incluso ITE (Integrated Test environment) come strumento di debug per lo sviluppo.
- Le procedure e le illustrazioni per la configurazione delle collaborazioni per i flussi di chiamate emesse sono state modificate per le interfacce delle finestre di dialogo aggiornate e le finestre di dialogo delle proprietà di re-implementazione come editor.
- L'accesso a InterChange Server Express ora include un utente con nome con dei ruoli assegnati. Questo RBAC (role-based access control) è una parte importante per mantenere un ambiente sicuro e la sua influenza viene documentata in questo libro.
- L'EDK (e-Business Development Kit) è stato ridenominato in SADK (Server Access Development Kit).
- E' stato risolto un problema nel codice di esempio Capitolo 5 del servlet.
- La documentazione relativa alle sequenze di eventi è stata ampliata. Il controllo degli utenti delle richieste sincronizzate contro quelle asincrone è stata dettagliata.
- E' stato presentato l'XML necessario per aggiungere la sezione di accesso al file InterChangeSystem.cfg

Novità nel rilascio 4.3.1

Questo è il primo rilascio di questo manuale.

Parte 1. Guida introduttiva

Capitolo 1. Introduzione all'Interfaccia di accesso al server

L'**Interfaccia di accesso al server** del sistema di IBM WebSphere Business Integration Server Express è un API che consente un'elaborazione esterna per richiedere l'esecuzione di una collaborazione all'interno di IBM WebSphere Business Integration Server Express. Tale processo esterno, chiamato **client di accesso**, invia una richiesta di accesso per iniziare un flusso di chiamate emesse.

Questo capitolo fornisce una panoramica sull'Interfaccia di accesso al server, su come abilita la connettività business-to-business e su come iniziare a sviluppare soluzioni specifiche per il sito utilizzando l'API dell'Interfaccia di accesso del server.

Il capitolo contiene le seguenti sezioni:

Flusso attivato da chiamata

L'Interfaccia di accesso al server è un API che consente un processo esterno per richiedere l'esecuzione di una collaborazione all'interno di IBM WebSphere Business Integration Server Express. Una **collaborazione** rappresenta un processo aziendale che può coinvolgere diverse applicazioni. Utilizzando l'Interfaccia di accesso al server, questo processo esterno, chiamato **client di accesso**, può ottenere dati da applicazioni che InterChange Server Express gestisce attraverso l'esecuzione di una collaborazione.

L'interfaccia di accesso al server abilita InterChange Server Express a ricevere direttamente le richieste per l'esecuzione di una collaborazione, senza ricevere da un connettore un evento di attivazione. Le richieste inviate dal client di accesso vengono chiamate **richieste di accesso**. Per inviare una richiesta di accesso, un client di accesso emette un richiamo ad un metodo nell'interfaccia del server di accesso, anziché inviare realmente un evento. Quindi, l'attivazione del flusso avviata da un client di accesso viene chiamata **flusso attivato da chiamate**, invece del flusso attivato dagli eventi avviati da un connettore (consultare Figura 1).

Il flusso attivato da chiamate viene gestito dall'economia e trasparenza di un flusso attivato da eventi. La principale distinzione operativa sta nel fatto che i flussi attivati da chiamate vengono elaborati in modalità sincronizzata e quindi *non* sono persistenti nel sistema di InterChange Server Express. Al contrario, i flussi attivati da eventi vengono processati in modalità asincrona e sono persistenti. Per ulteriori

informazioni relative alla modalità di elaborazione nel sistema di tali flussi, consultare la *Guida all'implementazione del sistema*.

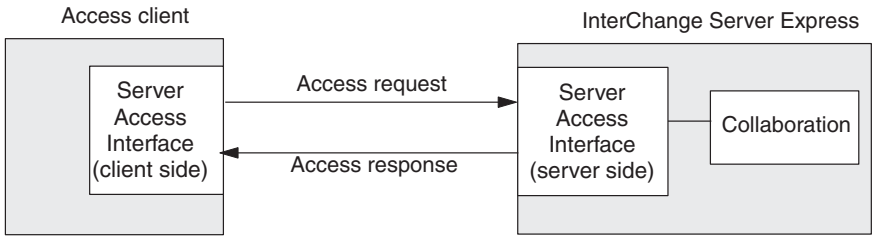

Figura 1. Flusso attivato da chiamata

Come mostra Figura 1, una richiesta di accesso avviata da un client di accesso richiede la seguente procedura:

1. Il client di accesso crea l' **attivazione dei dati di accesso**, che poi invia all'InterChange Server Express durante la richiesta di accesso. Tali dati attivano la collaborazione specificata; ossia, la collaborazione richiede questi dati per avviare l'esecuzione.
2. Il client di accesso richiama un metodo dell'API dell'Interfaccia di accesso al server per inviare un' **attivazione di una chiamata di accesso** al all'Interfaccia di accesso al server all'interno di InterChange Server Express. L'attivazione della chiamata di accesso include l'attivazione dei dati di accesso e il nome della collaborazione da eseguire. Con il richiamo di questo metodo, il client di accesso esegue un richiesta di accesso, la quale avvia il flusso attivato da chiamate.
3. L'interfaccia di accesso al server in IBM WebSphere Business Integration Server Express riceve l'attivazione della chiamata di accesso, eseguendo qualsiasi conversione necessaria del richiamo dei dati di accesso ad un oggetto business del sistema. Per ulteriori informazioni relative a tale conversione dei dati, consultare "Il ruolo dei gestori dati di IBM WebSphere Business Integration Server Express" a pagina 5.
4. L'interfaccia di accesso al server in InterChange Server Express invia il richiamo dei dati di accesso alla collaborazione per attivarne l'esecuzione.
5. Una volta che la collaborazione avrà completato, invierà l'oggetto business risultante all'interfaccia di accesso al server.
6. L'interfaccia di accesso al server esegue qualsiasi conversione necessaria dall'oggetto business risultante al formato originale dell'attivazione dei dati di accesso quindi esegue **risposta di accesso** per rinviare i dati della risposta di accesso al client di accesso. Per ulteriori informazioni su tale conversione dei dati, consultare "Il ruolo dei gestori dati di IBM WebSphere Business Integration Server Express" a pagina 5.

Questa sezione fornisce di seguito, ulteriori informazioni relative al flusso delle chiamate emesse:

- Il ruolo dei gestori dati di IBM WebSphere Business Integration Server Express
- Esempio di flusso di chiamate emesse

Il ruolo dei gestori dati di IBM WebSphere Business Integration Server Express

Un gestore dati di IBM WebSphere Business Integration Server Express converte tra dati serializzati e un oggetto business di IBM WebSphere. Tali gestori dati supportano una varietà di formati di dati per i dati serializzati. L'API dell'interfaccia di accesso al server consente al client di accesso di inviare un evento di attivazione in uno dei diversi vari formati. Se i dati dell'attivazione dell'accesso si trovano in XML, l'interfaccia di accesso al server in InterChange Server Express effettua delle chiamate al gestore dati XML, che analizza i dati di attivazione dell'accesso e li converte in un formato dati IBM WebSphere: un oggetto business. Facoltativamente, il client di accesso può trasmettere l'oggetto business risultante da una risposta della collaborazione all'interfaccia di accesso al server, che richiama il gestore dati appropriato per riconversione al formato "in entrata" (in questo caso, XML).

Per invocare il gestore dati, l'interfaccia di accesso al server deve prima localizzare un gestore dati di livello superiore meta-oggetto che utilizza per creare un'istanza di un gestore dati. Il meta-oggetto di livello superiore per InterChange Server Express è `MO_Server_DataHandler` ed è collocato nello stesso computer di InterChange Server Express. Il software di sviluppo dell'interfaccia di accesso al server include il gestore dati XML, il gestore dati EDI, il gestore dati NameValue, il gestore dati FixedWidth e il gestore dati Delimiter. Supporta inoltre, lo sviluppo di gestori dati personalizzati. Per impostazione predefinita, il meta-oggetto `MO_Server_DataHandler` è configurato in maniera tale che l'interfaccia di accesso al server richiami automaticamente il gestore dati XML quando riceve dati serializzati da un client di accesso. Se il proprio client di accesso utilizza dati serializzati in un formato diverso da XML, è necessario assicurarsi che il meta-oggetto `MO_Server_DataHandler` sia modificato per supportare il gestore dati appropriato. Per ulteriori informazioni, consultare la *Guida al gestore dati*.

Esempio di flusso di chiamate emesse

L'interfaccia di accesso al server supporta le transazioni business-to-business che richiedono un accesso dall'esterno sicuro e affidabile da parte di fornitori oppure unità corporative in rete per supportare le applicazioni. Quanto segue è un esempio di business-to-business che comprende due ipotetiche aziende, Azienda A e Azienda B.

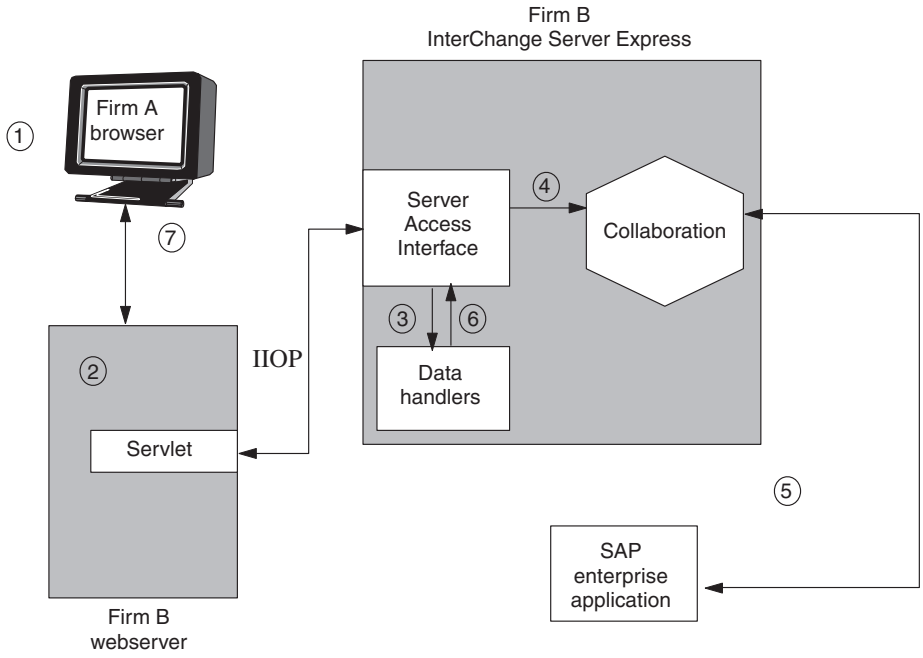

Figura 2. Esempio di Business-to-business

In questo esempio, l'Azienda A desidera ordinare 1.000 IC dall'Azienda B. Per i fornitori autorizzati come l'Azienda A, l'Azienda B supporta i flussi di chiamate emesse al proprio backend integrato di IBM WebSphere Business Integration Server Express. Il processo di dispiega nella seguente maniera:

1. Un impiegato di un'Azienda A si registra al sito Web dell'Azienda B, immettendo un Id account e una password. L'impiegato, quindi, effettua un ordine di 1.000 IC. Il server Web dell'azienda B autentica l'utente come un vendor autorizzato.
2. Il client di accesso avvia un flusso di chiamate emesse al server e-business dell'Azienda B (IBM WebSphere Business Integration Server Express). L'interfaccia di accesso al server dell'Azienda B riceve ed elabora le chiamate dell'API dal client di accesso. La chiamata di accesso emesso indica che i dati sono in formato XML.
3. Il flusso di chiamate emesse dell'Azienda A trasferisce i dati al Gestore dati XML. Questo gestore dati converte i dati serializzati nel formato generico di oggetto business dell'Azienda B. Le definizioni dell'oggetto business vengono estratte dai DTD nel flusso di dati XML e dal meta-oggetto del gestore dati.
4. Il client di accesso dell'Azienda A esegue la collaborazione all'interno dell'InterChange Server Express dell'Azienda B, lanciando un processo `Order_Generation`. L'oggetto business utilizza una collaborazione IBM WebSphere che sia configurata correttamente - uno che sia collegato ad una porta con una funzione di client di accesso e che abbia una mappa che trasmetta dati da e per quella porta.
5. L'oggetto business viene indirizzato ad un connettore IBM WebSphere per SAP, il quale accede all'applicazione SAP/R3 dell'Azienda B ed effettua gli ordini. (L'Azienda B indirizza l'ordine al proprio fornitore perché vengano evasi). Il risultato —conferma dell'ordine— viene generato e, attraverso un connettore, viene ritrasmesso al client di accesso.
6. Il client di accesso dell'Azienda A invia l'oggetto business risultante al gestore dati XML. Il gestore dati XML analizza e converte il risultato in un flusso di dati XML.

7. Il risultato viene a sua volta trasmesso al server Web, che lancerà un processo separato per inviare un'e-mail all'impiegato con la conferma dell'avvenuta transazione, includendo il numero d'ordine.

Panoramica della procedura per lo sviluppo del client di accesso

Per sviluppare un client di accesso, si codificherà il file di origine del client di accesso e si completeranno altre attività. L'attività di creazione di un client di accesso include i seguenti passi:

1. Impostazione dell'ambiente di sviluppo. Installare il software di IBM WebSphere Business Integration Server Express includendo il file `AccessInterfaces.idl` e quindi utilizzare il programma di utilità per generare sia frammenti di codice Java or C++ dal file `AccessInterfaces.idl`.
2. Configurare una porta di una collaborazione per accesso ed esecuzione con un flusso di chiamate emesse. Questo passo comporta la configurazione di porte esterne della collaborazione, in grado di gestire i client di accesso.
3. Implementare ed eseguire il debug al client di accesso (come un servlet web) che esegue le chiamate dell'API dell'interfaccia di accesso al server. Importare le classi `IdlAccessInterfaces.*`, ed implementare il codice Java per eseguire quanto segue:
   - Ottenere una sessione di accesso a IBM WebSphere Business Integration Server Express.
   - Inviare un'attivazione di chiamata di accesso ad un collaborazione specificata, includendo le chiamate del gestore dati
   - Eseguire una collaborazione.
4. Configurare il meta-oggetto del gestore dati di livello superiore `MO_Server_DataHandler` per puntare alle istanze del gestore dati necessarie alla conversione dei dati dal formato esterno (inviato dal client di accesso) al formato oggetto business IBM WebSphere. Per ulteriori informazioni, consultare la *Guida al gestore dati*.

Figura 3 fornisce una panoramica del processo di sviluppo del client di accesso ed un riferimento rapido ai capitoli dove è possibile trovare delle informazioni relative a specifici argomenti. Da notare che se si dispone di un gruppo di persone per lo sviluppo del client di accesso, la maggior parte delle attività dello sviluppo del client di accesso può essere eseguita in parallelo dai diversi membri del gruppo.

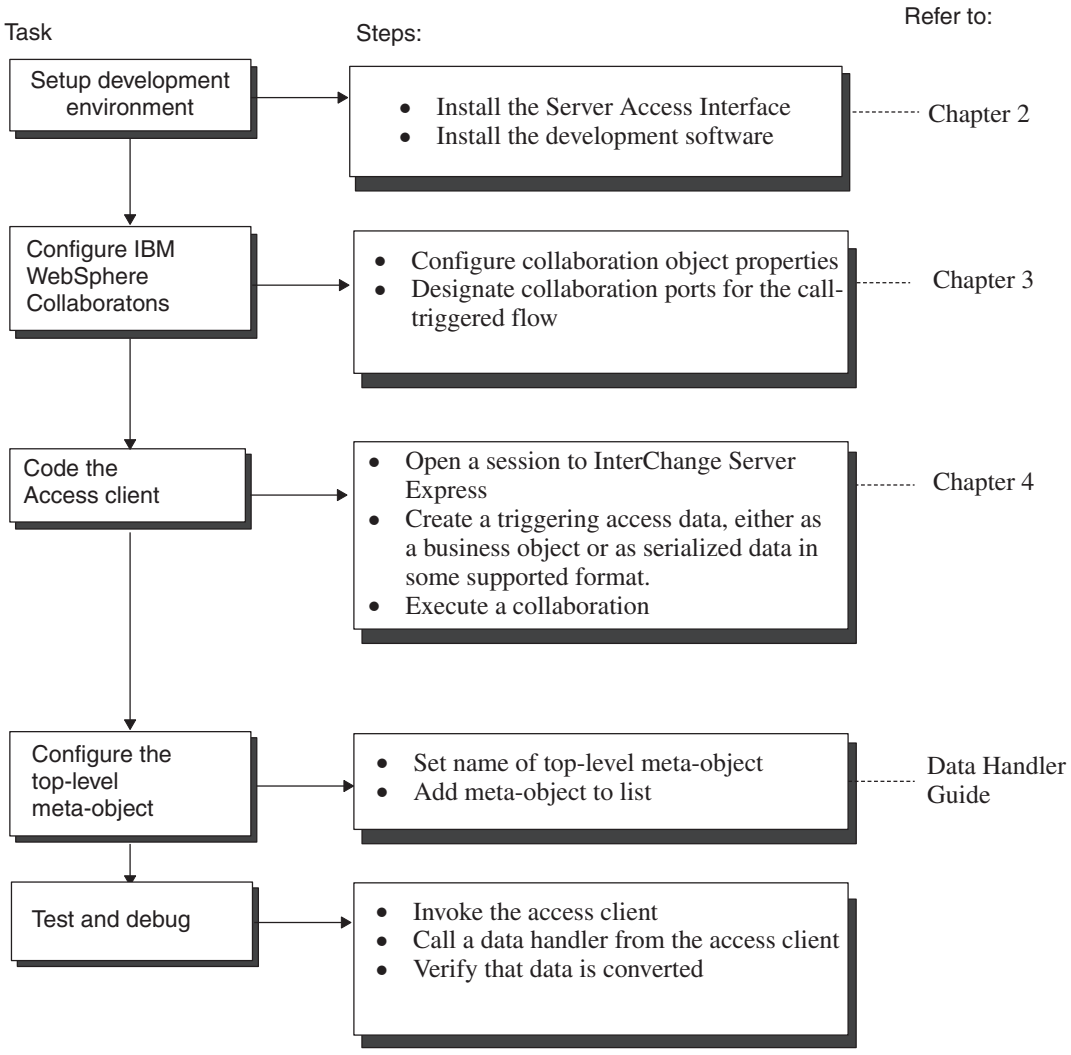

Figura 3. Panoramica dell'attività di sviluppo del client di accesso

Strumenti per lo sviluppo del client di accesso

Dato che i client di accesso vengono scritti in Java, è possibile svilupparli sia su un sistema Windows che Linux. La seguente tabella elenca gli strumenti che IBM WebSphere fornisce per lo sviluppo dei client di accesso.

Strumento IBM WebSphere	**Descrizione**
SADK (Server Access Development Kit)	Sono inclusi i seguenti argomenti: • Esempio di gestori dati • Frammento di file per l'estensione della classe `DataHandler`
API interfaccia di accesso al server	Contiene classi Java per accedere all'InterChange Server Express dall'interno di un client di accesso.
API del gestore dati	Contiene una classe singola, `DataHandler`, che è possibile estendere per creare un gestore dati personalizzato.

Strumento IBM WebSphere	Descrizione
JCDK (IBM WebSphere Java Connector Development Kit)	Comprende le classi Java per lavorare con gli oggetti business.
ITE (Integrated Test Environment)	E' lo strumento di debug per lo sviluppo che supporta il simulatore di interfaccia di accesso chiamato "Simulatore di client" per verificare la collaborazione da un client di interfaccia di accesso.

Server Access Development Kit

Il SADK (Server Access Development Kit) fornisce agli sviluppatori degli strumenti per sviluppare il software personalizzato nella seguente tabella.

Software personalizzato	Directory secondaria di DevelopmentKits\sadk
Connettore	`ConnectorAgent`
Gestore dati	`DataHandler`
Gestore protocollo	`ProtocolHandler`
Client di accesso	`ServerAccessInterfaces`
Programmi di utilità, compreso XMLBORGEN (utilizzato dal gestore dati XML)	`Programmi di utilità`

Come mostra la tavola precedente, gli strumenti per lo sviluppo di client di accesso si trovano nella directory `ServerAccessInterfaces`, sotto la directory secondaria `DevelopmentKits\sadk` della directory *ProductDir*.

Esempio di client di accesso

Per assistere allo sviluppo di un client di accesso, SADK comprende un esempio di client di accesso nella directory IBM WebSphere: `DevelopmentKits\sadk\ServerAccessInterfaces\AccessSample`.

