IBM WebSphere
Business Integration Server Express ed Express Plus

# Guida allo sviluppo dell'oggetto business

*versione 44*

IBM WebSphere
Business Integration Server Express ed Express Plus

# Guida allo sviluppo dell'oggetto business

*versione 44*

**Nota!**
Prima di utilizzare queste informazioni ed il prodotto supportato, consultare “Informazioni particolari” a pagina 297.

**22 aprile 2005**

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

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

L’IBM può utilizzare o divulgare le informazioni ricevute dagli utenti secondo le modalità ritenute appropriate, senza nessun obbligo nei loro confronti.

# Indice

# Informazioni

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

In questo documento viene descritto come utilizzare Business Object Designer Express per creare le definizioni di oggetti business, sia manualmente che utilizzando un ODA (Object Discovery Agent). Gli Object Discovery Agents vengono creati per "rilevare" requisiti di oggetti business specifici per un'origine dati e per generare le definizioni per tali requisiti. Business Object Designer Express fornisce un'interfaccia grafica utente (GUI) agli Object Discovery Agents disponibili e agevola nella gestione dei processi di rilevamento e generazione delle definizioni. In questo documento inoltre viene illustrato come utilizzare ODK (Object Discovery Agent Development Kit) per creare gli Object Discovery Agents.

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 relative varianti fa riferimento a entrambi i prodotti.

## A chi è diretto questo manuale

Questo documento si rivolge ai clienti, consulenti o rivenditori IBM he creano o modificano oggetti business. Prima di iniziare, occorre avere dimestichezza con tutti i concetti illustrati in *IBM WebSphere Business Integration Server Express System Implementation Guide*.

## Documentazione correlata

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

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

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

## Convenzioni tipografiche

Questo manuale utilizza le seguenti convenzioni:

| | |
|---|---|
| `carattere courier` | Indica un valore letterale, come il nome di un comando, un nome file, informazioni immesse dall'utente o le informazioni visualizzate sullo schermo. |
| *corsivo*, *`corsivo`* | Indica un nuovo termine la prima volta che appare, un nome di variabile o un riferimento incrociato. |
| **grassetto** | Indica un elemento GUI. |
| *riquadro blu* | Uno schema blu, visibile solo quando si visualizza il manuale in linea, indica un collegamento ipertestuale a riferimento incrociato. Fare clic all'interno del riquadro per passare all'oggetto del riferimento. |
| { } | 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. |
| `...` | In una riga di sintassi, le ellissi indicano una ripetizione del parametro precedente. Ad esempio, `option[,...]` significa che è possibile inserire più opzioni separate da virgola. |
| `< >` | Nelle convenzioni di denominazione, le parentesi angolari racchiudono singoli elementi di un nome per distinguerli l'uno dall'altro, ad esempio `<nome_server><`*`nome_connettore`*`>tmp.log.` |
| /, \ | In questo documento, le barre rovesciate (\) vengono utilizzare come convenzione per i percorsi di directory. Per installazioni Linux sostituire le barre (/) con barre rovesciate (/). Tutti i nomi percorso di prodotti IBM sono relativi alla directory dove il prodotto IBM è installato sul sistema. |
| *ProductDir* | Rappresenta la directory in cui viene installato il prodotto. |

**Linux**
Le sezioni con tali istruzioni indicano note che elencano differenze tra sistemi operativi.

| | |
|---|---|
| *`%testo%`* e *`$testo`* | Il testo compreso tra segni percentuali (%) indica il valore della variabile di sistema del `testo` di Windows o della variabile utente. Una notifica equivalente in un ambiente Linux è *`$testo`*, che indica il valore della variabile d'ambiente *`text`* di Linux. |

# Novità di questo rilascio

Questa sezione fornisce un riepilogo delle funzioni nuove e modificate di IBM WebSphere Business Integration Server Express e di IBM WebSphere Business Integration Server Express Plus e dei programmi a loro associati per lo sviluppo di oggetti business. Queste funzioni e questi programmi sono trattati in dettaglio in questa guida.

## Novità nel rilascio 4.4

Questo rilascio fornisce le seguenti nuove funzionalità: functionality:

- Supporto per l'immissione e la visualizzazione di script bidirezionali in Business Object Designer Express.
- Object Discovery Agents supporta testo bidirezionale o metadati passando per un'applicazione esterna.

## Novità nel rilascio 4.3.1

Questo manuale non è stato modificato nel rilascio 4.3.1.

## Novità del rilascio 4.3

Questo è il primo rilascio di questo manuale.

# Parte 1. Progettazione e sviluppo di oggetti business

# Capitolo 1. Oggetti business

Un sistema di integrazione business utilizza oggetti business per trasportare dati ed istruzioni di elaborazione tra broker di integrazione e connettori o un client di accesso. InterChange Server Express è il broker di integrazione di WebSphere Business Integration Server Express. Gli oggetti business rappresentano una richiesta proveniente da InterChange Server Express, un evento in un'applicazione o server Web, oppure una chiamata da un sito esterno. In questo manuale sono contenute informazioni di sviluppo e progettazione di oggetti business, ed anche sullo sviluppo di un proprio ODA (object discovery agent). Gli argomenti principali di questo capitolo sono:



In questo capitolo si assume che si abbia una conoscenza di base di InterChange Server Express:

*Tabella 1. Documenti prerequisiti*

| Broker di integrazione | Documenti prerequisiti |
|---|---|
| InterChange Server Express | • *WebSphere Business Integration Server Express - Guida all'installazione per Windows* o *per Linux*<br>• *System Implementation Guide* |

## Oggetti business nel sistema WebSphere Business integration Server Express

Il sistema WebSphere Business Integration Server Express comprende i seguenti componenti:

- Un insieme di adattatori

  Un *adattatore* consiste in una serie di moduli software che comunicano con Interchange Server Express e con applicazioni o tecnologie per eseguire attività quali l'esecuzione di logiche di applicazione e lo scambio di dati.
- Un broker di integrazione

  InterChange Server Express è il broker di integrazione del sistema WebSphere Business Integration Server Express. L'attività di InterChange Server Express consiste nell'integrare i dati tra applicazioni eterogenee.

Nel sistema WebSphere Business Integration Server Express, l'informazione inviata o ricevuta dai componenti viene impacchettata nella forma di un *oggetto di business*, come riportato di seguito:

- Per i dati trasferiti tra un adattatore ed un broker di integrazione, si deve progettare un *oggetti business specifici dell'applicazione* che modellano entità appropriate dell'applicazione.
- Per i dati elaborati all'interno di una logica business dell'oggetto di collaborazione di InterChange Server Express, si deve progettare un *oggetti business generici* che contengono un insieme di informazioni per le entità delle

applicazioni che hanno bisogno di comunicare. Le mappe trasformano i dati tra oggetti business generici ed oggetti business specifici dell'applicazione, permettendo agli adattatori di comunicare con le loro applicazioni utilizzando entità specifiche dell'applicazione, mentre gli oggetti di collaborazione possono applicare logiche di business in modo indipendente dall'applicazione.

Entrambi gli oggetti business specifici dell'applicazione che quelli generici sono modellati al momento della progettazione come *definizioni di oggetti business*, che vengono archiviati nel sistema di integrazione del business. Al momento dell'esecuzione, il dato è riempito in una *istanza di oggetto business* (spesso chiamata "oggetto business"), che si basa su una definizione appropriata. L'oggetto business si sposta nel sistema di integrazione di business così come definito nel suo instradamento e nelle regole delle logiche di business.

## Definizioni di oggetti business

Una *definizione di un oggetto business* consiste in una maschera per il dato che può essere gestita come un'unità collettiva. Comprende un 'intestazione dell'oggetto business, che specifica il nome e la versione della definizione dell'oggetto business. Inoltre, la definizione di oggetto di business contiene le seguenti informazioni:

- "Attributi dell'oggetto business e proprietà degli attributi"
- "Verbi dell'oggetto business" a pagina 8
- "Informazioni specifiche dell'applicazione dell'oggetto business" a pagina 8

Figura 1 mostra le parti di una definizione di oggetto business.

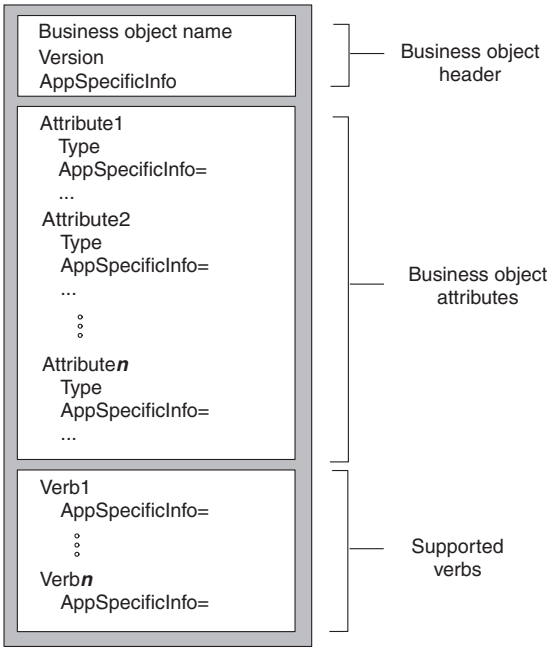



*Figura 1. Parti della definizione dell'oggetto business*

### Attributi dell'oggetto business e proprietà degli attributi

Un oggetto business contiene attributi, ogni *attributo* rappresenta una entità di dati. Nella definizione dell'oggetto business, si definisce il nome di ogni attributo così come altre *proprietà dell'attributo*. L'istanza dell'oggetto business mantiene un valore per ogni attributo (oppure indica che l'attributo non ha un valore).

**Nota:** Si possono definire valori predefiniti e dati stringa all'interno di un'oggetto business in uno script bidirezionale. Tutti i nomi degli attributi definiti **devono** utilizzare solo caratteri nella serie di caratteri US English.

Le definizioni degli oggetti business comprendono diverse proprietà che si applicano agli attributi. Queste proprietà forniscono al connettore, al gestore dati, ed ad altri componenti le informazioni sul tipo, la dimensione ed i valori predefiniti degli attributi. Le proprietà degli attributi sono affrontate nelle sezioni seguenti.

**Proprietà Name:** Ogni attributo dell'oggetto business deve avere un nome univoco all'interno di una definizione dell'oggetto business. Il nome dovrebbe descrivere i dati che sono contenuti nell'attributo. Il nome può contenere fino ad 80 caratteri e non può iniziare con un numero. Può contenere caratteri alfanumerici ed il carattere di sottolineatura, ma non può contenere spazi, punteggiatura o caratteri speciali quali:

- @
- ~
- $
- %
- ^
- &
- (
- )
- |
- ?
- ,
- '
- '
- ;
- {
- [
- ]
- }

**Note:**

1. Nella progettazione di un oggetto business specifico dell'applicazione, controllare la guida per l'utente del suo adattatore oppure la *Data Handler Guide* per consigli e requisiti specifici di nomenclatura.
2. Gli attributi nome devono usare *solo* caratteri definiti nella serie caratteri associata con la locale U.S. English (`en_US`).

**Proprietà Type:** La proprietà Type definisce il tipo di dato dell'attributo:

- Per un attributo semplice, i tipi supportati sono `Boolean`, `Integer`, `Float`, `Double`, `String`, `Date`, and `LongText`.
- Per un attributo complesso, il tipo è una definizione di oggetto di business:
  - Se l'attributo rappresenta un oggetto business secondario, specificarne il tipo come nome della definizione dell'oggetto business secondario ed impostare la cardinalità ad 1 (cardinalità singola).

– Se l'attributo rappresenta una serie di oggetti business figli, specificarne il tipo come nome della definizione dell'oggetto business ed impostare la cardinalità ad `n` (cardinalità multipla).

**Nota:** Tutti gli attributi che rappresentano un oggetto business secondario hanno inoltre una proprietà `ContainedObjectVersion` (che specifica il numero di versione della definizione dell'oggetto business secondario) ed una proprietà `Relationship` (che specifica il valore `Containment`).

**Proprietà Cardinality:** Ogni attributo semplice ha una cardinalità singola (cardinalità 1). Ogni attributo complesso, che rappresenta un oggetto business secondario o un insieme di oggetti business figli, ha rispettivamente una cardinalità singola o multipla (cardinalità `n`). Per ulteriori informazioni, consultare "Oggetti business gerarchici" a pagina 12.

**Nota:** Quando specificata per un attributo obbligatorio, la cardinalità singola indica che un oggetto business secondario *deve* esistere, e la cardinalità multipla indica da zero a più istanze di oggetti business secondario.

**Proprietà Key:** Almeno un attributo in ogni oggetto business deve essere specificato come la chiave. L'attributo chiave contiene un valore che identifica univocamente l'oggetto business. Per definire un attributo come chiave, impostare la proprietà Key su `true`.

**Nota:** Un valore chiave di un oggetto business viene spesso referenziato come *chiave primaria*.

Se si specifica una chiave come attributo complesso :

- Se l'attributo rappresenta un oggetto business secondario, la chiave è la concatenazione della chiave e dell'oggetto business secondario.
- Se l'attributo rappresenta una serie di oggetti business secondario, la chiave è la concatenazione delle chiavi dell'oggetto business secondario alla posizione 0 della serie.

**Proprietà chiave esterna:** La proprietà Chiave esterna è di solito utilizzata in negli oggetti business specifici dell'applicazione per specificare che il valore di un attributo contiene la chiave primaria di un altro oggetto business, che collega oggetti business. L'attributo che contiene la chiave primaria di un altro oggetto business è chiamato *chiave esterna*. Definire la proprietà Foreign Key come `true` per ogni oggetto che rappresenta una chiave esterna.

Si può utilizzare la proprietà Foreign Key anche per altre istruzioni di elaborazione. Ad esempio, per specificare quale tipo di ricerca di chiave esterna il connettore effettua. In questo caso si può impostare la proprietà Foreign Key come `true` per fare in modo che il connettore verifichi l'esistenza dell'entità nel database, e crei una relazione solo se il record dell'entità esiste.

**Proprietà Required:** La proprietà Required specifica se un attributo deve contenere un valore. Se un determinato attributo nell'oggetto business deve contenere un valore per riuscire ad elaborare il dato dell'oggetto business, impostare la proprietà Required dell'attributo come `true`.

**AppSpecificInfo:** La proprietà AppSpecificInfo può contenere una `Stringa` di non più di 1000 caratteri che viene definita principalmente per oggetti business specifici dell'applicazione. Per ulteriori informazioni su questa proprietà, consultare "Informazioni specifiche dell'applicazione dell'oggetto business" a pagina 8

**Note:**

1. L'informazione specifica dell'applicazione *non* è disponibile nel processo di mappatura.
2. Si può specificare questo dato in uno script bidirezionale, se necessario.

**Proprietà Max Length:** La proprietà Max Length è impostata sul numero di byte che un attributo tipo `Stringa` può contenere. Sebbene questo attributo non sia imposto dal sistema WebSphere Business Integration Server Express, determinati connettori o gestori dati utilizzano questo valore. Controllare la guida dell'adattatore o del gestore dati che elabora l'oggetto business per determinare la lunghezza minima e massima.

**Importante:** La proprietà Max Length quando si utilizza il gestore dati a larghezza fissa.

**Nota:** L'attributo lunghezza *non* è disponibile nel processo di mappatura.

**Proprietà Default value:** La proprietà Default Value può specificare un valore predefinito per un attributo.

Se questa proprietà viene specificata per un un oggetto business specifico dell'applicazione e la proprietà di configurazione del connettore `UseDefaults` è impostata su `true`, il connettore può utilizzare i valori predefiniti specificati nella definizione dell'oggetto business per fornire valori agli attributi che non ne hanno al momento dell'esecuzione.

**Note:**

1. Il valore predefinito dell'attributo può utilizzare un qualsiasi carattere definito nella serie caratteri associata con la locale corrente.
2. Per un attributo il cui tipo è Stringa, è possibile definire un carattere vuoto come valore predefinito.

**Proprietà Comments:** La proprietà Comments permette di definire un commento per un attributo. A differenza della proprietà AppSpecificInfo, che è utilizzata per elaborare un oggetto business, la proprietà Comments fornisce unicamente informazioni di documentazione, che aiutano gli sviluppatori nella comprensione delle scelte di progettazione altrui.

**Nota:** Per il commento dell'attributo si può utilizzare un qualsiasi carattere definito nella serie caratteri associata con la locale corrente compresi caratteri bidirezionali. Il carattere nuova riga non è valido.

**Attributo ObjectEventId:** L'attributo `ObjectEventId` non è solo obbligatorio, ma deve anche essere l'ultimo attributo in ogni oggetto business. Il sistema WebSphere Business Integration Server Express utilizza questo attributo per identificare e seguire il flusso di un evento nel sistema.

L'attributo `ObjectEventId` memorizza un valore univoco che identifica ogni evento nel sistema WebSphere Business Integration Server Express. La struttura del connettore genera valori per questo attributo nell'oggetto business principale ed in ogni secondario.

**Importante:** *Non* mappare l'attributo `ObjectEventId` e non farlo riempire da un connettore o gestore dati. Il sistema di integrazione business gestisce il valore di questo attributo.

## Verbi dell'oggetto business

La definizione dell'oggetto business comprende un elenco dei verbi supportati dall'oggetto business. I verbi corrispondono ad operazioni che sono valide sul dato all'interno dell'oggetto business. Al momento dell'esecuzione, un oggetto business contiene un'istruzione attiva, che descrive le operazioni da effettuare sul dato nell'oggetto business definito.

Tabella 2 elenca le istruzioni di base che una definizione dell'oggetto business supporta.

*Tabella 2. Istruzioni di base*

| Istruzione | Funzione |
|---|---|
| Create | Crea una nuova entità nell'applicazione. |
| Retrieve | Tramite l'utilizzo di valori chiave, restituisce un oggetto business completo. |
| Update | Cambia il valore in uno o più campi nell'entità dell'applicazione. |
| Delete | Elimina l'entità dall'applicazione. Questa operazione consiste in una effettiva cancellazione fisica. |

In aggiunta alle istruzioni di base in Tabella 2, una definizione di un oggetto business dovrà poter supportare una o più delle seguenti istruzioni:

- RetrieveByContent—Tramite l'utilizzo di valori non chiave, restituisce un oggetto business completo.
- Exist—Verifica l'esistenza di una determinata entità ma non la recupera.
- Custom—Esegue un'operazione specifica dell'applicazione.

## Informazioni specifiche dell'applicazione dell'oggetto business

Una definizione di oggetto business può fornire le *informazioni specifiche dell'applicazione* il cui contenuto a sua volta fornisce i *metadati* al componente che elabora l'oggetto business. Un uso comune di informazioni specifiche dell'applicazione è quello di fornire al connettore o al gestore dati informazioni dipendenti dall'applicazione su come elaborare un oggetto business. L'informazione specifica dell'applicazione è una stringa inserita al momento della progettazione dell'oggetto business e letta al momento dell'esecuzione da un connettore o da un gestore dati.

**Nota:** I connettore che sono progettati per utilizzare le informazioni specifiche dell'applicazione nelle definizioni degli oggetti business specifici dell'applicazione sono chiamati *connettore controllato da metadati*. Poiché le informazioni di elaborazione sono configurabili, e non codificate in maniera sicura, un connettore controllato da metadati è più flessibile e semplice da gestire di uno che non lo sia.

All'interno di una definizione dell'oggetto business, si possono fornire informazioni specifiche dell'applicazione ad uno dei seguenti tre livelli:

- Definizione dell'oggetto business
- Attributo nella definizione dell'oggetto business
- Istruzione dell'oggetto business

Le informazioni specifiche dell'applicazione sono sono memorizzate in un campo della definizione dell'oggetto di business chiamato proprietà AppSpecificInfo. Il valore della proprietà AppSpecificInfo è una stringa di testo che può contenere

qualsiasi informazione riguardante l'oggetto business o l'applicazione. Figura 2 descrive gli elementi principali della definizione dell'oggetto business e la proprietà specifica dell'applicazione per ciascun elemento.

**Nota:** La stringa può contenere caratteri bidirezionali, se richiesto dalla propria locale.

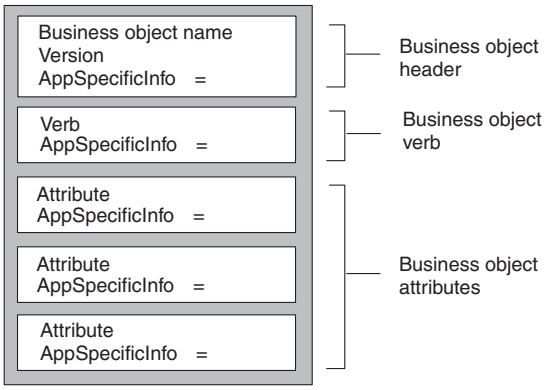



*Figura 2. Definizione dell'oggetto business che mostra la proprietà specifica dell'applicazione per ciascun elemento*

Questa sezione tratta i seguenti argomenti:

- Informazioni specifiche dell'applicazione per un oggetto business
- Informazioni specifiche dell'applicazione per un attributo
- Informazioni specifiche dell'applicazione per un'istruzione

**Informazioni specifiche dell'applicazione per un oggetto business:**
L'informazione specifica dell'applicazione a livello di oggetto business fornisce informazioni che il connettore o il gestore dati utilizzano per l'elaborazione del dato. L'informazione specifica dell'applicazione a livello di oggetto business è utilizzata ogni volta che le istruzioni di elaborazione sono rilevanti per la gerarchia dell'intero oggetto business. Ad esempio può effettuare una delle seguenti operazioni:

- Definire l'ambito dell'elaborazione della transazione dell'oggetto business
- Per applicazioni che richiedono l'elaborazione dell'oggetto nell'estensione dell'applicazione, può contenere il nome delle funzione da chiamare per gestire l'oggetto business
- Specificare il nome della tabella o modulo a cui appartiene il record
- Specificare il nome di un attributo di un oggetto business che rappresenta una eliminazione logica o "soft"

Figura 3 descrive l'informazione specifica dell'applicazione che identifica il nome di una tabella o di un modulo di una applicazione. Il connettore può ricavare il nome della tabella o del modulo dalla proprietà AppSpecificInfo ed utilizzarla in una chiamata API per recuperare dati dall'applicazione.

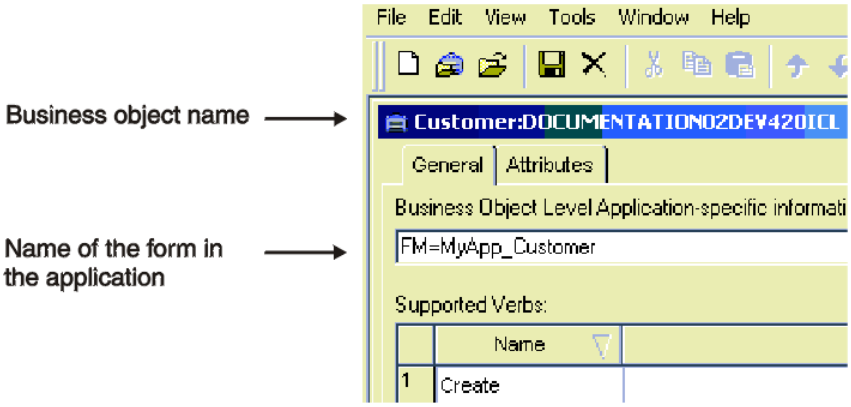



*Figura 3. Informazioni specifiche dell'applicazione per un oggetto business*

**Informazioni specifiche dell'applicazione per un attributo:** Ogni attributo della definizione di un oggetto business può avere associata un informazione specifica dell'applicazione. Informazioni specifiche dell'applicazione a livello attributo sono utilizzate ogni volta che le istruzioni di elaborazione sono rilevanti per un singolo attributo. Ad esempio, questa informazione può specificare un campo in un modulo, una colonna in una tabella, o tutto ciò che sia necessario ad un connettore per posizionare o lavorare con l'attributo. Se determinati attributi dell'oggetto business sono posizionati in un determinato modulo secondario dell'applicazione, la proprietà AppSpecificInfo ha molte possibilità di essere prescelta per la codifica di questa informazione.

Figura 4 descrive la proprietà AppSpecificInfo dell'attributo. In questo esempio, l'informazione specifica dell'applicazione definisce il nome del modulo secondario e del campo.

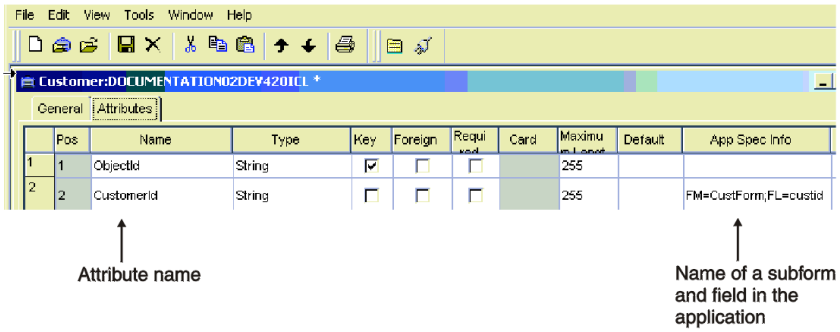



*Figura 4. Informazioni specifiche dell'applicazione per un attributo.*

Figura 5 descrive le relazioni tra modulo, modulo secondario ed il nome del campo presenti nell'informazione specifica dell'applicazione a livello di attributo e di oggetto. Questo esempio assume che l'applicazione della fatturazione si basa su moduli e che il suo modo di comunicare con le fatture è tramite il modulo di fattura, che è un modulo secondario del modulo principale CustAccount. Il modulo secondario di Fattura ha i seguenti campi: CustName, CustAddr, InvNum, DollarAmount, e Terms.

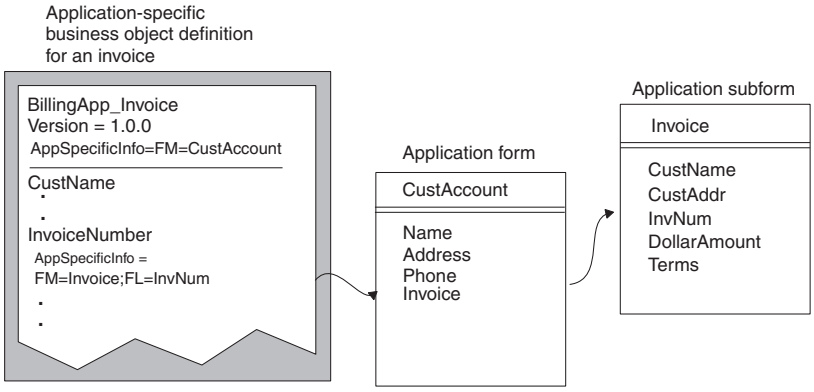



*Figura 5. Utilizzo dell'informazione specifica dell'applicazione della definizione dell'oggetto business*

Nella figura 5 si utilizza la proprietà AppSpecificInfo a livello di attributo per archiviare il nome del modulo secondario di Fattura ed il nome campo corrispondente dell'attributo. L'esempio utilizza coppie di valori nome per specificare l'informazione.

**Informazioni specifiche dell'applicazione per un'istruzione:** Ciascuna definizione di istruzione può contenere informazioni specifiche dell'applicazione che forniscono al connettore o al gestore dati direttive su come elaborare l'oggetto business quando l'istruzione è attiva.

**Nota:** Il gestore dell'oggetto business, che è parte di un connettore che gestisce richieste inviate dal broker di integrazione al connettore, può essere progettato per utilizzare informazioni specifiche dell'applicazione nei verbi della propria definizione dell'oggetto business specifico dell'applicazione. Tali gestori di oggetti business sono chiamati *gestori di oggetti business controllati da metadati*. Siccome le informazioni di elaborazione sono configurabili, e non codificate in maniera sicura, un gestore di oggetti business controllato da metadati è più flessibile e semplice da gestire di uno che non lo sia.

Ad esempio, se il connettore utilizza un'API per la gestione degli aggiornamenti del database dell'applicazione, l'informazione specifica dell'applicazione può dare al connettore le informazioni per l'esecuzione dell'API.

L'informazione specifica dell'applicazione dell'istruzione può inoltre definire il nome di una funzione da chiamare nell'applicazione per gestire l'elaborazione dell'oggetto business.

## Istanze dell'oggetto business

Mentre la definizione dell'oggetto business rappresenta la maschera di una raccolta di dati, un'istanza dell'oggetto business (spesso chiamata semplicemente "*oggetto business*") è un'entità di esecuzione che contiene il dato effettivo. L'oggetto business è ciò che i componenti di un sistema di integrazione business si scambiano.

L'oggetto business contiene le seguenti informazioni:

- Attributi, ciascuno dei quali contiene dati del oggetto business associato. Uno degli attributi è spesso un attributo chiave, che contiene un valore che identifica univocamente l'oggetto business tra tutti quelli con la stessa definizione.
- Un'istruzione attiva, che dovrebbe essere una delle istruzioni supportate dalla definizione dell'oggetto business

Figura 6 mostra la definizione dell'oggetto business del `Cliente` ed una istanza dell'oggetto business corrispondente per questa definizione.

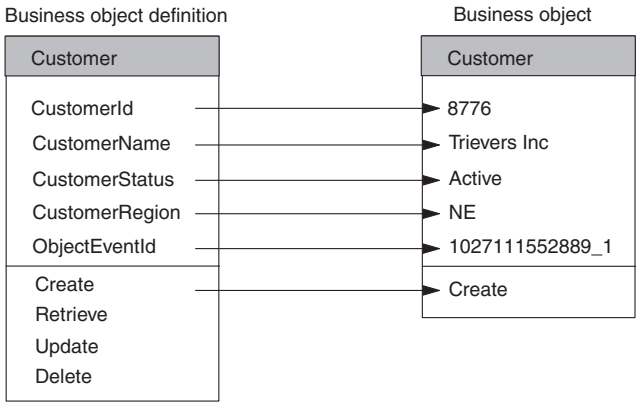



*Figura 6. Definizione dell'oggetto business e oggetto business di esempio.*

## Struttura oggetto business

La struttura di un oggetto business può essere una delle seguenti:

- Oggetti business senza nessun effetto
- Oggetti business gerarchici

La seguente sezione mostra esempi di strutture di oggetti business senza alcun effetto e gerarchici, e fornisce informazioni su come la struttura dell'oggetto business influenzi la logica del connettore.

### Oggetti business senza nessun effetto

La definizione dell'oggetto business di un *oggetto business senza nessun effetto* contiene uno o più attributi semplici ed un elenco di istruzioni supportate. Un *attributo semplice* rappresenta un valore, quale `String` o `Integer` o `Date`. Tutti gli attributi semplici hanno un unico cardinalità. L'oggetto business cliente nella figura 6 è un esempio di oggetto business senza nessun effetto. Per ulteriori informazioni, consultare "Attributi dell'oggetto business e proprietà degli attributi" a pagina 4.

### Oggetti business gerarchici

*Le definizioni di un oggetto business gerarchico* definiscono la struttura delle entità a relazione multipla, incapsulando non solo le singole entità ma anche gli aspetti delle loro relazioni. Oltre a contenere almeno un attributo semplice, un oggetto business della gerarchia ha uno o più attributi che sono *complessi*; in altre parole, l'attributo stesso contiene uno o più oggetti business denominati *oggetti business secondari*. L'oggetto business che contiene l'attributo complesso è chiamato *oggetto business principale*.

Ci sono due tipi di relazioni tra oggetti business principali e secondari:

- *Cardinalità singola*—Quando un attributo in un oggetto business principale rappresenta un *singolo* oggetto business secondario. Il tipo dell'attributo viene impostato sul nome dell'oggetto business secondario, e la cardinalità è impostata ad 1.

- *Cardinalità multipla*—Quando un attributo in un oggetto business principale rappresenta un *vettore* di oggetti business secondari. Il tipo dell'attributo viene impostato sul nome dell'oggetto business secondario, e la cardinalità è impostata ad `n`.

A turno, ogni oggetto business secondario potrà contenere attributi che contengono un oggetto business secondario o un array di oggetti business, e così via. L'oggetto business alla sommità della gerarchia, che non avrà un oggetto principale, viene chiamato *oggetto business di livello superiore*. Ogni singolo oggetto business, indipendente dai suoi oggetti business secondari che potrà contenere (o che potranno contenerlo) è chiamato *oggetto business individuale*.