Questa directory contiene:

- L'esempio del client di accesso, `ATPServlet.java`, è un servlet che consente di convertire dati HTML in un oggetto business, che può essere poi mandato ad una collaborazione in InterChange Server Express.
- Un gestore dati personalizzato, `HtmlDataHandler.java`, gestisce la conversione tra dati HTML e un oggetto business InterChange Server Express.
- Il file `SampleRepos.jar`, che contiene le definizioni del repository dei componenti utilizzati dall'esempio di accesso.
- Le directory secondarie elencate nella seguente tabella contiene ulteriori file di esempio.

Nome	Descrizione
`collaborations`	Contiene collaborazioni configurate per i flussi delle chiamate emesse.
`DLMs`	Contiene le mappe native richieste.

Nota: E' bene considerare che questo esempio *non* fornisce esempi di tutte le funzionalità supportate nell'API dell'interfaccia di accesso al server.

Per ulteriori informazioni, consultare Capitolo 5, “Un servlet di esempio con funzioni di gestione dati HTML”, a pagina 31.

API dell’interfaccia di accesso al server IBM WebSphere

L’API dell’interfaccia di accesso al server IBM WebSphere fornisce le seguenti interfacce:

Interfaccia di accesso al server	Descrizione	Per ulteriori informazioni
`IAccessEngine`	Fornisce un metodo per collegare il client di access to a InterChange Server Express	Capitolo 6, “Interfaccia IAccessEngine”, a pagina 53
`IInterchangeAccessSession`	Fornisce metodi per controllare l’acceso ad una sessione di accesso in InterChange Server Express	Capitolo 7, “Interfaccia IInterchangeAccessSession”, a pagina 57
`IBusinessObject`	Fornisce metodi per eseguire operazioni di oggetti business come ottenere, impostare, e confrontare un valore di attributo	Capitolo 8, “Interfaccia IBusinessObject”, a pagina 67
`IBusinessObjectArray`	Fornisce i metodi che consentono ad un client di accesso per interagire e modificare gli array dell’oggetto business	Capitolo 10, “Interfaccia IBusinessObjectArray”, a pagina 97

Nota: I metodi nelle interfaccia elencati nella tabella precedente emettono le eccezioni descritte in Capitolo 9, “Eccezioni dell’interfaccia di accesso al server”, a pagina 93.

API del gestore dati IBM WebSphere

L’API del gestore dati IBM WebSphere fornisce una classe singola, chiamata `DataHandler`. La classe di base astratta `DataHandler` facilita lo sviluppo del gestore dati personalizzato. Questa classe contiene i metodi che vanno a popolare l’oggetto business con valori estratti da dati di input serializzati e metodi che serializzano un oggetto business in una stringa o flusso. La classe include anche metodi del programma di utilità che un gestore dati personalizzato può utilizzare. Si otterrà così un gestore dati personalizzato da questa classe `DataHandler`. Per ulteriori informazioni sui metodi nella classe `DataHandler`, consultare la *Guida al gestore dati*.

Nota: Si rende necessario sviluppare un gestore dati personalizzato se il proprio client di accesso formatta i suoi dati serializzati in formati diversi da quelli supportati gestori dati IBM WebSphere Business Integration Server Express esistenti. Per un elenco di questi gestori dati, consultare “Il ruolo dei gestori dati di IBM WebSphere Business Integration Server Express” a pagina 5.

JCDK (IBM WebSphere Java Connector Development Kit)

Se si sviluppa un gestore dati personalizzato, è necessario utilizzare dei metodi di alcune delle classi nel JCDK (IBM WebSphere Java Connector Development Kit) per lavorare con gli oggetti business. Se si sviluppa un gestore dati personalizzato, potrebbe essere necessario importare ulteriori classi JCDK, come `CxCommon.CxObjectContainerInterface` or `CxCommon.CxObjectAttr`.

Capitolo 2. Impostazione degli ambienti del client di accesso

Questo capitolo mostra come impostare il proprio ambiente di sviluppo ed eseguire i client di accesso. Il capitolo contiene le seguenti sezioni:

Impostazione dell'ambiente di sviluppo

L'ambiente di sviluppo per il client di accesso richiede che si abbia accesso agli stub dell'API dell'interfaccia di accesso al server, che sono parte del software che il programma di installazione IBM WebSphere installa. Quindi, per includere le chiamate all'API dell'interfaccia di accesso al server nel proprio client di accesso, è necessario avere acceso al seguente software:

- Un ambiente di sviluppo IBM Java ORB (versione 4.5 o successiva; consultare la *Guida all'installazione di IBM WebSphere Business Integration Server Express* per il rilascio corrente)

  Nota: E' possibile lavorare con qualsiasi ORB che sia compatibile con CORBA 2.3. Verificare con il venditore ORB che il proprio ORB sia compatibile con CORBA 2.3.
- Un ambiente di sviluppo Java e JDK 1.4.2
- Il rilascio corrente del software IBM WebSphere
- InterChange Server Express avviato e in esecuzione
- Un repository IBM WebSphere con le collaborazioni configurate per il flusso delle chiamate emesse (Per ulteriori informazioni sull'esecuzione di tale configurazione, consultare Capitolo 3, "Configurazione delle collaborazioni per i flussi attivati dalle chiamate", a pagina 19)

Una volta che si ha accesso al software sopra elencato, l'impostazione dell'ambiente di sviluppo per un client di accesso comprende i seguenti passi:

- "Installazione dell'interfaccia di accesso al server IBM WebSphere"—Installazione dell'interfaccia di accesso al server sul computer per lo sviluppo.
- "Compilazione del client di accesso" a pagina 14—Creazione di un eseguibile per il client di accesso.

Installazione dell'interfaccia di accesso al server IBM WebSphere

Per poter sviluppare un client di accesso, è necessario installare l'interfaccia di accesso al server sul computer per sviluppo. Il programma di installazione di IBM WebSphere installa i file associati all'interfaccia di accesso al server di IBM WebSphere. Installa le directory e i file mostrati in Tabella 1 a pagina 14.

Tabella 1. Struttura dei file installati per l'interfaccia di accesso al server di IBM WebSphere

Directory	Descrizione
`DevelopmentKits\sadk\ ServerAccessInterfaces`	Contiene il file `AccessInterfaces.idl` per i client di accesso.
`DevelopmentKits\sadk\ ServerAccessInterfaces\ AccessSample`	Contiene il codice di origine per l'esempio il client di accesso.
`repository\sadk`	Contiene i file per il meta-oggetto `MO_Server_DataHandler` che definisce quale gestore dati l'interfaccia di accesso al server supporta.

Il programma di installazione di IBM WebSphere installa automaticamente i file in Tabella 1 quando installa il software di IBM WebSphere. Per essere sicuri che L'API dell'interfaccia di accesso al server è installato, assicurarsi che il componente Server e strumenti sia selezionato sul pannello Selezione componenti del programma di installazione di IBM WebSphere. Quando il programma di installazione installa questo componente, installa automaticamente le directory e i file elencati in Tabella 1. Per informazioni sul programma di installazione di IBM WebSphere, consultare la *Guida all'installazione di IBM WebSphere Business Integration Server Express per Linux* o *per Windows*.

Nota: Il programma di installazione di IBM WebSphere installa anche i file necessari ai gestori dati distribuiti da IBM WebSphere. Per ulteriori informazioni, consultare il capitolo sull'installazione nella *Guida al gestore dati*.

Compilazione del client di accesso

Quando si è pronti per la compilazione del proprio client di accesso, bisogna accertarsi che i percorsi ai seguenti file siano nel proprio classpath:

- Il file `crossworlds.jar` di IBM WebSphere
- I file jar di IBM Java ORB (Object Request Broker)

E' possibile utilizzare il programma di compilazione `javac` o qualsiasi IDE (Integrated Development Environment).

Impostazione dell'ambiente di run-time

Al run time, il client di accesso non necessita di essere su di un computer che contiene IBM WebSphere Business Integration Server Express, né di essere sullo stesso computer dell'ambiente di sviluppo. Tuttavia, il client di accesso per localizzare l'istanza di InterChange Server Express di cui ha bisogno al run time, deve poter localizzare il server dell'ORB (Object Request Broker), che mantiene traccia delle ubicazioni di diversi oggetti CORBA (comprese le istanze InterChange Server Express) e comunica tali informazioni ai client ORB (come un client di accesso). Per ottenere l'ubicazione del server ORB, il client di accesso può utilizzare Il file di riferimento all'oggetto interoperabile che la sua istanza di InterChange Server Express genera. Quando InterChange Server Express viene avviato o riavviato, genera u file di riferimento all'oggetto interoperabile, che un'estensione `.ior`. Il client di accesso può utilizzare tale file per individuare il server ORB, e, a turno, per comunicare con la propria istanza di InterChange Server Express.

Quindi, perché il client di accesso individui la sua istanza InterChange Server Express, è necessario che vengano eseguiti i seguenti passi:

1. Richiedere che InterChange Server Express generi un file .ior persistente.
2. Assicurarsi che il computer sul quale si trova il client di accesso sia in grado di individuare il file .ior per la sua istanza InterChange Server Express.

Ciascuno di questi passi viene descritto più dettagliatamente nella seguente sezione.

Generazione di un file .ior persistente

Quando InterChange Server Express, versione 3.1.0 o successive, viene riavviato, genera un nuovo file .ior. Tuttavia, InterChange Server Express assegna dinamicamente un numero di porta per il server ORB. Se il numero della porta cambia ogni volta che il server viene riavviato, il client di accesso non può dipendere dal file .ior per individuare il server ORB. Quindi, un client di accesso ha bisogno di InterChange Server Express per generare un file .ior **persistente**.

Perché InterChange Server Express generi un file .ior persistente, è necessario modificare il file di configurazione di InterChange Server Express (InterchangeSystem.cfg) in un editor XML e aggiungere un sezione secondari per CORBA, se non ne esistesse già uno. Figura 4 mostra il codice XML che definisce una sezione secondaria *vuota* di CORBA (una *senza* parametri di configurazione definiti).

```
<tns:property>
   <tns:name>CORBA</tns:name>
   <tns:isEncrypted>false</tns:isEncrypted>
   <tns:updateMethod>system restart</tns:updateMethod>
   <tns:location>
      <tns:reposController>false</tns:reposController>
      <tns:reposAgent>false</tns:reposAgent>
      <tns:localConfig>true</tns:localConfig>
   </tns:location>
     Le definizioni XML delle proprietà di CORBA vanno qui
</tns:property>
```

Figura 4. Definizione XML della sezione secondaria di CORBA

La sezione secondaria di CORBA specifica il numero statico della porta con l'parametro di configurazione OAport, che contiene la seguente sintassi:

```
OAport=portNumber
```

Ad esempio, se il numero statico della porta deve essere 15000, assegnare un valore di 15000 al suo parametro OAport nella sezione secondaria di CORBA. Il seguente frammento XML apparirebbe all'interno delle tag <tns:property> per la sezione secondaria di CORBA, nell'ubicazione indicata da Figura 4 con la stringa "*Le definizioni XML delle proprietà di CORBA vanno qui*":

```
<tns:property>
  <tns:name>OAport</tns:name>
  <tns:value xml:space="preserve">15000</tns:value>
  <tns:isEncrypted>false</tns:isEncrypted>
  <tns:updateMethod>system restart</tns:updateMethod>
  <tns:location>
     <tns:reposController>false</tns:reposController>
     <tns:reposAgent>false</tns:reposAgent>
     <tns:localConfig>true</tns:localConfig>
  </tns:location>
</tns:property>
```

Importante: Il file di configurazione di InterChange Server Express è un file XML. Per aggiungere una sezione secondaria di CORBA e i suoi parametri di configurazione, è necessario utilizzare un editor XML o formattare correttamente le tag XML appropriate.

Per ulteriori informazioni sulla sezione secondaria di CORBA nel file di configurazione, consultare la *Guida all'installazione di WebSphere Business Integration Server Express per Linux* o *per Windows*.

Localizzazione del file .ior

Per individuare il server ORB al runtime il client di accesso deve essere in grado di individuare il file `.ior` per la sua istanza InterChange Server Express. L'individuazione di questo file non rappresenta un problema se il client di accesso e l'InterChange Server Express si trovano sullo stesso computer. Tuttavia, se questi due componenti *non* sono sullo stesso computer, è necessario intraprendere *una* delle seguenti azioni per assicurarsi che il computer della macchina client di accesso può accedere al file `.ior`:

- Copiare il file `.ior` che InterChange Server Express ha generato nel computer sul quale si trova il client di accesso.
- Creare un directory condivisa sul computer con InterChange Server Express e puntare il computer del client di accesso alla directory.

Attivazione o disattivazione della sequenza degli eventi per le richieste di accesso

Quando vengono inviate richieste simultanee alla collaborazione utilizzando il framework di accesso, la sequenza delle richieste può non essere importante, specialmente quando si sta cercando di migliorare le prestazioni. Per impostazione predefinita la sequenza degli eventi viene messa a punto a livello di collaborazione per richieste di accesso simultanee.

Nel rilascio 4.3 di InterChange Server Express, sono state aggiunte nuove proprietà che aumentano il controllo che gli utenti possono esercitare sulla sequenza degli eventi. E' stata aggiunta una nuova casella di controllo al pannello delle proprietà Generale dell'oggetto della collaborazione del System Manager. Questa casella di controllo viene denominata `Isolamento evento`, ed è selezionata per impostazione predefinita.

La sequenza degli eventi per le richieste di accesso simultanee, viene controllata sia dal System Manager che modificando la sezione `ACCESS` del file `InterchangeSystem.cfg` per impostare il valore di `EVENT_SEQUENCING` su `FALSE`. (Da non dimenticare che ora il file `InterchangeSystem.cfg` ora è in formato XML).

Il comportamento del sistema dipende dalla natura della richiesta, il valore del parametro `EVENT_SEQUENCING` nel file di configurazione e dallo stato della casella di controllo nel System Manager. Tabella 2 a pagina 17 spiega come la sequenza degli eventi varia in base alle diverse impostazioni.

Tabella 2. L'effetto di relative impostazioni sulla sequenza degli eventi

Si tratta di una richiesta esterna (SAI)?	**Proprietà EVENT_SEQUENCING nella sezione ACCESS di InterchangeSystem.cfg**	**La casella di controllo "Isolamento evento" nelle proprietà generali della collaborazione**	**E' stata abilitata la sequenza degli eventi nella collaborazione?**
Sì	True/non specificato	Selezionato	Sì
		Non selezionato	No
	False	Nessun effetto	No
No	Nessun effetto	Selezionato	Sì
		Non selezionato	No

Capitolo 3. Configurazione delle collaborazioni per i flussi attivati dalle chiamate

In questo capitolo è riportato come configurare le collaborazioni per i flussi emessi dalle chiamate. E' necessario configurare le collaborazioni *prima* di eseguirle dal client di accesso. Gli argomenti di questo capitolo comprendono:

- "Utilizzo di System Manager per implementare un'opzione del flusso di chiamate emesse"
- "Designazione delle porte della collaborazione per i flussi delle chiamate emesse"
- "Associazione di oggetti business e mappe" a pagina 21
- "Configurazione delle proprietà dell'oggetto della collaborazione" a pagina 22

Importante: Per configurare le collaborazioni per un flusso di chiamate emesse, è necessario avere installato tutto il software IBM WebSphere e avere l'InterChange Server Express in esecuzione.

Utilizzo di System Manager per implementare un'opzione del flusso di chiamate emesse

E' necessario utilizzare System Manager per configurare una collaborazione per un flusso di chiamate emesse. Per implementare un'opzione del flusso di chiamate emesse per una collaborazione, è necessario creare un nuovo oggetto della collaborazione da uno dei modelli di collaborazione nel repository utilizzando la procedura guidata Creazione di una nuova Collaborazione.

Per informazioni relative all'utilizzo della procedura guidata, consultare *Guida all'implementazione del sistema*. La procedura guidata comprende dei pannelli per la designazione e il collegamento delle porte della collaborazione, associazione degli oggetti business e delle mappe e le proprietà dell'impostazione per il nuovo oggetto della collaborazione.

Una volta creato l'oggetto della collaborazione, è possibile modificare le impostazioni della configurazione per un flusso delle chiamate emesse seguendo i passi riportati nella sezione qui di seguito.

Nota: Una collaborazione può avere più porte configurate per un flusso delle chiamate emesse.

Designazione delle porte della collaborazione per i flussi delle chiamate emesse

Per ciascuna collaborazione che si desidera configurare per un flusso delle chiamate emesse, è necessario configurare la porta sull'oggetto della collaborazione.

Per configurare una porta della collaborazione per un flusso delle chiamate emesse:

1. Fare doppio clic sull'oggetto della collaborazione che si desidera configurare nella cartella degli oggetti della collaborazione o nella cartella delle librerie del del componente di integrazione oppure nella cartella dei progetti dell'utente nel System Manager.
2. Per modificare i collegamenti della porta nella vista grafica, fare clic con il pulsante destro sull'icona per una porta e selezionare Collega porta dal menu di contesto. Per modificare i collegamenti della porta nella vista della struttura, fare clic con il pulsante destro del mouse sull'icona di una porta e selezionare Collega porta dal menu di contesto. Per le informazioni su "Viste dell'oggetto della collaborazione," consultare la *Guida all'implementazione del sistema*.

   Si apre la finestra di dialogo Configurazione porta. Le impostazioni predefinite per il tipo di porta è Interna ed abilita i collegamenti ai connettori e alle collaborazioni.

   La figura 5 mostra come modificare il collegamento della porta nella vista struttura (a destra) e la finestra di dialogo Configurazione che viene aperta (a sinistra).

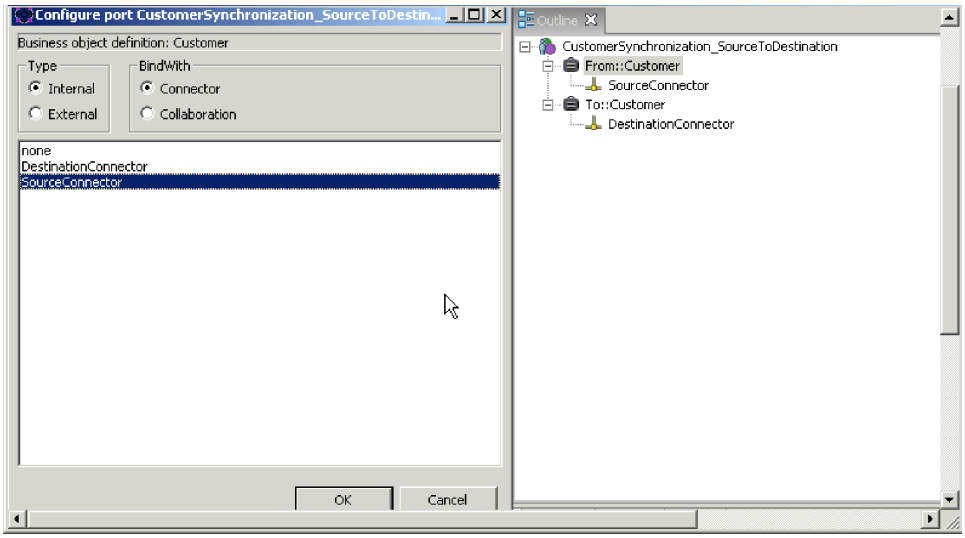

Figura 5. Vista Struttura e finestra di dialogo Configura porta (interna)

3. Fare clic su Esterna nell'area Tipo della finestra di dialogo di Configura porta.

   Questa visualizza la finestra di dialogo Configura porta (Esterna) come mostrato in Figura 6. Nell'area Configura come, la finestra di dialogo visualizza il tipo di porta scelto da configurare: In entrata se le richieste dell'oggetto business vengono ricevute dalla porta o In uscita se le risposte dell'oggetto business vengono inviate dalla porta.

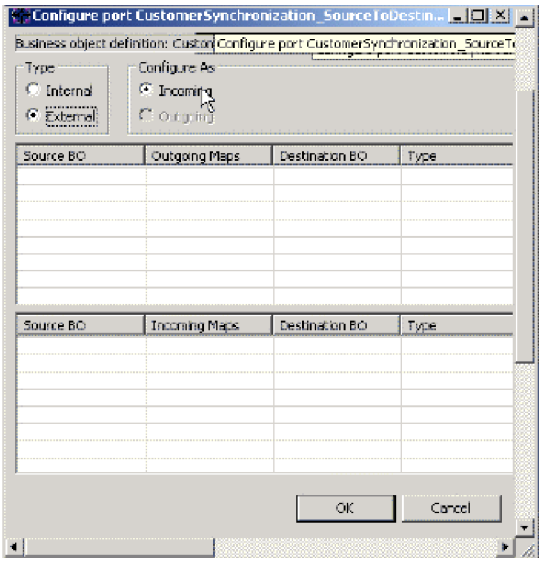

Figura 6. Finestra di dialogo Configura porta (esterna)

Associazione di oggetti business e mappe

E' possibile associare facoltativamente uno o più mappe per collegamenti esterni delle porte. Seguire i seguenti passi:

1. Passare alla finestra di dialogo Configura porta.
2. Effettuare una delle seguenti operazioni:
   - Trascinare e rilasciare una definizione dell'oggetto business dalla cartella delle librerie del componente integrativo sia nel nella colonna Mappe in entrata che quelle in uscita nella finestra di dialogo Configura porta, rispettivamente se la porta riceve le richieste dell'oggetto business o se le risposte dell'oggetto business vengono inviate dalla porta.

     Quando si apre la finestra di dialogo Tipo oggetto business, selezionare l'Oggetto business dell'origine oppure l'oggetto business della destinazione, in base al fatto che si l'oggetto business trascinato e rilasciato sia un oggetto di origine o di destinazione nella mappa che lo converte.
   - Trascinare e rilasciare una definizione di mappa che converte il tipo di oggetto business dalla cartella delle librerie del componente integrativo sia nel nella colonna Mappe in entrata che quelle in uscita nella finestra di dialogo Configura porta, rispettivamente se la porta riceve le richieste dell'oggetto business o se le risposte dell'oggetto business vengono inviate dalla porta.

   Nota: In alternativa, è possibile fare clic con il pulsante destro del mouse sulla griglia ed utilizzare il menu di contesto per aggiungere le mappe.
3. Fare clic su OK.

Direzione del flusso: nella collaborazione

Vengono applicate le seguenti azioni quando la direzione del flusso si dirige nella collaborazione:

- Trascinamento di un oggetto business —Utilizzare l'oggetto business come tipo di destinazione per la collaborazione.

- Trascinamento di una mappa—Utilizzare la mappa quando viene fatta la chiamata alla collaborazione. Selezionare una mappa che supporti la destinazione dell'oggetto business.

Nota: In alternativa, utilizzare il menu di contesto. Per ulteriori informazioni, consultare "Associazione di oggetti business e mappe" a pagina 21.

Direzione di flusso: all'esterno della collaborazione

Vengono applicate le seguenti azioni quando la direzione del flusso va all'esterno della collaborazione:

- Trascinamento dell'oggetto business—Utilizzare le opzioni dell'oggetto business quando la collaborazione sta restituendo il risultato.
- Trascinamento di una mappa—Utilizzare la mappa quando la collaborazione restituisce i dati o gli attributi del processo di richiesta.

Nota: In alternativa, utilizzare il menu di contesto. Per ulteriori informazioni, consultare "Associazione di oggetti business e mappe" a pagina 21.

Configurazione delle proprietà dell'oggetto della collaborazione

Per configurare le proprietà generali di un'oggetto della collaborazione per un flusso delle chiamate emesse, eseguire i passi seguenti:

1. Accedere al pannello Proprietà generali dell'editor della collaborazione eseguendo una delle seguenti operazioni:
   - Se l'editor della collaborazione non è aperto, fare clic con il pulsante destro del mouse sull'oggetto della collaborazione nella cartella delle librerie del componente integrativo e selezionare Proprietà dal menu di contesto. Quest'operazione aprirà l'editor della collaborazione con la seconda scheda attivata e verrà visualizzato il pannello Proprietà generalvolerei.

     In alternativa, fare doppio clic con il tasto destro del mouse sull'oggetto della collaborazione nella cartella delle librerie del componente integrativo. Quest'operazione consentirà di aprire l'editor della collaborazione con la scheda grafica impostata come predefinita. Fare clic sulla seconda scheda, Proprietà generali per visualizzarne il pannello.
   - Se l'editor della collaborazione è già aperto in una vista grafica, fare clic sulla seconda scheda, Proprietà generali, per visualizzarne il pannello.
2. Configurare le proprietà dell'oggetto della collaborazione come preferito.

   Importante: Assicurarsi che la proprietà `Numero massimo di eventi simultanei` sia impostato su di un valore pari a `0`. I flussi delle chiamate emesse sono impostati a più thread per impostazione predefinita, per cui impostando questa proprietà su `0` ci si assicura che ulteriori thread vengano richiamati dall'InterChange Server Express per fornire la capacità del thread multiplo. Per ulteriori dettagli relativi a questa proprietà, consultare la *Guida alla gestione del sistema*.
3. Dal menu File, selezionare Salva (oppure utilizzare Ctrl+S).

   Nota: Selezionando Salva aggiornerà tutte e tre le schede, non solo la scheda Proprietà generali.

Capitolo 4. Implementazione di un client di accesso

Questo capitolo fornisce una panoramica sulle modalità di implementazione di un client di accesso, che potrebbe richiedere l'esecuzione di una collaborazione con InterChange Server Express mediante L'API dell'interfaccia di accesso al server. Gli argomenti di questo capitolo comprendono:

Creazione di una sessione di accesso

Prima che un client di acceso possa emettere una richiesta di accesso, bisogna che prima stabilisca una **sessione di accesso** con InterChange Server Express. Per consentire al client di accesso di connettersi a InterChange Server Express, l' interfaccia `IAccessEngine` fornisce il `login()` e `metodi securelogin()`. Tali metodi creano la sessione di accesso, che consentono al client di accesso di accedere all' Interfaccia di accesso al server all'interno di InterChange Server Express. E' necessario fornire un nome utente e password InterChange Server Express validi ai metodi `login()` e `securelogin()` come argomenti.

Per una spiegazione più dettagliata dell'interfaccia `IAccessEngine`, consultare Capitolo 6, "Interfaccia IAccessEngine", a pagina 53.

Emissione della richiesta di accesso

Una volta che il client ha creato una sessione di accesso, può inviare una richiesta di accesso all'InterChange Server Express. La richiesta di accesso è ciò che avvia il flusso di chiamate emesse all'interno di InterChange Server Express. Prima che possa inviare la sua chiamata di attivazione dell'accesso, il client di accesso deve generare i dati di attivazione dell'accesso che vengono inviati alla collaborazione. L'interfaccia di accesso al server consente al client di accesso le seguenti due modalità per emettere una richiesta di accesso, in base al formato dei dati di attivazione dell'accesso:

Invio di un oggetto business

Il client di accesso può inviare i dati di attivazione dell'accesso incorporati in un oggetto business IBM WebSphere Business Integration Server Express. L' interfaccia `IInterchangeAccessSession` fornisce i metodi per la creazione di oggetti business e

l'esecuzione di collaborazioni. Per una spiegazione dettagliata relativa a questa interfaccia, consultare Capitolo 7, "Interfaccia IInterchangeAccessSession", a pagina 57.

L'invio di un oggetto business come dati di attivazione dell'accesso comprende i seguenti passi:

- "Creazione dell'oggetto business"
- "Operazione sull'oggetto business"
- "Richiesta di esecuzione della collaborazione" a pagina 25

Creazione dell'oggetto business

Tabella 3 mostra i metodi che l'API dell'interfaccia di accesso al server fornisce nell'interfaccia `IInterchangeAccessSession` per consentire al client di accesso di creare un oggetto business.

Tabella 3. Metodi IInterchangeAccessSession per la creazione di oggetti business

Creazione dell'oggetto business	Metodo IInterchangeAccessSession
Creazione di un oggetto business	IcreateBusinessObject()
Creazione di un oggetto business con un verb che specifica un'operazione sugli attributi dell'oggetto business.	IcreateBusinessObjectWithVerb()
Creare un array oggetti business che contenga uno o più attributi, ognuno dei quali ha un oggetto business come proprio tipo.	IcreateBusinessObjectArray()
Creare un oggetto business da dati che vengono formattati in un tipo MIME specificato.	IcreateBusinessObjectFrom()

Operazione sull'oggetto business

Una volta che il client di accesso ha creato l'oggetto business, può utilizzare le interfacce in Tabella 4 per eseguire qualsiasi operazione richiesta per inserire i dati di attivazione dell'accesso in questo oggetto.

Tabella 4. Interfacce di accesso ad un oggetto business

Tipo di oggetto business	API interfaccia di accesso al server	Per ulteriori informazioni
Oggetto business (cardinalità singola)	`IBusinessObject` consente al client di accesso di eseguire le operazioni dell'oggetto business come ottenere, impostare e confrontare i valori degli attributi.	Capitolo 8, "Interfaccia IBusinessObject", a pagina 67
array oggetti business	`IBusinessObjectArray` consente al client di accesso di interagire con gli array degli oggetti business e di manipolarli. I metodi comprendono l'impostazione o il richiamo di elementi dell'array di oggetti business, copiando un array, aggiungendo un oggetto business ad un array oppure prendendo il numero di elementi in un array di oggetti business.	Capitolo 10, "Interfaccia IBusinessObjectArray", a pagina 97

Richiesta di esecuzione della collaborazione

L'interfaccia `IInterchangeAccessSession` fornisce il IexecuteCollaboration() metodo per l'invio di un oggetto business come dati di attivazione dell'accesso nella chiamata di attivazione dell'accesso. Questo metodo fa si che l'interfaccia di accesso al server in InterChange Server Express invii l'oggetto business come dati di attivazione dell'accesso alla collaborazione specificata.

Nota: La collaborazione, la porta e l'oggetto business devono essere configurati e mappati per per la chiamata di accesso diretto e la modifica.

Invio dati serializzati

Il client di accesso può inviare i dati di attivazione dell'accesso come dati serializzati in un tipo MIME specificato. L' Interfaccia di accesso al server nel InterChange Server Express esegue la conversione dei dati necessari dai dati serializzati ad un oggetto business di IBM WebSphere. L'invio di dati serializzati comprende una chiamata ad un metodo singolo dell'API dell'interfaccia di accesso al server, IexecuteCollaborationExtFmt(). Tale metodo fornisce le seguenti attività per il client di accesso:

- Specificare un gestore dati (in base al tipo MIME dei dati serializzati) per convertire i dati serializzati in un per l'oggetto business.
- Creare l' oggetto business che attiva la collaborazione.
- Impostare l' verb ad un valore specifico.
- Eseguire la collaborazione.

Locale e codifica

Per impostazione predefinita, la sessione di accesso utilizza il valore della Locale dell'InterChange Server Express. Tuttavia, è possibile modificare il valore della Locale così che coincida con il valore della Locale dell'oggetto business o della collaborazione che si sta creando o eseguendo mediante la sessione di accesso.

I dati di input inviati all'interfaccia di accesso al server devono essere in codifica Unicode.

Per una panoramica delle Locale, consultare l'Appendice A, Considerazioni sull'internazionalizzazione.

Per una descrizione del metodo per impostare i valori delle Locale, consultare setLocale(stringa) in Capitolo 7, "Interfaccia IInterchangeAccessSession", a pagina 57.

Richiamo della risposta di accesso

Una collaborazione restituisce una risposta di accesso al client di accesso mediante il valore restituito di uno dei metodi in Tabella 5. Il formato di questa richiesta d'accesso dipende dal metodo che il il client di accesso utilizza per inviare la richiesta di accesso.

Tabella 5. Metodi per il richiamo della risposta di accesso

Richieste di accesso	Metodo dell'interfaccia di accesso al server	Formato della risposta di accesso
Invia dati di attivazione dell'accesso come oggetto business	IexecuteCollaboration()	Oggetto business

Tabella 5. Metodi per il richiamo della risposta di accesso (Continua)

Richieste di accesso	Metodo dell'interfaccia di accesso al server	Formato della risposta di accesso
Invia i dati di attivazione dell'accesso come dati serializzati in un tipo MIME specifico	IexecuteCollaborationExtFmt()	dati serializzati (nello stesso formato MIME della richiesta di accesso)

Nota: Se la risposta di accesso è in forma di un oggetto business IBM WebSphere Business Integration Server Express , è possibile utilizzare i metodi delle interfacce elencate in Tabella 4 a pagina 24 per operare su questo oggetto business.

Chiusura della sessione di accesso

Quando il client di accesso ha completato la sua richiesta di accesso, dovrebbe entrare in Tabella 6.

Tabella 6. Chiusura della sessione di accesso

Attività	Metodo dell'Interfaccia di accesso al server
Le risorse del rilascio che l'Interfaccia di accesso al server in InterChange Server Express sta utilizzando per gli oggetti business e gli array degli oggetti business	Metodi `IInterchangeAccessSession`: IreleaseBusinessObject() IreleaseBusinessObjectArray()
Chiudere la sessione di accesso	Metodo `IAccessEngine`: logout()

Nota: Una chiamata a `logout()` libera le risorse che la sessione di accesso sta utilizzando.

Un esempio di implementazione di un flusso di chiamate emesse

Figura 7 mostra un flusso di chiamate emesse più dettagliato, iniziato, in questo caso, da un client di accesso che è un browser del client.

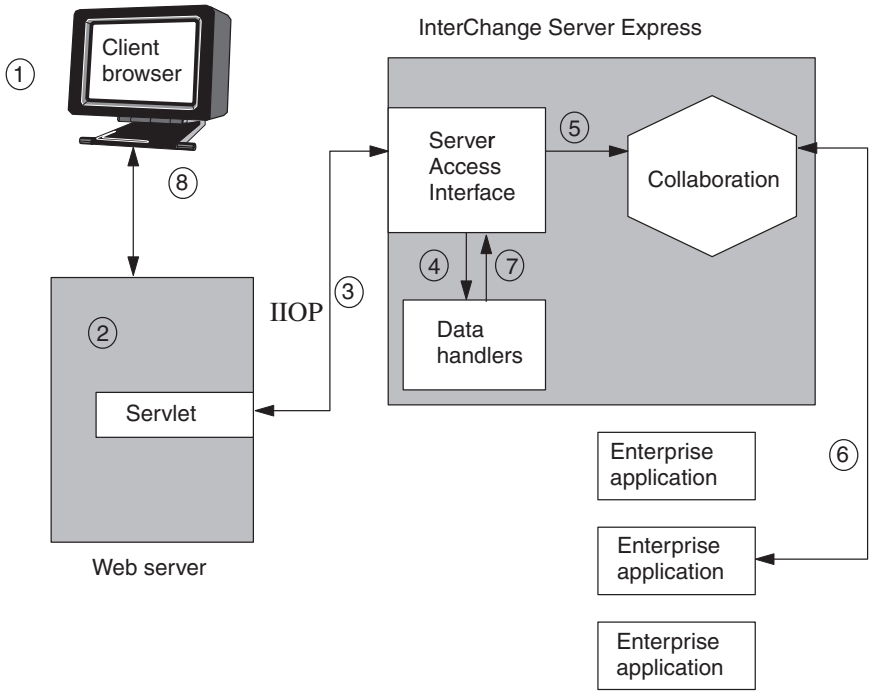

Figura 7. Esempio di un flusso di chiamate emesse iniziato da un browser del client

Come mostrato in Figura 7:

1. Il browser del client emette una richiesta in un protocollo e un formato specifici (ad esempio, un protocollo HTTP ed un formato dati XML).
2. Il server Web enterprise carica un servlet per gestire una richiesta. Questo servlet è il client di access. Il client è programmato per cercare il nome dell'Interchange Server Express compatibile con CORBA (da registro di CORBA).
3. Il client di accesso si connette a InterChange Server Express, mediante una connessione IIOP, creando una sessione di accesso con il metodo `IgetInterchangeAccessSession()` dell'interfaccia `IAccessEngine` dell'API dell'interfaccia di accesso al server.

   Nota: Per eseguire la collaborazione, InterChange Server Express *non* crea alcun thread proprio, ma utilizza il thread CORBA. Per informazioni relative all'utilizzo dei thread da parte delle collaborazioni, consultare la *Guida allo sviluppo della collaborazione*.
4. Il client di accesso utilizza il metodo `IcreateBusinessObjectFrom()` nell'interfaccia `IInterchangeAccessSession` per convertire i dati XML in un oggetto business generico di IBM WebSphere Business Integration Server Express. In risposta a questa chiamata del metodo, l'interfaccia di accesso al server in InterChange Server Express invoca il gestore dati XML per eseguire la conversione dei dati e poi rimandare l'oggetto business al client di accesso.
5. Il cliente di accesso utilizza il metodo `IexecuteCollaboration()` nell'interfaccia `IInterchangeAccessSession` per inviare la chiamata di attivazione dell'accesso, che contiene l'oggetto business come dati di attivazione dell'accesso. Questo processo richiede l'esecuzione di una collaborazione che modifica l'oggetto business.

Nota: L'API dell'interfaccia di accesso al server fornisce anche il metodo `IexecuteCollaborationExtFmt()`, che combina i passi 4 e 5 in chiamata al metodo unica.

6. Tramite i connettori, la collaborazione effettua delle richieste, riordina ed ottiene dati, modificando le applicazioni enterprise quando richiesto. La collaborazione restituisce i dati richiesti, o i risultati delle azioni richieste, al client di accesso nel formato oggetto business.
7. Se il client di accesso ha utilizzato il metodo `IexecuteCollaborationExtFmt()` per emettere la richiesta di accesso, non necessita di eseguire esplicitamente le azioni previste al passo 6. Il metodo `IexecuteCollaborationExtFmt()` converte l'oggetto business automaticamente riportandolo al suo formato originale (in questo caso il formato XML) e restituisce questi dati serializzati al client di accesso.
8. I risultati vengono inviati al browser del client.

Come mostrato in Figura 7, il server Web gestendo la chiamata carica un servlet, che si connette all'InterChange Server Express.

Parte 2. Esempio

Capitolo 5. Un servlet di esempio con funzioni di gestione dati HTML

Questo capitolo presenta un tipico scenario di e-commerce e un codice di esempio che utilizza gli API dell'interfaccia di accesso al server. Gli argomenti compresi in questo capitolo sono::

Lo scenario

Un problema comune che si è riscontrato in ambienti e-commerce è quello della disponibilità degli articoli e la prospettiva di una consegna sicura entro la data richiesta. Questa classe di problemi è comunemente conosciuta come available to promise o, brevemente, ATP.

Un enterprise che utilizza un sistema di ottimizzazione della catena di forniture o il sistema ERP (enterprise resource planning) generalmente interrogherà il loro sistema per stabilire se un prodotto potrà essere consegnato per la data di consegna richiesta. Alcune ditte, particolarmente quelle con relazioni di scambio in linea con diversi fornitori, potrebbero voler stabilire la disponibilità dei prodotti prima di impegnarsi nell'ordine del prodotto.

Una funzione ATP significa effettivamente accedere all'ERP (enterprise resource planning) di una ditta o al sistema di ottimizzazione della catena di forniture. Nell'esempio seguente, gli API dell'interfaccia di accesso al server vengono utilizzati per eseguire le seguenti attività:

- **Conversione dei dati** - Converte una quota di oggetti in entrata dal formato HTML ad un oggetto IBM WebSphere business.
- **Esecuzione della collaborazione** - Esegue il trigger di una collaborazione che richiama i dati ATP per ciascun elemento riscontrato negli oggetti quota.
- **Richiamo dei risultati** - Restituisce dei risultati in una tabella in formato HTML.

Figura 8 descrive una singola disponibilità per assicurare la collaborazione.

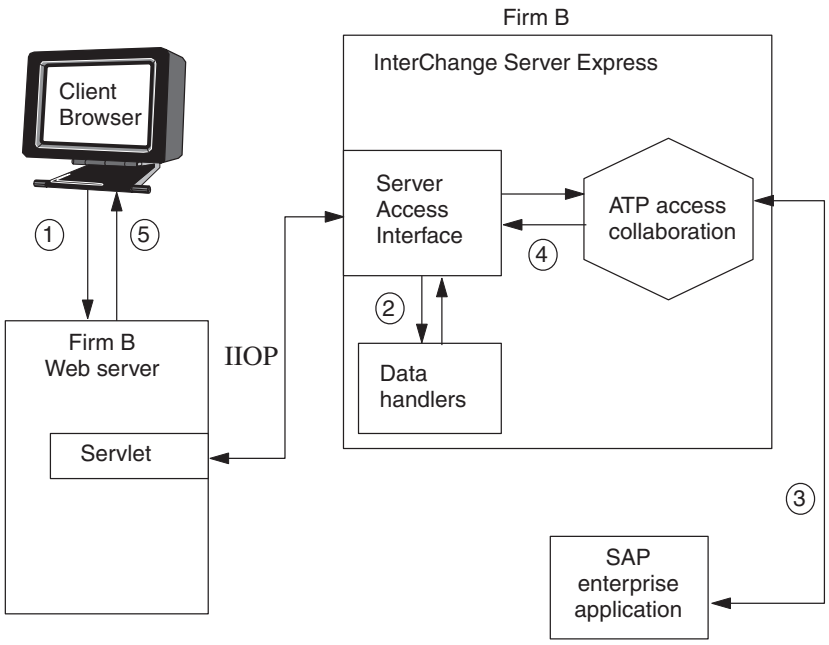

Figura 8. Uno scenario e-commerce available-to-promise

1. Il client del browser invia un modulo HTML contenente i dati corrispondenti ad un oggetto IncomingQuote. Gli oggetti IncomingQuote sono dei dati in formato HTML forniti da una terza applicazione.
2. Il servlet (vedere esempio di seguito) utilizza gli API dell'interfaccia di accesso al server per convertire l'HTML in un oggetto SalesQuote generico e poi inviarlo alla collaborazione.
3. La collaborazione di accesso ATP, quindi richiama la data available-to-promise dal connettore SAP.
4. La collaborazione restituisce tali informazioni al servlet.
5. Il servlet costruisce una tabella HTML contenente una data ATP per ciascuna articolo richiesto e visualizza tale tabella sul browser del client.