In una gerarchia di oggetti business, la definizione di un oggetto business di livello superiore contiene uno o più attributi semplici, uno o più attributi che rappresentano un oggetto business secondario o un array, ed un elenco di istruzioni supportate. Figura 7 mostra una tipica gerarchia di oggetti business. L'oggetto business di livello superiore, Customer, ha sia attributi di cardinalità singola che attributi di cardinalità multipla con oggetti business secondari:

- Il suo attributo Address è un attributo complesso con cardinalità multipla. Customer è l'oggetto business principale per ciascuno degli oggetti business secondari di Address.
- Il suo attributo CustProfile è un attributo complesso con cardinalità singola. Customer è un oggetto business principale per il singolo oggetto business secondario CustProfile.

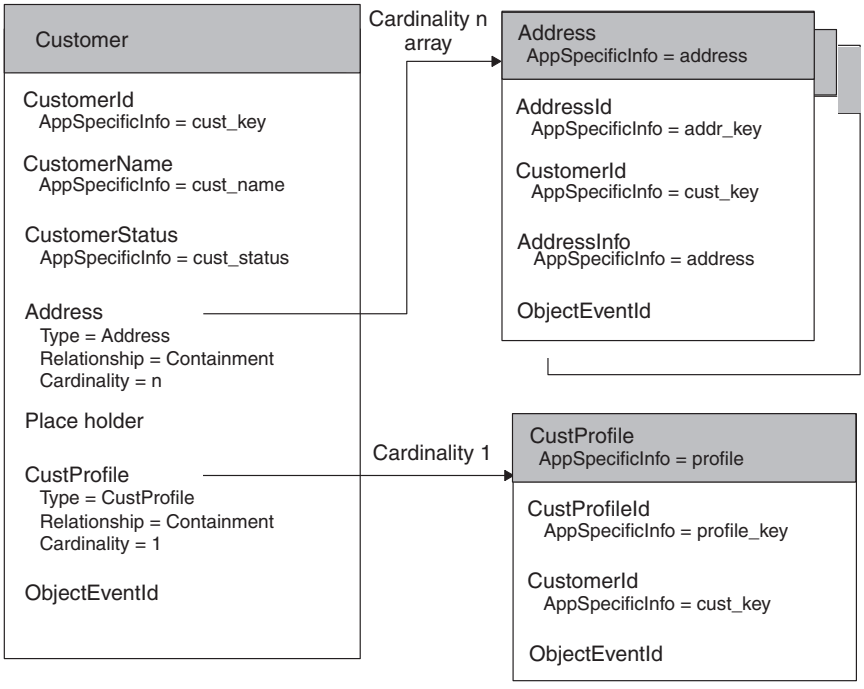



*Figura 7. Esempio di definizione di oggetto business gerarchico*

In Figura 7, gli oggetti business Customer e CustProfile, così come ogni oggetto business Address sono oggetti business individuali.

**Nota:** Quando un oggetto business di livello superiore contiene informazioni per l'elaborazione dei suoi oggetti business secondari, viene chiamato per l'oggetto business. Ad esempio, il connettore XML richiede un oggetto

business wrapper per contenere le informazioni che determinano il formato del suo oggetto business dati secondario e l'instradamento.

Quando si progetta la struttura di un oggetto business specifico dell'applicazione gerarchico, si deve determinare:

- Come il dato dell'entità è rappresentato nell'oggetto business
- Come l'entità dell'applicazione primaria si relaziona con entità secondarie
- Se un'entità dell'applicazione include dati provenienti da diverse entità, si deve decidere:
  - Se l'oggetto business specifico dell'applicazione ha bisogno di includere dati relativi
  - Come definire la relazione tra dati relativi

Per ulteriori informazioni, consultare "Considerazioni sulla progettazione di entità multiple" a pagina 25.

# Panoramica sul processo di sviluppo

Questa sezione fornisce una panoramica sul processo di sviluppo di un oggetto business.

## Impostazione dell'ambiente di sviluppo

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

- Il sistema WebSphere Business Integration Server Express deve essere installato su un computer a cui si ha accesso.

  Per informazioni su come installare ed avviare il sistema WebSphere Business Integration Server Express, fare riferimento alla guida all'installazione di WebSphere Business Integration Server Express appropriata.
- InterChange Server Express e il relativo server di database del repository siano in esecuzione. Questo passo è necessario solo quando si è pronti per salvare la definizione nel repository o ad eliminarne una. Solo per lo sviluppo è possibile eseguire localmente Business Object Designer Express, senza collegarsi a InterChange Server Express.
- Se si prevede di creare una definizione di oggetto business utilizzando un Object Discovery Agent, si deve avviare l'Object Activation Daemon (OAD) prima di utilizzare questo ODA per la generazione di definizioni di oggetti business. Per ulteriori informazioni, consultare "Prima di usare un ODA" a pagina 71.

## Fasi di sviluppo di oggetti business

Le fasi di sviluppo dell'oggetto business sono le seguenti:

1. Comprendere i requisiti dei dati che sono critici per l'integrazione del processo di business.
   - Se si sta creando un oggetto business specifico dell'applicazione, comprendere le relazioni tra connettori, gestori dati ed gli oggetti business specifici dell'applicazione supportati.
   - Se si sta creando un oggetto business generico da utilizzare con InterChange Server Express, comprendere le relazioni tra oggetti di collaborazione ed oggetti business.
2. Sviluppare le definizioni dell'oggetto business in uno dei seguenti modi:
   a. Generazione da un'origine dati—il sistema WebSphere Business Integration Server Express contiene strumenti che facilitano la generazione di una

definizione dell'oggetto business per alcuni connettori. Questi strumenti sono Object Discovery Agents oppure strumenti di linea comandi che sono progettati per la connessione ad una applicazione e per "scoprire" i requisiti di un oggetto business specifici per un'entità di business, e per generare definizioni da tali requisiti. Business Object Designer Express contiene un'interfaccia grafica utente per Object Discovery Agents, ed aiuta nella gestione dei processi di scoperta e generazione della definizione. Consultare la guida per l'utente dell'adattatore e del gestore dati che si utilizzeranno per determinare se hanno uno strumento un programma di utilità disponibile. Si può inoltre consultare la Connector Feature Checklist, che si trova nella pagina documentazione principale sotto Connectors category. Se si sta sviluppando un adattatore personalizzato per la comunicazione con l'applicazione, si può utilizzare Object Discovery Agent Development Kit per creare un Object Discovery Agent personalizzato per l'adattatore.

b. Manuale—Business Object Designer Express è un'interfaccia grafica utente che facilita la creazione manuale di definizioni di oggetti business. Questa interfaccia è particolarmente utile per lo sviluppo di oggetti business generici da utilizzare con InterChange Server Express, siccome non esiste un'applicazione in cui si possa effettuare la ricerca di oggetti.

3. Se si necessita di uno strumento per la creazione automatica di definizioni di oggetti business da un'origine dati, verificare che la struttura generata e le informazioni specifiche dell'applicazioni siano conformi ai requisiti. Consultare la guida per l'utente dell'adattatore per il connettore che utilizza la definizione dell'oggetto business, determinare se ci sono configurazioni particolari da effettuare manualmente.
4. Verificare ed effettuare il debug dell'oggetto business eseguendolo nel sistema e modificarlo se necessario.

Tabella 3 è una panoramica visiva del processo di sviluppo dell'oggetto business, e fornisce una guida di riferimento rapido ai capitoli dove è possibile trovare informazioni su argomenti specifici.

*Tabella 3. Processo di sviluppo di oggetti business*

| **Attività:** | **Passi:** | **Fare riferimento a:** |
|---|---|---|
| Progettazione di oggetti business | Identificazione dei requisiti dei dati<br>Informazioni sulle relazioni tra connettori, gestori dati ed oggetti business specifici dell'applicazione<br>Se si utilizza InterChange Server Express , informazioni sulle relazioni tra collaborazioni ed oggetti business generici | Capitolo 2 |
| Informazioni su Business Object Designer Express | Avvio di Business Object Designer Express<br>Utilizzo di Business Object Designer Express<br>Operazioni locali o in collegamento con InterChange Server Express | Capitolo 3 |

*Tabella 3. Processo di sviluppo di oggetti business (Continua)*

| **Attività:** | **Passi:** | **Fare riferimento a:** |
|---|---|---|
| Operazioni con le definizioni degli oggetti business | Creazione manuale di un oggetto business<br>Creazione di una definizione di un oggetto business utilizzando un Object Discovery Agent (ODA)<br>Eliminazione di una definizione di un oggetto business | Capitolo 4 |
| Creazione di un Object Discovery Agent | Informazioni sull'applicazione ed i suoi requisiti<br>Apprendimento della struttura di un ODA<br>Informazioni sulla relazione tra ODA e Business Object Designer Express<br>Apprendimento delle classi ODK | Capitolo 5<br>Capitolo 6<br>Capitolo 7<br>Capitolo 8<br>Capitolo 9<br>Capitolo 10<br>Capitolo 11<br>Capitolo 12<br>Capitolo 13<br>Capitolo 14<br>Capitolo 15<br>Capitolo 16 |

# Capitolo 2. Progettazione di oggetti business

La chiave di sviluppo di oggetti business è quella di sviluppare la definizione di un oggetto business che modelli quanto più precisamente (ed efficientemente) possibile i dati che hanno bisogno di essere trasmessi tra componenti del sistema di integrazione del business:

- Per i dati trasferiti tra un connettore e InterChange Server Express, vengono progettati *oggetti business specifici dell'applicazione* che modellano entità appropriate dell'applicazione. Tali entità potrebbero corrispondere alle strutture dei dati o a standard tecnologici usati dal server web.
- Per i dati elaborati all'interno di business logic di un oggetto di collaborazione InterChange Server Express, si progettano *oggetti business generici* che contengono un insieme di informazioni per le entità delle applicazioni che hanno bisogno di comunicare. Quando l'oggetto di collaborazione scambia informazioni con un'applicazione, le mappe convertono i dati tra oggetto generici di business e le strutture dell'oggetto business specifiche dell'applicazione.

In questo capitolo vengono presentate le informazioni sulla struttura di oggetti business per il sistema WebSphere Business Integration Server Express e indica le raccomandazioni per la progettazione di oggetti business generici e specifici dell'applicazione. Il materiale qui presentato presuppone una comprensione del concetto basilare di oggetto descritto in *System Implementation Guide*:

Gli argomenti principali di questo capitolo sono:

- "Determinazione della struttura dell'oggetto business"
- "Progettazione degli oggetti business specifici dell'applicazione" a pagina 32
- "Progettazione di oggetti business generici (solo InterChange Server Express)" a pagina 41
- "Determinazione dei requisiti di mappatura per gli oggetti business (Solo InterChange Server Express)" a pagina 45

## Determinazione della struttura dell'oggetto business

Lo scopo dell'oggetto business è quello di trasportare dati tra componenti del sistema business integration e le applicazioni che esso integra. Pertanto, l'oggetto business deve modellare i dati che hanno bisogno di essere trasportati. Questi dati vengono di solito associati ad un'entità in un'applicazione in una tecnologia che il sistema business integration integra. La struttura di un oggetto business può essere una delle seguenti:

- "Rappresentazione di un'entità"
- "Rappresentazione di entità multiple" a pagina 18

Inoltre, questa sezione fornisce "Considerazioni sulla progettazione di entità multiple" a pagina 25.

### Rappresentazione di un'entità

Il progetto più semplice di un oggetto business è un oggetto business piatto che rappresenta una entità. Tutti gli attributi di un oggetto business piatto sono

semplici (cioè, ogni attributo rappresenta un valore, come `String` o `Integer` o `Date`). Per ulteriori informazioni, consultare “Oggetti business senza nessun effetto” a pagina 12.

Nel caso di un oggetto business specifico dell'applicazione, un oggetto business piatto può rappresentare una entità in un'applicazione o in uno standard tecnologico. Ad esempio, si supponga che un'applicazione abbia una tabella di database che descriva un record. Si supponga inoltre che tale tabella abbia cinque colonne che si chiamano ObjectID, UserName, TimeStamp, Detail, e Status (consultare Figura 8). L'ObjectID è la chiave primaria per ciascuna riga, ed il suo valore è generato dall'applicazione. Questa tabella non ha relazioni con altre tabelle.

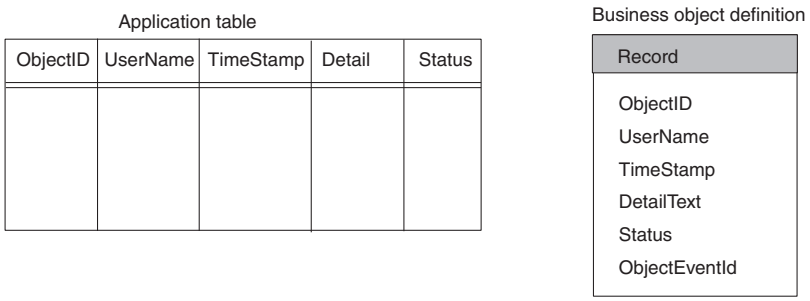



*Figura 8. L'ogetto business piatto rappresenta una entità*

Come illustrato in Figura 8, l'oggetto business Registra progettato per rappresentare la tabella potrebbe avere cinque attributi, uno per ogni colonna, con l'attributo chiave corrispondente alla colonna ObjectID.

L'uso di oggetti business piatti può semplificare il progetto del connettore corrispondente nei modi di seguito riportati:

- In un'operazione di creazione, il connettore potrebbe eseguire un ciclo attraverso gli attributi, estraendo i valori dell'attributo che non sono chiave dall'istanza dell'oggetto business ed estraendo le istruzioni di elaborazione dalla definizione di oggetto business. Una volta assemblate le informazioni necessarie all'elaborazione dell'oggetto business, il connettore potrebbe avviare una chiamata ad una funzione dell'applicazione o un'istruzione SQL per creare una nuova riga per il record all'interno della tabella. Il connettore restituisce un valore per la chiave al sistema di integrazione del business.
- In un'operazione di richiamo, il connettore potrebbe estrarre la chiave primaria dalla richiesta dell'oggetto business, usare il valore chiave per richiamare l'impostazione corrente dei dati per la riga e ritornare all'oggetto business con l'insieme completo dei valori.

Questo tipo di oggetto business è semplice nella sua struttura e nella logica del connettore richiesta per l'elaborazione. Tipicamente, tuttavia, le entità dell'applicazione sono più complesse ed includono informazioni che vengono memorizzate in altri oggetti.

## Rappresentazione di entità multiple

Un oggetto business può rappresentare entità di applicazioni che includono dati di altre entità in uno dei modi riportati in Tabella 4 a pagina 19.

*Tabella 4. Rappresentazione di entità multiple*

| Struttura dell'oggetto business | Tipo di organizzazione dei dati | Tipo di relazioni principale/secondario |
|---|---|---|
| L'oggetto business principale può avere uno o più oggetti business secondari che rappresentano le altre entità. | Una-a-una<br>Una-a-molte | Strutturale |
| L'oggetto business principale può avere uno o più attributi di chiave esterni che fanno riferimento ad altri oggetti business di livello superiore che rappresentano le altre entità. | Una-a-una<br>Una-a-molte<br>Molte-a-molte<br>Molte-a-una | Semantico |
| Se un'applicazione e la sua interfaccia lo permettono, un oggetto business piatto può includere attributi che fanno riferimento direttamente ad altre entità. | Una-a-una | Nessuno |

Quando si decide il modo in cui strutturare gli oggetti business che rappresentano entità multiple, tenere in considerazione le seguenti linee guida:

- Se la relazione tra entità è di tipo una-a-molte, rappresentare i dati nelle entità subordinate come oggetti business secondari. Ad esempio:
  - Quando si lavora con le tabelle del database, se la riga di un'entità è correlata ad una o più righe in un'altra entità ed è la sola entità correlata all'entità subordinata, creare un oggetto business secondario separato per ciascuna riga correlata.
  - Quando si lavora con un DTD, se un elemento XML ha un attributo con una cardinalità di *, si crea un oggetto business secondario separato per ciascun attributo dell'elemento correlato.
- Se la relazione tra entità è di tipo molte-a-molte, rappresentare i dati nelle entità correlate come oggetti business di livello superiore referenziati dal processo principale piuttosto che come contenuti dal processo principale.
- Se la definizione di un oggetto business per un'entità include molti attributi provenienti da un'altra entità e gli attributi di una seconda entità formano un raggruppamento logico, è possibile che si voglia creare la definizione di un oggetto business secondario per la seconda entità piuttosto che individuare tutti gli attributi per entrambe le entità nella stessa definizione di oggetto business.
- Se un oggetto business esistente contiene già altri oggetti business secondari, la creazione di uno o più oggetti business secondari che rappresentano nuove entità rende consistente la struttura dell'oggetto business.

Le seguenti sezioni descrivono nei dettagli ciascuna di queste rappresentazioni.

## Relazioni strutturali

In una relazione strutturale, l'oggetto business principale contiene fisicamente l'oggetto business secondario. Un oggetto business di questo tipo è un oggetto business gerarchico: almeno uno dei suoi attributi è complesso (cioè, contiene un oggetto business secondario o un array di oggetti business secondari). La proprietà dell'attributo della relazione per questo attributo si definisce di `contenimento`, ad indicare una relazione di contenimento. Il tipo di questo attributo è il tipo di oggetto business secondario (o di oggetti) che rappresenta. Per ulteriori informazioni, consultare "Oggetti business gerarchici" a pagina 12.

I seguenti oggetti business gerarchici rappresentano relazioni strutturali:

- Poiché un ordine è composto da elementi di riga, un oggetto business Ordine contiene un array di oggetti business LineItem. La relazione di contenimento ha cardinalità multiple perché ciascun ordine può contenere elementi multipli di riga. Questa struttura rappresenta una relazione una-a-molte.
- Poiché un impiegato è associato ad un indirizzo di casa, l'oggetto business Impiegato contiene un oggetto business Indirizzo. La relazione di contenimento ha una cardinalità singola poiché ciascun impiegato può essere associato ad un solo indirizzo di casa. Questa struttura rappresenta una relazione una-a-una.

In entrambi i casi, poiché l'oggetto business principale contiene il processo secondario o l'array di secondari, la relazione viene definita strutturale.

Una relazione strutturale presuppone che l'oggetto business principale possegga i dati all'interno dell'oggetto secondario. In tal modo, quando viene creato un nuovo impiegato, viene inserita una nuova riga nella tabella dell'indirizzo per contenere gli indirizzi degli impiegati. Analogamente, quando un impiegato viene cancellato, viene cancellato anche il suo indirizzo dalla tabella dell'indirizzo.

## Relazioni semantiche

In una relazione semantica, l'oggetto business principale fa riferimento al secondario oppure fa riferimento al principale. Quando l'oggetto business fa riferimento ad altro, memorizza un valore che identifica univocamente l'altro, ma che non contiene l'altro. In questo caso, il componente che elabora l'oggetto business ricava la relazione semanticamente.

Una relazione semantica è tipicamente definita da un attributo semplice che serve come *chiave esterna*. L'attributo della chiave esterna è collocata in un oggetto business e contiene l'identificativo univoco (definito *chiave primaria*) dell'altro. In altre parole, l'oggetto business ha un attributo di chiave primaria che contiene il suo identificativo univoco. Inoltre, uno degli oggetti business ha anche un attributo di chiave esterna che contiene il valore della chiave primaria dell'altro. La chiave esterna stabilisce semanticamente il collegamento tra principale e secondario.

Le relazioni semantiche sono importanti quando ci sono relazioni molte-a-molte o molte-a-una tra entità; in altre parole, quando più di un principale ha relazioni con lo stesso secondario. Il mettere in relazione semanticamente le entità piuttosto che strutturalmente, isola i dati secondari, il che è importante per mantenere la coerenza dei dati.

Poiché il principale non contiene il secondario in una relazione definita semanticamente, il connettore che gestisce le richieste per il principale ed il secondario li riceve in operazioni separate. In altre parole, le richieste vengono inviate separatamente al connettore, che gestisce il principale ed il secondario in operazioni separate. Per ulteriori informazioni consultare "Proprietà dei dati nelle relazioni" a pagina 25 e "Scelta tra una relazione semantica e una strutturale" a pagina 27.

Considerare le opzioni di progettazione in Tabella 5 per specificare una relazione semantica.

*Tabella 5. Opzioni di progettazione per relazioni semantiche.*

| Opzioni di progettazione | Tipo di relazione |
|---|---|
| "Memorizzazione di una chiave esterna nell'oggetto principale" a pagina 21 | una-a-una<br>Molte-a-una |

*Tabella 5. Opzioni di progettazione per relazioni semantiche. (Continua)*

| Opzioni di progettazione | Tipo di relazione |
|---|---|
| "Memorizzazione di una chiave esterna in un oggetto secondario" a pagina 22 | Una-a-molte |
| "Memorizzazione di chiavi esterne in un array di oggetti secondari" a pagina 22 | Una-a-molte<br>Molte-a-molte |
| "Memorizzazione di una chiave esterna in un albero di oggetto business" a pagina 23 | Una-a-una |

**Memorizzazione di una chiave esterna nell'oggetto principale:** Nell'uso più semplice di chiavi esterne, la chiave esterna che stabilisce la relazione viene memorizzata nella principale. In questo caso, una principale può contenere un riferimento solo ad una secondaria di un dato tipo. La relazione tra principale e secondaria è chiaramente definita nella principale. Pertanto, questa struttura rappresenta una relazione una-a-una. Tuttavia, oggetto business principali possono far riferimento allo stesso oggetto business secondario per implementare relazioni molte-a-una.

**Nota:** Quando la chiave esterna che stabilisce la relazione viene memorizzata nella principale, una principale può contenere attributi multipli e ciascuno contiene un riferimento al secondario, ma ognuno di questi attributi contiene di solito un tipo differente di secondario.

In Figura 9, l'oggetto business Cliente ha due attributi (`AddressId` e `CustInfo`) ciascuno dei quali contiene un riferimento all'oggetto business secondario. Gli attributi della chiave esterna in Cliente identificano prontamente la relazione principale alle due secondarie.

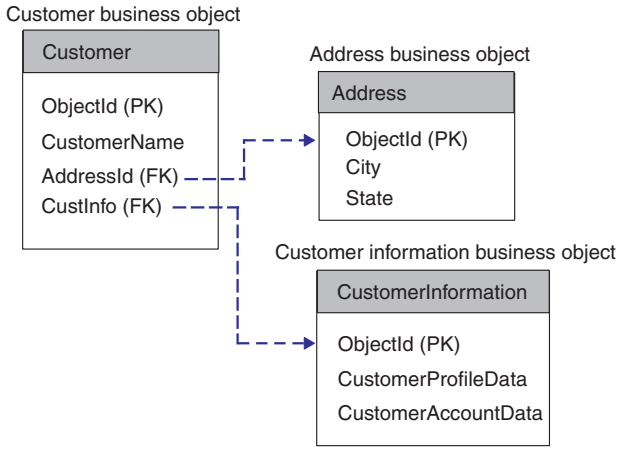



*Figura 9. una-a-una: Attributi multipli della chiave esterna memorizzati nell'oggetto business principale*

**Nota:** In Figura 9, l'acronimo "PK"viene usato per indicare una chiave primaria e "FK" è usato per indicare una chiave esterna. Inoltre, questi oggetti business seguono le convenzioni di ridenominazione per oggetti business generici ridenominando gli attributi delle loro chiavi primarie `ObjectId`. In un oggetto business specifico dell'applicazione, è di solito meglio assegnare il nome all'attributo dopo il nome o il suo campo equivalente o dopo la colonna nell'applicazione.

usando InterChange Server Express, è possibile esaminare l'oggetto business generico Ordine distribuito per un esempio di oggetto principale che memorizza un riferimento di chiave esterna in un altro oggetto. Contiene l'attributo CustomerId, che fa riferimento all'oggetto business generico Cliente di livello superiore. Consultare Figura 11 per una illustrazione dell'oggetto business Ordine.

**Memorizzazione di una chiave esterna in un oggetto secondario:** In alternativa, la chiave esterna che stabilisce la relazione può essere memorizzata nel secondario. Questo caso rappresenta una relazione una-a-molte; cioè, secondarie multiple possono fare riferimento alla stessa principale. Tuttavia, poiché la relazione tra principale e secondaria è definita nella secondaria, la relazione è invisibile quando si esamina solo la principale. Pertanto, se un oggetto business principale esegue il trigger del flusso di una relazione, quelle secondarie non possono essere richiamate—non c'è riferimento ad esse nell'oggetto business principale che sta viaggiando attraverso il sistema.

In Figura 10, l'attributo della chiave esterna viene memorizzato in ciascun oggetto business secondario, piuttosto che in quello principale. Questa struttura consente secondarie multiple da mettere in relazione semanticamente alla stessa principale. In questo caso, tuttavia, poiché l'oggetto business principale non ha attributi che contengano un riferimento ad un oggetto business secondario, non c'è modo di identificare la relazione principale alla sua secondaria o, data la principale, di richiamare tutte le sue secondarie correlate.

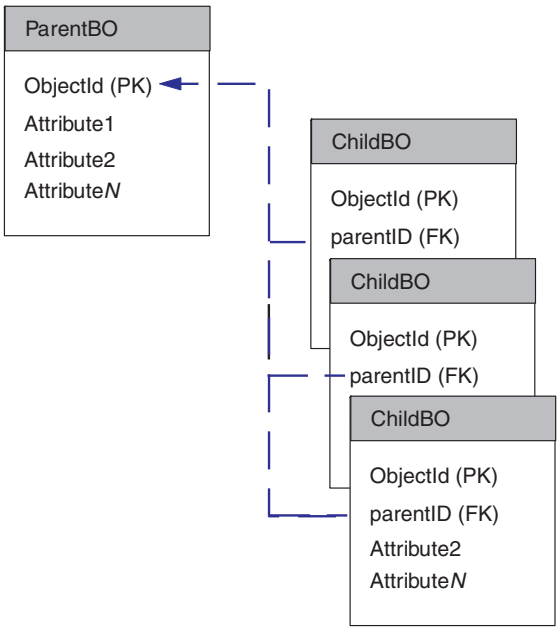



*Figura 10. Molte-a-una: Chiave esterna memorizzata in oggetto business multipli secondari*

**Nota:** In Figura 10, l'acronimo "PK" è usato per indicare una chiave primaria e "FK" per indicare una chiave esterna.

**Memorizzazione di chiavi esterne in un array di oggetti secondari:** Per rappresentare una relazione una-a-molte, la chiave esterna che al momento stabilisce la relazione, è memorizzata in un attributo semplice in un oggetto business secondario. L'oggetto business principale contiene strutturalmente un array dei secondari. In altre parole, il principale contiene un array di oggetti business principali, ciascuno dei quali contiene una chiave esterna che fa

riferimento ad un altro oggetto business di livello superiore. In aggiunta, oggetti business multipli principali possono far riferimento allo stesso oggetto business secondario nel loro array di oggetti business secondari per implementare una relazione molte-a-molte.

**Nota:** Con InterChange Server Express, ci sono svariati oggetti business che si possono esaminare per vedere un esempio di relazione principale-secondario di questo tipo. L'Ordine generico e gli oggetti business ContactRef forniscono un esempio di tale opzione. L'ordine contiene l'attributo OrderContactRef, che contiene un array di oggetti business generici ContactRef. Ciascun oggetto business ContactRef contiene un attributo ContactId che mantiene un riferimento all'oggetto business generico Contatto di livello superiore.

In Figura 11, l'oggetto business Ordine contiene un riferimento ad un oggetto business Cliente e strutturalmente contiene un array di oggetti business ContactRef business. Ciascun oggetto business ContactRef contiene un riferimento ad un oggetto business Contact.

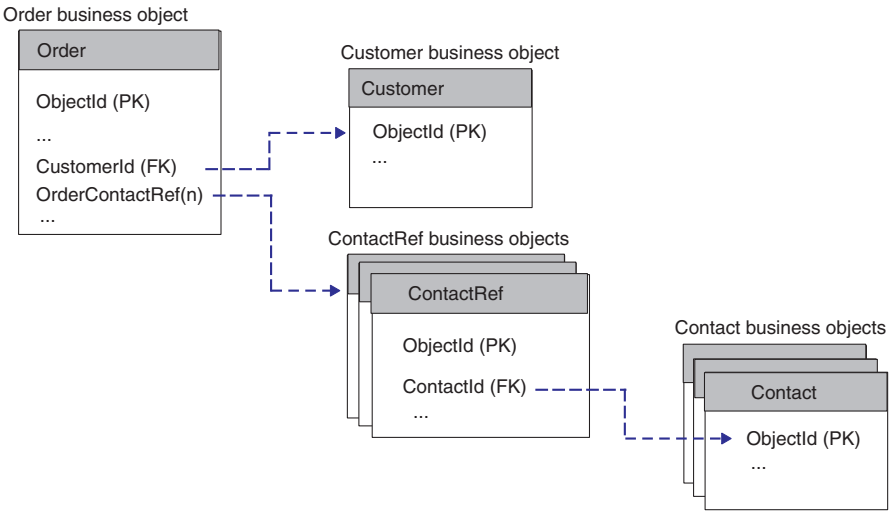



*Figura 11. L'oggetto business contenente un oggetto business secondario che memorizza le chiavi esterne*

**Nota:** In Figura 11, l'acronimo "PK" è usato per indicare una chiave primaria e "FK" è usato per indicare una chiave esterna.

**Memorizzazione di una chiave esterna in un albero di oggetto business:** Nella progettazione, la chiave esterna che stabilisce la relazione viene memorizzata in un oggetto business "secondario" il cui principale è un altro business del suo stesso tipo. Con InterChange Server Express, è possibile esaminare l'oggetto business generico InstalledProduct per un esempio di questa progettazione. Questo oggetto business contiene l'attributo ParentId, che può contenere un riferimento ad un altro oggetto business InstalledProduct, che è il principale diretto dell'oggetto business corrente.

In Figura 12, l'attributo ParentId di un oggetto business InstalledProduct contiene un riferimento all'attributo della chiave primaria (ObjectId) del suo immediato

oggetto business principale InstalledProduct. In testa a questa gerarchia c'è l'oggetto business il cui attributo ParentId non contiene un valore.

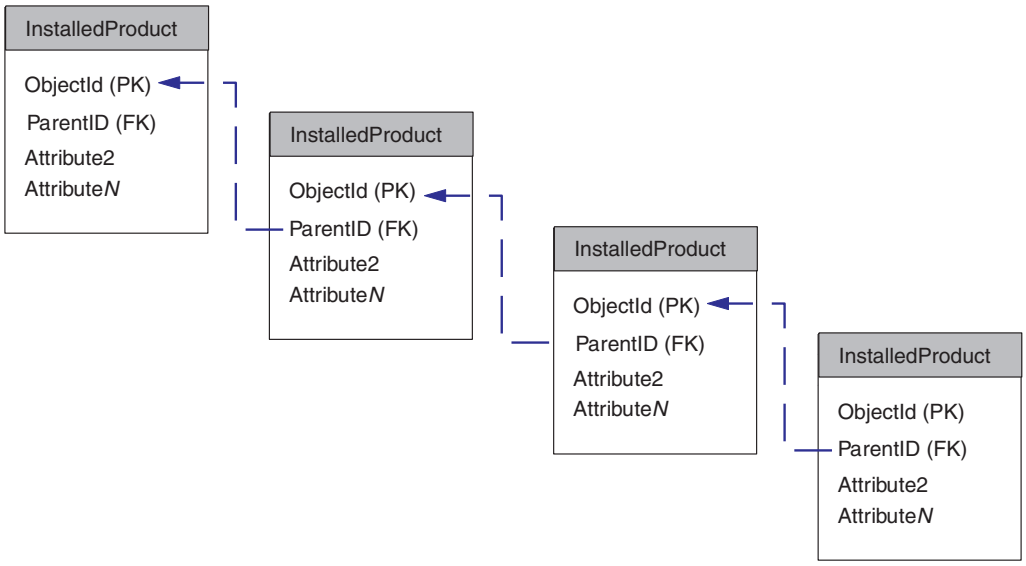



*Figura 12. Oggetto business durante la memorizzazione di una chiave esterna nel suo principale dello stesso tipo*

**Nota:** In Figura 12, l'acronimo "PK" è usato per indicare una chiave primaria e "FK" è usato per indicare una chiave esterna.

Poiché un oggetto business InstalledProduct può contenere un riferimento al suo oggetto business principale, il sistema di integrazione del business può sincronizzare i prodotti installati che sono parte di di un'ampia gerarchia. Il sistema di integrazione del business può gestire i componenti di una gerarchia complessa di prodotti installati come gli oggetto business individuali InstalledProduct. Con InterChange Server Express, per ulteriori informazioni è possibile visualizzare la documentazione della maschera di collaborazione *InstalledProductSync*.

## Oggetto business Piatto che rappresenta le entità correlate

Se l'interfaccia dell'applicazione fornisce la possibilità d unificare entità multiple di applicazione, è possibile definire un oggetto business piatto che contenga gli attributi che fanno riferimento all'entità primaria e alle entità correlate. Se la relazione tra entità è di tipo una-a-una, dove una istanza dell'entità primaria può essere associata ad una istanza di ciascuna entità correlata, gli attributi provenienti da entità multiple possono essere inclusi in un oggetto business.

Quando si progetta un oggetto business di questo tipo, specifico dell'applicazione, si potrebbe aver bisogno di usare informazioni specifiche dell'applicazione per specificare la posizione dei dati dell'attributo nell'applicazione in modo che il connettore possa trovare ed elaborare correttamente i dati.

Figura 13 fornisce un esempio di oggetto business piatto del sistema WebSphere Business Integration Server Express che rappresenta dati in due entità, una come tabella contenente i dati dell'indirizzo e l'altra come tabella contenente dati di ricerca per le abbreviazioni di stato/provincia e nazione.

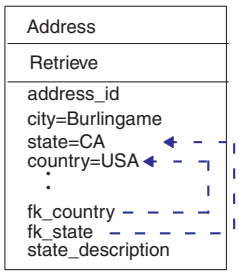



*Figura 13. Oggetto business piatto che rappresenta due entità*

Questo esempio usa informazioni specifiche dell'applicazione per stabilire una chiave esterna tra le entità. In questo caso, il connettore esegue una ricerca da un valore in un attributo che rappresenta una tabella per fornire un valore per un attributo che rappresenta un'altra tabella. Per richiamare questi dati, il connettore esegue due letture di tabella.

Sebbene gli oggetti business piatti possano incapsulare informazioni provenienti da o incluse in entità di applicazioni multiple, problemi di integrazione di applicazioni incrociate spesso richiedono una logica di integrazione delle strutture di dati più complessa e più complicata di quella che gli oggetti business piatti possono rappresentare. Per gestire una maggiore complessità nelle entità dell'applicazione e nei requisiti di integrazione, il sistema WebSphere Business Integration Server Express fornisce oggetti business gerarchici.

## Considerazioni sulla progettazione di entità multiple

Questa sezione fornisce le seguenti considerazioni durante la progettazione di oggetti business per entità multiple:

- "Proprietà dei dati nelle relazioni"
- "Scelta tra una relazione semantica e una strutturale" a pagina 27

### Proprietà dei dati nelle relazioni

Il modo in cui si progettano i propri oggetti business per rappresentare entità multiple produce un effetto sulla proprietà dei dati:

- Una relazione strutturale presume che l'oggetti business principale possegga i dati secondari.
- Una relazione semantica *non* presume che l'oggetti business principale possegga i dati all'interno dell'oggetto secondario.

Questa distinzione è significativa quando si considera la coerenza dei dati di un'entità condivisa da oggetti business multipli.

Ad esempio, si supponga che un cliente ed contatto condividano un indirizzo. Se gli oggetti business Cliente e Contatto contengono un riferimento ad un oggetto business Indirizzo (una relazione semantica) invece di contenere l'oggetto business (una relazione strutturale), le modifiche all'Indirizzo possono essere apportate indipendentemente dalle modifiche a Cliente o a Contatto.

Tuttavia, se ciascun oggetto business Cliente e Contatto contiene l'oggetto business Indirizzo, le modifiche all'Indirizzo fatte dal Cliente possono ricoprire quelle fatte dal Contatto. In tal caso, due diversi oggetti di collaborazione (CustomerSync e ContactSync) potrebbero aggiornare gli stessi dati dell'indirizzo contemporaneamente, comportando un'incoerenza nei dati.

Se il Cliente e il Contatto hanno una relazione con l'oggetto business Indirizzo di tipo semantica piuttosto che strutturale, è possibile limitare le modifiche ai dati dell'indirizzo ad una terza interfaccia. Per esempio, si potrebbe avere una interfaccia per ciascun oggetto business Contatto e Cliente. Entrambe queste interfacce potrebbero delegare la gestione degli oggetti business Indirizzo ad una terza interfaccia. Con InterChange Server Express ciò viene fatto con la chiamata AddressSync degli oggetti di collaborazione CustomerSync e ContactSync attraverso un oggetto di collaborazione wrapper piuttosto che facendo direttamente le modifiche da sé. Per ulteriori informazioni sulla progettazione di oggetti business per la gestione della coerenza dei dati per InterChange Server Express scenari di integrazione, consultare "Designing for Parallel Execution" in *Collaboration Development Guide*.

Figura 14 illustra le differenze tra semantico e strutturale, definendo la relazione ad un oggetto business secondario.

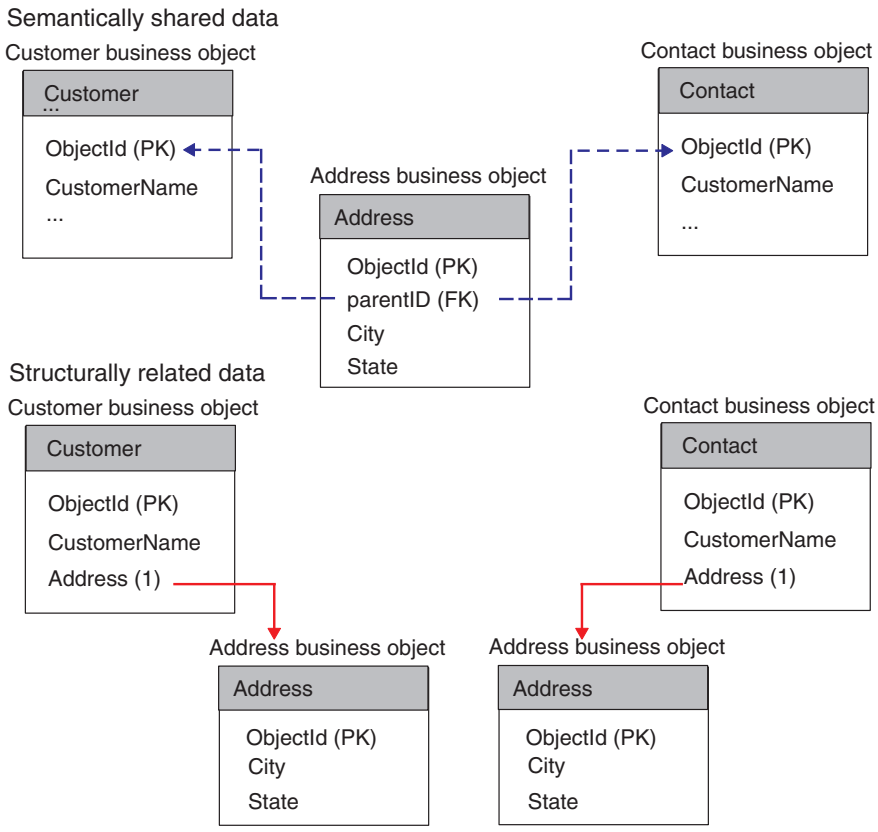



*Figura 14. Confronto delle relazioni semantiche e strutturali*

La figure precedente illustra due tipi di relazioni ai dati secondari:

- Semantica—L'oggetto business secondario Indirizzo, collegato semanticamente sia a Cliente che a Contatto, contiene il valore della sua chiave primaria principale in un semplice attributo di chiave esterna. In questo caso, il nome dell'attributo della chiave principale in entrambi i principali è la stessa, il che semplifica il collegamento da secondario a principale.
- Strutturale—I due oggetti business che si collegano strutturalmente ad Indirizzo, hanno un attributo che rappresenta un'istanza di secondario. In questo caso, i dati nel secondario sono correlati solo al principale che lo contiene e non sono condivisi.

## Scelta tra una relazione semantica e una strutturale

Come mostra Tabella 4 a pagina 19, entrambe le relazioni una-a-una e una-a-molte può essere rappresentata da una relazione strutturale o semantica. Tabella 6 riepiloga queste rappresentazioni strutturali e semantiche.

*Tabella 6. Rappresentazioni delle relazioni una-a-una e una-a-molte*

| Tipo di relazione | Rappresentazione strutturale | Rappresentazione semantica |
|---|---|---|
| Una-a-una(cardinalità singola) | Un attributo in un oggetto business principale rappresenta un oggetto business secondario. | Un attributo in un oggetto business principale è semplice e contiene la chiave esterna per far riferimento all'oggetto business secondario. |
| Una-a-molte (cardinalità multipla) | Un attributo in un oggetto business principale rappresenta un array di oggetti business secondari. | Ciascun oggetto business secondario multiplo contiene un attributo di chiave esterna che memorizza la chiave primaria principale. |

Figura 9 e Figura 10 illustrano gli oggetti business le cui relazioni singole e di cardinalità multiple sono definite semanticamente. Gli oggetti business nell'esempio potrebbero rappresentare dati memorizzati nel database. Le relazioni tra oggetto business che rappresentano questo tipo di dati possono essere definiti sia semanticamente che strutturalmente. Per questo tipo di dati, la relazione tra principale e secondario può essere definita sia dal punto di vista semantico che strutturale negli stessi due oggetti business.

**Scelta di una relazione semantica:** Per implementare una relazione semantica, l'applicazione sottostante deve poter supportare chiavi esterne. Ad esempio, quando un oggetto business rappresenta i dati del database, può stabilire la relazione tra entità sia semanticamente che strutturalmente. Questi tipi di oggetti business sono progettati in modo ridondante. In altre parole, il componente che li elabora può individuare la secondaria tramite la principale e la principale tramite ciascuna secondaria.

Ad esempio, si supponga che un'applicazione abbia una tabella che rappresenti ordini di acquisto. Questa tabella è correlata tramite chiavi esterne ad una tabella che contiene elementi di riga per gli ordini di acquisto. Righe multiple nella tabella degli elementi di riga fanno riferimento ad una riga nella tabella degli ordini di acquisto. Figura 15 illustra queste tabelle.

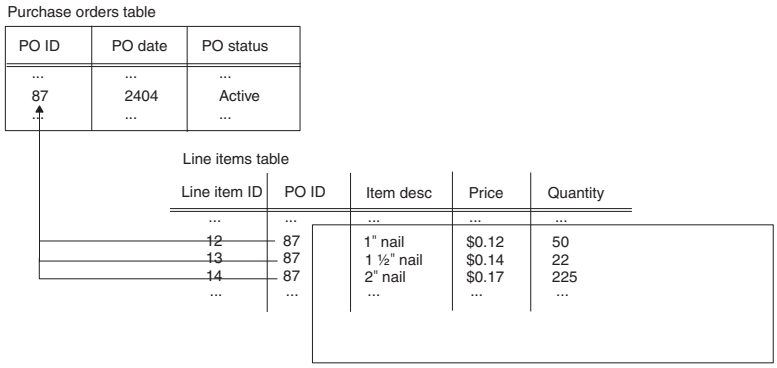



*Figura 15. Esempio di tabelle di applicazioni con una relazione semantica una-a-molte.*

Figura 16 illustra oggetti business che potrebbero corrispondere a queste tabelle. Questa figura mostra un oggetto business PurchaseOrder di livello superiore e tre oggetti business secondari LineItem.

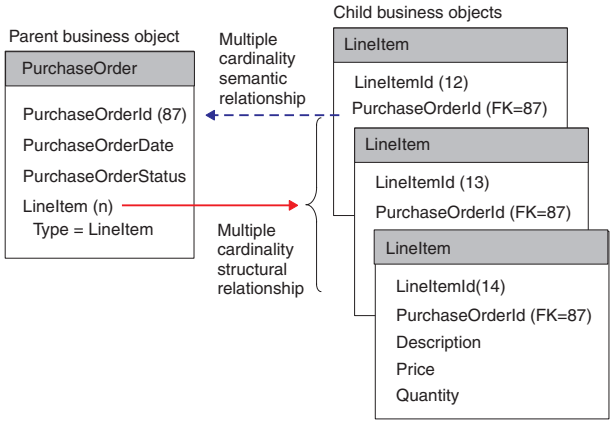



*Figura 16. Oggetti business di esempio con relazioni semantiche e strutturali di cardinalità multipla.*

L'oggetto business illustrato PurchaseOrder ha una relazione sia semantica che strutturale al suo secondario LineItem. L'attributo PurchaseOrderId in ciascun secondario crea il collegamento semantico di chiave esterna al principale dal secondario. L'attributo LineItem nel principale, definito con `cardinalità n`, crea il collegamento strutturale al secondario dal principale.

**Nota:** IBM non distribuisce oggetti business con chiavi esterne memorizzate nel secondario. Questo documento espone l'esempio precedente solo per illustrare i modi diversi di collegare dati principali e secondari.

**Scelta della relazione strutturale:** Se l'applicazione sottostante non supporta chiavi esterne, c'è probabilmente bisogno di implementare la relazione strutturale. Ad esempio, un DTD, che rappresenta un documento XML *non* supporta informazioni di chiavi esterne. Pertanto, devono essere definite strutturalmente

relazioni una-a-una o una-a-molte. Il seguente DTD Ordine, che contiene elementi che corrispondono all'entità di una applicazione `Ordine` , illustra relazioni di cardinalità singole e multiple:

```
<!-- Ordine -->
<!-- Dichiarazione elemento -->
<!ELEMENT Ordine (Unità+)>
<!ELEMENT Unità (PartNumber?, Quantità, Prezzo, Accessori*)>
<!ELEMENT PartNumber (#PCDATA)>
<!ELEMENT Quantità (#PCDATA)>
<!ELEMENT Prezzo (#PCDATA)>
<!ELEMENT Accessori (Quantità, Tipo)>
<!ATTLIST Nome accessorio CDATA >
<!ELEMENT Tipo (#PCDATA)>
```

Figura 17 illustra un oggetto business che rappresenta un DTD Ordine. L'oggetto business di livello superiore contiene l'oggetto business con una relazione una-cardinalità, e Ordine contiene oggetti business Unità secondari con una relazione a cardinalità multipla. Subito dopo, Unità contiene oggetti business Accessori con una relazione di cardinalità multipla.

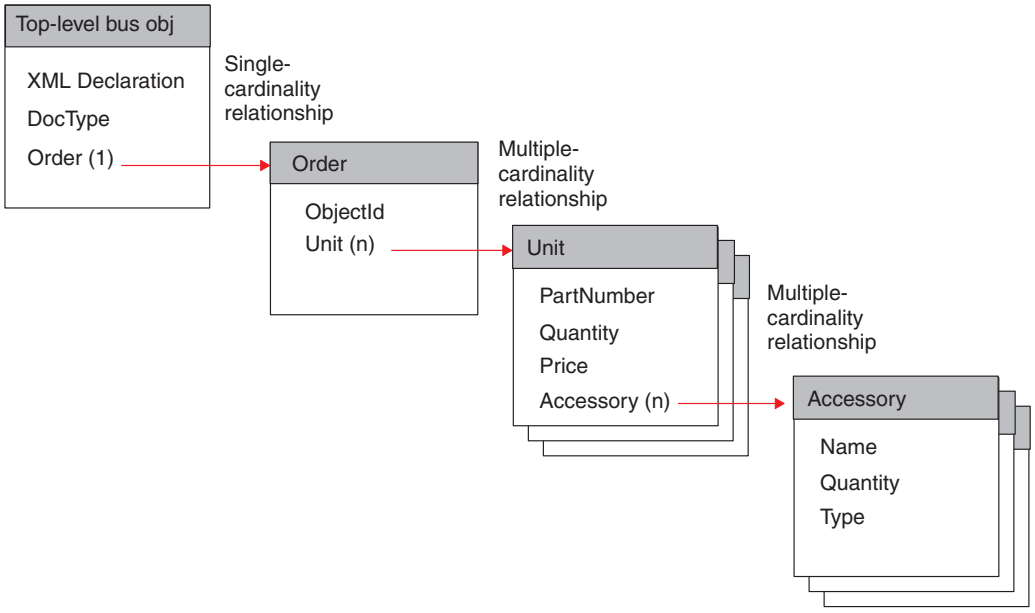



*Figura 17. Relazioni strutturali a cardinalità singola e multipla*

Le relazioni tra oggetti business illustrate in Figura 17 sono definite strutturalmente; cioè, ciascun oggetto business principale contiene un attributo il cui tipo è lo stesso del secondario e la cui relazione è specificata come `contenimento`.

**Importante:** Il programma di gestione dei dati XML ha requisiti specifici dell'oggetto business di livello superiore che rappresenta un DTD. Per informazioni su questi requisiti, consultare *Data Handler Guide*.

## Abilitazione di oggetti business per script bidirezionali

Se richiesti, gli oggetti business possono supportare script bidirezionali come valori per gli attributi della stringa o come parte dei metadati dell'oggetto. Usare i metodi descritti in Capitolo 17, "Classe CxBiDiEngine", a pagina 243 per trasformare le stringhe o gli oggetti business.

Il supporto per gli script bidirezionali avviene sia sulla maschera dell'oggetto business sia su ciascuna istanza dell'oggetto business.

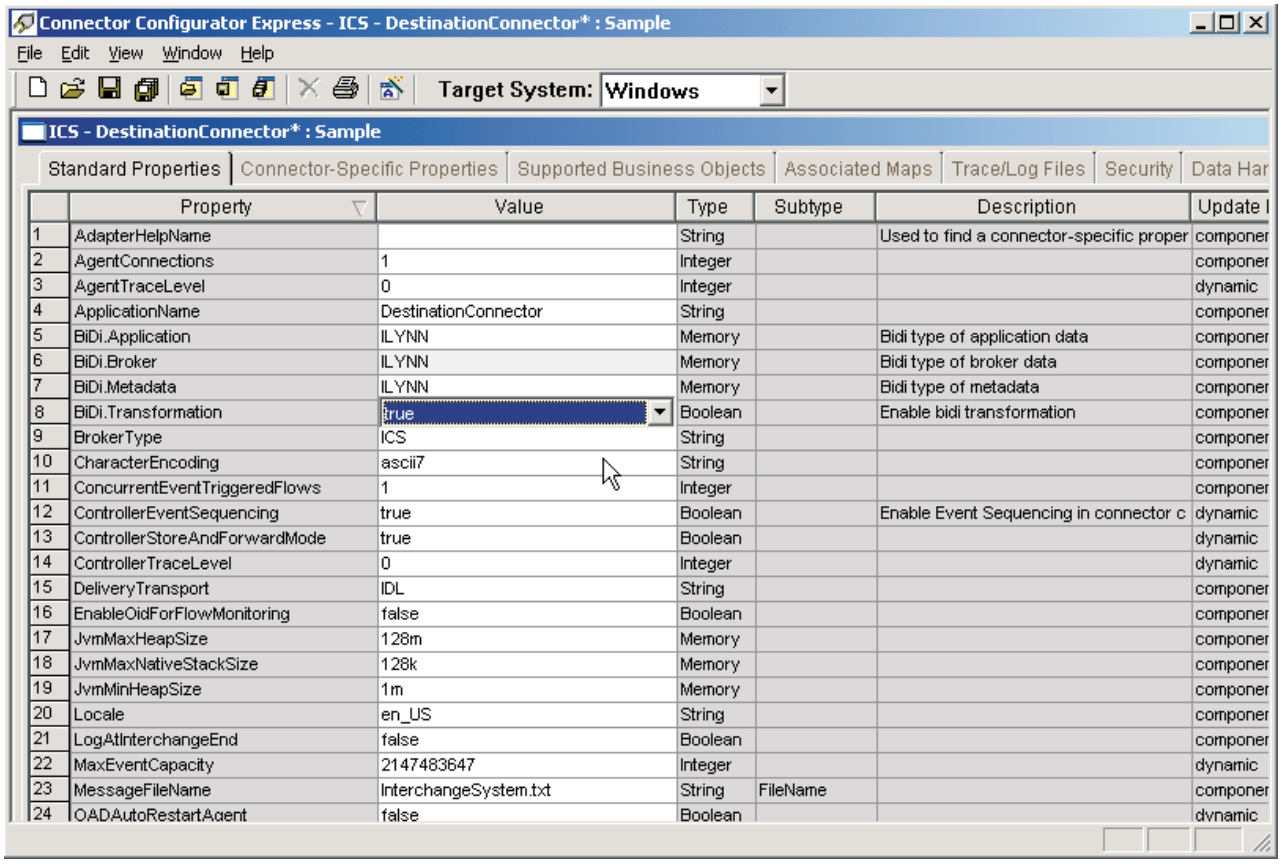
Il supporto dello script bidirezionale in una maschera di oggetto business è automatico; non c'è bisogno di eseguire configurazioni aggiuntive. Il supporto sulle maschere consente la digitazione e la visualizzazione corretta di caratteri in lingue bidirezionali.

Per abilitare il supporto script bidirezionale per le istanze oggetto business, configurare le connessioni come descritto in "Abilitazione dei connettori per script bidirezionali".

## Abilitazione dei connettori per script bidirezionali

Per abilitare un connettore a supportare uno script bidirezionale:

1. Impostare la proprietà BiDiTransformation su `true` nella scheda **Standard** di Connector Configurator Express. (Consultare Figura 18 ). Impostando il valore su `true` si consente la visualizzazione di altri parametri che supportano gli script bidirezionali per il connettore.




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

*Figura 18. La proprietà BiDiTransformation property in Connector Configurator Express*

2. Specificare le opzioni bidirezionali per quanto di seguito riportato:
   - BiDi.Application
   - BiDi.Broker
   - BiDi.MetaData

Ciascuna proprietà visualizza una finestra di dialogo (consultare Figura 19) in cui si scelgono i parametri bidirezionali da supportare. Consultare Tabella 7 per una descrizione dei parametri.

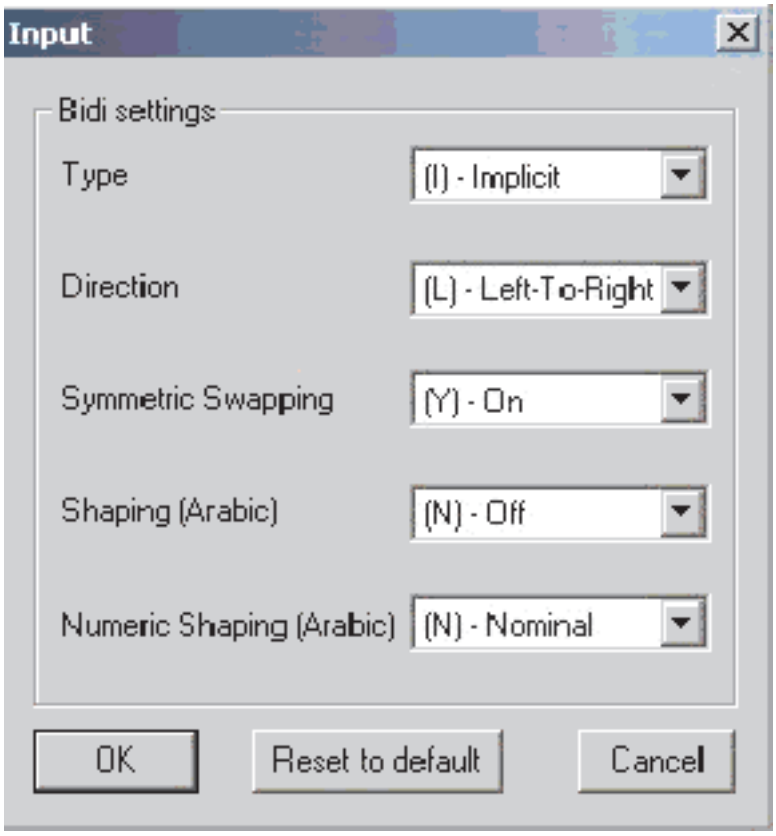



*Figura 19. Finestra di dialogo per l'immissione del parametro script bidirezionale*

*Tabella 7. Valori per stringhe di formato bidirezionale*

| Posizione lettera | Scopo | Valori | Descrizione | Predefinito |
|---|---|---|---|---|
| 1 | Tipo | I | Implicito (Logico) | I |
| | | V | Visivo | |
| 2 | Direzione | S | Sinistra a destra | S |
| | | R | Destra a sinistra | |
| 3 | Inversione simmetrica | Y | Inversione simmetrica attiva | Y |
| | | N | Inversione simmetrica non attiva | |
| 4 | Struttura | Y | Il testo è strutturato | N |
| | | N | Testo non strutturato | |
| 5 | Struttura numerica | H | Hindi | N |
| | | C | Contestuale | |
| | | N | Nominale | |

3. Distribuzione del connettore. (Consultare *System Implementation Guide* per ulteriori informazioni sui connettori di distribuzione).

## Progettazione degli oggetti business specifici dell'applicazione

Un oggetto business specifico di un'applicazione contiene dati, azioni da eseguire sui dati (istruzioni), ed informazioni sui dati (informazioni specifiche sull'applicazione). Molti metodi di connettori passano un oggetto business specifico dell'applicazione come un argomento. Ad esempio:

- Quando si verifica un evento dell'applicazione, alcuni connettori richiamano i gestori dei dati per convertire il formato dei dati in un oggetto business, che il connettore invia a InterChange Server Express.
- Quando InterChange Server Express invia una richiesta ad un connettore, la struttura del connettore invia l'oggetto business come argomento al gestore dell'oggetto business del connettore. In questo caso, alcuni connettori richiamano il gestore dei dati per convertire le informazioni nell'oggetto business nel formato usato dall'applicazione, che abilita il connettore ad eseguire operazioni nelle applicazioni.

La progettazione delle relazioni tra connettore, gestore dei dati e gli oggetti business specifici dell'applicazione è una delle attività nello sviluppo del connettore e nella gestione dati. Poiché la progettazione di oggetti business specifici dell'applicazione possono generare requisiti per la logica di programmazione di gestione dati e connettori che devono essere integrati nel processo di sviluppo del connettore, gli sviluppatori del connettore, i gestori dei dati e gli oggetti business specifici dell'applicazione devono lavorare insieme per sviluppare specifiche per quei componenti. La struttura ed il progetto di oggetti business specifici dell'applicazione devono essere determinati dal connettore o dal gestore dati che lo elabora.

**Nota:** Per ottenere prestazioni ottimali con InterChange Server Express, gli oggetti business specifici dell'applicazione devono essere inferiori a 1 MB, quando possibile e mai superiori a 5 MB. Gli oggetti business di grandi dimensioni causano problemi nelle prestazioni a causa di limitazioni in Java Virtual Machine su cui è in esecuzione InterChange Server Express.

Questa sezione descrive i seguenti argomenti:


- "Contenuto delle definizioni dell'oggetto business specifico dell'applicazione" a pagina 33
- "Progettazione di un connettore esistente o di un gestore dati" a pagina 40

## Contenuto delle definizioni dell'oggetto business specifico dell'applicazione

La definizione di oggetto business include la seguente informazione:

*Tabella 8. Contenuto della definizione di oggetto business.*

| Contenuto della definizione di oggetto business | Descrizione | Per ulteriori informazioni |
|---|---|---|
| Struttura dell'oggetto business | La struttura di un oggetto business specifico dell'applicazione è di solito progettata perché corrisponda quanto più è possibile all'entità dell'applicazione (struttura dati) in modo che il connettore o il gestore dati interagisce con l'applicazione (come il livello di tabella, il livello API o a livelli differenti all'interno di un'API). | "Struttura di oggetti business specifici dell'applicazione" a pagina 33 |
| Proprietà dell'attributo | Gli attributi contengono parti individuali di dati all'interno dell'entità di un'applicazione. Hanno inoltre proprietà per fornire informazioni come il tipo di dati, la cardinalità e il valore predefinito. Le proprietà dell'attributo specificano anche se è richiesto l'attributo o la chiave. | "Attributi in un oggetto business specifico dell'applicazione" a pagina 34 |
| Informazioni specifiche sull'applicazione | La definizione di un oggetto business specifico dell'applicazione spesso include stringhe di testo che dicono al connettore quanto è rappresentato l'oggetto business nell'applicazione o come elaborarlo. | "Informazioni specifiche sull'applicazione dell'oggetto busines" a pagina 35 |

### Struttura di oggetti business specifici dell'applicazione

Il modo in cui un connettore o un gestore di dati elabora gli oggetti business è determinato in parte dalla struttura degli oggetti business che supporta. Man mano che si progetta l'oggetto business specifico dell'applicazione, c'è bisogno di determinare quale struttura rappresenta meglio una particolare entità di applicazione e in che modo questa struttura influisce sulla progettazione del connettore e sulla logica di gestione dei dati o in che modo la struttura venga elaborata da un connettore esistente o da un gestore di dati.

Mentre lo scopo della progettazione di un connettore e di un gestore dati è quello di codificare un connettore o un gestore dati in modo che possa gestire oggetti business nuovi e modificati senza apportare modifiche, è difficile creare un connettore o un gestore dati che possa gestire un qualunque oggetto business.

Di solito, un connettore o un gestore di dati è strutturato in modo da creare i presupposti sulla struttura dei suoi oggetti business, sulle relazioni tra oggetti business principali e secondari e sulla rappresentazione possibile dell'applicazione degli oggetti business. Se si sta eseguendo una progettazione per un connettore esistente o per il gestore dati, la funzione è comprendere questi presupposti e progettare di conseguenza gli oggetti business.

Un insieme iniziale di domande da considerare in merito alla struttura di un oggetto business specifico dell'applicazione è:

- Qual è l'organizzazione o lo schema del database per l'entità dell'applicazione che verrà incapsulato nell'oggetto business. L'entità dell'applicazione rappresenta dati gerarchici o relazioni una-a-molte?
- Un oggetto business rappresenta un'entità di applicazione o più di un'entità di applicazione? In altre parole, i valori dell'attributo in un oggetto business individuale possono essere memorizzati in entità di applicazioni differenti?
- Che tipo di relazioni tra oggetti business gestisce il connettore o il gestore dati? In che modo sono modellate le relazioni negli oggetti business o in che modo vengono elaborate dal connettore o dal gestore dati?

Per ulteriori informazioni sulle strutture dell'oggetto business che possono rappresentare entità singole o multiple, consultare "Determinazione della struttura dell'oggetto business" a pagina 17.

**Nota:** Alcuni connettori possono richiedere l'oggetto business di livello superiore per contenere informazioni specifiche. Ad esempio, il connettore XML richiede il suo oggetto business di livello superiore per contenere attributi semplici per un URL, tipo di MIME e per il prefisso dell'oggetto business nonché attributi complessi per contenere un oggetto business di richiesta e un oggetto business di risposta. Se si sta progettando un oggetto business per un connettore esistente, fare riferimento alla guida per l'utente del suo adattatore per i requisiti specifici della struttura. Per ulteriori informazioni, consultare "Progettazione di un connettore esistente o di un gestore dati" a pagina 40.

## Attributi in un oggetto business specifico dell'applicazione

Gli attributi contengono parti individuali di dati in un'entità dell'applicazione. Quando si definiscono gli attributi per un oggetto business specifico dell'applicazione, considerare quanto di seguito riportato:

- Che parti di dati ciascun attributo rappresenterà nell'entità dell'applicazione? Per un oggetto business che rappresenta un'entità del database, ciascun attributo rappresenterà un campo, come la colonna di una tabella? Per un oggetto business che rappresenta un documento XML, ciascun attributo rappresenterà un elemento?
- e' necessario creare un attributo per ogni singolo campo nell'entità dell'applicazione? Alcune parti di dati in un'entità dell'applicazione potrebbero non essere significative nell'interazione di altre applicazioni; lasciandole fuori dalla definizione di oggetto business si può ridurre la complessità della progettazione e si può prevenire il trasferimento di dati non necessari.
- L'oggetto business specifico dell'applicazione ha pochi semplici attributi rispetto all'entità dell'applicazione? Ad esempio, c'è bisogno di un attributo per ogni colonna della tabella del database?
- In che modo opererà il connettore quando l'oggetto business individuale avrà più attributi semplici di quanti non ne abbiano le colonne della tabella del database corrispondente o le tag del DTD corrispondente? In altre parole, alcuni attributi nell'oggetto business non sono rappresentati nel database o nel DTD. Nella maggior parte dei casi, questi attributi trasmettono informazioni su meccanismi di accesso specifici o sono usati per separare attributi ce rappresentano oggetti business secondari. Il connettore o la mappa può impiegare una logica speciale che richiede che gli attributi specifici del connettore gestiscano certi oggetti business specifici dell'applicazione.