Esecuzione dell'esempio su di un server Web

E' possibile caricare ed eseguire l'esempio di codice Server Access Interface code samples. Questa sezione mostra come fare.

1. Installare il software Server Access Development e andare su `DevelopmentKits\sadk\ServerAccessInterfaces\AccessSample` per trovare quanto segue:
   - I due esempi di codice java:

     `HtmlDataHandler.java`

     `ATPServlet.java`
   - Il modulo Richiesta di quote delle vendite: `Example2.html`
   - L'esempio di repository: `SampleRepos.jar`
   - La directory secondaria `collaborations` contiene le collaborazioni e le classi.
   - La directory `DLMs` contiene le classi map native.
2. Caricare `SampleRepos.jar` con il programma di utilità `repos_copy`. Per la guida al caricamento dei file nel repository, consultare la *Guida alla gestione del sistema*.

3. Compilare il file del servlet ATPServlet.java.
4. Distribuire il servlet compilato nel server web. Impostare correttamente i parametri di inizializzazione per la configurazione. Fare riferimento al alla documentazione del server web per i dettagli relativi alla distribuzione e all'inizializzazione dei servlet.
5. Se si possiede un sistema operativo Solaris o HP-UX, aggiungere ibmorb.jar, posizionato in <ProductDirectory>jre\lib\ext (file di classe IBM Java ORB) al percorso di classe del client e del server web. Riavviare il server se necessario. Per i dettagli, consultare la documentazione del server web.
6. Rendere Example2.html disponibile sul server web.
7. Copiare la directory AccessSample\collaborations in *ProductDir*\collaborations.
8. Copiare la directory AccessSample\DLMs in *ProductDir*\DLMs.
9. Compilare HtmlDataHandler.java.
10. Creare un file .jar e salvarlo con il nome HtmlDataHandler.jar, mantenendo la struttura della directory di output.
11. Copiare il HtmlDataHandler.jar in *ProductDir*\lib.
12. Modificare il file batch start_server, aggiungendo *ProductDir*\lib\HtmlDataHandler.jar al percorso di classe.
13. Riavviare il server InterChange Express.
14. Rendere il file Interoperable Object Reference (.ior) disponibile sul proprio server Web.

    Per ulteriori informazioni, consultare "Impostazione dell'ambiente di run-time" a pagina 14.
15. Lanciare una finestra di browser ed aprire la pagina example2.html (consultare Figura 9).
16. Avviare il connettore di prova ed aprire ed aggiungere il profilo "SampleSapConnector". Premere il tasto Connetti per aprire il connettore.
17. Immettere i dati almeno in una riga dei campi (consultare "Esempio di programma di gestione dati HTML" per ulteriori informazioni sulla pagina HTML di esempio) ed eseguire l'operazione Richiama.

La seguente descrizione descrive i dati di gestione e il servlet utilizzato in questo esempio:

- "Esempio di programma di gestione dati HTML"
- "Codice del servlet—ATP Java di esempio" a pagina 42

Esempio di programma di gestione dati HTML

Nell'esempio, il gestore dati HTML converte la stringa della query HTML in entrata in un oggetto aziendale IBM WebSphere Business Integration Server Express. Per ulteriori informazioni sulla capacità del gestore dati IBM WebSphere, consultare la *Guida del gestore dati*. Queste sono le funzioni degne di nota del componente gestore dati:

- **La classe di base datahandler** - Il gestore dati HTML d'esempio estende la classe di base DataHandler fornita da IBM WebSphere Business Integration Server Express e viene caricata automaticamente al runtime quando si rileva una richiesta di accesso per un tipo MIME di "text/html".
- **Configurazione basata sui metadati** - I metadati suggeriscono al sistema dove trovare il gestore dati e come richiamarlo. Di conseguenza, più gestori dati possono operare contemporaneamente su di un solo server InterChange Express.

- **Trasformazione generica** - Il gestore dati HTML in origine è generico e può essere riutilizzato senza modifiche per trasformare qualsiasi tipo di stringe di query HTML.

Figura 9 mostra la pagina HTML così come potrebbe apparire su di un browser del client. Il gestore dati HTML dipende dalle proprietà associate al caselle di testo della pagina.

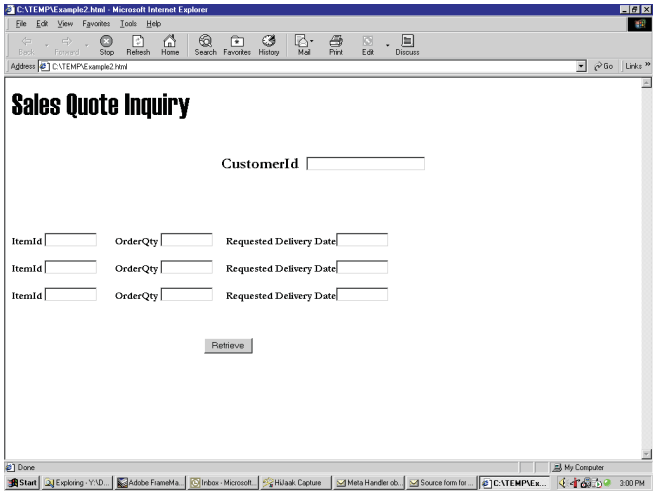

Figura 9. La pagina di richiesta delle quote di vendita HTML

In Figura 9, ciascuna casella di testo ha una proprietà HTML associata. La proprietà della casella di testo HTML contiene la grammatica dell'oggetto aziendale IBM WebSphere Business Integration Server Express. Tale grammatica abilita il gestore dati HTML a convertire i dati associati alla proprietà nell'oggetto aziendale.

Ad esempio, le proprietà associate al primo elemento sono le seguenti:

- **ItemId** - OrderItems[0].ItemID
- **OrderQty** - OrderItems[0].orderQty
- **Requested delivery date** - OrderItems[0].deliveryDate

Come mostrato in Figura 10, il gestore dati converte i dati sulla pagina HTML in un oggetto aziendale gerarchico `SalesQuote` con oggetti aziendali secondari (`orderQty`, `deliveryDate`, e così via).

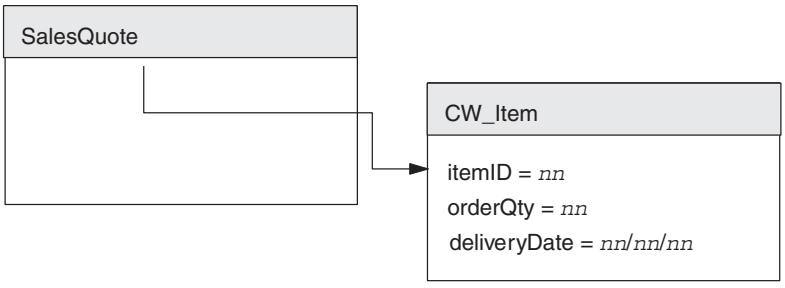

Figura 10. Oggetti aziendali principali-secondari gerarchici

Metaoggetti del gestore dati

Il software IBM WebSphere business integration fornisce due meta-oggetti del gestore dati di livello superiore, uno per il server ed uno per il connettore. Inoltre, vi è un meta-oggetto secondario per ogni gestore dati, diversi dei quali vengono forniti con IBM WebSphere business integration. Quando si configura l'ambiente, è possibile:

- Modificare il nome dell'attributo meta-object del server di livello superiore.

  Il meta-oggetto del gestore dati di livello superiore utilizzato con i gestori dati richiamati nel contesto dell'interfaccia di accesso al server è `MO_Server_DataHandler.`
- Modificare i valori predefiniti di un meta-oggetto secondario per riflettere l'istanza del gestore dati che bisogna creare.

E' possibile definire un attributo nel meta-oggetto di livello superiore per i tipi MIME e qualsiasi tipo secondario (`BOPrefix`) si desidera supportare. Tale attributo rappresenta un meta-oggetto secondario, che ha degli attributi da fornire al nome classe e le proprietà di configurazione richieste dal gestore dati per eseguire il suo lavoro.

Figura 11 a pagina 36 mostra il formato di testo di meta-oggetti:

- Il meta-oggetto del gestore dati di livello superiore, `MO_Server_DataHandler`.

  Da notare che tale meta-oggetto contiene un attributo denominato per il il MIME supportato dal gestore dati HTML (`text.html`). Questo attributo rappresenta il meta-oggetto del gestore dati secondario per il gestore dati HTML, `MO_DataHandler_DefaultHtmlConfig`.
- Il meta-oggetto del gestore dati secondario per il gestore dati HTML, `MO_DataHandler_DefaultHtmlConfig`.

  Il meta-oggetto secondario dichiara un attributo `ClassName`, la cui proprietà `DefaultValue` elenca il nome della classe del gestore dati (`com.crossworlds.DataHandlers.Html.HtmlDataHandler`) da utilizzare per richiamare il gestore dati HTML.

```
[BusinessObjectDefinition]
Nome = MO_Server_DataHandler
Versione = 1.0.0

     [Attribute]
     Name = text.html
     Type = MO_DataHandler_DefaultHtmlConfig
     ContainedObjectVersion = 1.0.0
     Relationship = Containment
     Cardinality = 1
     MaxLength = 1
     IsKey = true
     IsForeignKey = false
     IsRequired = false
     IsRequiredServerBound = false
     [End]
     [Attribute]
     Name = ObjectEventId
     Type = String
     MaxLength = 255
     IsKey = false
     IsForeignKey = false
     IsRequired = false
     IsRequiredServerBound = false
     [End]

     [Verb]
     Name = Create
     [End]

     [Verb]
     Name = Delete
     [End]

     [Verb]
     Name = Retrieve
     [End]

     [Verb]
     Name = Update
     [End]
[End]
```

Figura 11. Formato di testo del meta-oggetto HTML (Parte 1 di 2)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xsd:schema elementFormDefault="qualified"
   targetNamespace="http://www.ibm.com/websphere"
   xmlns:bx="http://www.ibm.com/websphere"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation><xsd:documentation>

Tue Mar 11 14:25:46 PST 2003

</xsd:documentation>

</xsd:annotation>

<xsd:element name="TestChildBO">d
<xsd:annotation>
<xsd:appinfo>
<bx:boDefinition version="3.0.0" />
</xsd:appinfo>
</xsd:annotation>
<xsd:complexType><xsd:sequence>
<xsd:element name="FirstName" minOccurs="0">
<xsd:annotation>
<xsd:appinfo>
<bx:boAttribute>
<bx:attributeInfo isForeignKey="false" isKey="false" />
</bx:boAttribute>
</xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="LastName" minOccurs="1">
<xsd:annotation>
<xsd:appinfo><bx:boAttribute>
<bx:attributeInfo isForeignKey="false" isKey="true" />
</bx:boAttribute>
|</xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="ObjectEventId" type="xsd:string" minOccurs="0" />
</xsd:sequence>
<xsd:attribute name="version" type="xsd:token" default="0.0.0" />
<xsd:attribute name="delta" type="xsd:boolean" default="false" />
<xsd:attribute name="verb" use="required"><xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="Create" />
<xsd:enumeration value="Delete" />
<xsd:enumeration value="Retrieve" />
<xsd:enumeration value="Update" />
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Figura 11. Formato di testo del meta-oggetto HTML (Parte 2 di 2)

Codice di esempio per il gestore dati HTML

Viene qui riportato il codice di esempio Java del gestore dati HTML.

```
/**
 * @(#) HtmlDataHandler.java
 *
 * Copyright (c) 1997-2000 CrossWorlds Software, Inc.
 * Tutti i diritti riservati.
 *
 * Questo software è materiale riservato e di proprietà dell'IBM, Inc.
 * Non è possibile divulgare tali informazioni riservate ed è possibile
 * farne uso solo in conformità dei termini dell'accordo di licenza
 * con CrossWorlds Software.
*/
import com.crossworlds.DataHandlers.*;
import com.crossworlds.DataHandlers.Exceptions.*;
import AppSide_Connector.JavaConnectorUtil;

import CxCommon.BusinessObjectInterface;

// java classes
import java.util.*;
import java.io.*;
/**
 ** Questo è un gestore dati html che converte una stringa della
 ** query html in un oggetto aziendale Crossworlds. Tale esempio
 ** presuppone che la query html in entrata sia strutturata in un
 ** formato specifico così come spiegato nel programma seguente.
 ** Consultare i commenti associati con il metodo parse() in questa classe.
 */
public class HtmlDataHandler extends DataHandler
{

     /*
      **  Un metodo del programma di utilità per convertire una stringa della query HTML
      **  in un ogg. azien. (BO) crossworld. Consultare commenti associati al met. parse() per spiegazione dettagliata
      **  dati serializzati del @param String
      **  il tipo mime in entrata del @param Object
      */
     public BusinessObjectInterface getBO(String serializedData,
       Object config)
          throws Exception
     {

          HashMap nameValuePairs = parse((String) serializedData);

          /*
           ** Ottenere il BO da creare dalla tag nascosta BusObjName
           */
          String boName = (String) nameValuePairs.get("BusObjName");
          if (boName == null)
               throw new Exception("Unable to find business object name in "
                    + "serialized business object");

          BusinessObjectInterface bo = JavaConnectorUtil.createBusinessObject(boName);
          String verb = (String) nameValuePairs.get("Verb");
          if (verb == null)
               throw new Exception("Unable to find verb in serialized business object");

          bo.setVerb(verb);

          /*
           ** Get the elements from the HashMap and set it into the BO
           */
          setValues(bo, nameValuePairs);
          return bo;
     }
```

```
/*
**  Analizzare una stringa della query HTML cercando token del modulo &name=value.
**  Il formato della stringa della query in entrata deve essere conforme al formato di &name=value
**  così come la seguente semantica:
**        se il nome non contiene la sintassi del modulo name[X].attribute si
**        presuppone che sia il nome di un attributo nell'oggetto principale, altrimenti
**        l'espres. verrà utilizzata così com'è per impostare il valore di un ogg. e un attr.
**        secondari.
**
**  Ad esempio, è possibile analizzare la seguente stringa della query
**  con il seguente metodo:
**
** CustomerID=&items[0].itemID=44&items[0].orderQty=25&items[0].
**          deliveryDate1=12/12/00
**  &items[1].itemID=67&items[1].orderQty=2&items[1].
** deliveryDate=12/12/00&Verb=Retrieve&
**  BusObjName=SalesQuote&SubObjName=CwItem
**
**
**  query @param String inviata dal server web da analizzare
**  @return HashMap una mappa hash contenente la coppia di valori del nome
*/

private HashMap parse(String queryString)
{
     HashMap nameValuePairs = new HashMap();
     String content = queryString.replace('+', ' ');
     StringTokenizer st = new StringTokenizer(content,"&");

     while (st.hasMoreTokens())
     {
          String token = st.nextToken();
          int i = token.indexOf("=");
          String name = token.substring(0, i);
          String value = token.substring(i+1);

          /*
           **  HTTP codificherà alcuni valori ASCII con i loro equivalenti esadecimali.
           **  Riconvertire qualsiasi di queste codifiche in ASCII sia per la stringa
           **  del nome che del valore (es. alla destra di = e alla sinistra
           **  di =)
           */
          name = replaceHexEncodedWithAscii(name);
          value = replaceHexEncodedWithAscii(value);

          /*
           **  Archiviare tali valori nell'hashmap così che il chiamante potrà
           **  cercarli.
           */
          nameValuePairs.put(name, value);
     }

     return(nameValuePairs);
}

/*
 *  Dato un Hashmap della coppia nome/valore, eseguire il conteggio tra gli
 *  oggetti aziendali e impostare ciascun attributo nel BO con il valore
 *  corrispondente dalla Hashtable
 *  @param IBusinessObject destinazione dell'insieme
 *  @param Hashmap contiene la coppia nome/valore
 */
private void setValues(BusinessObjectInterface bo, HashMap nameValuePairs)
     throws Exception
{
     String SubObjName = null;
```

```
     Iterator aIterator = nameValuePairs.keySet().iterator();
     // Save the SubObject name so we need to save it
     while (aIterator.hasNext())
     {
          String name = (String) aIterator.next();
          /*
           **  Ignorare qualsiasi chiave nascosta analizzata fuori da HTML
           **  archiviata nella mappa di hash
           */

          if (name.equalsIgnoreCase("BusObjName")  ||
                    name.equalsIgnoreCase("Verb") ||
                    name.equalsIgnoreCase("SubObjName") ||
                    name.equalsIgnoreCase("ContainerAttrName"))
          {
               System.out.println("Skipping Item : " + name);
               continue;
          }

          /*
           **  Tutti gli oggetti secondari hanno un grammatica nella forma di object[X].attribute
           **  dove X è l'indice degli oggetti secondari contenuti.  Quindi, se il
           **  nome non contiene tale stringa integrata, è un attributo
           **  dell'oggetto principale
           */
          if (name.indexOf("[") == -1)
           bo.setAttrValue(name, (String) nameValuePairs.get(name));
          else
           bo.setAttributeWithCreate(name, (String) nameValuePairs.get(name));
     }
}

/*
 *  Sostituire qualsiasi byte in codice esadecimali con l'equivalente carattere ASCII
 *  e rimandare la nuova stringa al chiamante.
 *  @param name la stringa da convertire.
 */
private String replaceHexEncodedWithAscii(String name)
{
     int nameLength = name.length();

     /*
      **  Sostituire qualsiasi valore esadecimale (HTTP può ricoprire un valore esadecimale
      **  intero con il modulo di %XX per alcuni caratteri) con i loro
      **  caratteri equivalenti corrispondenti.
      */
     StringBuffer nameBuffer = new StringBuffer();
     for (int i = 0; i < nameLength; ++i)
     {
          char c = name.charAt(i);
          switch (c)
          {
               case '%':
                    byte[] b = { Byte.parseByte(name.substring(i+1, i+3),
                         16) };
                    nameBuffer.append(new String(b));
                    i += 2;
                    break;
               default:
                    nameBuffer.append(c);
          }
     }
     return(nameBuffer.toString());
}

/**
 **  Implementazione di metodi astratti nella classe del gestore dati
```

```
     **  @param BusinessObjectInterface l'oggetto aziendale corrente
     **  Configurazione @param Object
     **  @return String stringa di rappresentazione del BO
     */

    public String getStringFromBO(BusinessObjectInterface theObj, Object config)
         throws Exception
    {
         throw new Exception("Not implemented");
    }

    /**
     **  Implementazione di metodi astratti nella classe del gestore dati
     *  @param Reader dati reali
     *  @param BusinessObjectInterface l'oggetto aziendale reale
     *  Configurazione @param Object
     */
    public void getBO(Reader serializedData, BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }

    /**
     ** Implementazione di metodi astratti nella classe del gestore dati
     *  @param String dati reali
     *  @param BusinessObjectInterface l'oggetto aziendale reale
     *  Configurazione @param Object
     */
    public void getBO(String serializedData, BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
  }

    /**
     ** Implementazione di metodi astratti nella classe del gestore dati
     *  @param BusinessObjectInterface l'oggetto aziendale reale
     *  @return InputStream un handle per il flusso
     */
    public InputStream getStreamFromBO(BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }

    /**
     ** Implementazione di metodi astratti nella classe del gestore dati
     *  @param Reader dati reali
     *  @param BusinessObjectInterface l'oggetto aziendale reale
     *  @return BusinessObjectInterface il BO tradotto
     */
    public BusinessObjectInterface getBO(Reader serializedData, Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }
}
```