Come regola, conservare la struttura dell'oggetto business allo stesso modo della struttura dell'entità dell'applicazione corrispondente (come le tabelle del database o DTD). Se l'oggetto business è grande (contiene molti attributi), definire solo gli attributi usati nel processo business per il quale si sta progettando l'oggetto business. Tuttavia, se l'oggetto business è piccolo, definire tutti gli attributi in modo che siano disponibili per un uso futuro. La definizione del numero di attributi dipende dalla dimensione degli oggetti business e dalla complessità delle relazioni tra di loro.

Inoltre per identificare quali entità di applicazione devono esistere come attributi nell'oggetto business, bisogna esaminare il processo business per determinare se sono richiesti attributi aggiuntivi. Come parte dell'analisi del processo di business, identificare i requisiti dell'oggetto business. Procedendo nel processo business si rivela il modo in cui viene gestito l'oggetto business ed il modo in cui vengono usati gli attributi richiesti. Le variazioni nel processo business e nella gestione delle eccezioni potrebbe identificare gli attributi aggiuntivi richiesti per elaborare l'oggetto business. Questi attributi aggiuntivi potrebbero non corrispondere ai dati richiamati o aggiornati nell'applicazione.

Ad esempio, si potrebbe aver bisogno di attributi che:

- servano come indicatore di priorità il cui valore è derivato dall'elaborazione secondo il valore dell'attributo
- servano come ricerca che contenga informazioni di instradamento su uno o più valori dell'attributo

## Informazioni specifiche sull'applicazione dell'oggetto busines

Dopo aver definito la struttura di una definizione di oggetto business specifico dell'applicazione ed aver definito l'insieme degli attributi che contiene la definizione dell'oggetto business, è possibile determinare se il connettore o il gestore dei dati ha bisogno di informazioni aggiuntive sul modo in cui elaborare l'oggetto business per abilitarlo alla gestione delle richieste che riceve da InterChange Server Express. La definizione di oggetto business può includere queste informazioni aggiuntive nelle informazioni specifiche dell'applicazione.

L'informazione specifica dell'applicazione fornisce al connettore o al gestore di dati istruzioni dipendenti dall'applicazione su come elaborare gli oggetti business. L'approccio raccomandato per progettare relazioni tra oggetti business e connettori è di memorizzare informazioni nella definizione di oggetti business che aiutino un connettore ad interagire con un'applicazione o un sorgente dati. Un tale tipo di informazione, denominato **metadati**, può essere specificata nelle informazioni specifiche dell'applicazione di ciascun oggetto business, attributo dell'oggetto business e istruzione dell'oggetto business.

L'informazione specifica sull'applicazione è una stringa immessa durante la progettazione dell'oggetto business al momento dell'esecuzione da un connettore o da un gestore dati. Il connettore o il gestore dati usa i metadati nella definizione dell'oggetto business per elaborare istanze di oggetti business. Poiché il connettore o il gestore dati ha accesso alle sue definizioni di oggetto business al momento dell'esecuzione, può determinare dinamicamente come elaborare un particolare oggetto business.

Considerare i seguenti vantaggi e limitazioni delle informazioni specifiche dell'applicazione quando si progetta un oggetto business:

- L'informazione specifica dell'applicazione abilita un oggetto business ad essere auto-contenuto con tutte le informazioni richieste per l'elaborazione.

Le informazioni specifiche dell'applicazione nella definizione dell'oggetto business possono includere tabelle e nomi di colonne, istruzioni di elaborazione, nomi di funzioni o altre informazioni su come elaborare i dati nell'applicazione.

Poiché un oggetto business specifico dell'applicazione contiene tutte le informazioni di cui ha bisogno per l'elaborazione, il connettore può gestire oggetti business nuovi o modificati senza richiedere modifiche al codice sorgente del connettore. Il connettore può essere scritto in modo generico, con un gestore di oggetti business che non contiene logica hard-coded per l'elaborazione di oggetti business specifici.

- Un connettore di metadati può costruire chiamate di funzioni dell'applicazione o o istruzioni SQL dai valori in un'istanza oggetto business e le informazioni specifiche sull'applicazione nella definizione dell'oggetto business.

  Le chiamate di funzioni o le istruzioni SQL eseguono le modifiche richieste nel database dell'applicazione per l'oggetto business e per l'istruzione che il connettore sta elaborando.
- L'applicazione che un oggetto business rappresenta determina il quantitativo di informazioni specifiche dell'applicazione che la definizione dell'oggetto business può contenere.

  Secondo l'applicazione e la sua interfaccia di programmazione, un connettore ed il suo oggetto business potrebbe essere progettato in modo che il connettore sia quasi totalmente guidato dalle informazioni specifiche dell'applicazione nel suo oggetto business. In questo caso, il connettore può richiedere solo un gestore dell'oggetto business per trasformare gli oggetti business in richieste per le operazioni dell'applicazione.

  Per alcune applicazioni, tuttavia, l'interfaccia di applicazione può avere restrizioni che forzano interamente la logica differente di elaborazione per i diversi oggetti business e, pertanto, l'implementazione di gestori di oggetti business multipli. Per queste applicazioni, è possibile solo una parziale implementazione dei metadati o l'implementazione di nessun dato.

  Secondo le applicazioni, gli oggetti business variano in rapporto a quante informazioni specifiche dell'applicazione possono contenere. La maggior parte degli oggetti business specifici dell'applicazione può essere progettata in modo da contenere alcune informazioni che assistano il connettore o il gestore dei dati con l'elaborazione di oggetti business.

**Formato suggerito di informazioni specifiche dell'applicazione:** Si raccomanda di usare la sintassi di coppia nome-valore quando si definiscono le informazioni specifiche dell'applicazione. Questa sintassi specifica il nome della proprietà ed il suo valore associato, separato da un segno di uguale (=), come mostra la seguente sintassi:

```
name1=value1;name2=value2
```

Ad esempio, la seguente coppia nome-valore definisce la proprietà di "nome tabella":

```
TN=TableName
```

Le coppie Nome-valore consentono ai valori di essere specificate in modo casuale. Il connettore valuta il nome di ciascun parametro prima di interpretare il valore. Si raccomanda di separare le coppie nome-valore con un delimitatore che:

- abbia come valore predefinito un punto e virgola (**;**)
- sia configurabile

**Nota:** Se si sta creando un oggetto business per un connettore esistente, controllare la guida per l'utente all'adattatore per determinare la sintassi richiesta. Non tutti i connettori hanno come valore predefinito un punto e virgola come delimitatore, o possono essere configurati in tale modo.

Tabella 9 fornisce esempi di parametri che possono essere inclusi in informazioni specifiche dell'applicazione dell'attributo. Tali parametri sono rilevanti solo per un oggetto business che rappresenta dati in una tabella di database.

*Tabella 9. Esempio di parametro nome-valore per informazioni specifiche dell'applicazione dell'attributo.*

| Parametro | Descrizione |
|---|---|
| TN=*TableName* | Il nome della tabella del database. |
| CN=*col_name* | Il nome della colonna del database per questo attributo. |
| FK=[..]*fk_attributeName*] | Il valore della proprietà della chiave esterna definisce una relazione principale/secondaria. |
| UID=AUTO | Questo parametro notifica al connettore di generare l'ID univoco per l'oggetto business e di caricare il valore in questo attributo. |
| CA=*set_attr_name* | La proprietà Copia attribuito istruisce il connettore di copiare il valore di uno degli attributi in un altro. Se set_attr_name è impostato sul nome di un alto attributo all'interno dell'oggetto business corrente individuale, il connettore usa il valore dell'attributo specificato per impostare il valore di questo attributo prima che aggiunga l'oggetto business al database durante l'operazione di creazione. |
| OB=[ASC\|DESC] | Se un valore viene specificato per Ordine per parametro e l'attributo è in un oggetto business secondario, il connettore usa il valore dell'attributo nella clausola di richiamo delle informazioni ORDER BY per determinare se richiamare l'oggetto business secondario in ordine ascendente o discendente. |
| UNVL=*value* | Specifica il valore che usa il connettore per rappresentare un valore nullo quando richiama un oggetto business con attributi valutati nulli. |

Un'informazione specifica dell'applicazione dell'attributo potrebbe combinare diversi esempi di parametri tra quelli precedenteente riportati. Questo esempio usa i delimitatori del punto e virgola (;) per separare i parametri:

```
TN=LineItems;CN=POid;FK=..PO_ID
```

L'informazione specifica dell'applicazione di questo esempio specifica il nome della tabella, il nome della colonna e che l'attributo corrente è una chiave esterna che collega l'oggetto business secondario al suo principale.

**Contenuto dell'informazione specifica dell'applicazione:** Il contenuto dell'informazione specifica dell'applicazione può variare considerevolmente la sua complessità. Alcuni esempi sono:

- Le informazioni specifiche dell'applicazione nella definizione di un oggetto business può codificare il nome della tabella cui corrisponde l'oggetto business e, per ogni attributo, può codificare il nome della colonna cui corrisponde l'attributo. Questa è un'implementazione relativamente semplice di informazione specifica dell'applicazione, ma può può essere tutto quello di cui ha bisogno un connettore.

- Una implementazione più complessa di informazioni specifiche dell'applicazione potrebbe contenere un insieme di parametri che specificano in che modo il connettore gestisce le varie operazioni degli oggetti business.
- In modo più complesso, le informazioni specifiche dell'applicazione potrebbero includere condizioni, elaborazioni dirette di transazioni del connettore, potrebbero specificare metodi di richiamo dei dati e fornire capacità di pre-elaborazione.

Se la definizione dell'oggetto business include informazioni specifiche sull'applicazione ed il connettore è stato progettato per farne uso, il connettore può estrarre il contenuto dell'informazione specifica dell'applicazione dalla definizione dell'oggetto business ed usarlo per l'elaborazione.

**Esempio: In che modo un connettore elabora le informazioni specifiche dell'applicazione:** Come esempio del modo in cui le informazioni specifiche dell'applicazione elaborano un connettore, si supponga che la propria applicazione sia basata su una tabella e che si voglia lavorare con una tabella dell'applicazione denominata CURRENTCUST, che memorizza le informazioni sui clienti. Questa tabella ha due colonne: CSTName e CSTCity.

Nella proprietà AppSpecificInfo dell'intestazione dell'oggetto business, è possibile memorizzare il nome della tabella. Nella proprietà AppSpecificInfo di ciascun attributo, è possibile memorizzare i nomi delle colonne. In aggiunta, poiché il connettore per questa applicazione usa istruzioni SQL per interagire con il database, è possibile progettare l'istruzione dell'informazione specifica dell'applicazione per contenere l'istruzione SQL e le parole chiave. Figura 20 illustra come potrebbe apparire la definizione di oggetto business Cliente .

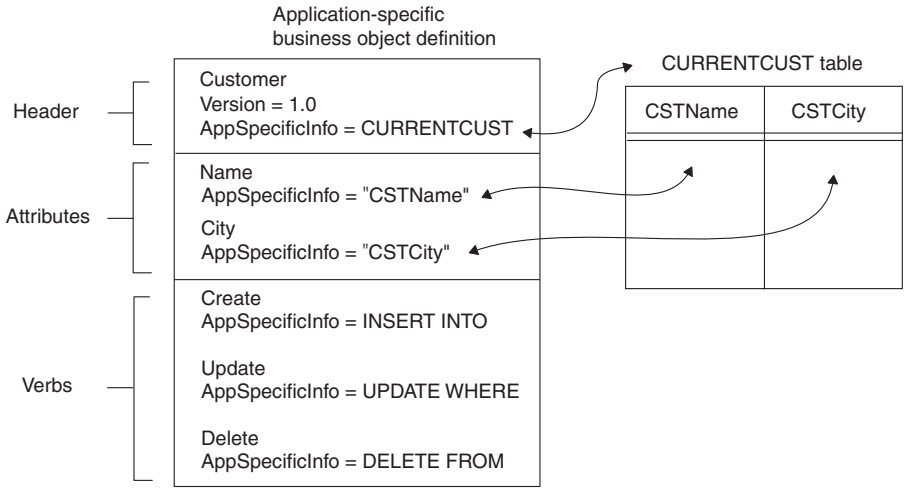



*Figura 20. Informazioni specifiche sull'applicazione in una definizione di oggetto business.*

Quando un connettore di metadati riceve un'istanza di questo oggetto business da InterChange Server Express, esso estrae il nome della tabella e i nomi delle colonne dalle proprietà delle informazioni specifiche dell'applicazione nella definizione dell'oggetto business e poi estrae i valori dell'attributo e dell'istruzione dall'istanza dell'oggetto business. Usando i nomi della tabella e della colonna, i valori dell'attributo e le parole chiave SQL nell'istruzione delle informazioni specifiche dell'applicazione, il connettore può costruire un'istruzione SQL.

Figura 21 mostra un esempio di questo tipo di elaborazione. Il connettore estrae l'istruzione di elaborazione dell'azione e i nomi della tabella e delle colonne per la definizione dell'oggetto business. Richiama quindi i valori degli attributi dall'istanza dell'oggetto business. Usando questa informazione, il connettore costruisce un'istruzione SQL INSERT per aggiornare la tabella CURRENTCUST con le nuove informazioni.

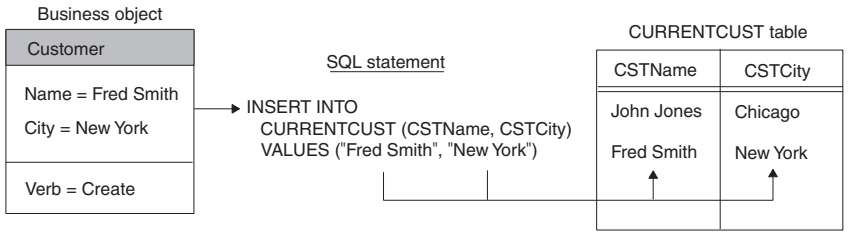



*Figura 21. Uso delle informazioni specifiche sull'applicazione per creare un'istruzione SQL per un'operazione di creazione*

Come precedentemente menzionato, la definizione di un oggetto business può includere testo AppSpecificInfo per oggetto business come un intero, e per i suoi attributi ed istruzioni. La seguente sezione fornisce ulteriori informazioni su come usare informazioni specifiche dell'applicazione in questi componenti di oggetti business.

**Importante**
La lunghezza delle informazioni specifiche dell'applicazione è ristretta a 1000 caratteri.

Figura 5 usa le proprietà di livello dell'attributo AppSpecificInfo per memorizzare il nome del sotto-modulo ed il nome del campo corrispondente all'attributo. L'esempio usa coppie nome-valore per specificare le informazioni:

**Suggerimenti sulla progettazione di informazioni specifiche sull'applicazione:** Quando si progettano oggetti business per portare al massimo il comportamento del connettore guidato dai metadati, seguire queste raccomandazioni di carattere generale per memorizzare informazioni specifiche sull'applicazione nella definizione dell'oggetto business:

- Memorizzare nomi di entità, come tabelle o nomi di moduli, nella proprietà AppSpecificInfo di livello oggetto-business dell'oggetto business di livello superiore. Memorizzare nomi dei sotto-moduli o i nomi delle tabelle nella proprietà AppSpecificInfo di livello oggetto-business dell'oggetto business secondario.
- Memorizzare nomi di campi, nomi di colonne ed altre informazioni correlate agli attributi dell'oggetto business nella proprietà AppSpecificInfo.
- Memorizzare le istruzioni di elaborazione dell'azione nell'azione della proprietà `AppSpecificInfo`.

L'uso attento della proprietà `AppSpecificInfo` abilita un connettore a gestire una varietà di oggetti business alla stesso modo. Se un'applicazione è coerentenel modo di gestire le operazioni dei dati, e se per tutte le operazioni il connettore esegue attività coerenti, l'oggetto business può essere designato ad abilitare un connettore completamente controllato dai metadati.

## Progettazione di un connettore esistente o di un gestore dati

Se si sta progettando un oggetto business specifico per l'applicazione per un connettore esistente o un gestore di dati, il primo passo è quello di consultare la sua guida per l'utente per i suoi requisiti, specificando informazioni specifiche sull'applicazione e usando gestori di oggetti business. Tener presente che quando si progettano oggetti business per un connettore esistente o un gestore dati.

- Per determinare se c'è disponibilità di Object Discovery Agent, controllare la guida per l'utente dell'adattatore per il connettore e la documentazione per il gestore dati che elaborerà l'oggetto business. L'uso di Object Discovery Agent può facilitare molto lo sforzo di progettazione dell'oggetto business, particolarmente quando le entità coinvolte sono ampie.
- Determinare se c'è un oggetto business esistente disponibile che modelli l'entità dell'applicazione, come un esempio. Determinare se lo sforzo per personalizzare l'oggetto business esistente è inferiore rispetto alla creazione di uno completamente nuovo ed in tal caso, considerare l'uso dell'oggetto business campione.

**Importante**

L'IBM non supporta oggetti business di esempio, ma possono essere molto utili come punto di partenza per la progettazione di oggetti business.

- Se non ci sono oggetti business esistenti per le entità che si ha bisogno di modellare e l'Object Discovery Agent non esiste già per l'applicazione, è possibile sviluppare un nuovo Object Discovery Agent per l'applicazione. Questo potrebbe non essere un approccio efficiente se ci sono pochi oggetti business richiesti per l'applicazione o se le entità sono molto piccole. Per ulteriori informazioni, consultare Capitolo 5, "Sviluppo di in Object Discovery Agent", a pagina 95.
- Sia che si usi un Object Discovery Agent che un oggetto business esistente, resta importante esaminare e confermare tutti i requisiti di definizione dei dati, come la chiave oggetto, le chiavi esterne, gli oggetti business secondari, i valori predefiniti, i tipi di dati e limitazioni nelle dimensioni. I seguenti fattore risultano in questi requisiti:
  - Object Discovery Agents può agevolare lo sforzo di progettazione, ma non può scoprire tutti i requisiti che fanno da contorno ad un'entità dell'applicazione.
  - Con oggetti business esistenti, il sintomo è che le applicazioni possono essere installate e configurate in modi diversi per accogliere le necessità specifiche del cliente. Un oggetto business che modella accuratamente un'entità in una installazione dell'applicazione, potrebbe non modellare accuratamente quella entità in un'altra installazione di quella applicazione.

Mentre si progetta l'oggetto business specifico dell'applicazione, tener presente che il ruolo principale è quello di modellare l'entità nei dati sorgente. E' anche importante identificare il modo in cui il suo connettore associato o gestore di dati gestisce il suo processo e quali sono i requisiti del processo business cui partecipa.

## Progettazione di oggetti business generici (solo InterChange Server Express)

Un *oggetto business generico* riflette un insieme di informazioni che rappresenta entità usate da varie applicazioni multiple o entità programmatiche. Gli oggetti di collaborazione InterChange Server Express usano oggetti business generici in modo da fornire informazioni su una varietà di applicazioni diverse. Pertanto, la progettazione di oggetti business generici è parte dell'attività di collaborazione allo sviluppo dell'oggetto. Quando si progetta un oggetto business generico, tenere in considerazione quanto segue:

- Comprensione dei requisiti dei dati critici all'integrazione del processo di business.
- Studio della logica business cui partecipa l'oggetto business e di tutti i requisiti basati su quella logica.

  Le seguenti due considerazioni illustrano la complessità delle analisi logiche di business:

  – Elaborazione dei dati prerequisiti

    A volte l'applicazione fa scattare l'oggetto della collaborazione non fornisce tutti i dati richiesti per elaborare il triggering dell'oggetto business. I dati aggiuntivi potrebbero risiedere in altre applicazioni, incluse le applicazioni di destinazione.

    Ad esempio, un'applicazione SFA (Sales Force Automation) - come la Siebel - potrebbe generare una citazione che necessita di essere registrata come Ordine in un sistema di gestione ordine (come il SAP). Tuttavia, prima che Citazione diventi un Ordine, potrebbe richiedere informazioni aggiuntive non disponibili nelle applicazioni SFA. Ad esempio, un Ordine potrebbe richiedere certi dati aggiuntivi quali lo stato del credito del cliente (da un sistema finanziario), informazioni sul contatto (da un sistema di supporto del cliente) o informazioni sulla disponibilità (da un sistema warehouse).

    Quando si progetta l'oggetto business generico Ordine, si potrebbe aver bisogno di includere gli attributi e di progettare strutture a supporto di dati non presenti necessariamente nell'applicazione sorgente, ma che possono essere presenti in altre applicazioni coinvolte nell'interfaccia.

  – Riferimenti incrociati tra entità di applicazioni individuali

    Determinare in che modo le entità di applicazioni individuali corrispondono tra di loro o sono incrociate genericamente tra loro nel processo business.

    Ad esempio, un cliente in Oracle è rappresentato come Cliente il cui indirizzo è rappresentato come Indirizzon. I SAP, un cliente è rappresentato come entità "SoldTo" il cui indirizzo è rappresentato come entità "ship-to". In Clarify un cliente è rappresentato come "Business Organization" il cui indirizzo è rappresentato come "Site".

    Studio della funzionalità e delle relazioni tra entità di applicazioni per determinare gli oggetti business e i processi coinvolti nell'integrazione dei dati tra applicazioni.

- Comprensione di quali sono i dati richiesti in comune o condivisi tra tutte le applicazioni che partecipano al processo business e di cosa è critico per l'integrazione di quel processo. Gli attributi e le loro relazioni devono determinare un minimo di attributi obbligatori (il minimo denominatore comune di attributi) e le trasformazioni che il sistema di integrazione business deve eseguire tra questi oggetti business specifici dell'applicazione business.

  Considerare le seguenti possibilità di integrazione:

  - Il processo business integra dati dal sistema ERP (Enterprise Resource Management) in un sistema CRM (Customer Relationship Management). In questo caso, l'oggetto business probabilmente non ha bisogno di essere molto complesso perché la maggior parte dei sistemi CRM non accoglie la maggior parte dei dati memorizzati in un sistema ERP.
  - Il processo business integra dati tra due sistemi ERP. In questo caso, è probabile che l'oggetto business diventi altamente complesso.
  - Il processo business integra dati da un sistema CRM ad un sistema ERP. In questo caso, il proprio progetto deve riflettere il quantitativo di dati originati al momento nel sistema CRM (e che quindi devono essere rappresentati da attributi in oggetti business generici) e quanti di essi possono essere destinati come valore predefinito alle applicazioni stesse (e forniti in tal modo come valori predefiniti nell'oggetto business specifico dell'applicazione).
- Se esistono, studiare gli oggetti business specifici dell'applicazione cui corrisponderà l'oggetto business generico. Analizzare la struttura e gli attributi di tutti gli oggetti business per far derivare un generico oggetto business che si adatti a tutte le applicazioni.
- Considerare se c'è uno standard per il tipo di oggetti business che si stanno progettando. Ad esempio, potrebbe esserci un modello appropriato per le entità fornite da iniziative EDI (Electronic Data Interchange), OAG (Open Applications Group) o OMG (Object Management Group).

## Standard di progettazione di oggetti business generici

Per essere coerenti con gli oggetti business generici distribuiti da IBM, usare i seguenti standard quando si progetta un oggetto business generico:

- Il primo attributo di ogni oggetto deve essere la sua chiave e deve chiamarsi `ObjectId`.
- Se un attributo rappresenta una chiave esterna, il suo nome deve concatenare il nome dell'oggetto business esterno e l'`Id`; ad esempio: `CustomerId`, `ItemId` e `OrderId`.
- Siate coerenti. Se si usa un'abbreviazione nel nome di un attributo, usare la stessa abbreviazione negli oggetti business principali e secondari. Se possibile, usare la stessa abbreviazione per tutti i nomi rilevanti dell'attributo. Ad esempio, se si abbrevia `Numero` in `Num`, fatelo costantemente.

## Progettazione per l'isolamento eventi

Quando si progetta un oggetto business generico, si raccomanda di tener presente le necessità di isolamento dell'evento, come spiegato i *Collaboration Development Guide* (nella sezione "Designing for Parallel Execution").

Per prevenire che più di un oggetto collaborazione venga aggiornato contemporaneamente, ciascun oggetto business dovrebbe essere modificato solo da un tipo di oggetto collaborazione. In altre parole, l'oggetto business Cliente deve essere modificato solo da un oggetto di collaborazione CustomerSync.

Se un oggetto collaborazione modifica un oggetto business che contiene un oggetto business secondario, e l'oggetto business secondario è contenuto anche da un oggetto business di livello superiore differente che ha il suo proprio oggetto di collaborazione modifica, progettare gli oggetti business di livello superiore in modo che contengano il secondario semanticamente piuttosto che strutturalmente. Sviluppare un terzo oggetto collaborazione per modificare il secondario condiviso.

Gli oggetti collaborazione che posseggono gli oggetti business di livello superiore devono delegare l'elaborazione del secondario condiviso al terzo oggetto collaborazione.

Ad esempio, se entrambi gli oggetti business Cliente e Contatto contengono lo stesso dato dell'indirizzo, progettare l'oggetto business Indirizzo come oggetto di livello superiore cui si fa riferimento per Cliente e Contatto ma che non sono contenuti in esso. Quindi sviluppare un separato oggetto collaborazione Indirizzo per modificare i dati dell'indirizzo.

In un altro esempio, tuttavia, se l'oggetto business Ordine è l'unico oggetto che modifica i dati OrderLineItem, è possibile progettare Ordine in modo che contenga gli oggetti business secondari OrderLineItem piuttosto che fare mero riferimento ad essi.

In altre parole, progettare gli oggetti business Cliente e Contatto in modo che contenga un attributo di chiave esterna che faccia riferimento all'oggetto business Indirizzo, cioè, ce contenga solo il valore chiave per Indirizzo. Non progettarli in modo che contengano un attributo che rappresenti un oggetto business Indirizzo valutato completamente. Ma progettare l'oggetto business Ordine in modo che contenga un attributo che rappresenti un oggetto business OrderLineItem valutato completamente.

**Nota:** Progettare oggetti business condivisi come riferimento piuttosto che contenuti, può semplificare la distribuzione di oggetti business. Se lo stesso oggetto business secondario viene definito oggetto business multiplo, il programma di utilità `repos_copy` tenta di caricare due volte lo stesso oggetto business durante l'installazione, causando il rollback. Per informazioni sui contrassegni di `repos_copy` che cambiano questo comportamento predefinito, consultare *System Administration Guide*.

## Attributi in un oggetto business generico

Quano si definiscono gli attributi per un oggetto business generico, studiare gli attributi degli oggetti business specifici dell'applicazione cui corrisponderanno oggetti business generici. Considerare le seguenti linee guida:

- Notare le similitudini tra entità negli attributi degli oggetti business specifici dell'applicazione. Definire gli attributi per oggetti business generici che più semplicemente corrispondono a quelli negli oggetti business specifici dell'applicazione.
- Notare le differenze tra entità negli attributi degli oggetti business specifici dell'applicazione. Se un oggetto business specifico dell'applicazione divide i dati in campi multipli mente un altro unisce gli stessi dati in un solo campo, determinare quale è il progetto che semplifica maggiormente la mappatura tra due entità di applicazioni. Per ulteriori informazioni, consultare "Progettazione di un connettore esistente o di un gestore dati" a pagina 40.
- Considerare i requisiti generati dall'elaborazione eseguita dall'oggetto collaborazione. Ad esempio, se l''oggetto collaborazione elabora prerequisiti così come sono descritti in "Progettazione di oggetti business generici (solo InterChange Server Express)" a pagina 41, assicurarsi che l'oggetto business generico contenga tutti gli attributi richiesti per memorizzare i dati prerequisiti.
- Sviluppare l'oggetto business generico e l'interfaccia per accontentare il più ampio numero di applicazioni coinvolte nell'interfaccia. Ad esempio, se ci sono quattro applicazioni coinvolte in un'interfaccia e tre di esse incapsulano dati in un oggetto secondario ma il quarto contiene quei dati dell'oggetto a livello

principale, progettare l'oggetto business generico in modo che incapsuli anche i dati in un oggetto secondario; risultando così nella mappatura ed in altre attività correlate.

- Tenere in considerazione lavori futuri di progettazione; si potrebbe voler progettare un oggetto business generico per venire incontro alle strutture dei dati che verranno richieste ad un certo punto per ridurre al minimo gli sforzi e cambiare l'impatto. Tuttavia, non diminuire in modo significativo l'ambito dello sviluppo per un progetto futuro che potrebbe non esistere mai.

In genere, la definizione di oggetto business generico dovrebbe includere attributi che catturino tutti gli elementi dei dati che devono essere trasformati tra tutti gli oggetti business specifici dell'applicazione cui corrisponde l'oggetto business generico.

I nomi degli attributi devono essere quanto più intuitivi possibile. Ad esempio, se svariate applicazioni fanno riferimento ad un'entità come `Cliente` ed un'applicazione si riferisce alla stessa entità come `Organizzazione commerciale`, usare la terminologia più comune per assegnare un nome agli attributi generici.

**Nota:** Il nome di un attributo può contenere solo caratteri alfanumerici e di sottolineatura (_).

## Valutazione degli oggetti business generici esistenti

Si potrebbe essere in grado di facilitare lo sviluppo di un oggetto business generico copiando e personalizzandone uno esistente.

Per valutare un oggetto business generico, esaminare i dati coinvolti nell'interfaccia. Una indicazione è che se l'80% o più dei dati esiste in un oggetto business generico distribuito, personalizzare l'oggetto esistente.

Quando si esegue questa analisi, è più importante guardare la struttura dell'oggetto business piuttosto che gli attributi. Gli attributi sono semplici da aggiungere ed eliminare, mentre le modifiche strutturali o gerarchiche possono richiedere molto sforzo.

Se si decide di personalizzare un oggetto business generico esistente, esaminare la definizione di oggetto business per determinare se uno o più attributi sono mancanti. Gli attributi mancanti diventano più evidenti durante la progettazione della mappatura. Se un oggetto business generico richiede uno o più attributi aggiuntivi, creare un oggetto business secondario che contenga attributi aggiuntivi. Isolando gli attributi personalizzati in oggetti business secondari si facilita l'aggiornamento futuro degli oggetti business distribuiti da IBM.

Se si inseriscono attributi personalizzati in un oggetto business distribuito da IBM, l'aggiornamento di una nuova versione d oggetto business richiede il reinserimento di quegli attributi nel nuovo oggetto business. L'isolamento degli attributi personalizzati nei loro oggetti business consente di aggiungere un attributo al nuovo oggetto business IBM; l'attributo che crea la relazione tra l'oggetto business secondario personalizzato e il principale. Se si sta personalizzando un oggetto business che richiede attributi aggiuntivi in entrambi il principale e secondario, creare oggetti business separati per ciascuno di esso.

Si raccomanda di assegnare un nome personalizzato ad attributi e oggetti business in un modo che siano di pronta identificazione. Una semplice convenzione è quella di aggiungere il suffisso `_x` a ciascun nome personalizzato. Ad esempio, se si crea

un oggetto business secondario personalizzato che aggiunge attributi all'oggetto business generico Ordine, assegnare come nome al secondario `Ordine_x`. Facendo ciò si consente all'elenco alfabetico di mantenere insieme i nomi correlati. Se è più importante identificare l'oggetto business personalizzato o gli attributi che mantenere l'ordine alfabetico dell'oggetto personalizzato con il suo oggetto generico, aggiungere il prefisso `x_` a ciascun nome personalizzato.

## Determinazione dei requisiti di mappatura per gli oggetti business (Solo InterChange Server Express)

Quando un oggetto business specifico dell'applicazione è stato progettato per corrispondere all'entità di un'applicazione, non può coincidere con il suo corrispondente oggetto business generico. Pertanto, bisogna creare mappe tra l'oggetto business specifico dell'applicazione e l'oggetto business generico in modo che i dati dell'applicazione possano essere trasportato attraverso il sistema WebSphere Business Integration Server Express.

Un oggetto business specifico dell'applicazione potrebbe non aver bisogno di includere tutti i campi o elementi di colonne nell'entità di un'applicazione. Usare i requisiti funzionali dell'applicazione e dei processi business in cui partecipa per identificare quali attributi appartengono all'oggetto business specifico dell'applicazione.

Si può anche esaminare la corrispondenza tra oggetti business generici e l'entità dell'applicazione. E' possibile scegliere di includere campi nell'oggetto business specifico dell'applicazione che corrispondano a quelli nel generico, che consente agli elementi di questi dati di partecipare al processo business.

Quando si progetta l'oggetto business, notare la differenza tra l'entità dell'applicazione e l'oggetto business generico. Queste differenze definiscono che tipo di trasformazione dati ha bisogno di verificarsi. Si potrebbe aver bisogno di progettare la mappatura per:

- Associare campi multipli nell'entità dell'applicazione per riempire un attributo nell'oggetto business generico.
- Dividere un campo nell'entità dell'applicazione per riempire un attributo multiplo in un oggetto business generico.
- Ignorare un campo che è presente nell'oggetto business generico ma che non è rilevate all'entità dell'applicazione.
- Gestire le differenze nelle relazioni semantiche o strutturali tra un oggetto business specifico dell'applicazione ed un oggetto business generico.
- Gestire relazioni di chiavi esterne ed altri tipi di relazioni tra entità di applicazione.
- Stabilire associazioni tra dati, ad esempio:
  - Stabilire una ricerca tra dati in attributi non-chiave, come un'associazione che trasforma valori di codici (ad esempio, stato civile o conto corrente) tra applicazioni.
  - Stabilire associazioni di identità tra dati in oggetti business, come un'associazione che trasforma gli attributi chiave (ad esempio, identificatori univoci e codici di prodotto) tra applicazioni.

L'aiuto con i concetti di mappatura e progettazione, le relazioni tra campi in una tabella, gli attributi in un oggetto business specifico dell'applicazione e gli attributi in un oggetto business generico sono mostrati in modo molto semplificato in

Figura 22. Le differenze tra un oggetto business specifico dell'applicazione l'oggetto business generico sono gestite nella mappatura. Se l'oggetto business ha attributi che non sono rappresentati nel database, il connettore può fornire un valore predefinito per l'attributo.

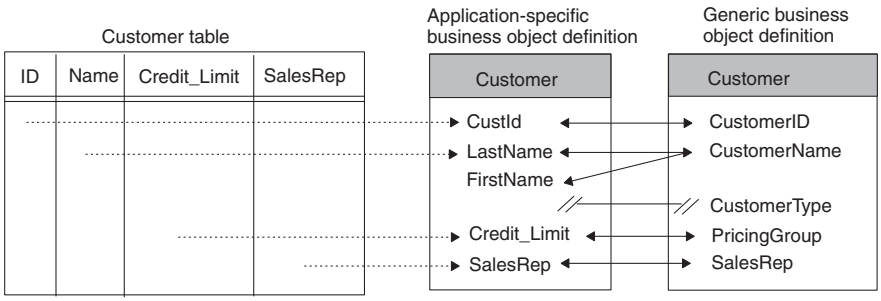



*Figura 22. Vista di livello superiore delle relazioni campo/attributo.*

Per informazioni sulla creazione di mappe, consultare *Map Development Guide*.

# Capitolo 3. Utilizzo di Business Object Designer Express

Lo strumento Business Object Designer Express viene utilizzato per creare, modificare e eliminare le definizione di oggetti business. In questo capitolo viene fornita una panoramica su come utilizzare Business Object Designer Express. Gli argomenti principali di questo capitolo sono:



## Operazioni con i progetti

Business Object Designer Express utilizza il concetto di *progetto* per definire un'area di lavoro virtuale in cui vengono create, modificate o eliminate le definizioni di oggetti business. A seconda dell'ambiente, il "progetto" cui fa riferimento Business Object Designer Express nelle caselle di dialogo può essere uno dei seguenti:

*Tabella 10. Progetti in Business Object Designer Express.*

| Ambiente Business Object Designer Express | Progetto |
|---|---|
| Non si sta eseguendo Business Object Designer Express da System Manager. | Un'area di lavoro virtuale in cui sono state importate definizioni di oggetti business da una directory locale su cui operare durante la sessione corrente di Business Object Designer Express. Inoltre è detto *progetto locale*. |
| Se si sta eseguendo Business Object Designer Express da System Manager. | Un ICL (Integration Component Library) sulla macchina Windows dove sono in esecuzione Business Object Designer Express e System Manager. Anche chiamato *progetto basato su ICL*. |

L'utilizzo di ciascun tipo di progetto viene illustrato in modo più dettagliato in seguito.

### Se Business Object Designer Express è in esecuzione senza System Manager

Se non si sta eseguendo Business Object Designer Express da System Manager, Business Object Designer Express utilizza un progetto locale come "il progetto". Un progetto locale è un'area di lavoro virtuale in cui possono essere importate definizioni di oggetti business con cui si desidera lavorare.

#### Come Business Object Designer Express opera con un progetto locale

Di seguito viene riportato un riepilogo di come le funzioni di Business Object Designer Express operano su un progetto locale. Ulteriori dettagli sull'esecuzione di queste attività viene fornito negli argomenti che iniziano con "Avvio di Business Object Designer Express" a pagina 50.

- **Modifica delle definizioni di oggetti business esistenti:** Per modificare una definizione di oggetti business esistenti, fare clic su **File > Apri da file**. Questo menu importa la definizione di oggetti business da una directory locale nel proprio progetto e lo apre perché possa essere modificato.

  Per modificare una definizione di oggetto business esistente che è già stato importata nel progetto, ma che ora è chiusa, fare clic su **File > Apri**.
- **Creazione di una nuova definizione di oggetto business:** Per creare una definizione di oggetti business, fare clic su **File > Nuovo o file > Nuovo utilizzando ODA**.
- **Salvataggio di una definizione di un oggetto business:** Per salvare una definizione di oggetti business nuova o modificata, fare clic su **Salva** sulla barra di menu. Viene richiesto di salvarla in una directory locale. Per salvare una definizione di oggetti business sotto un nome o una directory diversa, fare clic su
- **Eliminazione di definizioni di oggetti business:** Per eliminare una definizione di oggetti business dalla directory Windows in cui si trova, utilizzare lo strumento fornito da Windows. Non è possibile utilizzare la funzione **Elimina** in Business Object Designer Express a tale scopo. Per eliminare una definizione di oggetti business da un progetto locale, selezionare **File > Elimina**. Viene richiesto di selezionare una definizione di oggetti business per eliminarla dal progetto.

## Se Business Object Designer Express è in esecuzione da System Manager

Quando Business Object Designer Express viene eseguito da System Manager, si ha accesso a funzionalità aggiuntive, più sofisticate, per lo sviluppo e la gestione di definizioni di oggetti business. In System Manager, le definizioni di oggetti business, insieme ad altri componenti di integrazione business, quali le collaborazioni e le mappe, vengono memorizzate nelle ICL (*Integration Component Libraries*). Le ICL sono i depositi dei componenti di integrazione business, che possono utilizzare i blocchi per costruire soluzioni del integrazione di business. Ciascuna ICL contiene un insieme di cartelle, una per ciascun tipo di componenti di integrazione, come illustrato in Figura 23.

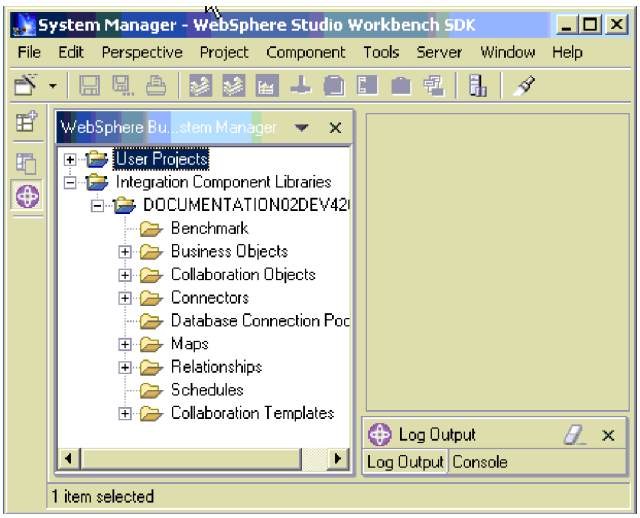



*Figura 23. ICL (Integration Component Libraries) in System Manager.*

Di seguito viene riportata la metodologia per lo sviluppo e la distribuzione di definizioni di oggetti business. Si sviluppa una definizione di oggetti business in Business Object Designer Express e la si salva nella cartella di una ICL. Quando si desidera utilizzare questa definizione di oggetto business in una soluzione integrata business, si associa la definizione con una o più *progetti utente*. Ciascun progetto utente comprende tutti i componenti di integrazione business necessari all'implementazione di una particolare soluzione di integrazione business. Ad esempio, in un progetto utente che contiene i componenti necessari all'implementazione di un adattatore PeopleSoft, la cartella di oggetti business contiene tutte le definizioni di oggetti business necessarie all'adattatore.

come una ICL, ciascun progetto utente contiene un insieme di cartelle di componenti di integrazione business. Tuttavia, un progetto utente contiene solo copie virtuali di componenti ICL. Quando si modifica una definizione di oggetti business, modificare l'istanza nella ICL. Le modifiche effettuate vengono automaticamente estese ad ogni progetto utente che comprende la definizione di oggetti business (consultare Figura 24). Quindi se una particolare definizione di oggetti business viene inclusa in due progetti utente e viene effettuata una modifica a quella definizione nella ICL (Integration Component Library), la modifica si riflette automaticamente nelle copie virtuali che risiedono nei progetti utente. Questo collegamento tra definizioni di oggetti business e una ICL e le relative copie virtuali nei progetti utente consentono di modificare e conservare le definizioni di oggetti business in una posizione centrale, mentre le definizioni vengono sviluppate in più soluzioni di integrazione business.

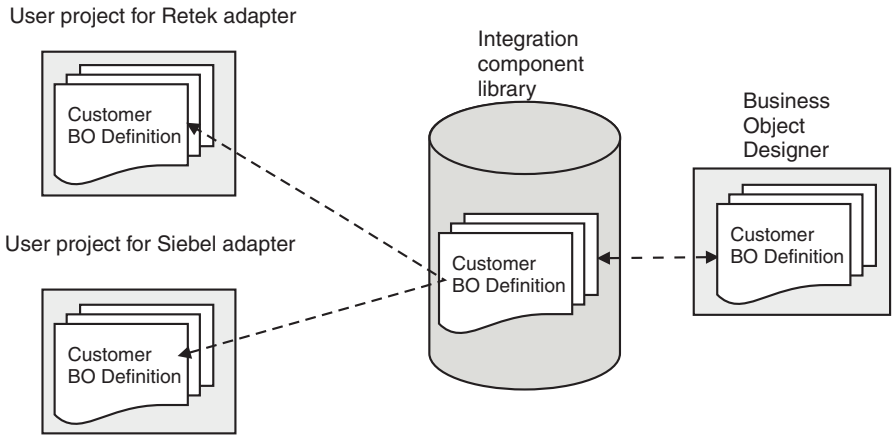



*Figura 24. Modifiche alle definizioni di oggetti business in una ICL che si propagano automaticamente alle copie virtuali nei progetti utente.*

Per ulteriori informazioni sullo sviluppo di componenti di integrazione utilizzando le ICL, consultare la guida relativa al sistema che si sta utilizzando.

## Come Business Object Designer Express opera con un progetto basato su ICL

Quando si sta si sta eseguendo Business Object Designer Express da System Manager, utilizza come "progetto" la ICL selezionata. Di seguito viene riportato un riepilogo di come le funzioni di Business Object Designer Express operano su un progetto basato su una ICL. Ulteriori dettagli sull'esecuzione di queste attività viene fornito negli argomenti che iniziano con "Avvio di Business Object Designer Express" a pagina 50.

- **Modifica delle definizioni di oggetti business esistenti:** Per modificare una definizione di oggetti business memorizzata in un progetto, fare clic su **File > Apri da file**.
- **Creazione di nuove definizioni di oggetti business:** Per creare una definizione di oggetto business, fare clic su **File > Nuovo** oppure **File > Nuovo utilizzando ODA**.
- **Salvataggio di definizioni di oggetti business:** Per salvare una definizione di oggetti business nuova o modificata, fare clic su **File >Salva**. L'oggetto business viene salvato nella cartella di oggetti the business del progetto. Per salvare la definizione di oggetti business nuova o modificata utilizzando un altro nome, fare clic su **File > Salva con nome**.
- **Eliminazione di definizione di oggetti business:** Per eliminare una definizione di oggetti business, selezionare **Eliminare** dalla barra dei menu. Viene richiesto di selezionare una definizione di oggetti business per eliminarlo dal progetto.

Quando si esegue Business Object Designer Express senza System Manager, non si ha accesso alle ICL. In questo ambiente, Business Object Designer Express utilizza un progetto locale come descritto in "Se Business Object Designer Express è in esecuzione senza System Manager" a pagina 47.

## Avvio di Business Object Designer Express

È possibile aprire Business Object Designer Express in uno dei modi elencati in Tabella 11. Dopo aver aperto Business Object Designer Express, è possibile creare manualmente una definizione di oggetto business oppure utilizzare un ODA per generare una definizione di un oggetto business specifico di un'applicazione. Per ulteriori informazioni, consultare Capitolo 4, "Sviluppo delle definizioni dell'oggetto business", a pagina 61.

*Tabella 11. Come aprire Business Object Designer Express.*

| | |
|---|---|
| **Da System Manager** | • Selezionare la cartella di oggetti business in una ICL, e quindi eseguire una delle seguenti operazioni:<br>– Fare clic su **Business Object Designer Express** dal menu **Strumento**.<br>– Fare clic sull'icona della barra degli strumenti di **Business Object Designer Express**.<br>• Fare clic con il pulsante destro del mouse sulla cartella di oggetti business in una ICL.<br>• Fare doppio clic su una definizione di oggetti business. |
| **Utilizzare un collegamento (InterChange Server Express) di Windows** | Fare clic su **Programmi > IBM WebSphere Business Integration Server Express > Toolset Express > Development > Business Object Designer Express**. |
| **Da un altro strumento di sviluppo (solo InterChange Server Express)** | Effettuare una delle seguenti operazioni:<br>• Sul menu **Strumento**, fare clic su **Business Object Designer Express**.<br>• Dalla barra degli strumenti, fare doppio clic sull'icona Business Object Designer Express. |

Quando si apre Business Object Designer Express direttamente da System Manager, senza prima selezionare una definizione di oggetti business, viene visualizzata automaticamente la casella di dialogo Nuovo oggetto business. Se System Manager non è in esecuzione, Business Object Designer Express si apre ma non viene visualizzata la casella di dialogo Nuovo oggetto business.

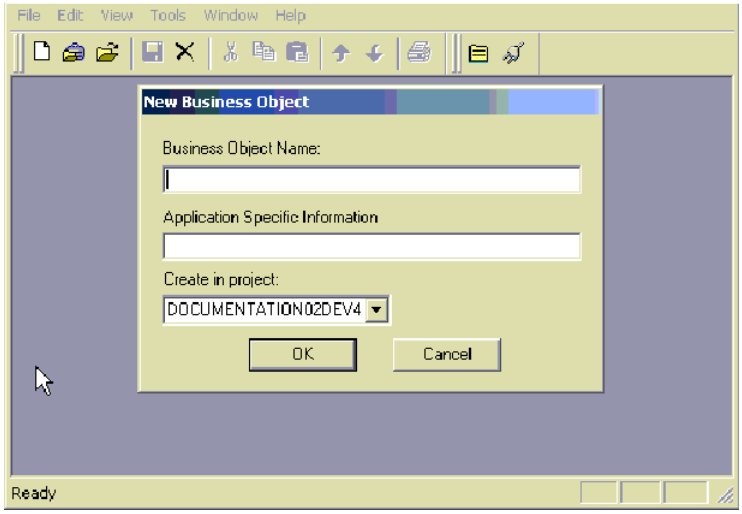



*Figura 25. Casella di dialogo Nuovo oggetto business*

Quando si apre Business Object Designer Express facendo doppio clic su una definizione di oggetti business, viene visualizzata la definizione selezionata nell'area di lavoro Business Object Designer Express.

## Apertura della definizione di oggetti business da Business Object Designer Express

Una volta aperto Business Object Designer Express, è possibile aprire le definizioni di oggetti memorizzate in un file. Se Business Object Designer Express è in esecuzione da System Manager, è anche possibile aprire le definizioni di oggetti business memorizzati in una ICL.

Questa sezione descrive:

- "Apertura di una definizione di oggetti business da un progetto"
- "Prevenzione dei nomi di definizione duplicati" a pagina 52
- "Apertura di una definizione da un file" a pagina 52

### Apertura di una definizione di oggetti business da un progetto

Se Business Object Designer Express è già aperto, è possibile effettuare le seguenti operazioni per aprire una definizione di oggetti business da un progetto.

**Nota:** Se Business Object Designer Express è in esecuzione da System Manager, si tratta di un progetto basato su ICL. Altrimenti, si tratta di un progetto locale, che contiene solo definizioni di oggetti business importate. Consultare "Operazioni con i progetti" a pagina 47 per ulteriori informazioni sui progetti in Business Object Designer Express.

1. Dall'elenco di definizioni di oggetti business nel progetto, evidenziare il nome della definizione che si desidera aprire.
2. Per selezionare più definizioni di oggetti business nel progetto, effettuare una delle seguenti operazioni:
   - Quando si selezionano nomi consecutivi, selezionare il primo nome e, mentre si preme il tasto `Maius` fare clic sull'ultimo nome.

- Quando si selezionano nomi non consecutivi, premere il tasto `Ctrl` e fare clic su ciascun nome.

3. Dopo aver selezionato le definizioni da aprire, fare clic con il pulsante destro del mouse e quindi fare clic su **Apri**.

   Business Object Designer Express visualizza una finestra per ciascuna definizione selezionata.

## Apertura di una definizione da un file

Per aprire una definizione di oggetti business che viene memorizzato in una directory locale, effettuare le seguenti operazioni:

1. Fare clic su **File > Apri da file**.

   La casella di dialogo Importa. La casella di dialogo imposta come valore predefinito il filtro di file di tipo XML Schema Definition (con un'estensione `.xsd`). È anche possibile selezionare un tipo di file differente dall'elenco **File di tipo** oppure è possibile selezionare tutti i tipi di file.

2. Nella casella di dialogo Importa, cercare tra le cartelle fino a individuare il file, selezionarlo e fare clic su **Apri**. Figura 26 illustra questa casella di dialogo.

   **Nota:** Se Business Object Designer Express non è in esecuzione da parte di System Manager, l'elenco **Nel progetto**, che consente di specificare il progetto basato su ICL per ricevere la definizione di oggetti business importata, viene omesso dalla casella di dialogo. Pertanto, la definizione di oggetti business viene importata nel progetto locale.

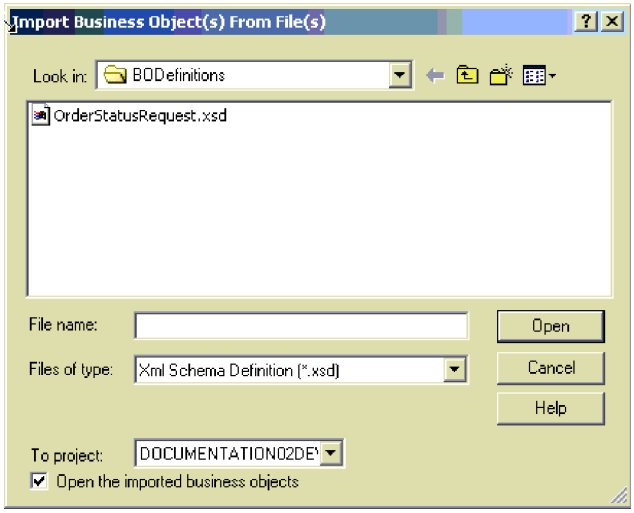



*Figura 26. Casella di dialogo Importa oggetti business*

Se viene selezionata la casella di dialogo **Apri gli oggetti business importati**, Business Object Designer Express apre anche la definizione di oggetti business perché venga modificata. Altrimenti, la definizione di oggetti business viene importata nel progetto, ma non aperta per la modifica. Per ulteriori informazioni, consultare "Operazioni con le definizioni di oggetti business" a pagina 54.

## Prevenzione dei nomi di definizione duplicati

Business Object Designer Express non consente di avere due oggetti business con lo stesso nome nello stesso progetto, mentre ciò può accadere in una delle seguenti situazioni:

- Si cerca di aprire una definizione da un file che è identico a quello che già è presente nel progetto.
- Si cerca di creare una nuova definizione che è identica ad una che già esiste nel progetto.

  In questo caso, Business Object Designer Express visualizza un messaggio di errore del tipo `Un oggetto business con questo nome esiste già.`

Se si cerca di aprire una definizione da un file e il progetto locale o basato su ICL contiene già una definizione con lo stesso nome, Business Object Designer Express visualizza la casella di dialogo Importa risultati, illustrata in Figura 27.

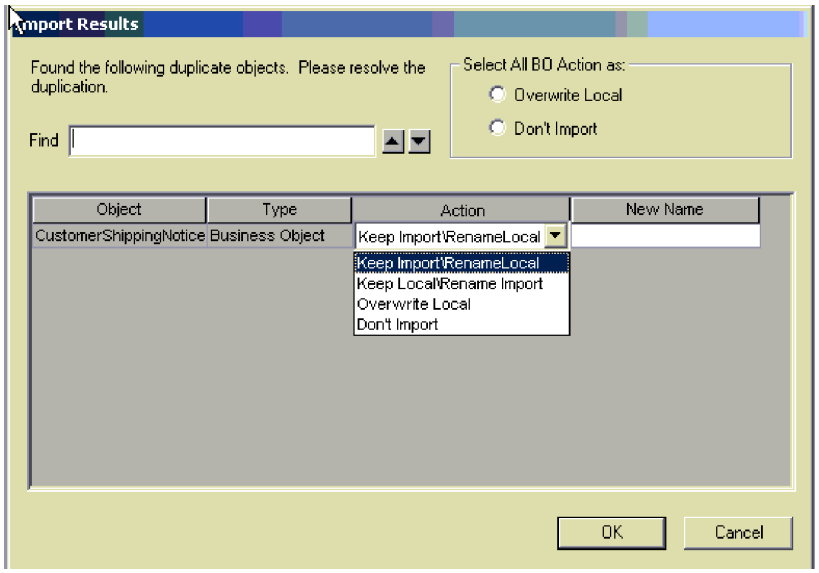



*Figura 27. Prevenzione dei nomi duplicati: Mantieni locale o Importa*

Nella casella di dialogo Importa risultati, eseguire una delle seguenti operazioni per ciascuna definizione di oggetti business che ha un nome duplicato:

1. Fare clic su **Azione** e selezionare un'azione dall'elenco. Una spiegazione di ciascun'azione viene fornita qui di seguito.
2. Se si seleziona **Mantieni importazione/Rinomina locale** oppure **Mantieni locale/Rinomina importato**, immettere un nuovo nome per la definizione di oggetti business nella colonna **Nome** come illustrato in Figura 27.

In alternativa, è possibile utilizzare **Seleziona tutte le azioni BO con nome** per specificare una delle due azioni per ogni definizione di oggetti business che ha un nome duplicato. Per sostituire tutte le definizioni di oggetti business nel progetto con le definizioni che si stanno importando, selezionare **Sostituisci locale**. Per impedire l'importazione di definizioni di oggetti business che hanno nomi duplicati, selezionare **Non importare**.

L'elenco **Azioni** nella finestra Risultati importazione fornisce le seguenti opzioni:

- **Mantieni importato\Rinomina locale**—Consente di modificare il nome della definizione nel progetto e lasciare non modificato il nome della definizione nel file.

  Per effettuare questa modifica, immettere il nuovo nome nella colonna **Nuovo nome** come illustrato in Figura 27.

- **Mantieni locale\Rinomina importato**—Consente di modificare il nome della definizione nel progetto e lasciare non modificato il nome della definizione nel progetto.

  Per effettuare questa modifica, immettere il nuovo nome nella colonna **Nuovo nome** come illustrato in Figura 27.
- **Sostituisci locale**—Sostituisce la definizione attualmente memorizzata nel progetto con quella memorizzata nel file.
- **Non importare**—Annulla l'azione di importazione della definizione memorizzata nel file.

## Operazioni con le definizioni di oggetti business

Business Object Designer Express fornisce una casella di dialogo con schede con due pannelli per la creazione e la modifica di una definizione:

- Scheda **Generale** tab — specifica o modifica le informazioni specifiche dell'applicazione e i verbi a livello di oggetti business.
- La scheda **Attributi** — specifica o modifica le proprietà degli attributi.

Quando si crea prima o si apre una definizione, viene aperta la scheda **Attributi**.

Figura 28 illustra l'ambiente per la definizione e la modifica di attributi.

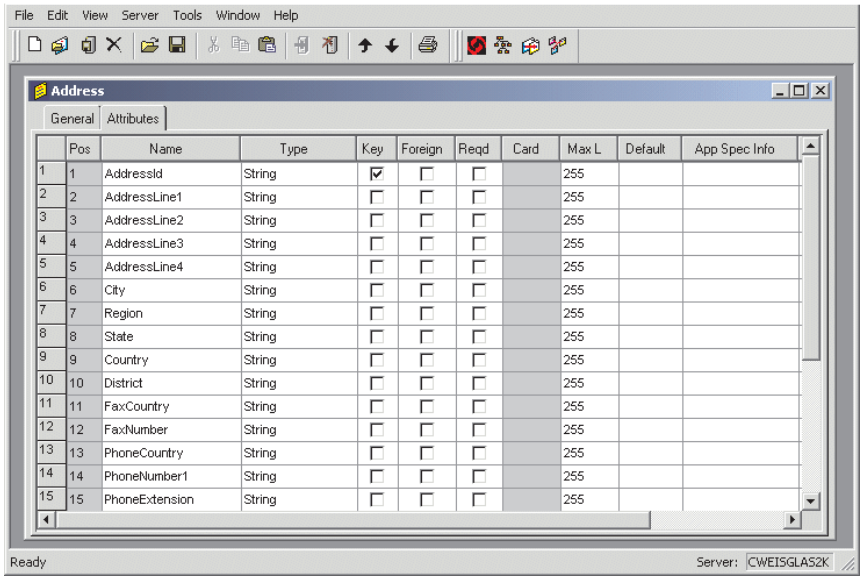



*Figura 28. Definizione e modifica di attributi*

Per informazioni sull'utilizzo delle schede **Generale** e **Attributi**, consultare "Creazione di una definizione di oggetto business" a pagina 61.

## Apertura di una definizione di oggetti business e degli elementi secondari contenuti

Business Object Designer Express consente di aprire finestre separate per modificare definizioni di oggetti business principali e i relativi secondari contenuti.

Figura 29 illustra le finestre separate per la modifica di oggetti business principali e secondari.

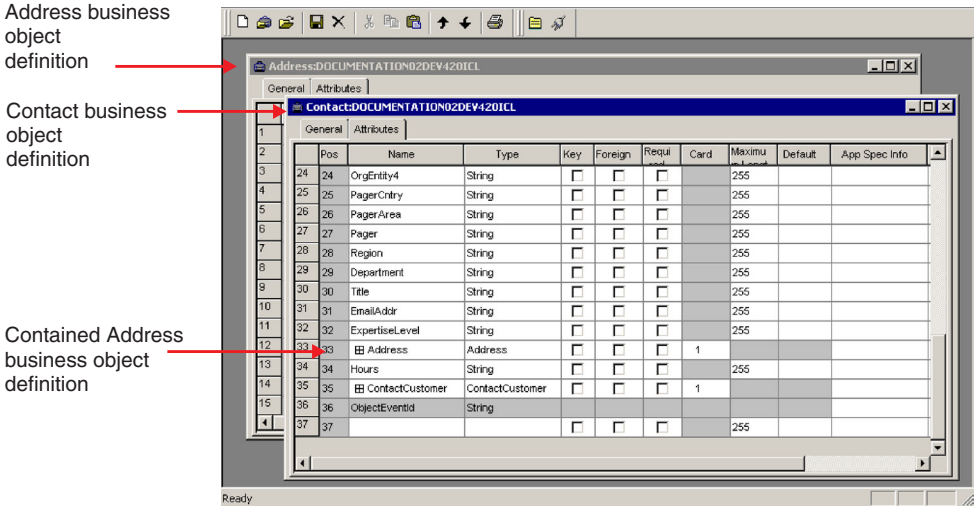



*Figura 29. Finestra Separa per oggetti business principali e secondari*

Si noti che l'attributo Indirizzo in Contatta oggetto business viene compresso in Figura 29. È possibile espandere l'attributo così che la finestra Contatta visualizzi tutti gli attributi dell'oggetto business Indirizzo, che consente di modificare l'elemento secondario direttamente da quello principale. Per evitare di modificare la stessa definizione in due posti, tuttavia, lo strumento chiude automaticamente una finestra degli oggetti business secondari ogni qualvolta si espande un oggetto business secondario nell'oggetto business principale.

Figura 30 illustra lo strumento dopo che l'attributo Indirizzo contatto è stato espanso e è stata chiusa la finestra Indirizzo.

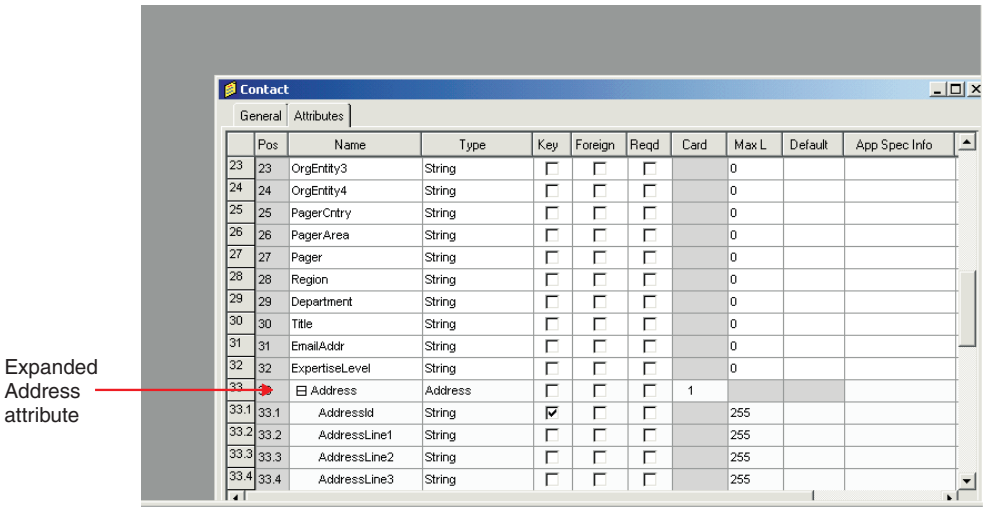



*Figura 30. Espansione di un attributo di oggetto business principale che rappresenta l'oggetto business secondario*

# Funzionalità di Business Object Designer Express

È possibile accedere alle funzioni di Business Object Designer Express in uno dei seguenti modi:

- Dalla barra dei menu:
- Utilizzando le icone della barra degli strumenti

Nelle sezioni seguenti viene fornita una panoramica dei vari menu e opzioni di menu di Business Object Designer Express.

## Menu File

Il menu **File** contiene le seguenti voci:

- **Nuovo BO** — Crea una definizione di oggetti business manualmente. Per ulteriori informazioni, consultare "Creazione di una definizione di oggetto business" a pagina 61.
- **Nuovo utilizzando ODA** — Visualizza la Procedura guidata dell'oggetto Business, che consente di creare una definizione di oggetti business da un ODA (Object Discovery Agent). Per ulteriori informazioni, consultare "Uso di Object Discovery Agent per creare una definizione di oggetto business" a pagina 70.
- **Apri** — Apre una definizione di oggetti business localizzata nel progetto. Se Business Object Designer Express è in esecuzione da System Manager, il progetto è un progetto basato su ICL. Altrimenti, si tratta di un progetto locale. Consultare "Operazioni con i progetti" a pagina 47 per ulteriori informazioni sui progetti.
- **Apri da file** — Importa e apre facoltativamente una definizione di oggetti business da una directory locale.
- **Salva** — Salva la definizione di oggetti business nel progetto nel modo seguente:

  Se si modifica una definizione di oggetti business esistenti:

  – Se il progetto è basato su ICL, la definizione di oggetti business viene salvata nel progetto che l'ha originata.
  – Se il progetto è locale, la definizione di oggetti business viene salvata nel file esistente.

  Se si crea una nuova definizione di oggetti business:

  – Se Business Object Designer Express è in esecuzione da System Manager, viene richiesto di selezionare la ICL in cui si desidera salvare a definizione di oggetti business.
  – Altrimenti, viene richiesto di specificare la directory di destinazione locale e il nome file per la definizione di oggetti business.

  La definizione di oggetti business può essere salvata come un file di tipo:

  **.xsd** XML Schema Definition. Questo è il tipo di file predefinito.

  **.in oppure .txt** InterChange Server Express

  **.xls** Foglio di calcolo
- **Salva con nome** — Salva la definizione di oggetti business con un nuovo nome. Il nome del file che contiene la definizione di oggetti business deve essere unica all'interno del progetto di destinazione.