Codice del servlet—ATP Java di esempio

Viene qui riportato l'esempio di servlet ATP descritto in "Lo scenario" a pagina 31.

```
/**
 * @(#) ATPServlet.java
 *
 * Copyright (c) 1997-2000 CrossWorlds Software, Inc.
 * Tutti i diritti riservati.
 *
 * Questo software è materiale riservato e di proprietà dell'IBM.
 * Non è possibile divulgare tali informazioni riservate ed è possibile
 * farne uso solo in conformità dei termini dell'accordo di licenza
 * con IBM Software.
*/
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import java.util.*;
import java.text.*;
import IdlAccessInterfaces.*;
import CxCommon.BusinessObject;

/**
* Servlet di esempio ATP (Available To Promise)
*/
public class ATPServlet extends HttpServlet
{

     // Definisce alcuni valori statici
     public static String DEFAULT_SERVER = "CrossWorlds";
     public static String DEFAULT_IOR = "CrossWorlds.ior";
     public static String DEFAULT_USER = "admin";
     public static String DEFAULT_PASSWD = "null";
     // Nome utente per effettuare il login nel server IC
     private String userName = DEFAULT_USER;
     // Password
     private String passWord = DEFAULT_PASSWD;
     // Nome server
     private String serverName = DEFAULT_SERVER;
     // File IOR
     private String iorFile = DEFAULT_IOR;
     // Sessione di accesso
     private IInterchangeAccessSession accessSession = null;
     // Motore di accesso
     private IAccessEngine accessEngine = null;

     // Contesto del servlet per il richiamo di informazioni sulla config
     private ServletContext ctx;
     // Un programma di formattazione per stampare il prezzo con precisione.
     private static DecimalFormat formatter;
     // Tipo MIME
     private String mimeType = "text/html";

     /**
     * Il metodo init. Tale metodo viene utilizzato dal server web
     * quando il servlet viene caricato per la prima volta.
     * @param ServletConfig informazioni sulla configurazione
     * associata al servlet.
     * @exception ServletException viene restituita quando
     * il servlet non può essere inizializzato
     */
     public void init(ServletConfig aConfig)throws ServletException
     {
        super.init(aConfig);
        // Il programma di formattazione per stampare i prezzi nel formato corretto
        formatter = new DecimalFormat();
```

```
      formatter.setDecimalSeparatorAlwaysShown(true);

      // Leggere i parametri iniziali così che ci si possa
      // connettere al server ICS corretto
      String configuredServer = null;
      String configurediorFile = null;
      String configuredUser = null;
      String configuredpassWord = null;
      configuredServer = aConfig.getInitParameter("ICSNAME");
      if ( configuredServer != null)
             {
             this.serverName = configuredServer;
             }
      else
             {
             this.log(
                "No Interchange Server configured, using
                  default of CROSSWORLDS");
             }
      configurediorFile = aConfig.getInitParameter("IORFILE");
      if (iorFile != null)
             {
             this.iorFile = configurediorFile;
             }
      else
             {
             this.log(
               "IOR file not defined, will use CrossWorlds.ior
                from home directory");
             }

      try
             {
             initAccessSession();
             }
      catch(Exception e)
             {
             this.log("Encountered Initialization error", e);
             throw new ServletException(e.toString());
             }
  }
  /**
  * Metodo cleanup richiamato quando il servlet scaricato dal server web
  */
  public void destroy()
{
       // Rilasciare la sessione
       if ( ( accessEngine != null) && (accessSession != null))
             {
             accessEngine.IcloseSession(accessSession);
             accessEngine = null;
             accessSession = null;
             }
  }

  /*
  **  Metodo del progr. di util. che crea una sess. di accesso con srvr InterChange.
  **  Se ne è stata stabilita una, restituirla.
  **  @exception Exception quando si verifica un errore mentre si tenta
  **  la connessione al server InterChange.
  */
  private synchronized void initAccessSession() throws Exception
  {
       try
             {
             /*
             **  Se la sessione di accesso è stata già stabilita, allora
```

```
**  controllare se la sessione è ancora valida (es. il server
**  InterChange potrebbe essere stato riavviato dall'ultima volta
**  che la sessione è stata utilizzata).
**  Se non è più valida, allora aprirne una nuova.
*/
if (accessSession != null)
     {
     try {
          accessSession.IcreateBusinessObject("");
     } catch (ICxAccessError e) {
          /*
          ** La sessione memorizzata nella cache è ancora valida. Si prevede
             di rilevare questa eccezione
          */
               return;
     }
     // Catturare Corba SystemException
     catch (org.omg.CORBA.SystemException se) {
          /*
           **  La sessione non è valida.
           **  Aprirne una nuova di seguito
           */
          this.log("Re-establishing sessions to ICS");
     }
     }
/**
* Aggiungere le proprietà IBM ORB relative per inizializzare
* l'IBM ORB.
*/
Properties orbProperties = new java.util.Properties();
orbProperties.setProperty("org.omg.CORBA.ORBClass",
     "com.ibm.CORBA.iiop.ORB");
orbProperties.setProperty("org.omg.CORBA.ORBSingletonClass",
     "com.ibm.rmi.corba.ORBSingleton");
org.omg.CORBA.ORB  orb =
     org.omg.CORBA.ORB.init((String[])null, orbProperties);
/*
**  Utilizzare il file che contiene il riferimento Inter-Orb di internet
**  Tale riferimento all'oggetto diventerà un riferimento all'oggetto CORBA
**  serializzato al server Interchange in esecuzione con il quale
**  si desidera conversare.
*/
LineNumberReader input =
     new LineNumberReader(new FileReader(iorFile));
/*
**  Creare un riferimento all'oggetto CORBA residente nella memoria dal IOR
**  nel file
*/
org.omg.CORBA.Object object = orb.string_to_object
   (input.readLine());

/*
**  Ora creare una vera sessione con l'oggetto in esecuzione
*/
private IAccessEngine accessEngine accessEngine =
     IAccessEngineHelper.narrow(object);
Properties props = new Properties();
     props.put("username","admin");
     props.put("password","admin");
IInterchangeAccessSession accessSession =
     SecureLoginUtility.login(accessEngine,props);
if (accessEngine == null)
     throw new Exception("Unable to communicate with server
       " + serverName + " using IOR from " + iorFile);

/*
**  Ora che si ha un riferimento dell'oggetto ad un server in
```

```
                **  esecuzione, è necessario eseguire l'autenticazione prima che
                **  di poter avere una sessione sessione utile.
                */
                accessSession = accessEngine.IgetInterchangeAccessSession(
                     userName,
                     passWord);
SCM
                Properties props = new Properties();
                    props.put("username","admin");
                /*
                **  Qui l'utente deve fornire il nome utente.
                **  In questo esempio viene utilizzato "admin" come nome utente.
                */
                props.put("password","admin");
                /*
                **  Qui l'utente deve fornire la password corrispondente.
                **  In questo esempio si utilizza "admin" come password.
                */
                accessSession = SecureLoginUtility.login(accessEngine ,
                      props);
                /*
                **  Bisogna assicurarsi che "Crossworlds.jar" sia nel
                **  percorso di classe.
                **  La classe del programma di utilità del login denominata "SecureLoginUtility"
                    **  viene fornita con "Crossworlds.jar".
                */

                if (accessSession == null)
                     throw new Exception("Invalid user name and password");

                }
           catch (Exception e)
                {
                this.log("Encountered orb Initialization error" , e);
                if (e instanceof org.omg.CORBA.SystemException)
                     throw new Exception(e.toString());
                else
                     throw e;
                }
     }

     /**
     * Metodo get richiamato dal Server Web ogni volta che un'azione GET
     * viene richiesta da un pagina HTML.
     * @param HttpServletRequest handle alla richiesta http
     * object@param HttpServletResponse handle alla risposta http
     * object @exception ServletException viene restituita quando il servlet
     * rileva un errore @exception restituito quando il
     * server web non riesce a comunicare con la pagina
     * html in chiamata
     */
     public void doGet(HttpServletRequest req, HttpServletResponse res)
          throws ServletException, IOException
     {
          // La stringa serializedHTMLQuote = null;

          // Un BO per contenere il BO in entrata dalla
          // pagina HTML richiedente
          IBusinessObject aBO = null;

          // Un BO per contenere il BO risultante
          // dall'esecuzione della collaborazione

          IBusinessObject returnedQuoteBusObj = null;

          /*
          **  Assicurarsi prima che si abbia un sessione di accesso valida
```

```
**  con il server interchange
*/
try
     {
      initAccessSession();
     }
catch(Exception e)
     {
      throw new ServletException
         ("InitAccessSession Failed " + e.toString());
     }

// Creare un BO dai dati forniti dalla pagina HTML
try {
     aBO =
          accessSession.IcreateBusinessObjectFrom
            (req.getQueryString(),
              mimeType);
} catch (ICxAccessError e) {
       throw new ServletException
         (" Creating Business Object Failed : " +
           e.IerrorMessage);
}

if (aBO == null)
     {
      throw new ServletException("Attempting to use Null Bo ");
     }
/*
**  Eseguire la collaborazione.  Si otterrà un oggetto
**  aziendale CrossWorlds che contiene un data ATP
**  per ciascun articolo.
*/
try
     {
     returnedQuoteBusObj = accessSession.IexecuteCollaboration(
          "ATPExample","From", aBO);
     }
catch(IExecuteCollaborationError ae)
     {
     String error = "Collaboration Error :
          " + ae.IerrorMessage
                        + ae.status;
     this.log("Collaboration Error", ae);
     throw new ServletException(error);
     }
/*
**  Ora, creare una tabella da rimandare che abbia:
**    NumeroArticolo    Quantità    Prezzo
*/
res.setContentType(mimeType);
PrintWriter out = res.getWriter();
out.println("<body>");
out.println("<TABLE BORDER=\"1\">");
out.println("<caption align=\"center\" > " +
     "<font face=e=\"Haettenschweiler\" size=\"7\">" +
     "Sales Quote Response</caption>");

out.println("<TR> <TH>Item ID" +
     "<TH> Descrizione dell'articolo"    +
     "<TH> Quantità " +
     "<TH> Prezzo dell'articolo" +
     "<TH> Data di disponibilità" +
     "<TH> Prezzo totale " +
     "</TH> </TR>");
IBusinessObjectArray itemContainer = null;
try {
```

```
     itemContainer =
          returnedQuoteBusObj.
           IgetBusinessObjectArrayAttribute
            ("OrderItems");
} catch (IInvalidAttributeTypeException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IInvalidAttributeNameException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IAttributeBlankException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IAttributeNotSetException e) {
     throw new ServletException(e.IerrorMessage);
}

// Un oggetto secondario per contenere ciascun articolo
IBusinessObject item = null;

int size = itemContainer.IgetSize();
// Girare nell'array e stampare ciascun articolo
// separatamente
String attr = null;
int itemQuantity = 0;
double itemPrice = 0;
//Girare nell'array degli articoli restituiti
for (int i = 0;i < size; i++)
     {
     try
          {
          // Richiamare l'elemento BusinessObject al
              current indexitem =
              itemContainer.IgetBusinessObjectAtIndex(i);
          if (item != null)
               {
               // Creare una riga della tabella che inizi con l'attributo
               // ITemID
               try {
                    attr = item.IgetStringAttribute("ItemID");
                    out.print("<TR> <TD> " +
                         attr +
                         "</TD>" + "<TD>");
                    // E' stato stampato il valore,
                    // impostarlo nuovamente su nullo
                    attr = null;
               } catch (IAttributeNotSetException e) {
                    attr = "N/A";
                    out.print("<TR> <TD> ");
                    out.print(attr + "</TD>" + "<TD>");
               } catch (IInvalidAttributeNameException e) {
                         attr = "N/A";
                         out.print("<TR> <TD> ");
                         out.print(attr + "</TD>" + "<TD>");
               } catch (IInvalidAttributeTypeException e) {
                    attr = "N/A";
                    out.print("<TR> <TD> ");
                    out.print(attr + "</TD>" + "<TD>");
               }
               // Richiamare l'attributo ItemType
               try {
                    attr = item.IgetStringAttribute
                      ("itemType");
                    out.print(attr + "</TD>" + "<TD>");
                    // E' stato stampato il valore,
                    // impostarlo nuovamente su nullo
                    attr = null;
               } catch (IAttributeNotSetException e) {
                    attr = "N/A";
                    out.print(attr + "</TD>" + "<TD>");
```

```
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
// Richiamare l'attributo orderQty
try {
     attr = item.IgetStringAttribute
        ("orderQty");
     try {
          itemQuantity = Integer.parseInt(attr);
     } catch (NumberFormatException e) {
          itemQuantity = -1;
     }
     out.print(attr + "</TD>" + "<TD>");
     // E' stato stampato il valore,
     // impostarlo nuovamente su nullo
     attr = null;
} catch (IAttributeNotSetException e) {
     attr = "N/A";
     itemQuantity = -1;
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
// Richiamare l'attributo ItemPrice
try {
     attr = item.IgetStringAttribute("itemPrice");
     int indexOfDollar = attr.indexOf("$");
     String priceToParse = null;
     // Verificare se vi è "$" nel valore
     if (indexOfDollar == -1)
          priceToParse = attr;
     else
          priceToParse = attr.substring
              (indexOfDollar + 1);
     // Formattare il prezzo così che appaia come $NNNN.NN
     try {
          itemPrice = Double.parseDouble
          (priceToParse);
     } catch (NumberFormatException e) {
          itemPrice = -1;
     }
     out.print(attr + "</TD>" + "<TD>");
     // E' stato stampato il valore,
        impostarlo nuovamente su nullo
     attr = null;
} catch (IAttributeNotSetException e) {
     attr = "N/A";
     itemPrice = -1;
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
// Richiamare la ATPDate e stamparla
try {
     attr = item.IgetStringAttribute("ATPDate");
```

```
                                out.print(attr + "</TD>" + "<TD>");
                            } catch (IAttributeNotSetException e) {
                                attr = "N/A";
                                out.print(attr + "</TD>" + "<TD>");
                            } catch (IInvalidAttributeNameException e) {
                                attr = "N/A";
                                out.print(attr + "</TD>" + "<TD>");
                            } catch (IInvalidAttributeTypeException e) {
                                attr = "N/A";
                                out.print(attr + "</TD>" + "<TD>");
                            }
                            /*
                            **  Ora, stampare il prezzo totale per l'articolo.
                            **  Se non si posseggono informazioni sufficienti
                            **  stampare N/D
                            */
                            if ((itemPrice == -1) || (itemQuantity == -1))
                                {
                                out.println(attr + "</TD>" + "<TD>");
                                // E' stato stampato il valore,
                                // impostarlo nuovamente su nullo
                                attr = null;
                                }
                            else
                                {
                                double totalPrice = itemQuantity
                                  * itemPrice;
                                out.println("$" + formatter.format
                                  (totalPrice).trim()
                                     + "</TD>"
                                     + "<TD>");
                                }
                            } // end if (Item != null)
                    } // End try
                    catch (IAttributeBlankException e2) {
                         continue;
                    } catch (IInvalidIndexException e) {
                         throw new ServletException(e.getMessage());
                            } finally {
                                      if (item != null)
                             accessSession.IreleaseBusinessObject(item);
                            }
               }// End for loop
          // Il rilascio dell'oggetto aziendale, idealmente, dovrebbe essere eseguito in
          // in un blocco finale così che
                  // nel caso di eccezioni gli oggetti aziendali vengono rilasciati.
                  if (itemContainer != null)
             accessSession.IreleaseBusinessObjectArray(itemContainer);
                  if (returnedQuoteBusObj != null)
             accessSession.IreleaseBusinessObject(returnedQuoteBusObj);
                  if (aBO != null)
                     accessSession.IreleaseBusinessObject(aBO);
                  // fine del codice di rilascio
                      // Chiudere la tabella HTML
          out.println("</TABLE>");
          // Terminare il body della pagina
          out.println("</body></html>");
          } // fine del richiamo
}
```

Parte 3. Riferimento all'API dell'interfaccia di accesso al server

Capitolo 6. Interfaccia IAccessEngine

L'interfaccia `IAccessEngine` fornisce dei metodi per aprire e chiudere una sessione di accesso con InterChange Server Express. La Tabella 7 riepiloga i metodi disponibili nell'interfaccia `IAccessEngine`.

Tabella 7. Metodi disponibili dell'interfaccia IAccessEngine

Metodo	Descrizione	Pagina
IgetInterchangeAccessSession()	Crea una sessione di accesso a InterChange Server Express per il client di accesso.	53
IcloseSession()	Chiude la sessione di accesso con InterChange Server Express.	54
login()	Collega alla sessione	54
logout()	Scollega dalla sessione	55
securelogin()	Collegamento sicuro alla sessione	55
encryptedlogin()	Collegamento alla sessione se LDAP è il registro utente (API interna)	56

I metodi `IgetInterchangeAccessSession()` e `IcloseSession()` non sono più utilizzati nel rilascio 4.3. Essi sono stati sostituiti dai metodi `login()` e `logout()` che offrono una maggiore sicurezza. I metodi non più utilizzati sono ancora supportati, ma è preferibile non utilizzarli e migrare i codici esistenti ai nuovi metodi. Questi metodi verranno rimossi per i rilasci futuri.

Nota: Entrambi i metodi non più utilizzati sono stati modificati per richiamare automaticamente i metodi di sostituzione. Tuttavia, la migrazione completa dei nuovi metodi consente di crittografare i dati dell'utente e della password.

IgetInterchangeAccessSession()

Crea una sessione di accesso a InterChange Server Express per il client di accesso.

Sintassi

```
IInterchangeAccessSession IgetInterchangeAccessSession(
     string nomeutente, string password);
```

Parametri

nomeutente Il nome dell'utente di IBM WebSphere Business Integration Server Express.

password La password per l'utente di IBM WebSphere Business Integration Server Express.

Valori di restituzione

Un oggetto `IInterchangeAccessSession` per la sessione di accesso.

Eccezioni

ICxAccessError	Emesso quando viene rilevato un nome utente o una password non validi.

Notes

Il metodo IgetInterchangeAccessSession() verifica che i valori per *nomeutente* e *password* siano validi per l'istanza InterChange Server Express.

Importante: Il nome utente per questo metodo deve essere admin.

Esempio

```
// Apre la sessione di accesso
String userName = "admin";
String password = "null";
IInterchangeAccessSession aSession =
     serverAccessEngine.IgetInterchangeAccessSession(
     userName,
     password);
```

IcloseSession()

Chiude la sessione di accesso con InterChange Server Express.

Sintassi

```
void IcloseSession(IInterchangeAccessSession sessione);
```

Parametri

sessione	L'oggetto sessione da accesso da chiudere.

Valori di restituzione

Nessuno.

Esempio

```
// Chiude la sessione di accesso
serverAccessEngine.IcloseSession(aSession);
```

login()

Consente di eseguire il collegamento a una sessione di accesso a InterChange Server Express per il client di accesso.

Sintassi

```
IInterchangeAccessSession login(
     string nomeutente, string password);
```

Parametri

nomeutente	Il nome dell'utente di IBM WebSphere Business Integration Server Express.
password	La password per l'utente di IBM WebSphere Business Integration Server Express.

Eccezioni

ICxAccessError Emesso quando viene rilevato un nome utente o una password non validi.

Esempio

```
// Apre la sessione di accesso
String userName = "admin";
String password = "null";
IInterchangeAccessSession aSession =
     serverAccessEngine.login(
     userName,
     password);
```

logout()

Esegue lo scollegamento dalla sessione di accesso con InterChange Server Express.

Sintassi

```
void logout(IInterchangeAccessSession sessione);
```

Parametri

sessione L'oggetto sessione da accesso da terminare.

Esempio

```
// Esegue lo scollegamento dalla sessione di accesso
serverAccessEngine.logout(aSession);
```

securelogin()

Consente di eseguire il collegamento sicuro a una sessione di accesso a InterChange Server Express per il client di accesso.

Sintassi

```
IInterchangeAccessSession securelogin(
     string nomeutente, string hashofpwd);
```

Parametri

nomeutente Il nome dell'utente di IBM WebSphere Business Integration Server Express.

hashofpwd Il valore hash della password corrispondente al *nomeutente*. Il valore hash viene calcolato mediante l'algoritmo SHA1. L'utente immette la versione in testo semplice della password, ma ne viene comunicato il relativo valore hash che fornisce un ambiente più sicuro.

Eccezioni

ICxAccessError Emesso quando viene rilevato un nome utente o un valore hash della password non valido.

Esempio

```
// Apre la sessione di accesso
String userName = "admin";
String password = "passwd";
String hashofpwd = SHA1 algorithm calculated hash value of "passwd";
IInterchangeAccessSession aSession =
     serverAccessEngine.securelogin(
     userName,
     hashofpwd);
```

---

encryptedlogin()

Consente il collegamento della sessione di accesso utilizzando gli stessi parametri di login quando viene utilizzato LDAP come registro utente.

Notes

Questa è un'API interna che l'utente non dovrebbe utilizzare direttamente.