  Se si modifica una definizione di oggetti business esistenti:

  – Se Business Object Designer Express è in esecuzione da System Manager, la definizione di oggetti business viene salvata nel progetto basato su ICL.

– Se la definizione di oggetti business è stata aperta da un file, la definizione di oggetti business modificata viene salvata in questo file.

Se si crea una nuova definizione di oggetto business:

– Se Business Object Designer Express è in esecuzione da System Manager, viene richiesto di selezionare la ICL in cui si desidera salvare a definizione di oggetti business.

– Altrimenti, viene richiesto di specificare una directory di destinazione locale e il nome file per la definizione di oggetti business.

La definizione di oggetti business può essere salvata come un file di tipo:

| | |
|---|---|
| **.xsd** | XML Schema Definition. Questo è il tipo di file predefinito. |
| **.in oppure .txt** | InterChange Server Express |
| **.xls** | Foglio di calcolo |

- **Salva tutto**—Salva tutte le definizioni di oggetti business aperte come descritto per la voce di menu **Salva** in 56.
- **Salva copia in file**—Esporta una copia della definizione di oggetti business in un file separato.
- **Copia tutto in un file**—Esporta tutte le definizioni di oggetti business in un progetto come un file nel formato repos-copy.
- **Chiudi**—Chiude la definizione selezionata. Questa voce di menu non è disponibile se non vi sono definizioni aperte.
- **Chiudi tutto**—Chiude tutte le definizioni aperte. Questa voce di menu non è disponibile se non vi sono definizioni aperte.
- **Elimina** — Consente di eliminare una definizione di oggetti business da un progetto.

  **Nota:** Business Object Designer Express consente solo di eliminare le definizioni di oggetti business dal progetto. Se il progetto è basato su ICL, le definizioni di oggetti business eliminate vengono rimosse dalla ICL specificata. Se il progetto è locale, le definizioni di oggetti business eliminate vengono rimosse dal progetto locale, ma non vengono interessati i file che contengono definizioni di oggetti business nelle directory locali. Per eliminare i file locali, utilizzare gli strumenti forniti da Windows.
- **Imposta stampante** — Consente di specificare la stampante e le proprietà di stampa.
- **Anteprima stampa** — Visualizza un'anteprima delle definizione da stampare. Questa voce di menu non è disponibile se non vi sono definizioni aperte.
- **Stampa** — Consente di stampare la definizione selezionata. Questa voce di menu non è disponibile se non vi sono definizioni aperte.
- **Esci** — Consente di uscire da Business Object Designer Express.

## Menu Modifica

Tutte le opzioni del menu **Modifica** non sono disponibili se non vi sono definizioni aperte. Il menu **Modifica** contiene le seguenti voci:

- **Taglia** — Elimina un attributo dalla definizione o il testo dalla colonna. Questa voce di menu non è disponibile se non è stato selezionato testo in una colonna oppure non sono stati selezionati attributi (facendo clic nella colonna all'estrema sinistra). Consultare Figura 28 a pagina 54 per una descrizione degli attributi di modifica nella finestra.

- **Copia** — Copia un attributo nella definizione o il testo in una colonna. Questa voce di menu non è disponibile se non è stato selezionato testo in una colonna oppure non sono stati selezionati attributi (facendo clic nella colonna all'estrema sinistra). Consultare Figura 28 a pagina 54 per una descrizione degli attributi di modifica nella finestra.
- **Incolla** — Incolla un attributo tagliato o copiato nella definizione, oppure il testo tagliato o copiato nella colonna selezionata. Per impostazione predefinita, lo strumento incolla un attributo in buffer alla fine della definizione. Tuttavia, se si inserisce una riga vuota in una posizione specifica, è possibile incollare l'attributo in buffer nella riga vuota.
- **Elimina riga** — Elimina un attributo dalla definizione. Questa voce di menu non è disponibile se non è stato selezionato un attributo (facendo clic nella colonna all'estrema sinistra). Consultare Figura 28 a pagina 54 per una descrizione degli attributi di modifica nel pannello.
- **Seleziona tutti** — Seleziona tutti gli attributi nella definizione.
- **Inserisci sopra** — Inserisce una riga vuota al di sopra dell'attributo selezionato.
- **Inserisci sotto** — Inserisce una riga vuota al di sotto dell'attributo selezionato.
- **Sposta su** — Sposta l'attributo selezionato su di una riga. Questa voce di menu non è disponibile se non vi sono attributi selezionati.
- **Sposta giù** — Sposta l'attributo selezionato giù di una riga. Questa voce di menu non è disponibile se non vi sono attributi selezionati.

**Nota:** È possibili accedere alle voci di menu **Inserisci sopra, Inserisci sotto, Taglia, Copia, Incolla** e **Elimina** facendo clic con il pulsante destro del mouse nella colonna all'estrema destra di un attributo.

## Menu Visualizza

Le operazioni del menu **Visualizza** sono valide quando prima viene aperto Business Object Designer Express e quando l'area di lavoro è relativa alla visualizzazione dei diagrammi di attività. Molte di queste operazioni possono essere attivate o disattivate.

Il menu **Visualizza** visualizza le seguenti opzioni:

- **Espandi tutto** — Visualizza tutti gli attributi in tutti gli oggetti business secondari. Questa voce di menu non è disponibile se non vi sono definizioni aperte.
- **Comprimi tutto** — Chiude la visualizzazione di tutti gli attributi in tutti gli oggetti business secondari. Questa voce di menu non è disponibile se non vi sono definizioni aperte.
- **Preferenze** — Apre la finestra Preferenze oggetto business che consente di disattivare la conferma dell'eliminazione di oggetti business.
- **Barra degli strumenti** — Contiene un menu secondario con voci che controllano la visualizzazione delle due barre degli strumenti di Business Object Designer Express. Le opzioni di menu includono:
  - **Standard** — Quando si seleziona questa voce di menu, Business Object Designer Express visualizza i pulsanti per la barra degli strumenti Standard.
  - **Programmi** — Quando si fa clic su questa voce di menu, Business Object Designer Express visualizza i pulsanti per accedere ad altri Express programmi Toolset.
- **Barra di stato** — Quando si fa clic su questa voce di menu, Business Object Designer Express visualizza un messaggio di stato di una riga nella parte inferiore della finestra principale.

**Nota:** È possibile accedere alle voci **Espandi**, **Comprimi**, e **Apri nella finestra** quando si fa clic con il pulsante destro del mouse nella colonna all'estrema sinistra di un attributo che rappresenta un oggetto business secondario oppure un array degli oggetti business secondari.

## Menu Strumenti

Il menu **Strumenti** contiene le seguenti voci:

- **Log Viewer** — Apre Log Viewer.
- **Connector Configurator Express** — Apre Express.
- **System Manager** — Apre System Manager.

## Menu Finestra

Il menu **Finestra** funziona come in un ambiente Windows standard. Utilizzare le opzioni di menu per controllare le funzioni di visualizzazione quali affiancamento, sovrapposizione e attivazione di finestre aperte.

# Capitolo 4. Sviluppo delle definizioni dell'oggetto business

Questo capitolo conduce attraverso i passi base necessari alla creazione e alla cancellazione della definizione di un oggetto business. Dopo aver completato questo capitolo, si acquisirà familiarità con i passi per la creazione di una definizione sia manualmente sia usando un ODA (Object Discovery Agent). Ogni ODA genera definizioni per una specifica applicazione.

Sebbene questo capitolo esponga i meccanismi di creazione delle definizioni di oggetti business, bisogna comprendere i concetti di progettazione prima di poterne realmente crearne uno. Per ulteriori informazioni, consultare Capitolo 2, "Progettazione di oggetti business", a pagina 17. Per informazioni su come creare un Object Discovery Agent, consultare Capitolo 5, "Sviluppo di in Object Discovery Agent", a pagina 95.

Gli argomenti principali di questo capitolo sono:

- "Creazione di una definizione di oggetto business"
- "Cancellazione di una definizione di un oggetto business" a pagina 68
- "Uso di Object Discovery Agent per creare una definizione di oggetto business" a pagina 70

## Creazione di una definizione di oggetto business

Esistono due modi per creare una definizione di oggetto business:

- Manualmente—Utile quando si crea un oggetto business generico o un oggetto business semplice oppure quando si modifica la definizione generata da un Object Discovery Agent. Oggetto business
- Express fornisce un'interfaccia grafica per la creazione manuale di una definizione di oggetto business. Questa sezione rappresenta un supporto didattico che spiega quanto segue:
  - "Creazione manuale della definizione di un oggetto business"
  - "Creazione di una definizione di oggetto business gerarchico manualmente" a pagina 67
- Uso di un Object Discovery Agent—Utile per la creazione di un oggetto business specifico dell'applicazione. L' *Object Discovery Agent* esamina le entità specificate nell'applicazione, "scopre" gli elementi di quegli oggetti che corrispondono agli attributi dell'oggetto business e alle proprietà di ciascun attributo e genera la definizione di oggetto business. Per ulteriori informazioni, consultare "Uso di Object Discovery Agent per creare una definizione di oggetto business" a pagina 70.

### Creazione manuale della definizione di un oggetto business

Questa sezione descrive la creazione manuale della definizione di oggetto business cui è stat assegnato il nome `Hello`. In InterChange Server Express, questo oggetto business è creato con la collaborazione di `SampleHello`, la cui creazione è descritta nella *Collaboration Development Guide*.

Figura 31 illustra la definizione di oggetto business `Hello` che è stata creata e mostra i valori che Interchange Server Express potrebbe aspettarsi dal suo oggetto business di evento triggering.

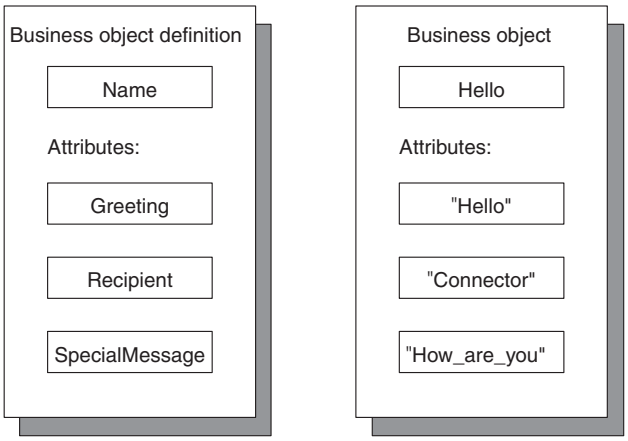



*Figura 31. Oggetto business Hello*

Per creare manualmente la definizione di oggetto business:

1. Avviare Business Object Designer Express; per ulteriori informazioni, consultare "Avvio di Business Object Designer Express" a pagina 50.
2. Fare clic su **File > Nuovo**.

   Business Object Designer Express visualizza la casella di dialogo Nuovo oggetto business. Figura 32 mostra la versione della casella di dialogo Nuovo oggetto business che si vede se si sta eseguendo Business Object Designer Express da System Manager. Se non si sta eseguendo Business Object Designer Express da System Manager, l'elenco **Crea nel progetto** viene omesso dalla casella di dialogo.

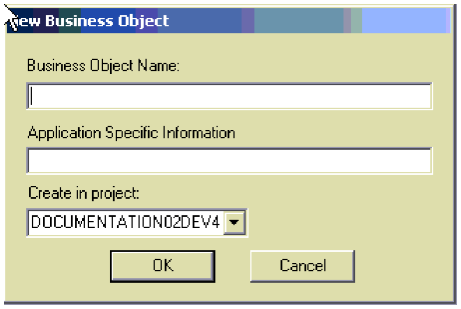



*Figura 32. Casella di dialogo Nuovo oggetto business*

3. Immettere il nome `Hello` per la definizione del nuovo oggetto business.

   I nomi sono di solito sensibili al maiuscolo-minuscolo, per cui immettere il nome esattamente come viene mostrato qui.

   **Nota:** Il nome della definizione dell'oggetto business può contenere solo caratteri alfanumerici e segni di sottolineatura (_). Questo nome deve usare *solo* caratteri definiti nella serie di codici associati alla locale U.S. English (`en_US`).
4. Lasciare vuota la casella **Informazioni specifiche dell'applicazione** e fare clic su OK.

Business Object Designer Express visualizza casella di dialogo della definizione di oggetto business, come illustrato in Figura 33..

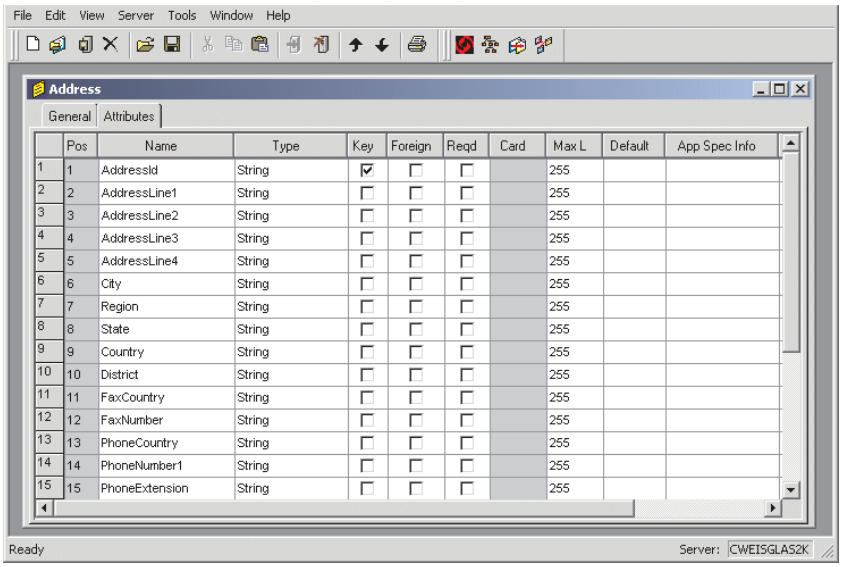



*Figura 33. Visualizzazione iniziale della nuova definizione di oggetto business*

**Nota:** Potrebbero esserci minori differenze nell'interfaccia di Business Object Designer Express. Tuttavia, la funzionalità base del tool è la stessa.

## Aggiunta di attributi

Ogni parte di informazione nell'oggetto business è rappresentato da un *attributo* nella definizione di oggetto business `Hello`. Bisogna fornire le definizioni dell'attributo per l'oggetto business `Hello`. Come illustrato in Figura 33,, Business Object Designer Express aggiunge automaticamente una voce per il contrassegno richiesto end-of-object, `ObjectEventId`.

**Importante**

Non cancellare, modificare, o spostare l'attributo `ObjectEventId`. Questo attributo è riservato ad un uso interno del sistema WebSphere Business Integration Server Express. Business Object Designer Express sposta automaticamente questo attributo quando si salva la definizione.

La riga per ciascun attributo definisce le proprietà dell'attributo. Per informazioni sulle proprietà dell'attributo, consultare "Attributi dell'oggetto business e proprietà degli attributi" a pagina 4.

Come mostra Figura 31 a pagina 62, la definizione di oggetto business `Hello` ha i seguenti attributi: `Saluti`, `Destinatario` e `SpecialMessage`. Definire gli attributi e le loro proprietà, uno alla volta.

**Aggiunta dell'attributo Saluti:** Per aggiungere l'attributo `Saluti`:

1. Immettere il nome dell'attributo `Saluti` nella colonna **Nome** della prima riga vuota disponibile, che è 2 per il primo attributo.

   **Nota:** Il nome questo attributo deve usare *solo* caratteri definiti nella serie di codici associati alla locale U.S. English (`en_US`).

2. Fare clic sulla colonna **Tipo** e selezionare **Stringa** per il tipo di attributo. Il tipo di un attributo è il suo tipo di dati.

   **Suggerimenti:**

   Se si hanno altri oggetti business aperti in Business Object Designer Express, i loro nomi appaiono nell'elenco **Tipo**. La visualizzazione di oggetti business esistenti tra le scelte di **Tipo** consente di creare un oggetto business gerarchico con un attributo il cui tipo è un altro oggetto business.

   Quando InterChange Server Express e System Manager sono in esecuzione, allora ogni definizione di oggetto business in Integration Component Library da cui si sta lavorando, viene automaticamente visualizzata nell'elenco.

   Se si sta usando InterChange Server Express ma System Manager non è in esecuzione, l'unico modo per aggiungere una definizione di oggetto business come secondario ad un'altra definizione di oggetto business è quello di importare quella definizione di oggetto business in Business Object Designer Express facendo clic su **File > Apri da File**.
3. Saltare le colonne **Chiave**, **Esterna**, **Reqd** (or **Richiesta**), e **Scheda**.

   Queste colonne specificano se l'attributo corrente è la chiave primaria dell'oggetto business o chiave esterna, se è richiesto il valore dell'attributo e se l'attributo rappresenta un oggetto business secondario o degli oggetti. Per una spiegazione di queste proprietà, consultare Capitolo 2, "Progettazione di oggetti business", a pagina 17.
4. Nella casella **Lunghezza massima**, lasciare il valore predefinito di `255`.

   Questa casella indica il numero minimo di tipi disponibili per valori di attributo.
5. Nella casella **Predefinita**, immettere `Hello`.

   Questo indica il valore da usare se non sono stati forniti altri valori per l'attributo al momento dell'avviamento.

   Sono state ora definite le seguenti proprietà per l'attributo `Saluti` :

| | |
|---|---|
| Nome: | `Saluti` |
| Tipo: | `Stringa` |
| Lunghezza massima: | `255` |
| Valore predefinito: | `Hello` |

6. Ignorare tutte le altre colonne e fare clic sulla colonna **Nome** della terza riga.

**Aggiunta di un attributo Destinatario:** Il secondo attributo, `Destinatario`, è una stringa.

In InterChange Server Express, l'oggetto di collaborazione `SampleHello` usa questo attributo nel seguente modo:

- Il connettore imposta il valore su `Collaborazione` quando invia un messaggio alla collaborazione.
- La collaborazione imposta il valore su `Connettore` quando invia un messaggio al connettore.

Almeno un attributo in ogni definizione di oggetto business deve essere un *attributo chiave*. Un attributo chiave contiene un valore tramite il quale il sistema WebSphere Business Integration Server Express identifica univocamente le istanze dell'oggetto business. Rendere l'attributo `Destinatario` attributo chiave.

Per aggiungere l'attributo `Destinatario`, immettere il testo `Destinatario` nella colonna **Nome** e seguire i passi per aggiungere l'attributo `Saluti` usando le seguenti proprietà:

| | |
|---|---|
| Nome: | `Destinatario` |
| Tipo: | `Stringa` |
| Lunghezza massima: | `255` |
| Valore predefinito: | `Collaborazione` |
| Chiave: | Sì<br>(Compare un segno di spunta nella colonna Chiave) |

Ignorare tutte le altre colonne vuote e fare clic sulla colonna **Nome** della quarta riga.

**Aggiunta dell'attributo SpecialMessage:** Il terzo attributo, `SpecialMessage`, è una stringa.

In InterChange Server Express, la collaborazione `SampleHello` si aspetta che il valore di questo attributo venga immesso dal responsabile di sistema o da un'altra persona con accesso alle proprietà di configurazione della collaborazione dopo essere stato creato l'oggetto di collaborazione. La collaborazione ottiene dinamicamente il valore della proprietà di configurazione e lo accoda al messaggio.

Per aggiungere l'attributo `SpecialMessage`, immettere il testo `SpecialMessage` nella colonna **Nome** e seguire i passi per aggiungere l'attributo `Saluti` usando le seguenti proprietà

| | |
|---|---|
| Nome: | `SpecialMessage` |
| Tipo: | `Stringa` |
| Lunghezza massima: | `255` |

Lasciare vuote le altre colonne.

I separatori degli **Attributi** ora visualizzano tre attributi definiti dall'utente: `Saluti` , `Destinatario` e `SpecialMessage`. Figura 34 illustra gli attributi dell'oggetto business `Hello`.

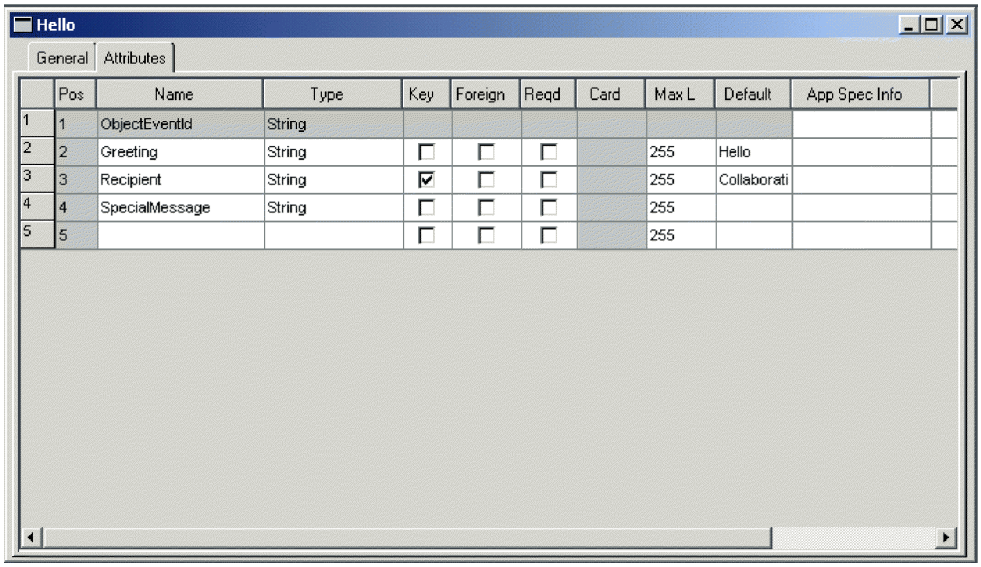



*Figura 34. Definizione di nuovo oggetto business con attributi*

## Modifica dell'ordine dell'attributo

E' possibile modificare graficamente l'ordine di sequenza degli attributi nella definizione dell'oggetto business. Ad esempio, per porre l'attributo chiave, `Destinatario`, sopra l'attributo `Saluti`, fare clic sulla prima colonna (quella più a sinistra) e trascinare il cursore di una riga sopra.

## Specifica delle istruzioni supportate

Bisogna ora specificare le istruzioni che l'oggetto business `Hello` supporta. Queste istruzioni rappresentano gli eventi triggering che l'oggetto business invia a InterChange Server Express. Fare clic sulla scheda **Generale** della finestra di dialogo per la definizione dell'oggetto business Hello per visualizzare il pannello in cui specificare le istruzioni. Figura 35 illustra questa scheda.

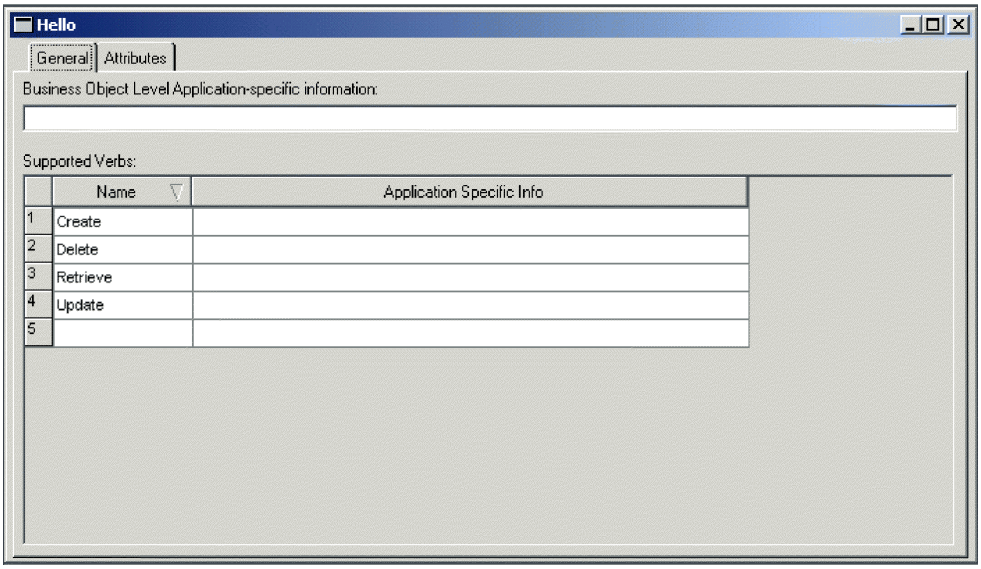



*Figura 35. Scheda di editazione Generale*

L'oggetto business supporta quattro valori predefiniti istruzione —Crea, Cancella, Richiama e Aggiorna; appaiono nella scheda **Generale** come valore predefinito. Per le funzioni di questo supporto didattico, è supportato solo un evento triggering: Crea. Pertanto, cambiare la definizione di oggetto business per supportare solo questa istruzione.

**Importante:** Bisogna specificare almeno un'istruzione per ciascuna definizione di oggetto business.

**Nota:** Il nome di una istruzione può contenere solo caratteri alfanumerici e di sottolineatura _). Questo nome deve usare *solo* caratteri definiti nell'insieme di codici associati alla locale U.S. English (en_US).

Per indicare che l'oggetto business Hello supporta solo l'istruzione Crea, si possono cancellare le rimanenti istruzioni simultaneamente oppure individualmente.

**Cancellazione di istruzioni multiple:** Per cancellare le istruzioni Cancella, Richiama e Aggiorna:

1. Selezionare l'istruzione Cancella e, tenendo premuto il tasto delle maiuscole, fare clic sull'istruzione Aggiorna.
2. Premere il tasto Cancella.

**Cancellazione delle istruzioni individuali:** Per cancellare individualmente ciascuna istruzione:

1. Fare clic sul numero a sinistra della riga Cancella nella tabella **Istruzioni supportate**.

   Viene selezionata la riga.
2. Premere il tasto Cancella.
3. Ripetere i passi 1 e 2 per le istruzioni Richiama e Aggiornanella tabella **Istruzioni supportate**.
4. Lasciare vuota la casella **Info specifiche sull'applicazione** per l'istruzione Crea.

La definizione per l'oggetto business Hello è terminata. Questo è il momento giusto per salvare le modifiche facendo clic su **File > Salva**. Se si sta usando un progetto basato su ICL, la definizione viene salvata in ICL. Se si sta usando un progetto locale, verrà richiesto di specificare un nome file e una directory locale in cui salvare la definizione.

## Creazione di una definizione di oggetto business gerarchico manualmente

Questa sezione descrive come creare la definizione gerarchica di un oggetto business definendo un attributo che rappresenti un oggetto business secondario o un array di oggetti business secondari.

Poiché la precedente sezione illustrava come definire un attributo semplice ed istruzioni supportate, questa sezione spiega solo la definizione di un attributo che rappresenta un oggetto business secondario. Questo esempio crea un oggetto business che si chiama HierarchicalBO che ha due attributi:

- Un attributo Chiave che serve come chiave richiesta dell'oggetto business.
- Un attributo Addr che rappresenta l'oggetto business Indirizzo con cardinalità 1.

Per creare manualmente la definizione gerarchica di un oggetto business:

1. Aprire Business Object Designer Express.
2. Fare clic su **File > Nuovo**.

   Business Object Designer Express visualizza la finestra di dialogo Nuovo oggetto business, come illustrato in Figura 32 a pagina 62.
3. Immettere il nome `HierarchicalBO` per la nuova definizione di oggetto business.
4. Lasciare vuota la colonna **Informazioni specifiche sull'applicazione** e fare clic su OK.

   Business Object Designer Express visualizza la finestra di dialogo di definizione dell'oggetto business come illustrato in Figura 33 a pagina 63.
5. Creare un attributo chiave nella prima riga vuota disponibile, che è 2 per il primo attributo. Chiamarlo `Chiave`, specificare eventuali tipi di dati semplici e fare clic sulla colonna **Chiave**.
6. Creare il prossimo attributo nella successiva riga vuota disponibile, che è la 3. Chiamarlo `Addr`.
7. Fare clic sull'elenco **Tipo** e selezionare **Indirizzo** per il tipo di attributo.

   **Nota:** Se l'oggetto business non esiste nell'elenco, è possibile crearlo selezionando **Nuovo oggetto business** nell'elenco **Tipo**. Bisogna avere un nuovo oggetto business secondario prima di poter completare questo passo.
8. Saltare le colonne **Chiave**, **Esterna**, e **Reqd** (o **Obbligatorio**). Fare clic sull'elenco **Scheda** e selezionare **1**.
9. Ignorare tutte le altre colonne. Definire le istruzioni supportate e salvare le definizioni.

## Cancellazione di una definizione di un oggetto business

Se si cancella la definizione di oggetto business con Business Object Designer Express o con System Manager in InterChange Server Express. Questa sezione descrive:

- "Cancellazione di una definizione usando Business Object Designer Express"
- "Cancellazione di una definizione usando System Manager" a pagina 70

**Importante**

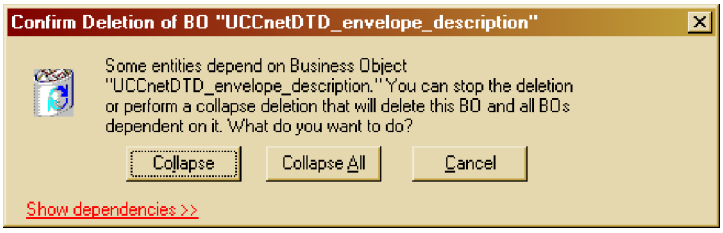
E' possibile cancellare le definizioni di oggetto business da una libreria del componente di integrazione tramite System Manager (se si sta usando ICS e System Manager è in esecuzione) o da un progetto in Business Object Designer Express. Non è possibile usare la funzione di cancellazione in Business Object Designer Express o in System Manager per cancellare file locali che contengono definizioni di oggetti business. Per eliminare i file locali, utilizzare gli strumenti forniti da Windows.

### Cancellazione di una definizione usando Business Object Designer Express

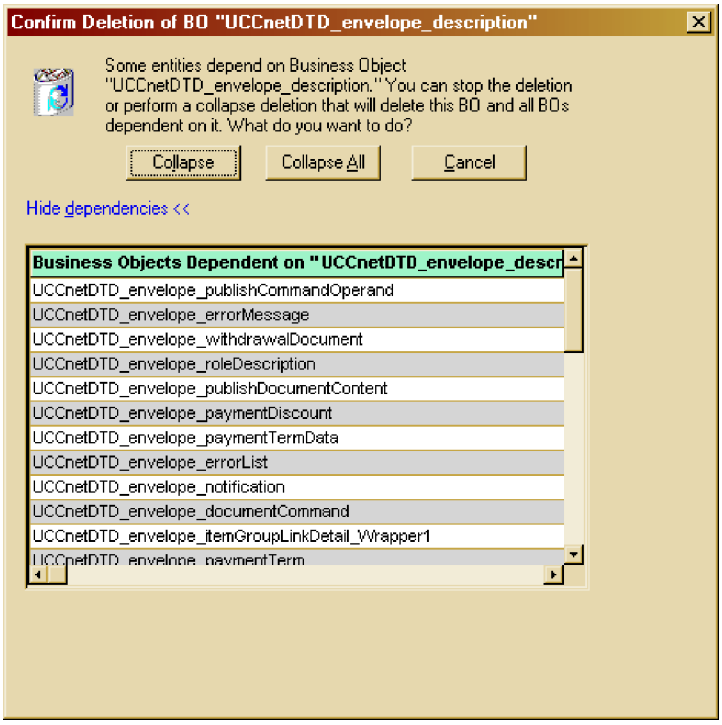
Per cancellare una definizione di oggetto business da un progetto usando Business Object Designer Express, fare quanto di seguito riportato:

1. Aprire Business Object Designer Express.
2. Dall'elenco delle definizione di oggetto business nel progetto, selezionare il nome della definizione che si vuole cancellare.

3. Per selezionare nomi multipli, fare una delle seguenti operazioni:
   - Per selezionare nomi consecutivi, fare clic sul primo nome e, tenendo premuto il tasto delle maiuscole, fare clic sull'ultimo nome.
   - Per selezionare nomi non consecutivi, premere il tasto `Ctrl` e fare clic su ogni nome.
4. Dopo aver selezionato le definizioni da cancellare, fare doppio clic con il tasto destro e poi fare clic su **Cancella**.
   - Business Object Designer Express visualizza il messaggio di conferma **Cancellazione in corso dell'oggetto business**. Fare clic su **Sì** per cancellare la definizione di oggetto business selezionata nel passo 2 a pagina 68 o per cancellare tutte le definizioni di oggetto business selezionate nel passo 3.
   - Se la definizione di oggetto business ha delle dipendenze con altri oggetti business, Business Object Designer Express visualizza un messaggio di conferma.