Capitolo 7. Interfaccia IInterchangeAccessSession

L'interfaccia `IInterchangeAccessSession` fornisce i metodi per la creazione di oggetti business e l'esecuzione di collaborazioni. La Tabella 8 riepiloga i metodi dell'interfaccia `IInterchangeAccessSession`.

Tabella 8. Metodi disponibili Interfaccia IInterchangeAccessSession

Metodo	Descrizione	Pagina
IcreateBusinessObject()	Crea un oggetto business da una definizione di oggetto business.	57
IcreateBusinessObjectArray()	Crea un'array di oggetti business che contiene uno o più elementi, ognuno dei quali ha un determinato oggetto business come tipo.	58
IcreateBusinessObjectFrom()	Converte i dati serializzati nel formato MIME specificato in un oggetto business IBM WebSphere Business Integration Server Express .	59
IcreateBusinessObjectWithVerb()	Crea un oggetto business con una istruzione specificata.	59
IexecuteCollaboration()	Esegue una collaborazione inviando un oggetto business come richiamo dei dati di accesso nella richiesta di accesso.	60
IexecuteCollaborationExtFmt()	Esegue una collaborazione, inviando i dati serializzati come richiamo dei dati di accesso nella richiesta di accesso.	61
IreleaseBusinessObject()	Rilascia le risorse di un oggetto business.	63
IreleaseBusinessObjectArray()	Rilascia le risorse di una array di oggetti business.	64
setLocale(String)	Imposta la locale.	64
login()	Esegue il collegamento al sistema	65
logout()	Esegue lo scollegamento dal sistema	65

IcreateBusinessObject()

Crea un oggetto business da una definizione di oggetto business.

Sintassi

```
IBusinessObject IcreateBusinessObject(string nomeoggbus);
```

Parametri

nomeoggbus Il nome della definizione di oggetto business da utilizzare nella creazione dell'oggetto business.

Valori di restituzione

Un oggetto `IBusinessObject` che contiene il nuovo oggetto business.

Eccezioni

ICxAccessError	Emesso quando la definizione di oggetto business specificata *non* è presente nel repository di IBM WebSphere Business Integration Server Express.

Notes

L'interfaccia di Server Access Interface crea un oggetto business di tipo *nomeoggbus* e lo invia al client di accesso.

Esempio

Il seguente frammento di codice crea un oggetto business:

```
// Questo metodo crea un oggetto business
// Dichiara l'oggetto
IBusinessObject exampleObj = null;
exampleObj = aSession.IcreateBusinessObject("PayablesNetChange");
```

IcreateBusinessObjectArray()

Crea un'array di oggetti business che contiene uno o più elementi, ognuno dei quali ha un determinato oggetto business come tipo.

Sintassi

```
IBusinessObjectArray IcreateBusinessObjectArray(string nomeoggbus);
```

Parametri

nomeoggbus	Il nome della definizione di oggetto business da utilizzare nella creazione dell'oggetto business nell'array di oggetti business.

Valori di restituzione

Un oggetto `IBusinessObjectArray` che contiene la nuova array di oggetti business.

Eccezioni

ICxAccessError	Emesso quando la definizione di oggetto business specificata *non* è presente nel repository di IBM WebSphere Business Integration Server Express.

Notes

L'interfaccia Server Access Methods crea un'array di oggetti business e la restituisce al client di accesso. Il metodo `IcreateBusinessObjectArray()` restituisce un oggetto `IBusinessObjectArray`. Altri metodi nell'interfaccia `IBusinessObjectArray` consentono di modificare l'array di oggetti business.

Esempio

Nel seguente esempio viene creata un'array di oggetti business:

```
// Dichiara l'array
IBusinessObjectArray exampleObjArray = null;
// Crea l'array di oggetti business che contiene gli oggetti business
```

```
// "CustomerAcct"
exampleObjArray =
     accessSession.IcreateBusinessObjectArray("CustomerAcct");
```

IcreateBusinessObjectFrom()

Converte i dati serializzati nel formato MIME specificato in un oggetto business IBM WebSphere Business Integration Server Express per l'oggetto business.

Sintassi

```
IBusinessObject IcreateBusinessObjectFrom(string datiserializzati,
     string tipomime);
```

Parametri

datiserializzati	I dati serializzati in ingresso.
tipomime	Il tipo MIME dei dati *datiserializzati*.

Valori di restituzione

Un oggetto `IBusinessObject` che contiene l'oggetto business che il gestore dati crea dai dati *datiserializzati*.

Eccezioni

ICxAccessError	Emesso quando i dati non possono essere convertiti in un oggetti business o se non è possibile accedere al gestore dati.

Notes

Il metodo `IcreateBusinessObjectFrom()` invia i dati *datiserializzati* al tipo MIME *tipomime* specificato a InterChange Server Express. L'interfaccia Server Access all'interno di InterChange Server Express richiama il gestore dati necessario per convertire il tipo MIME specificato in un oggetto business IBM WebSphere Business Integration Server Express compatibile con l'ambiente IBM WebSphere Business Integration Server Express. I dati *datiserializzati* devono specificare il nome della definizione dell'oggetto business da utilizzare nella creazione dell'oggetto. Il gestore dati analizza e converte i dati in un oggetto business, lo restituisce all'interfaccia Server Access all'interno di InterChange Server Express che a sua volta lo restituisce al client di accesso. Il formato esterno dei dati serializzati deve essere di un tipo supportato da un gestore dati (IBM WebSphere Business Integration Server Express o un gestore dati personalizzato). Per ulteriori informazioni, fare riferimento al manuale *Guida per il gestore dati*.

Esempio

```
// Dichiara l'oggetto
String custData = "exampleXmlData";
String mimeType = "text/Xml";
IBusinessObject exampleObj = null;
// Questo metodo crea l'oggetto business dai dati in formato XML
exampleObj =
     accessSession.IcreateBusinessObjectFrom(custData, mimeType);
```

IcreateBusinessObjectWithVerb()

Crea un oggetto business con una dell'oggetto business.

Sintassi

```
IBusinessObject IcreateBusinessObjectWithVerb(string nomeoggbus,
     string istruzione);
```

Parametri

nomeoggbus Il nome della definizione di oggetto business da utilizzare nella creazione dell'oggetto business.

istruzione L'istruzione per il nuovo oggetto business.

Valori di restituzione

Un oggetto `IBusinessObject` che contiene il nuovo oggetto business con il valore *istruzione* specificato.

Eccezioni

`ICxAccessError` Emesso quando la definizione di oggetto business specificata *non* è presente nel repository IBM WebSphere Business Integration Server Express o se l'*istruzione* inoltrata non è valida per la definizione dell'oggetto business.

Notes

L'interfaccia Server Access Interface crea un oggetto business di tipo `nomeoggbus` e lo inizializza con l'istruzione *istruzione*. Quindi lo restituisce al client di accesso. Sono valide soltanto le istruzioni supportate nelle definizioni di oggetti business.

Esempio

```
// Crea l'oggetto business
IBusinessObject exampleobj = null
exampleObj =
     accessSession.IcreateBusinessObjectWithVerb("AcctsRecCurrent",
     "Retrieve");
```

IexecuteCollaboration()

Esegue una collaborazione inviando un oggetto business come richiamo dei dati di accesso nella richiesta di accesso.

Sintassi

```
IBusinessObject IexecuteCollaboration
     (string nomecollab, string nomeporta, IBusinessObject oggbus);
```

Parametri

nomecollab Il nome della collaborazione da eseguire.

nomeporta Il nome della porta di collaborazione esterna a cui è collegato il client di accesso.

oggbus L'oggetto business generico che contiene i dati di accesso di richiamo per la collaborazione.

Valori di restituzione

Un oggetto `IBusinessObject` che contiene l'oggetto business restituito dalla collaborazione.

Eccezioni

`IExecuteCollaborationError`

Emesso quando la collaborazione non è attiva o se le associazioni non sono riuscite. Questa eccezione contiene un valore di stato impostato su una delle seguenti costanti per indicare i dettagli della chiamata quando si verifica l'eccezione. Per ulteriori informazioni su come accedere a questo stato, fare riferimento a "IExecuteCollaborationError" a pagina 94.

Nome costante	Descrizione
`UNKNOWNSTATUS`	Lo stato della chiamata al metodo `IexecuteCollaboration()` è sconosciuto.
`FAILEDTOREACHCOLLABORATION`	La richiesta di accesso non ha raggiunto la collaborazione.
`FAILEDINEXECUTIONOFCOLLABORATION`	La richiesta di accesso ha riportato un errore durante l'esecuzione della collaborazione.
`FAILEDINRETURNTOCLIENT`	La collaborazione è stata eseguita ma si è verificato un errore durante la distribuzione della risposta al client di accesso.

Notes

Il metodo `IexecuteCollaboration()` ha richiesto l'esecuzione della collaborazione *`nomecollab`*. Per iniziare la collaborazione, l'interfaccia Server Access Interface invia i dati di accesso di nell'oggetto business *`oggbus`* alla porta *`nomeporta`* della collaborazione *`nomecollab`*. Questa porta deve essere configurata come porta esterna in modo che possa supportare il flusso di chiamate emesse.

Nota: La collaborazione, la porta e l'oggetto business devono essere configurati e associati per il flusso di chiamate e per la modifica.

Esempio

```
String portName = "From";
IBusinessObject srcBO =
     accessSession.IcreateBusinessObject ("payableNetChange");

// imposta gli attributi srcBO, le istruzioni o entrambi
...
// Esegue la collaborazione
IBusinessObject resultantBO = null;
resultantBO = accessSession.IexecuteCollaboration(
     "getCustAcctPayable",
     portName,
     srcBO);
```

IexecuteCollaborationExtFmt()

Esegue una collaborazione, inviando i dati serializzati come richiamo dei dati di accesso nella richiesta di accesso.

Sintassi

```
string IexecuteCollaborationExtFmt(string nomecollab, string nomeporta,
          string datiserializzati, string tipomime, string istruzione);
```

Parametri

nomecollab	Il nome della collaborazione da eseguire.
nomeporta	Il nome della porta di collaborazione esterna a cui è collegato il client di accesso.
datiserializzati	I dati serializzati che rappresentano i dati di accesso di richiamo.
tipomime	Il formato esterno (come un tipo MIME) dei dati serializzati.
istruzione	Il valore per l'istruzione specificata dell'oggetto business.

Valori di restituzione

Una `stringa` che contiene la versione serializzata dell'oggetto business restituito dalla collaborazione. Questa stringa è nel formato esterno *tipomime*.

Eccezioni

`IExecuteCollaborationError`

Emesso quando la collaborazione non è attiva o se le associazioni non sono riuscite. Questa eccezione contiene un valore di stato impostato su una delle seguenti costanti per indicare i dettagli della chiamata quando si verifica l'eccezione. Per ulteriori informazioni su come accedere a questo stato, fare riferimento a "IExecuteCollaborationError" a pagina 94.

Nome costante	Descrizione
`UNKNOWNSTATUS`	Lo stato della chiamata al metodo `IexecuteCollaborationExtFmt()` è sconosciuto.
`FAILEDTOREACHCOLLABORATION`	La richiesta di accesso non ha raggiunto la collaborazione.
`FAILEDINEXECUTIONOFCOLLABORATION`	La richiesta di accesso ha riportato un errore durante l'esecuzione della collaborazione.
`FAILEDINRETURNTOCLIENT`	La collaborazione è stata eseguita ma si è verificato un errore durante la distribuzione della risposta al client di accesso.

Notes

Il metodo `IexecuteCollaborationExtFmt()` esegue la stessa attività di base del metodo `IexecuteCollaboration()`: richiede l'esecuzione della collaborazione *nomecollab*. La differenza principale sta nel fatto che questo metodo consente di eseguire le seguenti attività con un'unica chiamata:

- Convertire i dati *datiserializzati* in un oggetto business mediante il gestore dati appropriato per il tipo MIME *tipomime* dei dati. Questo oggetto business rappresenta i dati di accesso di richiamo per la collaborazione.
- Impostare l'istruzione dell'oggetto business sul valore `istruzione`.

- Inviare l'oggetto business alla porta *nomeporta* della collaborazione per avviare l'esecuzione della collaborazione. Questa porta deve essere configurata come porta esterna in modo che possa supportare il flusso di chiamate emesse.

Nota: Nessun oggetto CORBA inviato da o verso questo metodo.

La collaborazione e la porta devono essere configurati e associati per il flusso di chiamate e per la modifica.

Il parametro *tipomime* specifica il formato esterno dei dati serializzati per l'oggetto business. Server Access Interface utilizza questo tipo MIME per determinare quale gestore dati richiamare per analizzare e convertire i dati in un oggetto business IBM WebSphere Business Integration Server Express. Il formato esterno dei dati serializzati deve essere di un tipo supportato da un gestore dati (IBM WebSphere Business Integration Server Express o un gestore dati personalizzato). Per ulteriori informazioni sulla gestione di dati, fare riferimento al manuale *Guida per il gestore dati*.

Esempio

```
String portName = "From";
// Esegue la collaborazione
IBusinessObject resultantBO = null;
resultantBO = accessSession.IexecuteCollaborationExtFmt(
     "getCustAcctPayable",
     portName,
     serializedXMLData,
     "text/xml",
     "Create");
```

IreleaseBusinessObject()

Rilascia le risorse di una oggetto business.

Sintassi

```
void IreleaseBusinessObject(IBusinessObject oggettorilascio);
```

Parametri

oggettorilascio	L'oggetto business le cui risorse sono rilasciate.

Valori di restituzione

Nessuno.

Notes

Quando il client di accesso viene terminato mediante un oggetto business, il metodo `IreleaseBusinessObject()` dovrebbe liberare l'oggetto `IBusinessObject` nella memoria di InterChange Server Express.

Esempio

```
// Crea l'oggetto business
IBusinessObj anObject = null;
accessSession.IcreateBusinessObjectWithVerb("AcctsRecCurrent",
     "Retrieve");
// Rilascia l'oggetto
accessSession.IreleaseBusinessObject(anObject);
```

IreleaseBusinessObjectArray()

Rilascia le risorse di una array di oggetti business.

Sintassi

```
void IreleaseBusinessObjectArray(IBusinessObjectArray oggettorilascio);
```

Parametri

oggettorilascio	L'array di oggetti business le cui risorse sono rilasciate.

Valori di restituzione

Nessuno.

Notes

Quando il client di accesso viene terminato utilizzando un'array di oggetti business, il metodo `IreleaseBusinessObjectArray()` dovrebbe liberare l'oggetto `IBusinessObjectArray` nella memoria InterChange Server Express.

Esempio

```
// Crea l'array
IBusinessObjectArray exampleObjArray = null;
exampleObjArray =
     accessSession.IcreateBusinessObjectArray("CustomerAcct");
// Rilascia l'array di oggetti
accessSession.IreleaseBusinessObjectArray(exampleObjArray);
```

setLocale(String)

Imposta la locale dell'oggetto della sessione dell'interfaccia di accesso.

Sintassi

```
public String setLocale(String);
```

Parametri

Una stringa che definisce la locale, in questo formato:

ll_TT

dove *ll* è il codice della lingua a due caratteri (di solito in minuscolo) e *TT* è un codice del paese a due lettere facoltatvio (di solito in maiuscolo). Ad esempio, le seguenti stringhe sono locali valide:

```
en
de_DE
```

Notes

Il metodo `setLocale()` imposta la locale per l'oggetto della sessione dell'interfaccia di accesso. La locale definisce le convenzioni culturali per i dati in base alla lingua e al paese.

Per impostazione predefinita, la locale utilizzata all'inizio dell'oggetto di una sessione è la stesa utilizzata da InterChange Server Express. Quando viene

effettuata una chiamata al metodo setLocale() per modificare una nuova locale, le chiamate ai metodi successivi della sessione utilizzeranno la nuova locale.

login()

Esegue il collegamento a una sessione di accesso a InterChange Server Express.

Sintassi

```
IInterchangeAccessSession login (string nomeutente, string password)
```

Parametri

nomeutente
Il nome dell'utente di IBM WebSphere Business Integration Server Express.

password
La password per l'utente di IBM WebSphere Business Integration Server Express.

Esempio

```
// Apre la sessione di accesso
String userName = "admin";
String password = "null";
IInterchangeAccessSession aSession =
     serverAccessEngine.login(
     userName,
     password);
```

logout()

Esegue lo scollegamento dalla sessione di accesso con InterChange Server Express.

Sintassi

```
void logout (IInterchangeAccessSession sessione)
```

Parametri

sessione L'oggetto sessione da accesso da terminare.

Esempio

```
// Esegue lo scollegamento dalla sessione di accesso
serverAccessEngine.logout(aSession);
```

Capitolo 8. Interfaccia IBusinessObject

L'interfaccia `IBusinessObject` fornisce i metodi che operano su oggetti di tipo `BusinesssObject`. Tali oggetti rappresentano gli oggetti business di sistema IBM WebSphere Business Integration Server Express definiti nel repository IBM WebSphere. La Tabella 9 riepiloga i metodi disponibili nell'interfaccia `IBusinessObject`.

Tabella 9. Metodi disponibili Interfaccia IBusinessObject

Metodo	Descrizione	Pagina
Iduplicate()	Crea un clone dell'oggetto business.	69
Iequals()	Confronta i valori dell'attributo di questo oggetto business con quelli dell'oggetto business di immissione.	69
IequalsKeys()	Confronta i valori dell'attributo chiave di questo oggetto business con quelli dell'oggetto business di immissione.	70
IgetAppSpecificInfo()	Richiama le informazioni specifiche dell'applicazione per l'attributo.	70
IgetAttributeCount()	Richiama il numero di attributi per l'oggetto business.	71
IgetAttributeName()	Richiama il nome dell'attributo alla posizione specificata nella definizione dell'oggetto business.	71
IgetAttributeType()	Richiama il tipo di attributo.	72
IgetAttributeTypeAtIndex()	Richiama il tipo di attributo alla posizione specificata nella definizione dell'oggetto business.	73
IgetBooleanAttribute()	Richiama un valore `booleano` di un attributo.	73
IgetBOAppSpecification()	Richiama il valore di un attributo che è un'array di oggetti business (cardinalità multipla).	74
IgetBusinessObjectArrayAttribute()	Richiama il valore di un oggetto business attributo che è un'array di oggetti business (cardinalità multipla).	75
IgetBusinessObjectAttribute()	Richiama il valore di un attributo di cardinalità singola.	75
IgetDateAttribute()	Richiama il valore della data attributo.	76
IgetDefaultValue()	Richiama il valore predefinito attributo.	76
IgetDoubleAttribute()	Richiama un valore `doppio` di un attributo.	77
IgetFloatAttribute()	Richiama un valore `a virgola mobile` di un attributo.	77
IgetIntAttribute()	Richiama un valore `intero` di un attributo.	78
IgetLongTextAttribute()	Richiama un valore `longtext` di un attributo.	79

Tabella 9. Metodi disponibili Interfaccia IBusinessObject (Continua)

Metodo	Descrizione	Pagina
IgetName()	Richiama il nome della definizione di oggetto business.	80
IgetStringAttribute()	Richiama un valore `stringa` di un attributo.	80
IgetVerb()	Richiama il verb per l'oggetto di business.	80
IisAttributeMultipleCardinality()	Determina se l'attributo ha più cardinalità.	81
IisBlankValue()	Determina se il valore dell'attributo è vuoto.	81
IisIgnoreValue()	Determina se il valore dell'attributo è ignore.	82
IisKey()	Determina se l'attributo è critico.	83
IisRequired()	Determina se il valore dell'attributo è obbligatorio.	83
Iserialize()	Restituisce i dati dell'attributo in un formato leggibile (serializzato).	84
IsetAttributes()	Imposta gli attributi in un oggetto di business dai dati serializzati in un tipo MIME specificato.	84
IsetAttributeToBlank()	Imposta l'attributo in un oggetto di business su un valore vuoto.	84
IsetAttributeToIgnore()	Imposta un attributo in un oggetto di business su ignore.	85
IsetBooleanAttribute()	Imposta un attributo su un valore `booleano`.	85
IsetBusinessObjectArrayAttribute()	Imposta il valore di un attributo che è un'array di oggetti business (cardinalità multipla).	86
IsetBusinessObjectAttribute()	Imposta il valore di un attributo di cardinalità singola.	87
IsetDateAttribute()	Imposta un attributo su un valore della data.	87
IsetDoubleAttribute()	Imposta un attributo su un valore `doppio`.	88
IsetFloatAttribute()	Imposta un attributo su un valore `a virgola mobile`.	88
IsetIntAttribute()	Imposta un attributo su un valore `intero`.	89
IsetLongTextAttribute()	Imposta un attributo su un valore `long text`.	89
IsetStringAttribute()	Imposta un attributo su un valore `stringa`.	90
IsetVerb()	Imposta l' verb per l'oggetto di business.	90
ItoExternalForm()	Serializza i dati di un oggetto di business in un formato esterno del tipo MIME specificato.	91
ItoString()	Serializza i oggetto business dati utilizzando un formato IBM WebSphere Business Integration Server Express.	91

Iduplicate()

Crea un clone per l'oggetto business.

Sintassi

```
IBusinessObject Iduplicate();
```

Parametri

Nessuno.

Valori di restituzione

Un oggetto `IBusinessObject` che contiene l'oggetto di business duplicato.

Eccezioni

`ICxAccessError`	Emesso quando l'oggetto non viene trovato.

Notes

Il metodo `Iduplicate()` esegue una copia dell'oggetto di business e la restituisce. È necessario assegnare esplicitamente il valore di restituzione di questa chiamata al metodo a una variabile dichiarata di tipo `IBusinessObject`.

Esempio

Nel seguente esempio viene duplicato `sourceCustomer` in modo da creare `destCustomer`.

```
IBusinessObject destCustomer = sourceCustomer.Iduplicate();
```

Iequals()

Confronta i valori dell'attributo chiave di questo oggetto business con quelli dell'oggetto business di immissione.

Sintassi

```
boolean Iequals(IBusinessObject obj2);
```

Parametri

obj2 L'oggetto di business da confrontare.

Valori di restituzione

Restituisce `true` se i valori di *tutti* gli attributi e dei verbi sono gli stessi, altrimenti restituisce `false`.

Notes

Il metodo `Iequals()` confronta i valori degli attributi di questo oggetto aziendale con i valori nell'oggetto aziendale di input. Se gli oggetti sono contenuti in una gerarchia, il confronto include *tutti* gli attributi degli oggetti secondari. I valori dei verbi e degli attributi devono corrispondere.

Nel confronto, un valore null è considerato equivalente a qualsiasi valore a cui viene confrontato e *non* impedisce la restituzione del valore `true`.

Esempio

Nel seguente esempio sono confrontati i verbi e gli attributi di `order2` con tutti gli attributi di `order1`:

```
boolean isEqual = false;
IBusinessObject order1 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "create");
IBusinessObject order2 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "create");
isEqual = order1.Iequals(order2);
if(isEqual)
     System.out.println("order1 is the same as order2")
else
     System.out.println("order1 is not the same as order2");
```

IequalsKeys()

Confronta i valori dell'attributo chiave di questo oggetto business con quelli dell'oggetto business di immissione.

Sintassi

```
boolean IequalsKeys(IBusinessObject obj2);
```

Parametri

obj2 Un oggetto di business da valutare per il confronto.

Valori di restituzione

Restituisce `true` se i valori di *tutti* gli attributi chiave sono gli stessi, altrimenti restituisce `false`.

Notes

Il metodo `IequalsKeys()` esegue un confronto superficiale, ovvero *non confronta* tutti i valori degli oggetti di business secondari.

Esempio

Nel seguente esempio vengono confrontati gli attributi chiave di `order2` con gli attributi chiave di `order1`, escludendo gli attributi per gli oggetti di business secondari, se presenti:

```
boolean keyEqual = false;
IBusinessObject order1 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "retrieve");
IBusinessObject order2 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "retrieve");
keyEqual = order1.IequalsKeys(order2);
if(keyEqual)
     System.out.println("order1 is the same as order2")
else
     System.out.println("order1 is not the same as order2");
```

IgetAppSpecificInfo()

Richiama le informazioni specifiche dell'applicazione per attributo.

Sintassi

```
string IgetAppSpecificInfo(string nomeattributo)
```

Parametri

nomeattributo Il nome dell'attributo.

Valori di restituzione

Una `stringa` che contiene le informazioni sull'applicazione associate all'attributo specificato.

Eccezioni

`IInvalidAttributeNameException`	Emesso quando il nome dell'attributo non è valido.
`IValueNotSetException`	Emesso quando l'attributo non contiene informazioni specifiche dell'applicazione.

Notes

Il metodo `IgetAppSpecificInfo()` può restituire un risultato nullo.

Esempio

```
// Qeusto metodo determina le info specifiche dell'applicazione di un attributo
String appSpecificInfo;
appSpecificInfo = aBusObj.IgetAppSpecificInfor();
```

IgetAttributeCount()

Richiama il numero di attributi per l'oggetto business.

Sintassi

```
long IgetAttributeCount();
```

Parametri

Nessuno.

Valori di restituzione

Un valore intero per indicare il numero di attributi nell'oggetto aziendale corrente.

Esempio

```
long attributeCount = 0;
attributeCount = aBusObj.IgetAttributeCount();
```

IgetAttributeName()

Richiama il nome dell'attributo alla posizione specificata nella definizione dell'oggetto business.

Sintassi

```
string IgetAttributeName(long position);
```

Parametri

posizione La posizione di un attributo in una definizione di un oggetto aziendale.

Valori di restituzione

Una `stringa` che contiene il nome dell'attributo alla posizione specificata nella definizione dell'oggetto aziendale.

Eccezioni

`IInvalidIndexException` Emesso quando l'indice della posizione non è valido.

Esempio

```
int position = 1;
String attribute name;
attributeName = aBusObj.IgetAttributeName(position);
```

IgetAttributeType()

Richiama il tipo di attributo.

Sintassi

```
long IgetAttributeType(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui viene restituito il tipo.

Valori di restituzione

Un valore intero per indicare il tipo di dati dell'attributo specificato nell'oggetto aziendale, come mostrato di seguito:

0 `Oggetto`

1 `booleano`

2 `int`

3 `float`

4 `doppio`

5 `stringa`

6 `data`

7 `testo lungo`

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
String attributeName = "Nome";
long attributeType = 0;
attributeType = aBusObj.IgetAttributeType(attributeName);
```

IgetAttributeTypeAtIndex()

Richiama il tipo di attributo alla posizione specificata nella definizione dell'oggetto business.

Sintassi

```
long IgetAttributeTypeAtIndex(long position);
```

Parametri

posizione La posizione dell'attributo nella definizione dell'oggetto aziendale.

Valori di restituzione

Un valore intero per indicare il tipo di dati di un attributo alla posizione specificata nell'oggetto aziendale, come mostrato di seguito:

0 Oggetto

1 booleano

2 int

3 float

4 doppio

5 stringa

6 data

7 testo lungo

Eccezioni

IInvalidIndexException	Emesso quando l'indice della posizione non è valido.

Esempio

```
int indexPosition = 1;
long attributeType = 0;
attributeType = aBusObj.IgetAttributeTypeAtIndex(indexPosition);
```

IgetBooleanAttribute()

Richiama un valore booleano di un attributo.

Sintassi

```
boolean IgetBooleanAttribute(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo booleano del quale è stato richiamato il valore.

Valori di restituzione

Il valore booleano dell'attributo.

Eccezioni

`IAttributeNotSetException`	Emesso quando il valore dell'attributo non è impostato.
`IInvalidAttributeNameException`	Emesso quando il nome dell'attributo non è valido.
`InvalidAttributeTypeException`	Emesso quando l'attributo non è del tipo data `booleano`.
`IAttributeBlankException`	Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiama il metodo booleano
String booleanAttribute = "MyBooleanAttribute";
boolean value = exampleBusObj.IgetBooleanAttribute(booleanAttribute);
```

IgetBOAppSpecification()

Richiama le informazioni specifiche dell'applicazione.

Sintassi

```
public String IgetBOAppSpecificInfo();
```

Parametri

Questo metodo non contiene parametri di input.

Valori di restituzione

Un oggetto `IgetBOAppSpecificInfo()` che contiene informazioni specifiche dell'applicazione per l'applicazione aziendale.

Eccezioni

`IValueNotSetException`	Emesso quando il valore dell'attributo non è valido.

IgetBusinessObjectArrayAttribute()

Richiama il valore di un attributo che è un'array di oggetti business (cardinalità multipla).

Sintassi

```
IBusinessObjectArray IgetBusinessObjectArrayAttribute(
     string attributeName);
```

Parametri

nomeattributo
: Il nome dell'attributo multiple-cardinality di cui è stato richiamato il valore.

Valori di restituzione

Un oggetto `IBusinessObjectArray` che contiene il valore dell'attributo multiple-cardinality.

Eccezioni

`IAttributeNotSetException`	Emesso quando il valore dell'attributo non è impostato.
`IInvalidAttributeNameException`	Emesso quando il nome dell'attributo non è valido.
`InvalidAttributeTypeException`	Emesso quando il l'attributo non è un attributo single-cardinality (è qualche altro tipo di data).
`IAttributeBlankException`	Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiamare il metodo BusinessObjectArray e richiamare l'attributo
String arrayAttribute = "Account";
IBusinessObjectArray aBusObj =
     exampleBusObj.IgetBusinessObjectArrayAttribute(arrayAttribute);
```

IgetBusinessObjectAttribute()

Richiama il valore di un attributo di cardinalità singola.

Sintassi

```
IBusinessObject IgetBusinessObjectAttribute(string attributeName);
```

Parametri

nomeattributo
Il nome dell'attributo single-cardinality di cui è stato richiamato il valore.

Valori di restituzione

Un oggetto `IBusinessObject` che contiene il valore dell'attributo single-cardinality.

Eccezioni

`IAttributeNotSetException`	Emesso quando il valore dell'attributo non è impostato.
`IInvalidAttributeNameException`	Emesso quando il nome dell'attributo non è valido.
`InvalidAttributeTypeException`	Emesso quando il l'attributo non è un attributo single-cardinality (è qualche altro tipo di data).
`IAttributeBlankException`	Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiamare il metodo dell'oggetto aziendale e richiamare l'attributo
String busObjAttribute = "Customer";
IBusinessObject aBusObj =
     exampleBusObj.IgetBusinessObjectAttribute(busObjAttribute);
```

IgetDateAttribute()

Richiama il valore della data attributo.

Sintassi

```
string IgetDateAttribute(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo date di cui è stato restituito il valore.

Valori di restituzione

Una `stringa` che contiene il valore dell'attributo date.

Eccezioni

`IAttributeNotSetException` Emesso quando il valore dell'attributo non è impostato.

`IInvalidAttributeNameException` Emesso quando il nome dell'attributo non è valido.

`InvalidAttributeTypeException` Emesso quando l'attributo non è del tipo date.

`IAttributeBlankException` Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
//Richiamare il metodo date e richiamare l'attributo
String dateAttributeName = "DateOfBirth";
String aDate;
aDate = exampleBusObj.IgetDateAttribute(dateAttributeName);
```

IgetDefaultValue()

Richiama il valore predefinito attributo.

Sintassi

```
string IgetDefaultValue(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui è stato restituito il valore predefinito.

Valori di restituzione

Una `stringa` che contiene il valore predefinito dell'attributo.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

`IValueNotSetException` Emesso quando l'attributo non contiene alcun valore predefinito.

Esempio

```
// Richiamare il metodo del valore predefinito
String attributeName = "Nome";
String defaultAttributeValue;
defaultAttributeValue =
     exampleBusObj.IgetDefaultValue (attributeName);
```

IgetDoubleAttribute()

Richiama un valore `doppio` di un attributo.

Sintassi

```
double IgetDoubleAttribute(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui è stato richiamato il valore `double`.

Valori di restituzione

Il valore `double` di un attributo.

Eccezioni

`IAttributeNotSetException` Emesso quando il valore dell'attributo non è impostato.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

`InvalidAttributeTypeException`
Emesso quando l'attributo non è di tipo `double`.

`IAttributeBlankException` Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiamare il metodo double e richiamare l'attributo
double doubleValue = 0;
String doubleAttributeName = "Average";
doubleValue = exampleBusObj.IgetDoubleAttribute(doubleAttributeName);
```

IgetFloatAttribute()

Richiama un valore `float` value di un attributo.

Sintassi

```
float IgetFloatAttribute(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui è stato restituito il valore `float`.

Valori di restituzione

Il valore `float` dell'attributo.

Eccezioni

`IAttributeNotSetException` Emesso quando il valore dell'attributo non è impostato.

`IInvalidAttributeNameException` Emesso quando il nome dell'attributo non è valido.

`InvalidAttributeTypeException` emesso quando l'attributo non è del tipo `float`.

`IAttributeBlankException` Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiamare il metodo Float e richiamare l'attributo
float floatValue = 0.0;
String floatAttributeName = "Height";
floatValue = exampleBusObj.IgetFloatAttribute(floatAttributeName);
```

IgetICSVersion()

Richiama il numero di versione del framework di InterChange.

Sintassi

```
public String IgetICSVersion();
```

Parametri

Nessun parametro di input

Valori di restituzione

Restituisce il numero della versione del framework di InterChange.

Eccezioni

Questo metodo non emette eccezioni.

IgetIntAttribute()

Richiama un valore `intero` di un attributo.

Sintassi

```
long IgetIntAttribute(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui viene restituito il valore.

Valori di restituzione

Un valore `long` che contiene il valore intero dell'attributo.

Eccezioni

`IAttributeNotSetException` Emesso quando il valore dell'attributo non è impostato.

`IInvalidAttributeNameException` Emesso quando il nome dell'attributo non è valido.

`InvalidAttributeTypeException` Emesso quando l'attributo non è di tipo intero.

`IAttributeBlankException` Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiamare il metodo int e richiamare l'attributo
int intValue = 1;
String intAttributeName = "priority";
intValue = exampleBusObj.IgetIntAttribute(intAttributeName);
```

IgetLongTextAttribute()

Richiama un valore `longtext` di un attributo.

Sintassi

```
string IgetLongTextAttribute(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui è stato richiamato il valore `longtext`.

Valori di restituzione

Il valore `longtext` dell'attributo come stringa.

Eccezioni

`IAttributeNotSetException`
Emesso quando il valore dell'attributo non è impostato.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

`InvalidAttributeTypeException`
Emesso quando l'attributo non è di tipo `longtext`.

`IAttributeBlankException`
Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiamare il metodo LongText e richiamare l'attributo
long longValue = "net30";
String longAttributeName = "Customer";
longValue = exampleBusObj.IgetLongTextAttribute(longAttributeName);
```

IgetName()

Richiama il nome della definizione di oggetto business.

Sintassi

```
string IgetName();
```

Parametri

Nessuno.

Valori di restituzione

Un valore `string` che contiene il nome della definizione dell'oggetto aziendale.

Esempio

```
// Ottenere il nome della definizione dell'oggetto aziendale
String busObjName;
busObjName = exampleBusObj.IgetName();
```

IgetStringAttribute()

Richiama un valore `stringa` di un attributo.

Sintassi

```
string IgetStringAttribute(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui viene richiamato il valore `string`.

Valori di restituzione

Un valore `string` che contiene il valore dell'attributo.

Eccezioni

`IAttributeNotSetException`
: Emesso quando il valore dell'attributo non è impostato.

`IInvalidAttributeNameException`
: Emesso quando il nome dell'attributo non è valido.

`InvalidAttributeTypeException`
: Emesso quando l'attributo non è del tipo `string`.

`IAttributeBlankException`
: Emesso quando l'attributo contiene un valore vuoto.

Esempio

```
// Richiamare il metodo String e richiamare l'attributo
String stringValue = "declined";
String stringAttributeName = "SalesOrder";
stringValue = exampleBusObj.IgetStringAttribute(stringAttributeName);
```

IgetVerb()

Richiama il verb per l'oggetto di business.

Sintassi

```
string IgetVerb();
```

Parametri

Nessuno.

Valori di restituzione

Un valore `string` che contiene il verb dell'oggetto aziendale, il quale potrebbe essere `null`.

Eccezioni

`IVerbNotSetException`
Emesso quando il verb non è impostato.

Esempio

```
// Ottenere il verb dell'oggetto aziendale.
String busObjName;
busObjName = exampleBusObj.IgetVerb();
```

IisAttributeMultipleCardinality()

Determina se l'attributo ha più cardinalità.

Sintassi

```
boolean IisAttributeMultipleCardinality(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo di cui viene stabilita la cardinalità.

Valori di restituzione

Restituisce `true` se l'attributo contiene più cardinalità; altrimenti, restituisce `false`.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo di più cardinalità.
boolean multCard = false;
String busAttribute = "AttributeName";
multCard =
     exampleBusObj.IisAttributeMultipleCardinality(busAttribute);
if (multCard)
     System.out.println ("attribute is multiple cardinality");
else
     System.out.println ("attribute is not multiple cardinality");
```

IisBlankValue()

Determina se il valore dell'attributo valore vuoto.

Sintassi

```
boolean IisBlankValue(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo del quale si verifica che il valore sia vuoto o meno.

Valori di restituzione

Restituisce `true` se il valore dell'attributo è un valore vuoto; altrimenti, restituisce `false`.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Verificare se l'attributo è vuoto
boolean isBlank = false;
String busAttribute = "AttributeName";
isBlank = exampleBusObj.IisBlankValue(busAttribute);
if (isBlank)
     ...
```

IisIgnoreValue()

Determina se il valore dell'attributo è ignore.

Sintassi

```
boolean IisIgnoreValue(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo del quale si verifica se il valore è "ignore".

Valori di restituzione

Restituisce `true` se il valore dell'attributo è "ignore"; altrimenti, viene restituito `false`.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

`IValueNotSetException`
Emesso quando l'attributo non contiene alcun valore predefinito.

Esempio

```
// Richiamare il metodo ignore dell'attributo
boolean isIgnore = false;
String busAttribute = "AttributeName";
isIgnore = exampleBusObj.IisIgnoreValue(busAttribute);
if (isIgnore)
     ...
```

IisKey()

Determina se l'attributo è critico.

Sintassi

```
boolean IisKey(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo per il quale si verifica che sia una chiave o meno.

Valori di restituzione

Il metodo restituisce `true` se l'attributo è una chiave, altrimenti, restituisce `false`.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Verificare che l'attributo è una chiave
boolean isKey = false;
String busAttribute = "AttributeName";
isKey = exampleBusObj.IisKey(busAttribute);
if (isKey)
     ...
```

IisRequired()

Determina se il valore dell'attributo è obbligatorio.

Sintassi

```
boolean IisRequired(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo per il quale si verifica che sia richiesto o meno.

Valori di restituzione

Restituisce `true` se l'attributo è richiesto; altrimenti, restituisce `false`.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo isRequired
boolean isReq = false;
String busAttribute = "AttributeName";
isReq = exampleBusObj.IisRequired (busAttribute);
if (isReq)
...
```

Iserialize()

Serializza i dati dell'oggetto aziendale utilizzando il formato di serializzazione di IBM WebSphere Business Integration Server Express.

Sintassi

```
string Iserialize();
```

Parametri

Nessuno.

Valori di restituzione

Un valore `string` che contiene i dati serializzati per l'oggetto aziendale.

Esempio

```
// Richiamare il metodo di serializzazione dei dati
IBusinessObject srcBO =
     accessSession.IcreateBusinessObject("Customer");
...
String serializedCustomer = scrBO.Iserialize();
```

IsetAttributes()

Imposta gli attributi in un oggetto di business dai dati serializzati in un tipo MIME specificato.

Sintassi

```
void IsetAttributes(string serializedData, string mimeType);
```

Parametri

datiserializzati I dati serializzati nel formato specificato di tipo MIME.

tipomime Il tipo MIME identifica il formato esterno dei dati serializzati.

Valori di restituzione

Nessuno.

Eccezioni

`IMalFormedDataException`
: Emesso quando i dati non hanno il formato corretto.

Esempio

```
// Stabilisce il tipo di formato di dati
String externalData = "incomingData"
String mimeType = "text/xml";
exampleBusObj.IsetAttributes (externalData, mimeType);
```

IsetAttributeToBlank()

Imposta l'attributo in un oggetto di business su un valore vuoto.

Sintassi

```
void IsetAttributeToBlank(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato su vuoto.

Valori di restituzione

Nessuno.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo set-attribute-to-blank
String attributeName = "checkType";
exampleBusObj.IsetAttributeToBlank(attributeName);
```

IsetAttributeToIgnore()

Imposta un attributo in un oggetto di business su ignore.

Sintassi

```
void IsetAttributeToIgnore(string attributeName);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato su "ignore".

Valori di restituzione

Nessuno.