5. Se esistono dipendenze, fare clic sul collegamento **Mostra dipendenze**. Tutte le dipendenze con altri oggetti business sono elencati per le definizioni di oggetto che si vogliono cancellare.




6. Effettuare una delle seguenti operazioni:
   - Fare clic su **Comprimi** per cancellare la definizione di oggetto business selezionata nel passo 2 e tutti gli oggetti business che dipendono da questo.

- Se si selezionano oggetti business multipli nel passo 3, fare clic su **Comprimi tutto** per cancellare gli oggetti business selezionati e tutti gli oggetti business che dipendono da ciascuno di quegli oggetti business.
- Fare clic su **Annulla** per annullare la cancellazione della definizione di un oggetto business e delle sue dipendenze.

### Cancellazione di una definizione usando System Manager

Per cancellare una definizione di oggetto business usando System Manager, fare quanto segue:

1. Avviare System Manager.
2. Espandere **Integration Component Libraries** e poi espandere la libreria del componente di integrazione da cui si vuole cancellare la definizione di un oggetto business.
3. Aprire la cartella degli oggetti business e selezionare il nome della definizione di oggetto business da cancellare.
4. Cancellare la definizione di oggetto business facendo una delle seguenti operazioni:
   - Fare clic sull'icona della barra degli strumenti **Cancella**.
   - Fare clic con il tasto destro sulla definizione di oggetto business e selezionare **Cancella**.
5. Quando viene richiesto se si vuole cancellare, fare clic su **Sì**.
6. Se la definizione di oggetto business ha dipendenze con altri oggetti business, System Manager informa con un messaggio di errore. Bisogna usare Business Object Designer Express per rimuovere tali dipendenze prima di poter cancellare la definizione di oggetto business con System Manager.

## Uso di Object Discovery Agent per creare una definizione di oggetto business

Questa sezione descrive come usare un Object Discovery Agent (ODA) per generare definizioni di oggetto business per oggetti business specifici dell'applicazione Un ODA è un componente facoltativo di un adattatore. Quando si installa un adattatore predefinito che ha un ODA, il suo ODA viene installato automaticamente. Se si sta sviluppando un adattatore cliente e si vuole usare un ODA per creare definizioni di oggetti business, è possibile usare Object Discovery Agent Development Kit (ODK) per svilupparne uno. Per ulteriori informazioni sullo sviluppo di un ODA personalizzato, consultare Capitolo 5, "Sviluppo di in Object Discovery Agent", a pagina 95.

Per configurare ed eseguire ODA, usare la procedura guidata di Business Object in Business Object Designer Express. La procedura guidata di Business Object è un'interfaccia grafica per ODA che gestisce la scoperta e il processo di generazione del contenuto. Questa sezione fornisce le seguenti informazioni:

## Prima di usare un ODA

Prima di eseguire un ODA, verificare che si siano verificati i seguenti passaggi:

- I file di avvio del sistema sono disponibili e corretti.
- ODA è stato avviato.
- Business Object Designer Express è stato avviato.

### File di avvio del sistema

Perché ODA possa avviarsi, è necessario verificare che il proprio sistema abbia i requisiti necessari a ODA. Quando si installa un adattatore pre-definito che ha ODA, questi file di avvio del sistema ODA devono essere installati automaticamente. Se si sta sviluppando un adattatore personalizzato con un ODA personalizzato, questi file di avvio del sistema ODA devono essere creati come parte del processo di sviluppo ODA. tuttavia, IBM raccomanda che di confermare l'esistenza degli script di avviamento e che siano corretti per l'ODA di cui si dispone:

Ogni ODA richiede uno *scipt di avvio*, che avvii l'esecuzione di ODA. Prima di avviare un ODA per la prima volta, assicurarsi che le variabili siano correttamente impostate all'interno degli script di avvio. Aprire per la modifica la shell (`start_`*`ODAname`*`.sh`) o il file batch (`start_`*`ODAname`*`.bat`) e confermare che i valori descritti in Tabella 12 siano corretti.

*Tabella 12. Variabili di configurazione della shell ODA e del file batch*

| Variabile | Spiegazione | Esempio |
|---|---|---|
| `imposta AGENTNAME` | Nome di ODA | `imposta AGENTNAME=`*`ODAname`* |
| `imposta AGENT` | Nome del file jar di ODA | **Linux:** `set AGENT = ${`*`ProductDir`*`}/ODA/`*`srcDataName`*`/`*`ODAname`*`.jar`<br>**WINDOWS:** `set AGENT =` *`%ProductDir%`*`\ODA\`*`srcDataName`*`\`*`ODAname`*`.jar` |
| `imposta AGENTCLASS` | Nome della classe ODA Java | `imposta AGENTCLASS=com.ibm.oda.`*`srcDataName`*`.`*`ODAname`* |

Per informazioni sul nome ODA (*ODAname*) e sul suo nome dei dati sorgente (*srcDataName*), consultare "Denominazione dell'ODA" a pagina 173.

### avvio di ODA

E' possibile avviare ODA con lo script di avvio appropriato per il proprio sistema operativo.

**Linux**

`start_`*`srcDataName`*`ODA.sh`

**Windows**

`start_`*`srcDataName`*`ODA.bat`

Configurare ed eseguire ODA usando la procedura guidata di Business Object in Business Object Designer Express. Procedura guidata dell'oggetto Business individua ciascun ODA per il nome specificato nella variabile `AGENTNAME` di ogni script o file batch.

**Nota:** Per informazioni su come avviare istanze multiple di ODA, consultare "Utilizzo di più ODA contemporaneamente" a pagina 90.

### Avvio di Business Object Designer Express

Una volta avviato ODA, bisogna aprire Business Object Designer Express per configurarlo ed eseguirlo. Per informazioni sui modi di apertura di Business Object Designer Express, consultare "Avvio di Business Object Designer Express" a pagina 50. Per eseguire ODA, Business Object Designer Express fornisce Procedura guidata dell'oggetto Business, che guida attraverso ciascun passo.

Per avviare Procedura guidata dell'oggetto Business, fare quanto segue:

1. Aprire Business Object Designer Express usando uno dei metodi elencati in Tabella 11 a pagina 50.
2. Fare clic su **File > Nuovo uso di ODA**.

Procedura guidata dell'oggetto Business inizia a visualizzare la prima finestra di dialogo nella procedura guidata Seleziona agente. Tabella 13 riepiloga i passi di Procedura guidata dell'oggetto Business.

*Tabella 13. Passi della procedura guidata di Business Object*

| Attività | | Passi in Procedura guidata di Business Object |
|---|---|---|
| 1. | Seleziona l'ODA desiderata | Passo 1: Seleziona agente |
| 2. | Ottieni le proprietà di configurazione, comprese quelle che descrivono il sorgente dati da aprire. | Passo 2: Configura agente |
| 3. | Ottieni i dati sorgente per i quali ODA genera il contenuto. | Passo 3: Seleziona sorgente |
| 4. | Conferma che i nodi sorgente selezionati sono quelli desiderati per la generazione del contenuto. | Passo 4: Conferma i nodi sorgente |
| 5. | Inizia il processo di generazione del contenuto | Passo 5: Genera oggetti business<br>Proprietà degli oggetti business |
| 6. | Salva le definizioni di oggetto business in un formato specificato dall'utente. | Passo 6: Salva gli oggetti business |

Per un esempio del modo in cui Procedura guidata dell'oggetto Business esegue un ODA, consultare "Uso di un campione ODA".

## Uso di un campione ODA

IBM fornisce un campione di Object Discovery Agent che converte i soldati romani (in formato XML) in definizioni di oggetto business. Per acquisire familiarità con l'uso di ODA, la seguente descrizione dettagliata della generazione di definizioni di oggetti business usa questo semplice ODA.

**Nota:** Per informazioni sull'ubicazione e sui file di questo campione ODA, consultare "Supporto di sviluppo per gli ODA" a pagina 105.

Questa sezione include le seguenti attività:

## Avvio di un campione ODA

Se è stato installato l'ADK (Adapter Development Kit), il campione ODA ed il file per eseguirlo sono ubicati nella directory `DevelopmentKits\Odk\Samples` nella directory del prodotto. Questo file per eseguire il campione ODA dipende dall'ambiente del proprio sistema operativo, come mostra Tabella 14.

*Tabella 14. Script di avvio per un campione ODA di un esercito romano*

| Sistema operativo | Script di avvio |
|---|---|
| Windows | `start_Agent4.bat` |

**Nota:** Il campione ODA Esercito romano fornisce cinque versioni per illustrare le varie funzioni su un ODA. Questa sezione esegue la quarta versione del campione ODA, che usa lo script di avvio `start_Agent4` e il file di classe `ArmyAgent4`.

Poiché l'ODA dell'esercito romano fornisce cinque versioni di sé stesso, tutti gli script di avvio chiamano uno script di avvio comune che si chiama `start_AgentX`, passando il nome della classe ODA (che è assegnata alla variabile di configurazione `AGENTCLASS` in `start_AgentX`). Pertanto, lo script di avvio `start_Agent4` deve contenere una chiamata a `start_AgentX`, passando il seguente percorso come nome della classe ODA:

```
com.ibm.btools.ODK2.RomanArmy.ArmyAgent4
```

Per verificare le variabili di configurazione di questo campione ODA, controllare il file batch o script `start_AgentX` per confermare che le variabili di conferma corrispondono a quelle in Tabella 15. Se si sposta qualcuno dei file usati dalla versione 4 del campione ODA dell'esercito romano, accertarsi di spostare le corrispondenti variabili di configurazione.

*Tabella 15. Variabili di configurazione per il campione ODA di un esercito romano*

| Variabile | Valore per il campione ODA dell'esercito romano |
|---|---|
| `AGENTNAME` | `set AGENTNAME=Romano` |
| `AGENT` | **Linux:** `set AGENT = ${`*`ProductDir`*`}/DevelopmentKits/Odk/Samples/RomanArmy/ArmyODA.jar`<br><br>**WINDOWS:** `set AGENT = %`*`ProductDir`*`%\DevelopmentKits\Odk\Samples\RomanArmy\ArmyODA.jar` |
| `FILE_LOCATION` | **Linux:** `set FILE_LOCATION = ${`*`ProductDir`*`}/DevelopmentKits/Samples/Odk/RomanArmy/RomanArmy.xml`<br><br>**WINDOWS:** `set FILE_LOCATION = %`*`ProductDir`*`%\DevelopmentKits\Samples\Odk\RomanArmy\RomanArmy.xml` |

**Importante**

Bisogna avviare il campione ODA prima di provare a connetterlo tramite Procedura guidata dell'oggetto Business. Procedura guidata dell'oggetto Business può localizzare solo gli ODA che sono stati avviati.

## Uso di ODA per creare definizioni di oggetti business

Per avviare Procedura guidata dell'oggetto Business, fare quanto segue:

1. Aprire Business Object Designer Express usando un metodo elencato in Tabella 11 a pagina 50.
2. Fare clic su **File > Nuovo uso di ODA**.

   Business Object Wizard visualizza la prima finestra di dialogo, Seleziona Agente, mostrato in Figura 36..

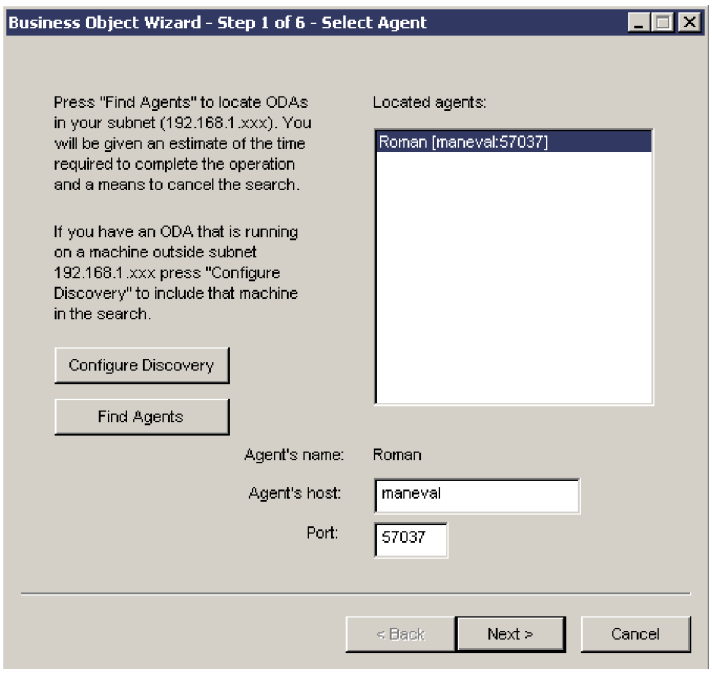



*Figura 36. casella di dialogo Seleziona agente*

3. Per selezionare l'ODA a cui si connette Procedura guidata dell'oggetto Business:
   a. Fare clic sul tasto **Trova agenti** per visualizzare gli ODA al momento in esecuzione (quelli che sono stati avviati con i loro script di avvio) nell'elenco **Agenti localizzati**.

      **Nota:** Se Procedura guidata dell'oggetto Business *non* localizza l'ODA desiderato, controllare l'avvio di ODA.

      Procedura guidata dell'oggetto Business identifica ciascun ODA in esecuzione attraverso il nome specificato per la variabile `AGENTNAME` o il suo script di avvio o tramite il file batch. Questo campione ODA si chiama `Romani`.
   b. Selezionare l' ODA dall'elenco **Agenti localizzati**. Procedura guidata dell'oggetto Business visualizza la selezione come **Nome agente**. In alternativa, è possibile trovare ODA specificando il suo nome host e il numero della porta.
4. Fare clic su **Avanti**. Procedura guidata dell'oggetto Business prova a collegarsi all'ODA specificato ODA. Se ODA è stato avviato, Procedura guidata dell'oggetto Business visualizza una finestra di stato appena si collega ad ODA, come mostra Figura 37.

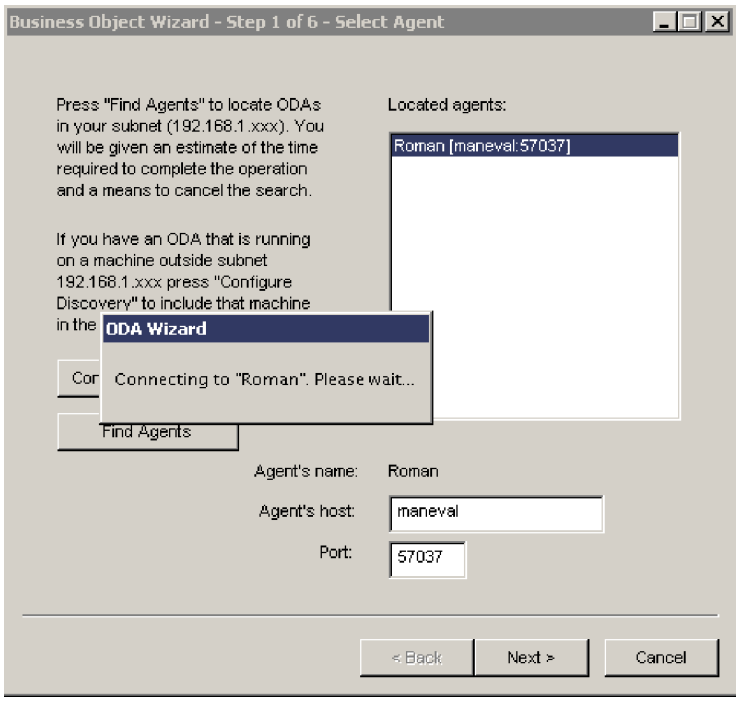



*Figura 37. Connessione a ODA.*

5. Quando Procedura guidata dell'oggetto Business si è collegato a ODA, viene visualizzata la seconda finestra di dialogo della procedura guidata, Configura agente, che è visualizzata in Figura 38. Questa finestra di dialogo mostra le proprietà della configurazione ODA richieste dal sorgente dati ed inizializza ODA.

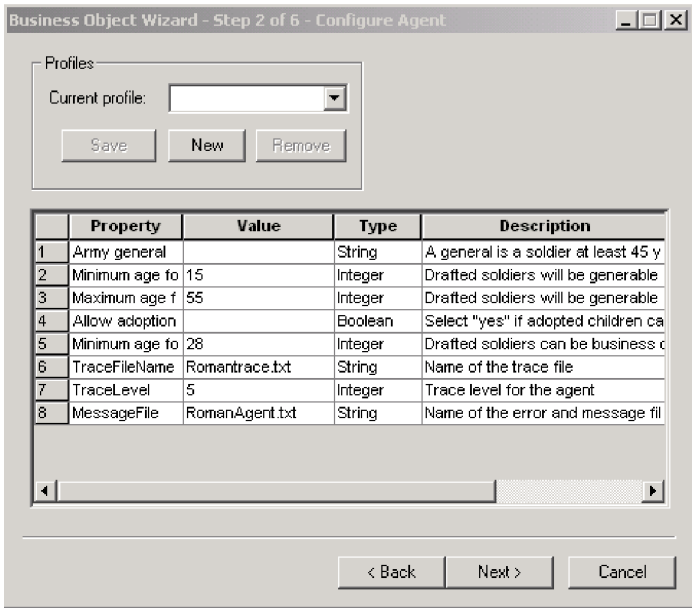



*Figura 38. casella di dialogo Configura agente*

6. Specificare i valori della configurazione ODA oppure selezionare un profilo per visualizzare i valori salvati. Una delle aree di configurazione richieste per

ODA è quella per l'impostazione della creazione log e della traccia. Per ulteriori informazioni, consultare "Impostazione di Creazione log e traccia" a pagina 82.

La prima volta che si usa un ODA particolare, specificare i valori per ciascuna delle sue proprietà di configurazione. Dopo aver fatto questo, si possono salvare i valori della proprietà in un profilo facendo clic sul pulsante **Salva**. La volta successiva che si usa lo stesso ODA, si può selezionare il profilo salvato dalla casella **Seleziona profilo**. Per ulteriori informazioni, consultare "Immissione dei valori e salvataggio di un profilo" a pagina 82.

7. Fare clic su **Avanti**. Procedura guidata dell'oggetto Business visualizza la terza finestra di dialogo, Seleziona sorgente, che è illustrato in Figura 39. La finestra di dialogo Seleziona sorgente visualizza la *gerarchia sorgente-nodo*, che è una struttura ad albero con gli oggetti di livello superiore in alto e al di sotto sono visualizzati gli oggetti secondari. Nella visualizzazione iniziale, la finestra di dialogo Seleziona sorgente di solito mostra i nodi sorgente di livello superiore.

**Importante**

Se ODA non riesce a procedere quando si fa clic su Avanti, verificare che il file di messaggio ODA specificato per la proprietà di configurazione `MessageFile` esista nella directory *`ProgramDir`*`\ODA\messages`. Per questo campione ODA, il nome predefinito di questo file di messaggio è `RomanAgent.txt`. Per ulteriori informazioni, consultare "Specifica del file di messaggio di ODA" a pagina 84.

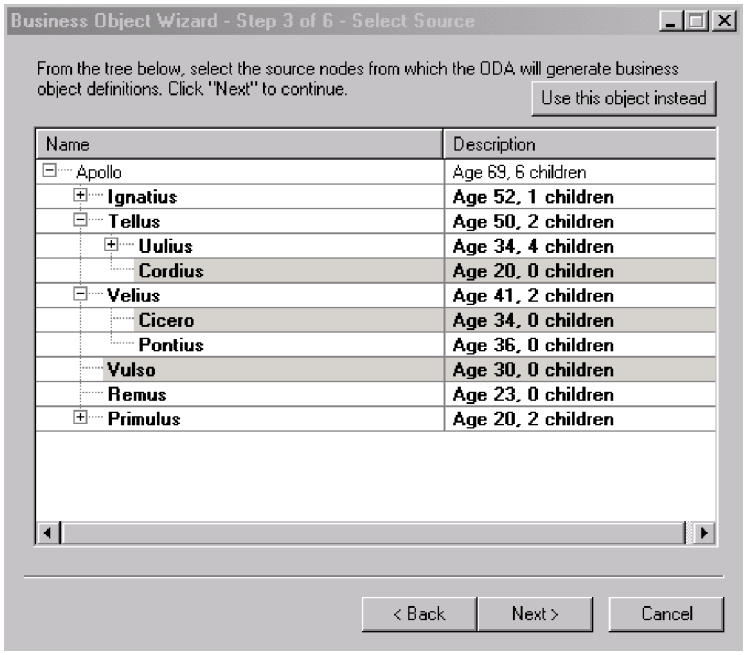



*Figura 39. Casella di dialogo Seleziona sorgente*

I nodi della gerarchia sorgente-nodi possono essere nomi di tabelle, nomi di oggetti business, schemi, o funzioni, a seconda del sorgente dati di ODA. Questo campione ODA genera nodi da oggetti all'interno di file XML denominati `RomanArmy.xml`. Figura 39 mostra il nodo sorgente di livello

superiore singolo per il generale Romano specificato per la proprietà di configurazione `generale Esercito` (consultare Figura 38 a pagina 75).

8. Selezionare gli oggetti nella gerarchia sorgente-codice per i quali si vuole che ODA generi definizione di oggetti business. Per selezionare un nodo sorgente, fare clic sul nome del nodo. Per selezionare nosi aggiuntivi, usare la chiave `Ctrl`. In Figura 40, sono stati espansi molti nodi sorgente e sono stati selezionati tre nodi sorgente (che corrispondono ad oggetti XML).

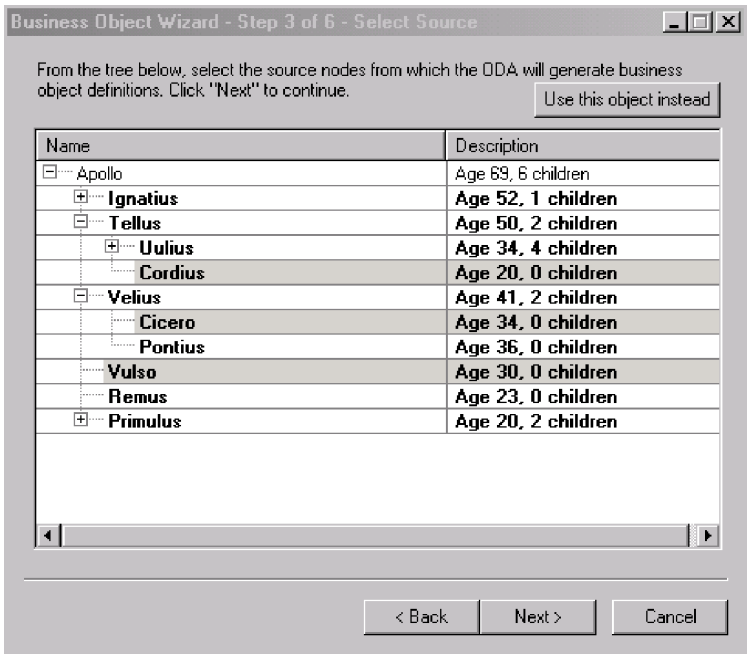



*Figura 40. Selezionare la casella di dialogo Sorgente con i nodi sorgente espansi e selezionati.*

Per espandere un nodo sorgente per visualizzare i suoi nodi secondari, effettuare quanto segue:

- Fare clic sul simbolo + a sinistra del nome del nodo.
- Fare clic con i tasto destro sul nome del nodo. Procedura guidata dell'oggetto Business visualizza il menu a comparsa mostrato in Figura 41. Per espandere il nodo selezionato, fare clic su **Richiama tutte le voci**. Procedura guidata dell'oggetto Business visualizza il livello successivo di nodi sorgente: i nodi secondari per il nodo principale espanso. Per aprire i livelli inferiori, ripetere questo processo.

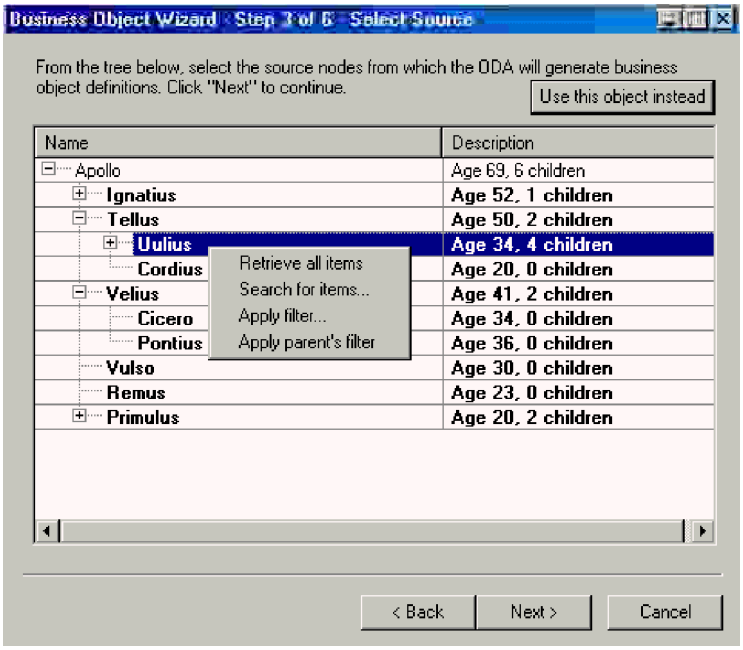



*Figura 41. Facendo clic con il tasto destro su un nodo*

**Nota:** Procedura guidata dell'oggetto Business fornisce svariati altri meccanismi per spostare i nodi della gerarchia sorgente-nodo. Per ulteriori informazioni, consultare "Spostamento tra la gerarchia sorgente-nodo" a pagina 85.

9. Dopo aver selezionato i nodi sorgente per i quali si devono generare le definizioni dell'oggetto business, fare clic su **Avanti**. Procedura guidata dell'oggetto Business visualizza la quarta casella di dialogo. Conferma sorgente, che viene visualizzata in Figura 42. Questa casella di dialogo consente di confermare le selezioni dei nodi sorgente. I nodi sorgente selezionati sono visualizzati in grassetto. In Figura 42, sono selezionati i nodi sorgente per **Cordius**, **Cicero**, e **Vulso**.

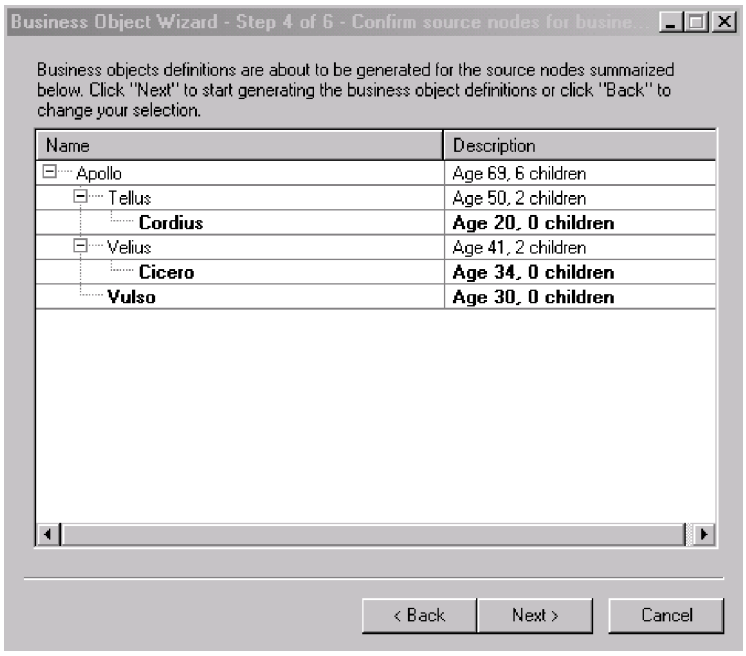



*Figura 42. Conferma degli oggetti per i quali generare definizioni di oggetti business*

Se le selezioni effettuate non sono corrette, fare clic su **Indietro** per ritornare alla precedente casella di dialogo ed apportare le modifiche necessarie.

10. Quando le selezioni sono corrette, fare clic su **Avanti**. Procedura guidata dell'oggetto Business visualizza il quinto pannello della procedura guidata. Generazione di oggetti business, visualizzata in Figura 43. Questo pannello informa che ODA sta generando le definizioni di oggetto business.

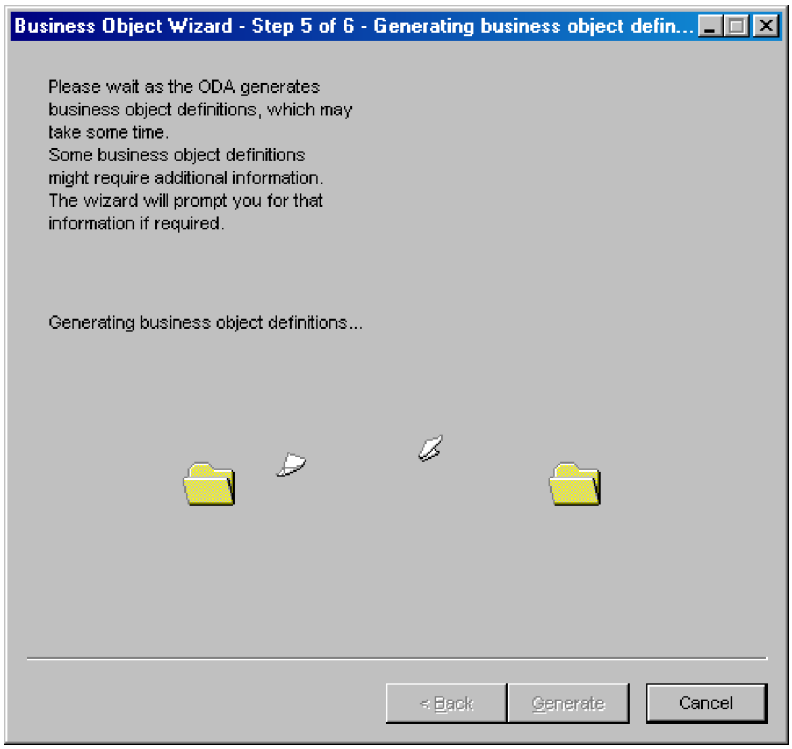



*Figura 43. Generazione delle definizioni*

Se ODA ha bisogno di informazioni aggiuntive, Procedura guidata dell'oggetto Business richiede tali informazioni visualizzando la casella di dialogo Proprietà BO. Tuttavia, questo campione ODA non richiede ulteriori informazioni. Per ulteriori informazioni sulla casella di dialogo delle proprietà BO, consultare "Come fornire informazioni aggiuntive" a pagina 89.

11. Quando ODA completa la generazione di definizioni di oggetti business, Procedura guidata dell'oggetto Business visualizza la casella di dialogo finale nella procedura guidata, Salva oggetti business, visualizzata in Figura 44. Questa casella di dialogo offre le seguenti opzioni per salvare le definizioni di oggetti business generate da ODA:
    - Memorizzare le definizioni di oggetti business in un progetto basato su ICL se Business Object Designer Express è in esecuzione da System Manager.
    - Memorizzare le definizioni di oggetti business in un file (per InterChange Server Express).
    - Aprire le definizioni di oggetti business per la modifica in Business Object Designer Express.
    - Chiudere ODA.

**Importante**

Se ODA genera una definizioni di oggetti business da un oggetto dati-sorgente che *non* identifica l'elemento chiave, la definizioni di oggetto business *non* avrà u attributo chiave. Ogni oggetto business deve avere *almeno una chiave*. Se ODA dovesse generare definizioni di oggetti business che non includono chiavi, si potrebbe scegliere l'opzione "Apri il nuovo BO in una finestra separata" invece di salvare le definizioni di oggetto business. All'interno di Business Object Designer Express, è possibile verificare che ogni definizione di oggetto business ha un attributo chiave, aggiungendone uno se non esiste. Business Object Designer Express non consente di salvare definizioni di oggetti business che non includano una chiave.