Eccezioni

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Impostare l'attributo predefinito su di un  valore CxIgnore
String attributeName = "Ignore";
exampleBusObj.IsetAttributeToIgnore(attributeName);
```

IsetBooleanAttribute()

Imposta un attributo su un valore `booleano`.

Sintassi

```
void IsetBooleanAttribute(string attributeName, boolean value);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato.

valore Il valore `booleano` per l'attributo.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è un tipo `booleano`.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo Boolean
String attributeName = "custID";
boolean value = false;
exampleBusObj.IsetBooleanAttribute(attributeName, false);
```

IsetBusinessObjectArrayAttribute()

Imposta il valore di un attributo che è un'array di oggetti business (cardinalità multipla).

Sintassi

```
void IsetBusinessObjectArrayAttribute(string attributeName,
     IBusinessObjectArray value);
```

Parametri

nomeattributo Il nome dell'attributo multiple-cardinality il cui valore è stato impostato.

valore L'array degli oggetti business che il valore per l'attributo.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è un array dell'oggetto business.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo dell'attributo BusinessObjectArray
String arrayAttribute = "CustomerAddress";
IBusinessObject CustomerAddress =
     accessSession.IcreateBusinessObjectArray ("Address");
IBusinessObject exampleBO =
     accessSession.IcreateBusinessObject ("Customer");
exampleBO.IsetBusinessObjectArrayAttribute(arrayAttribute,
     CustomerAddress);
```

IsetBusinessObjectAttribute()

Imposta il valore di un attributo di cardinalità singola.

Sintassi

```
void IsetBusinessObjectAttribute(string attributeName,
     IBusinessObject value);
```

Parametri

nomeattributo Il nome dell'attributo single-cardinality il cui valore è stato impostato.

valore L'oggetto business che è il valore per l'attribute.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è un oggetto business.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo dell'attributo BusinessObject
String attributeName = "AccountStatus";
String value = "delqnt";
exampleBusObj.IsetBusinessObjectAttribute(attributeName, value);
```

IsetDateAttribute()

Imposta un attributo su un valore della data.

Sintassi

```
void IsetDateAttribute(string attributeName, string value);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato.

valore Il valore data per l'attributo, in un formato stringa.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è una data.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo dell'attributo Date
String dateAttribute = "DateofBirth";
String dateValue = "11/18/1966";
exampleBusObj.IsetDateAttribute(dateAttribute, dateValue);
```

IsetDoubleAttribute()

Imposta un attributo su un valore `doppio`.

Sintassi

```
void IsetDoubleAttribute(string attributeName, double value);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato.

valore Il valore `doppio` per l'attributo.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è di tipo `doppio`.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo double
String doubleAttributeName = "Average";
double value = 5.75;
exampleBusObj.IsetDoubleAttribute(doubleAttributeName, value);
```

IsetFloatAttribute()

Imposta un attributo su di un valore `float`.

Sintassi

```
void IsetFloatAttribute(string attributeName, float value);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato.

valore Il valore `float` per l'attributo.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è di tipo `float`.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo Float
String floatAttributeName "FloatAttributeName";
float value = 0.999;
exampleBusObj.IsetFloatAttribute(floatAttributeName, value);
```

IsetIntAttribute()

Imposta un attributo su un valore `intero`.

Sintassi

```
void IsetIntAttribute(string attributeName, long value);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato.

valore Un valore `long` per l'attributo integer.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è un tipo intero.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo int
String intAttribute = "CustomerNumber";
int value = 5002;
exampleBusObj.IsetIntAttribute(intAttribute, value);
```

IsetLongTextAttribute()

Imposta un attributo su un valore `long text`.

Sintassi

```
void IsetLongTextAttribute(string attributeName, string value);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato.

valore Il valore per l'attributo, in un formato stringa.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è di tipo `longtext`.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo LongText
String longTextAttributeName = "Description";
String value = "A very long text"
exampleBusObj.IsetLongTextAttribute(longTextAttributeName, value);
```

IsetStringAttribute()

Imposta un attributo su un valore `stringa`.

Sintassi

```
void IsetStringAttribute(string attributeName, string value);
```

Parametri

nomeattributo Il nome dell'attributo il cui valore è stato impostato.

valore Il valore `stringa` l'attributo.

Valori di restituzione

Nessuno.

Eccezioni

`InvalidAttributeTypeException`
Emesso quando l'attributo non è di tipo `stringa`.

`IInvalidAttributeNameException`
Emesso quando il nome dell'attributo non è valido.

Esempio

```
// Richiamare il metodo String
String stringAttribute = "CustomerName";
String value = "Greatest Customer";
exampleBusObj.IsetStringAttribute(stringAttribute, value);
```

IsetVerb()

Imposta l' verb per l'oggetto di business.

Sintassi

```
void IsetVerb(string verb);
```

Parametri

istruzione
Verb per l'oggetto business

Valori di restituzione

Nessuno.

Eccezioni

`IInvalidVerbException`
Emesso quando il verb non è supportato da un oggetto business.

Esempio

```
// Impostare il verb
String verb = "Create";
exampleBusObj.IsetVerb(verb);
```

ItoExternalForm()

Serializza i dati di un oggetto di business in un formato esterno del tipo MIME specificato.

Sintassi

```
string ItoExternalForm(string mimeType);
```

Parametri

tipomime Il tipo MIME (del client di accesso) in cui convertire l'oggetto business.

Valori di restituzione

Una `stringa` che contiene la versione serializzata dell'oggetto business, nel tipo MIME specificato.

Eccezioni

`IMalFormedDataException`
Emesso quando si verifica un errore durante la conversione.

Notes

Il metodo `ItoExternalForm()` invoca un gestore dati, trasmettendogli il tipo MIME dei dati serializzati. Il gestore dati analizza e converte l'oggetto business InterChange Server Express in dati serializzati del tipo MIME richiesto, restituendo i dati serializzati al client di accesso. Il formato dei dati serializzati deve essere di un tipo che il software di IBM WebSphere Business Integration Server Express supporti oppure un gestore dati personalizzato. Per ulteriori informazioni, fare riferimento al manuale *Guida per il gestore dati*.

Esempio

```
// Serializzare i dati in html
String mimeType = "text/html");
String htmldata = exampleBusObj.ItoExternalForm(mimeType);
```

ItoString()

Restituisce il dump dell'oggetto business nel formato di serializzazione InterChange Server Express.

Sintassi

```
string ItoString();
```

Parametri

Nessuno.

Valori di restituzione

Una `stringa` che contiene i dati serializzati in un formato IBM WebSphere Business Integration Server Express-compatibile.

Esempio

```
// Convertire in formato IBM
String stringBusObj;
stringBusObj = exampleBusObj.ItoString();
```

Capitolo 9. Eccezioni dell'interfaccia di accesso al server

Questo capitolo descrive l' Eccezioni dell'interfaccia di accesso al server. Le eccezioni emesse dai metodi dell'interfaccia di accesso al server sono delle classi secondarie della seguente classe di eccezione:

```
org.omg.CORBA.UserException
```

Nota: `UserException` è una classe esterna. *Non* è una classe di eccezione IBM Crossworlds. Consultare la documentazione IBM Java ORB per i membri e i metodi di `UserException`.

Tutte le eccezioni dell'interfaccia di accesso al server contengono un membro del messaggio di errore in formato `string` chiamato `IerrorMessage`.

Tabella 10 riepiloga le eccezioni dell'interfaccia di accesso al server.

Tabella 10. Riepilogo eccezioni

Eccezione	Pagina
IAttributeBlankException	93
IAttributeNotSetException	93
ICxAccessError	93
IExecuteCollaborationError	94
IInvalidAttributeNameException	94
IInvalidAttributeTypeException	94
IInvalidBusinessObjectTypeException	95
IInvalidIndexException	95
IInvalidVerbException	95
IMalFormedDataException	95
IValueNotSetException	95
IVerbNotSetException	95

IAttributeBlankException

Questa eccezione viene emessa quando l'attributo contiene un valore vuoto.

Membri

```
string IerrorMessage;
```

IAttributeNotSetException

Questa eccezione viene emessa quando l'attributo non contiene un valore.

Membri

```
string IerrorMessage;
```

ICxAccessError

Questa eccezione viene emessa quando non è possibile accedere ad un oggetto.

Membri

```
string IerrorMessage;
```

IExecuteCollaborationError

Questa eccezione viene emessa quando l'esecuzione di una collaborazione non riesce.

Membri

```
string IerrorMessage;
long status;
```

Note

I seguenti due metodi , che richiedono l'esecuzione di una collaborazione, possono emettere l'eccezione `IExecuteCollaborationError`:

- IexecuteCollaboration()
- IexecuteCollaborationExtFmt()

Questa eccezione contiene una variabile pubblica `int` chiamata `status` per indicare i dettagli di quando si verifica l'eccezione. L'interfaccia di accesso al server fornisce continuamente lo stato dell'esecuzione per rappresentare i possibili valori di questa variabile `status` variabile. Gli stati di esecuzione continui per questa eccezione vengono elencati in Tabella 11

Tabella 11. Valori per lo stato IExecuteCollaborationError

Nome costante	Descrizione
`UNKNOWNSTATUS`	Lo stato della chiamata al metodo `IexecuteCollaboration()` o `IexecuteCollaborationExtFmt()`.
`FAILEDTOREACHCOLLABORATION`	La richiesta di accesso non ha raggiunto la collaborazione.
`FAILEDINEXECUTIONOFCOLLABORATION`	La richiesta di accesso ha riportato un errore durante l'esecuzione della collaborazione.
`FAILEDINRETURNTOCLIENT`	La collaborazione è stata eseguita ma si è verificato un errore durante la distribuzione della risposta al client di accesso.

Per ottenere tale valore, togliere il riferimento ad una variabile d'eccezione come segue:

`this_exception_name_caught``.status`

IInvalidAttributeNameException

Questa eccezione viene emessa quando il nome dell'attributo non è valido.

Membri

```
string IerrorMessage;
```

IInvalidAttributeTypeException

Questa eccezione viene emessa quando il tipo di attributo non è valido.

Membri

```
string IerrorMessage;
```

IInvalidBusinessObjectTypeException

Questa eccezione viene emessa quando il tipo di oggetto business non coincide con il contenitore.

Membri

```
string IerrorMessage;
```

IInvalidIndexException

Questa eccezione viene emessa quando l'indice non è valido.

Membri

```
string IerrorMessage;
```

IInvalidVerbException

Questa eccezione viene emessa quando il verb non è valido.

Membri

```
string IerrorMessage;
```

IMalFormedDataException

Questa eccezione viene emessa quando i dati sono corrotti.

Membri

```
string IerrorMessage;
```

IValueNotSetException

Questa eccezione viene emessa quando l'attributo non ha un valore predefinito.

Membri

```
string IerrorMessage;
```

IVerbNotSetException

Questa eccezione viene emessa quando il verb non è impostato.

Membri

```
string IerrorMessage;
```

Capitolo 10. Interfaccia IBusinessObjectArray

L'interfaccia `IBusinessObjectArray` fornisce i metodi per restituire un oggetto di business, un'array, un'attributo dell'array o per impostare gli attributi e gli oggetti all'interno di un'array. Tabella 12 riporta i metodi dell'interfaccia `IBusinessObjectArray` .

Tabella 12. Metodi disponibili Interfaccia IBusinessObjectArray

Metodo	Descrizione	Pagina
Iduplicate()	Restituisce una copia dell'array di oggetti business.	97
IdeleteBusinessObjectAtIndex()	Elimina l'oggetto di business all'indice specificato dell'array di oggetti business.	98
IgetBusinessObjectAtIndex()	Richiama un oggetto di business all'indice specificato dell'array di oggetti business.	98
IgetSize()	Restituisce la dimensione dell'array di oggetti business.	99
IremoveAllElements()	Rimuove tutti gli elementi (oggetti di business) nell'array di oggetti business.	99
IsetBusinessObject()	Imposta l'oggetto di business alla fine dell'array di oggetti business.	99
IsetBusinessObjectAtIndex()	Imposta l'oggetto di business all'indice specificato dell'array di oggetti business.	100

Iduplicate()

Restituisce una copia dell'array di oggetti business.

Sintassi

```
IBusinessObjectArray Iduplicate();
```

Parametri

Nessuno.

Valori di restituzione

Un oggetto `IBusinessObjectArray` che contiene la nuova array di oggetti di business duplicata.

Eccezioni

ICxAccessError	Emesso quando non è possibile accedere all'array di oggetti di business.

Esempio

Nel seguente esempio viene duplicato `sourceCustomer` in modo da creare `destCustomer`.

```
IBusinessObjectArray srcBOArray =
     accessSession.IcreateBusinessObjectArray ("Customer");
IBusinessObjectArray destBOArray = scrBOArray.Iduplicate();
```

IdeleteBusinessObjectAtIndex()

Elimina l'oggetto di business all'indice specificato dell'array di oggetti business.

Sintassi

```
void IdeleteBusinessObjectAtIndex(long indice);
```

Parametri

indice L'indice nell'array di oggetti di business dell'oggetto di business da eliminare.

Valori di restituzione

Nessuno.

Eccezioni

`IInvalidIndexException`
Emesso quando l'indice non è valido.

Esempio

```
//Elimina l'oggetto di business
long index = 5;
exampleBusObjArray.IdeleteBusinessObjectAtIndex(index);
```

IgetBusinessObjectAtIndex()

Richiama un oggetto di business all'indice specificato dell'array di oggetti business.

Sintassi

```
IBusinessObject IgetBusinessObjectAtIndex(long indice);
```

Parametri

indice L'indice nell'array di oggetti di business dell'oggetto di business da richiamare.

Valori di restituzione

Un oggetto `IBusinessObject` che contiene l'oggetto di business all'indice specificato dell'array di oggetti di business.

Eccezioni

`IInvalidIndexException`
Emesso quando l'indice non è valido.

Esempio

```
// Richiama l'oggetto di business al metodo dell'indice
IBusinessObject aBusinessObject = null;
long index = 1;
aBusinessObject = exampleBusObjArray.IgetBusinessObjectAtIndex(index);
```

IgetSize()

Restituisce la dimensione dell'array di oggetti business.

Sintassi

```
long IgetSize();
```

Parametri

Nessuno.

Valori di restituzione

Un numero intero che indica il numero di elementi (oggetti di business) nell'array di oggetti di business.

Esempio

```
// Richiama la dimensione dell'array
long = arraySize = 0;
arraySize = exampleBusObjArray.IgetSize();
```

IremoveAllElements()

Rimuove tutti gli elementi (oggetti di business) nell'array di oggetti business.

Sintassi

```
void IremoveAllElements()
```

Parametri

Nessuno.

Valori di restituzione

Nessuno.

Esempio

```
// Rimuove gli elementi dell'array
exampleBusObjArray.IremoveAllElements();
```

IsetBusinessObject()

Imposta l'oggetto di business alla fine dell'array di oggetti business.

Sintassi

```
void IsetBusinessObject(IBusinessObject valore);
```

Parametri

valore L'oggetto di business da impostare alla fine dell'array.

Valori di restituzione

Nessuno.

Eccezioni

`IInvalidBusinessObjectTypeException`
Emesso quando l'oggetto di business non è supportato.

Esempio

```
// Imposta l'oggetto di business alla fine dell'array
IBusinessObject srcBO = accessSession.IcreateBusinessObject(
     "PayableNetChange");
exampleBusObjArray.IsetBusinessObject(srcBO);
```

IsetBusinessObjectAtIndex()

Imposta l'oggetto di business all'indice specificato dell'array di oggetti business.

Sintassi

```
void IsetBusinessObjectAtIndex(long indice, IBusinessObject inObj);
```

Parametri

indice L'indice nell'array di oggetti di business.

inObj L'oggetto di business da inserire nell'array.

Eccezioni

`IInvalidIndexException`
Emesso quando l'indice non è valido.

`IInvalidBusinessObjectTypeException`
Emesso quando il tipo di oggetto di business non è supportato dall'array.

Esempio

```
// Imposta l'oggetto di business all'indice
long index = 1;
IBusinessObject aBusObj = accessSession.IcreateBusinessObject(
     "PayableNetChange");
exampleBusObjArray.IsetBusinessObjectAtIndex(index, aBusObj);
```

Appendice. Considerazioni sull'Internationalization

Un client di accesso internazionalizzato è un client progettato in maniera tale che può essere personalizzato per una locale particolare. Una locale è parte di un ambiente dell'utente che raccoglie informazioni sulle modalità di gestione dei dati specifici della nazione, della lingua o del territorio dell'utente.

Tale sezione fornisce le seguenti informazioni relative ad un client di accesso internazionalizzato:

Cos'è una locale?

Una **locale** è parte di un ambiente dell'utente che raccoglie informazioni sulle modalità di gestione dei dati specifici della nazione, della lingua o del territorio dell'utente. Normalmente la locale viene installata come parte del sistema operativo.

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

- Convenzioni culturali secondo la lingua e il paese (o territorio)
  - Formati dei dati:

    Date: definiscono nomi completi e abbreviati per i giorni della settimana e i mesi, così come la struttura della data (inclusi i separatori della data).

    Numeri: definiscono i simboli per i separatori delle migliaia e il punto dei decimali, e dove questi vengono collocati nei numeri stessi.

    Ore: definiscono gli indicatori per gli orari con l'intervallo di 12 ore (come gli indicatori AM e PM) così come la struttura dell'ora stessa.

    Valori monetari: definiscono simboli numerici e monetari, così come questi simboli vengono collocati all'interno dei valori monetari.
  - Ordine di collazione indica come ordinare i dati per la particolare serie di codici di caratteri e la lingua.
  - La gestione delle stringhe include attività come il confronto delle "dimensioni" delle lettere (maiuscola e minuscola), delle stringhe secondarie e la concatenazione.

Progettazione di un client di accesso per l'internazionalizzazione

Per utilizzare un client di accesso in un contesto internazionalizzato, implica la considerazione sia della locale che della codifica dei caratteri.

Considerazioni della locale

Per essere internazionalizzato, un client di accesso deve essere codificato perché sia sensibile alla locale; ciò vuol dire che il comportamento del client deve considerare le impostazioni della locale ed eseguire le attività appropriate a quella specifica locale.

Normalmente il client di accesso dovrebbe seguire questi principi di progettazione di sensibilità alla locale:

- Il testo di ciascun messaggio di errore, di stato e di traccia dovrebbe essere isolato dal componente specifico dell'applicazione in un file di messaggi e tradotto nella lingua della locale.
- L'ordinamento della collazione di dati utilizza un'ordinamento di collazione appropriata alla lingua e al paese della locale.
- L'elaborazione delle stringhe (come il confronto, stringhe secondarie e maiuscole/minuscole) è appropriata per i caratteri nella lingua della locale.
- I formati delle date, dei numeri e delle ore sono appropriati per la locale.

Codifica dei caratteri

L'interfaccia di accesso al server utilizza UCS-2, una specie di Unicode. I dati che il client di accesso trasmette all'interfaccia di accesso al server deve utilizzare la codifica dei caratteri Unicode.

Supporto per le lingue con script bidirezionale

Per i trasferimenti di dati bidirezionali, è necessario definire le proprietà per il modello della collaborazione in Process Designer Express e fornire i valori per queste proprietà alla distribuzione. Per ulteriori informazioni, consultare la *Guida allo sviluppo della collaborazione* e la *Guida all'implementazione del sistema*.

Indice analitico

J

L

M

O

P

R

S

T

V

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 IBM locale per informazioni sui prodotti e sui servizi disponibili nel proprio paese. Ogni riferimento relativo a prodotti, programmi o servizi IBM non implica che che solo quei prodotti, programmi o servizi IBM possano essere utilizzati. In sostituzione a quelli forniti dall'IBM, possono essere usati prodotti, programmi o servizi funzionalmente equivalenti che non comportino la violazione dei diritti di proprietà intellettuale o 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 nella presente pubblicazione. La fornitura di questa pubblicazione non implica la concessione di alcuna licenza su di essi. Chi desiderasse ricevere informazioni relative a licenze può rivolgersi 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 vengono 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. 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 sulle prestazioni riportati sono stati determinati in un ambiente controllato. Pertanto, i risultati ottenuti in altri ambienti operativi potrebbero variare significativamente. Alcune misurazioni possono essere state effettuate su sistemi del livello di sviluppo e non vi è alcuna garanzia che tali misurazioni resteranno invariate sui sistemi generalmente disponibili. Inoltre, alcune misurazioni possono essere state stimate tramite estrapolazione. I risultati reali possono 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. È possibile copiare, modificare e distribuire questi programmi di esempio sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in conformità alle API (application programming interface) di IBM per cui sono scritti i programmi di esempi. 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. Tali informazioni sono fornite per eseguire il debug del software applicativo.

Avvertenza: Non utilizzare queste informazioni come interfaccia di programmazione in quanto possono essere soggette a modifiche di vario genere.

Marchi

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. Nomi di altri prodotti, società e servizi possono essere marchi di altre società.

WebSphere Business Integration Server Express e Express Plus includono del software sviluppato dal Progetto Eclipse (http://www.eclipse.org/)

WebSphere Business Integration Server Express, Versione 4.4, e WebSphere Business Integration Server Express Plus, Versione 4.4