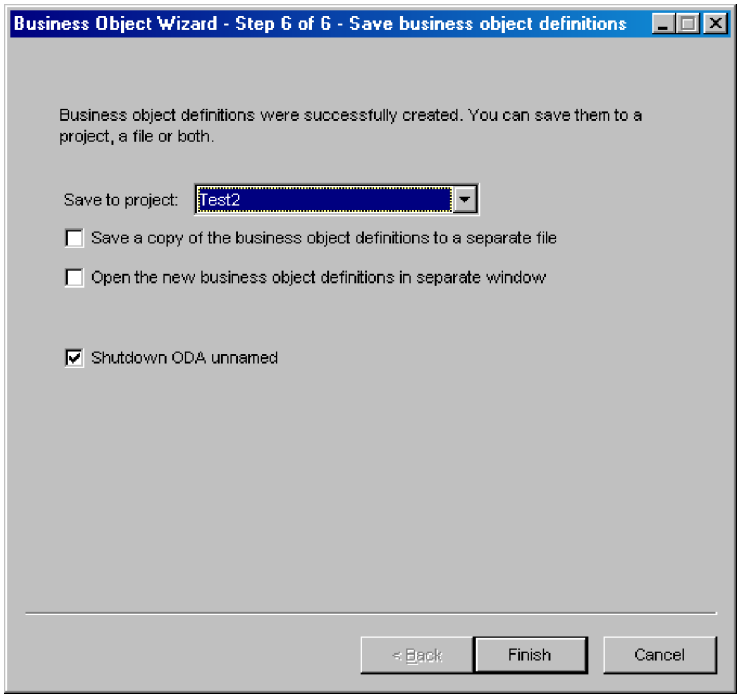



*Figura 44. Memorizzazione di definizioni di oggetti business*

Fare clic su **Fine** per salvare le definizioni di oggetti business o su **Annulla** per uscire senza salvare tali definizioni. In entrambi i casi Procedura guidata dell'oggetto Business si scollega da ODA. Questa finestra di dialogo fornisce l'opzione per far in modo che Procedura guidata dell'oggetto Business chiuda ODA dopo essersi scollegato. Se non c'è più bisogno di usare ODA, selezionare questa opzione.

Dopo aver fatto clic su **Fine**, se è stata selezionata l'opzione per salvare le definizioni di oggetti business in un file, si apre una finestra che consente di specificare il nome di questo file, dove memorizzarlo e che tipo di formato usare (file di testo o formato specifico di InterChange Server Express).

Le definizioni di oggetti business sono state create correttamente usando un Object Discovery Agent.

## Immissione dei valori e salvataggio di un profilo

E' possibile memorizzare u particolare insieme di valori di configurazione ODA in un profilo, in modo che possano essere disponibili per usi futuri di ODA. Per memorizzare il profilo:

1. Al passo 2, casella di dialogo Configurazione agente della procedura guidata di Business Object, fare clic sul tasto **Nuovo** in **Profili**.

   **Nota:** Per basare un profilo su uno già esistente, localizzare il profilo desiderato nell'elenco a scorrimento del profilo. *Non* fare clic sul tasto **Nuovo**.

2. Immettere un nome per il profilo nell'elenco **Corrente** (consultare Figura 38 a pagina 75 per un'illustrazione).

   **Nota:** Se si sta basando su uno esistente, sovrascrivere il nome del profilo esistente nell'elenco a scorrimento del profilo.

3. Immettere i valori della configurazione desiderata nella tabella **Configurazione agente**.
4. Fare clic sul tasto **Salva**.

   Procedura guidata dell'oggetto Business salva il profilo nella seguente directory:

   ```
   C:\Documents and Settings\All Users\Application Data\CrossWorlds\
      BusObjDesigner\profiles.bod
   ```

## Impostazione di Creazione log e traccia

Come parte della configurazione in ODA, bisogna impostare la creazione del log e della traccia. Le informazioni sulla creazione del log e della traccia per un ODA si specificano nella casella di dialogo Configurazione agente di Procedura guidata dell'oggetto Business. Procedura guidata dell'oggetto Business fornisce sempre lo standard delle proprietà di configurazione (illustrate in Tabella 16) per un ODA.

*Tabella 16. Proprietà di configurazione ODA standard.*

| Nome proprietà | Tipo proprietà | Descrizione |
|---|---|---|
| `TraceFileName` | Stringa | Specifica il file in cui ODA scrive le informazioni di traccia. Per ulteriori informazioni, consultare "Specifica del file di traccia e del livello di traccia" a pagina 83. |
| `TraceLevel` | Intero | Livello di traccia abilitato per ODA. Per ulteriori informazioni, consultare "Specifica del file di traccia e del livello di traccia" a pagina 83. |
| `MessageFile` | Stringa | Nome dell'errore di ODA e del file di messaggio. Usare questa proprietà per verificare o specificare un file esistente. Per ulteriori informazioni, consultare "Specifica del file di messaggio di ODA" a pagina 84. |

**Nota:** I valori predefiniti visualizzati in Business Object Designer Express per queste proprietà derivano dal file di descrizione della distribuzione di ODA. Consultare "Come lavorare con i file di messaggi di errore e di traccia" a pagina 91 per ulteriori informazioni.

Questa sezione fornisce le seguenti informazioni:

- "Specifica del file di traccia e del livello di traccia" a pagina 83
- "Specifica del file di messaggio di ODA" a pagina 84

## Specifica del file di traccia e del livello di traccia

Figura 45 mostra la casella di dialogo Configura agente in Procedura guidata dell'oggetto Business, in cui si specifica il nome del file di traccia e del livello di traccia.

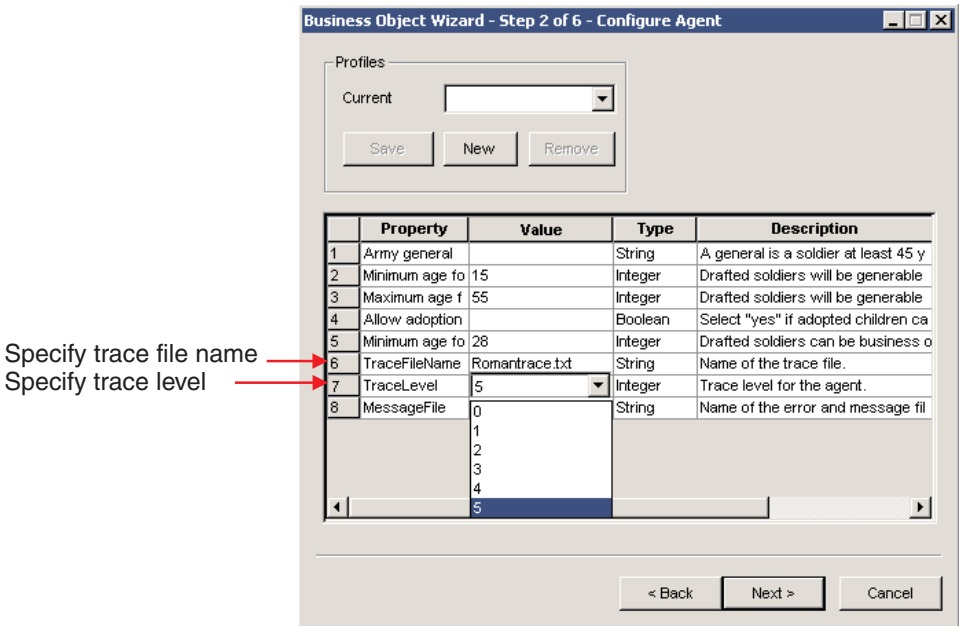



*Figura 45. Specifica delle informazioni di traccia*

**Specifica del file di traccia:** La `proprietà di configurazione TraceFileName` specifica il nome del file di traccia di ODA *file di tracciafile*. Questo file è la destinazione per *tutte* le tracce e i messaggi di errore che ODA registra. Come valore predefinito, l'esecuzione ODA assegna un nome al file di traccia secondo la convenzione di assegnazione nomi di seguito riportata:

`ODAnametrace.txt`

Nella riga precedente, *ODAname* è il nome che identifica univocamente ODA. Per ulteriori informazioni, consultare "Denominazione dell'ODA" a pagina 173. Ad esempio, se ODA si chiama `HTMLODA`, genera file di traccia che si chiamano `HTMLODAtrace.txt`.

**Nota:** Poiché ODK API fornisce un metodo per registrare *entrambi* i messaggi di traccia e di errore, un ODA ha solo un file per contenere entrambi questi tipi di messaggi. Pertanto, nonostante questo file sia chiamato file di traccia, contiene anche eventuali messaggi di errore generati da ODA.

Se il file di traccia specificato *non* esiste, ODA lo crea nella directory di esecuzione di ODA, che è la sottodirectory `ODA\srcDataName` della directory del prodotto. Se il file di traccia specificato già esiste, ODA lo mette in coda. Quando si configura ODA, è possibile usare un nome diverso per il file di traccia resettando la proprietà di `TraceFileName`.

**Impostazione del livello di traccia:** La `proprietà di configurazione TraceLevel` specifica il *livello di traccia di sistema* di ODA. Il metodo di traccia ODA invia il messaggio specifico al file di traccia quando il livello di traccia del messaggio è inferiore o uguale a questo sistema di livello di traccia. Pertanto, il livello di traccia del sistema determina il livello dei dettagli che i messaggi di traccia forniscono.

Tabella 17 elenca i livelli di traccia ed i loro comportamenti associati.

*Tabella 17. Livelli di traccia*

| Livello | Comportamento |
|---|---|
| 0 | Scrive messaggi di errore per file di traccia specificati. |
| 1 | Traccia in qualsiasi momento venga immesso un metodo—utile per i messaggi di stato e per le informazioni chiave di ciascuna definizione di oggetto business. |
| 2 | Traccia le proprietà dell'agente e i valori ricevuti. |
| 3 | • Traccia i nomi degli oggetti business.<br>• Traccia le proprietà degli oggetti business e i valori ricevuti. |
| 4 | • Traccia la creazione di tutti i processi.<br>• Traccia un messaggio in qualsiasi momento venga immesso o si trovi in uscita. |
| 5 | • Indica l'inizializzazione di Object Discovery Agent e registra i valori richiamati per tutte le proprietà di Object Discovery Agent.<br>• Traccia uno stato dettagliato di ciascun processo creato da Object Discovery Agent.<br>• Traccia il dump della definizione dell'oggetto business. |

Per informazioni su come generare i messaggi di traccia all'interno di ODA, consultare "Gestione dei messaggi di errore e di traccia" a pagina 163.

## Specifica del file di messaggio di ODA

La `proprietà di configurazione` `MessageFile` specifica il nome del *file di messaggio* di ODA. Un ODA può memorizzare i suoi messaggi di traccia e di errore in questo file di messaggi ODA. Può richiamare questi messaggi dal numero si messaggio, invece di creare un testo di messaggio da sé. Isolando i messaggi nei file di messaggio si fornisce un modo semplice ai messaggi ODA di essere tradotti in lingue con locali differenti che ODA può mettere in esecuzione.

Come valore predefinito, ODA esegue questo file di messaggio secondo le seguenti convenzioni di denominazione:

*ODAname*`Agent.txt`

Nella riga precedente, *ODAname* è il nome che identifica univocamente ODA. Per ulteriori informazioni, consultare "Denominazione dell'ODA" a pagina 173. Ad esempio, se ODA si chiama `HTMLODA`, il valore delle proprietà di `MessageFile` si chiama `HTMLODAAgent.txt`. Il file di messaggio deve risiedere nella seguente directory di file di messaggio:

*ProductDir*`\ODA\messages`

**Importante**

Se il file di messaggio specificato *non* esiste o se non esiste nella directory del file di messaggio, ODA genera un'eccezione di esecuzione. Bisogna assicurarsi che il file di messaggio (che specifica `MessageFile`) esista prima di continuare con l'esecuzione del file ODA.

Se ODA usa un diverso file di messaggio, impostare la proprietà `MessageFile` per specificare un nome diverso per il file di traccia.

Se si sta usando un file non-US English Procedura guidata dell'oggetto Business locale automaticamente cerca un file di messaggio ODA che includa il nome della locale nel nome file nel seguente modo:

*ODAname*Agent_*locale*.txt

dove *locale* ha il formato "*ll_TT*", con *ll* come nome della lingua a due caratteri (in minuscolo) e *TT* come nome della nazione a due caratteri (in maiuscolo). Ad esempio, se l'ODA che si chiama HTMLODA ha i suoi file di messaggio localizzati nella locale giapponese, il suo file di messaggio e, its si chiamerebbe:

HTMLODAAgent_ja_JP.txt

**Nota:** Quando ci si collega ad una locale non-US English, *non* bisogna specificare il nome non-US English nella proprietà MessageFile. Ad esempio, se si sta usando HTML ODA, si imposta MessageFile sul nome file US English (HTMLODAAgent.txt). Se si è collegati su una locale giapponese, Procedura guidata dell'oggetto Business locaizza il file di messaggio corretto per la locale giapponese: HTMLODAAgent_ja_JP.txt.

Se si creano istanze multiple dello script ODA o del file batch file e si fornisce un nome unico per ciascun ODA rappresentato, è possibile avere un file di messaggio per *ogni* istanza ODA. Per ulteriori informazioni, consultare "Utilizzo di più ODA contemporaneamente" a pagina 90.

## Spostamento tra la gerarchia sorgente-nodo

La casella di dialogo Seleziona sorgente di Business nella Procedura guidata dell'oggetto Business fornisce i seguenti meccanismi per consentire lo spostament tra nodi della gerarchia nodo-sorgente:

- "Limitazione della visualizzazione dei nodi secondari"
- "Specifica del percorso dell'oggetto" a pagina 87
- "Associazione di un file del sistema operativo" a pagina 88

### Limitazione della visualizzazione dei nodi secondari

I modi per espandere un certo nodo sorgente nel passo 8 a pagina 77 descrive come visualizzare *tutti* i nodi secondari di un nodo espandibile. Per limitare gli oggetti visualizzati, è possibile usare una delle seguenti voci di menu quando si fa clic con il tasto destro sul nome del nodo (consultare Figura 41 a pagina 78):

- **Applicare filtro**
- **Cerca per voce**

**Uso de filtro:** La voce di menu **Applica filtro** consente di specificare un *filtro*, il che può limitare quali dei nodi sorgente selezionati al momento, aprire. Quando si fa clic su questa voce di menu, Procedura guidata dell'oggetto Business visualizza la La casella di dialogo Applica filtro al nodo, come illustrato in Figura 46 a pagina 86.

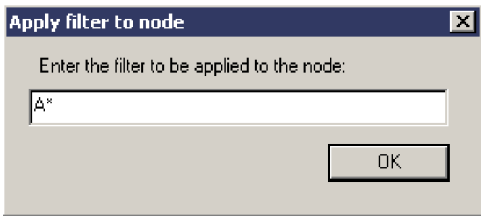



*Figura 46. Specifica del filtro per limitare i risultati*

Nel testo del filtro è possibile usare l'asterisco (*) come carattere jolly (per rappresentare zero o più caratteri corrispondenti). Il carattere jolly può apparire in qualunque posizione e in quante posizioni si vuole. Ad esempio, `SAP*`, `*SAP`, `*SAP*`, o `*S*AP*`.

Quando si fa clic su **OK**, Procedura guidata dell'oggetto Business ricerca i nodi secondari - richiamati al momento - dei nodi principali i cui nomi corrispondono al testo del filtro. Quando questo nodo principale si espande, visualizza solo quei nodi secondari i cui nomi corrispondono a questo testo.

**Importante:** Quando Procedura guidata dell'oggetto Business riceve un filtro, ricerca i corrispondenti nodi secondari dei nodi principali nel nodo sorgente richiamato al momento; cioè, *non* ricerca il sorgente dati per i odi secondari corrispondenti. Per far in modo che Procedura guidata dell'oggetto Business ricerchi il sorgente dei dati, si può specificare il modello di ricerca. Per ulteriori informazioni, consultare "Specifica dei modelli di ricerca" a pagina 87.

Ad esempio, nel campione ODA Romano, il nodo `Uulius` ha quattro nodi seondari: `Ares`, `Cronus`, `Atlas` e `Metis`. Se si applica il filtro in Figura 46 al nodo `Uulius` ("A*"), Procedura guidata dell'oggetto Business visualizza questo nodo, come mostrato in Figura 47 quando si espande il nodo.

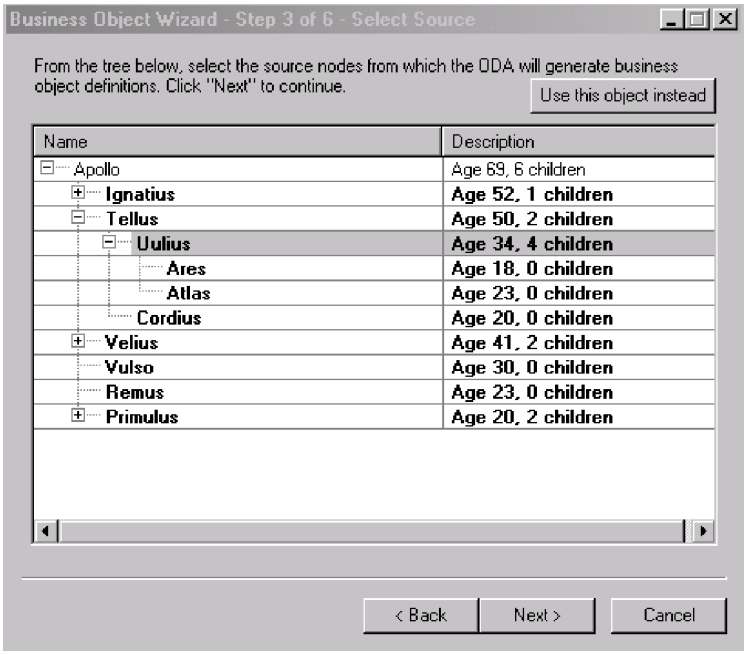



*Figura 47. Nodo con filtro dopo l'espansione*

Se si specifica un filtro all'inizio di un nodo e poi si espande il nodo, si può applicare lo stesso filtro agli oggetti secondari facendo clic con il tasto destro sul nodo e facendo clic su **Applica filtro del principale**. Se si fa clic sulla voce di menu **Richiama tutte le voci**, il filtro del nodo principale viene applicato a tutti gli elementi.

**Specifica dei modelli di ricerca:** La voce di menu **Ricerca per voci** consente di specificare un *modello di ricerca*, che può limitare i nodi sorgente che Procedura guidata dell'oggetto Business seleziona dal sorgente dati. . Quando si fa clic su **Ricerca per voci**, Procedura guidata dell'oggetto Business visualizza la finestra di dialogo Immetti modello di ricerca Figura 48 illustra questa casella di dialogo.

**Nota:** Un ODA deve supportare la funzione del modello di ricerca per abilitare la voce del menu **Ricerca per voci**. Se questa voce di menu non è disponibile, ODA *non* supporta modelli di ricerca.

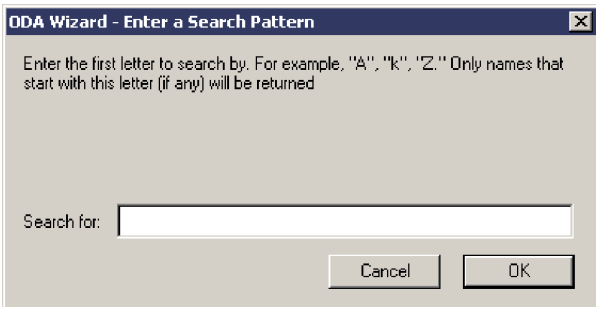



*Figura 48. Specifica del modello di ricerca per limitare i risultati richiamati*

La casella di dialogo Immetti un modello di ricerca fornisce una descrizione dei criteri di ricerca che può usare il modello di ricerca di cui si dispone. In Figura 48, il testo di questa casella di dialogo specifica che il modello di ricerca può consistere di una lettera. ODA fornisce una descrizione personalizzata dei modelli di ricerca. Accertarsi che il modello di ricerca immesso segua ill criterio di ricerca descritto. Altrimenti, ODA restituisce un'eccezione.

Quando si fa clic su **OK**, Procedura guidata dell'oggetto Business ricerca i dati della sorgente dati per i nodi secondari dei nodi principali i cui nomi corrispondono ai modelli di ricerca. Quando si espande questo nodo principale, vengono visualizzati solo quei nodi secondari i cui nomi corrispondono almodello.

**Importante:** Quando Procedura guidata dell'oggetto Business riceve un modello di ricerca, ricerca i corrispondenti nodi secondari dei nodi principali nella sorgente dati; cioè richiama un nuovo nodo tree dalla sorgente dati. *Non* ricerca semplicemente il nodo tree richiamato al momento per la corrispondenza dei nodi secondari. Per far in modo che Procedura guidata dell'oggetto Businessricerchi il nodo tree richiamato al momento, è possibile specificare un filtro. Per ulteriori informazioni, consultare "Uso de filtro" a pagina 85.

## Specifica del percorso dell'oggetto

Invece di spostarsi attraverso la gerarchia sorgente-nodo, è possibile specificare un percorso esatto per l'oggetto desiderato. Per far ciò, fare clic su **Sostituisci con questo oggetto**, nella parte superiore destra della finestra di dialogo per la selezione del sorgente. Procedura guidata dell'oggetto Business visualizza la

Finestra di dialogo Percorso oggetto, visualizzata in Figura 49, in cui si indica il percorso.

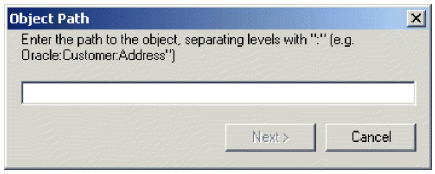



*Figura 49. Specifica del percorso di un oggetto.*

Si specifica il percorso dell'oggetto come il percorso completo del nodo sorgente (dal nodo principale di livello superiore al nodo desiderato). I nomi dei nodi all'interno di questo percorso sono separati dai due punti (:).

## Associazione di un file del sistema operativo

Per associare un file di sistema operativo con il corrente nodo della gerarchia sorgente-nodo, fare clic con il tasto destro su un nodo e fare clic su **File associati** (consultare Figura 50). Quando si associa un file ad un nodo sorgente, ODA usa il file come sorgente per quei dati del nodo sorgente (invece di usare la sorgente dati ODA).

**Nota:** ODA deve supportare la funzione dei file associati per la voce del menu **File associati** da abilitare. Se questa voce del menu non è disponibile, ODA *non* supporta l'associazione dei file con il nodo sorgente corrente.

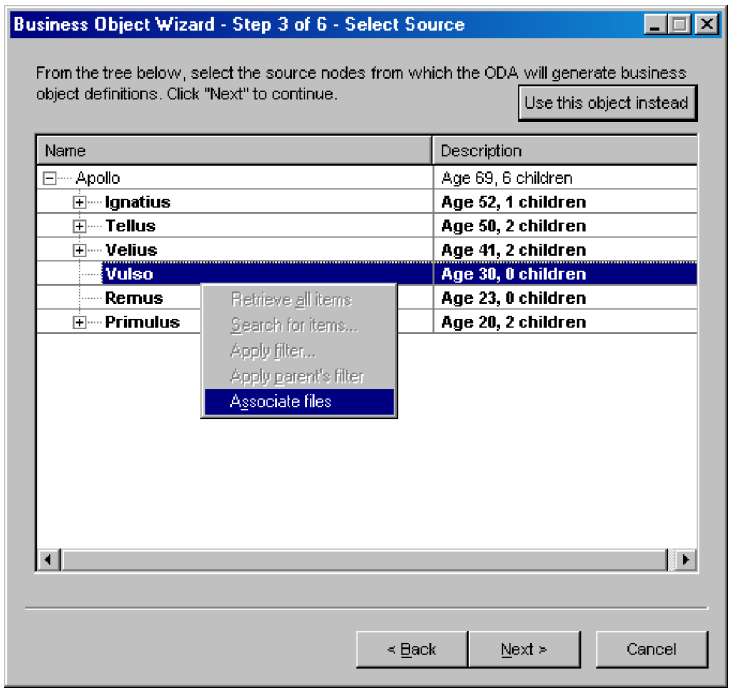



*Figura 50. Associazione di un file con un nodo sorgente*

Quando si fa clic sulla voce di menu **File associati**, Procedura guidata dell'oggetto Business visualizza la finestra Apri visualizzata in Figura 51. Da questa finestra, è possibile visionare la struttura del file e scegliere il file associato al nodo corrente.

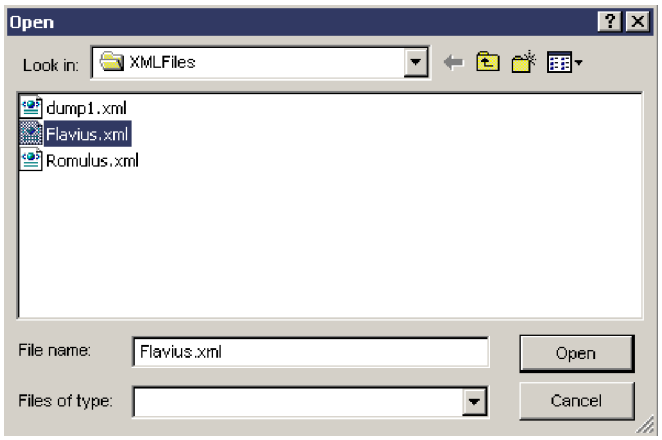



*Figura 51. Apri la finestra per selezionare il file associato*

Dopo aver selezionato il file da associare al nodo sorgente, fare clic su **Apri**. Quando Procedura guidata dell'oggetto Business restituisce il controllo alla casella di dialogo Seleziona sorgente, il file selezionato viene visualizzato sotto il nodo sorgente che è associato, come mostra Figura 52.

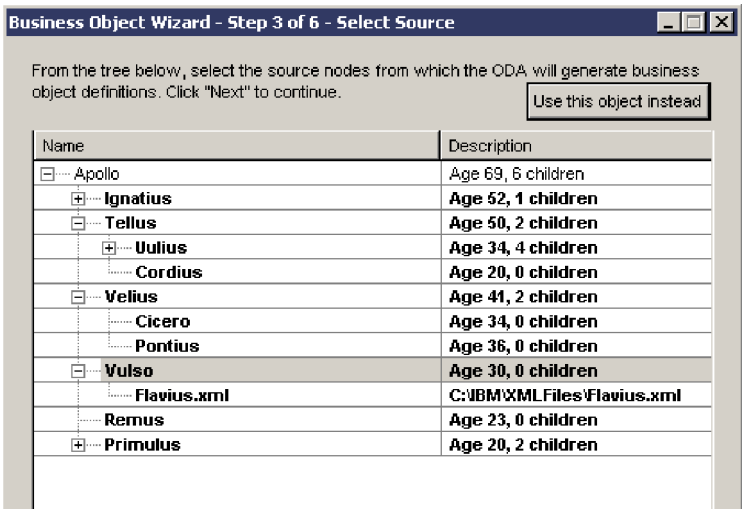



*Figura 52. File associati con un nodo sorgente*

## Come fornire informazioni aggiuntive

Nel Passo 5, Generazione di un oggetto business, se ODA ha bisogno di informazioni aggiuntive, si apre la finestra di dialogo delle proprietà BO, come mostrato in Figura 53..

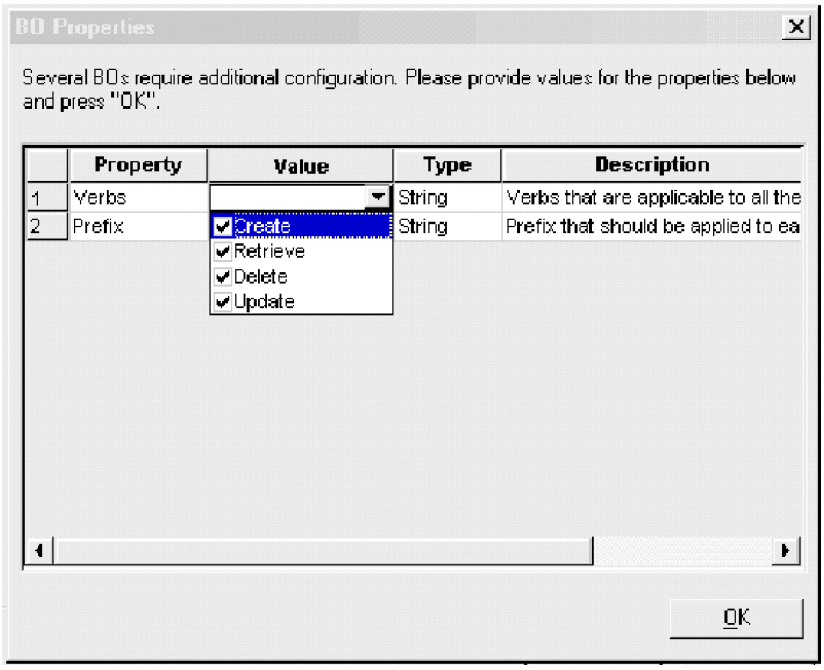



*Figura 53. Come fornire informazioni aggiuntive.*

**Nota:** Se una cella nella finestra di dialogo Proprietà BO ha valori multipli, quando si apre per la prima volta sembra vuota. Fare clic sulla cella per ottenere un elenco dei suoi valori.

Dopo aver fornito le informazioni richieste nella finestra di dialogo Proprietà BO, fare clic su **OK**. ODA continua con la sua generazione di definizioni di oggetti business.

## Utilizzo di più ODA contemporaneamente

E' possibile eseguire istanze multiple di ODA sia sulla macchina host locale che su una macchina host remota. Ogni istanza va in esecuzione su una porta unica. E' possibile specificare il numero della porta quando si avvia ciascun ODA dall'interno di Procedura guidata dell'oggetto Business.

Per eseguire simultaneamente Object Discovery Agents multipli in Business Object Designer Express, fare quanto segue:

1. Avviare ogni Object Discovery Agent eseguendo i suoi file `start_`*`ODAname`*`.bat` or `start_`*`ODAname`*`.sh`.
2. Open Business Object Designer Express.
3. Fare clic su **File > Nuovo uso di ODA**.

   La prima finestra di dialogo nella procedura guidata di Business Objec, Seleziona agente, apre (consultare Figura 36 a pagina 74).
4. Fare clic sul pulsante **Trova agenti** per visualizzare gli ODA al momento in esecuzione nell'elenco **Agenti localizzati**. E' anche possibile trovare ODA specificando il suo nome host e il numero della porta.
5. Selezionare il primo ODA dall'elenco visualizzato. La selezione effettuata viene elencata come **Nome agente**.
6. Fare clic di nuovo su **File > Nuovo uso di ODA**.

7. Fare clic sul tasto **Trova agenti** per visualizzare gli ODA al momento in esecuzione nell'elenc **Agenti rovati**, oppure trovare gli ODA che usano il loro nome host e il numero della porta.
8. Selezionare il secondo ODA dall'elenco visualizzato.
9. Procedere con la configurazione di ciascun ODA come descritto nel passo 4 di "Uso di ODA per creare definizioni di oggetti business" a pagina 73.

Se si creano istanze multiple dello script ODA o del file batch e si fornisce un nome unico per ciascun ODA rappresentato, è possibile un file di messaggio per *ogni* istanza ODA. In alternativa, si possono avere ODA con nomi differenti che usano lo stesso file di messaggio. Ci sono due modi per specificare un file di messaggio valido:

- Se ci cambia il nome di ODA e *non* si crea un file di messaggio per esso, bisogna cambiare il nome del file di messaggio in Procedura guidata dell'oggetto Business come parte della configurazione ODA. Procedura guidata dell'oggetto Business fornisce un nome per il file di messaggio ma non crea realmente il file. Se il file visualizzato come parte della configurazione ODA non esiste, cambiare il valore in modo che punti ad un file esistente.
- E' possibile copiare il file di messaggio esistente per un ODA specifico e modificarlo nel modo richiesto. Procedura guidata dell'oggetto Business presume che venga assegnato a ciascun file il nome secondo le convenzioni di assegnazione nomi. Ad esempio, se la variabile `AGENTNAME` (all'interno dello script di inizializzazione ODA) specifica `HTMLODA`, il programma presume che il nome del file di messaggio associato sia `HTMLODAAgent.txt`. Pertanto, quando Procedura guidata dell'oggetto Business visualizza il nome file per verifica come parte della configurazione ODA, il nome file è basato sul nome ODA. Verificare che il file di messaggi predefinito abbia il nome corretto e, se necessario, correggerlo.

## Come lavorare con i file di messaggi di errore e di traccia

Come impostazione predefinita, i file di messaggi di traccia e di errore si trovano nella sottodirectory \ODA\messages, sotto la directory del prodotto. Questi file usano le seuenti convenzioni di assegnazione dei nomi:

*AgentName*`Agent.txt`

Dove *AgentName* è l'ODA a cui appartiene il file.

Se si creano istanze multiple dello script ODA o del file batch e si fornisce un nome unico per ciascun ODA rappresentato, quelle istanze possono usare lo stesso file di messaggio. In alternativa, è possibile specificare file di messaggi differenti per ciascuna istanza ODA specificando i nomi i nomi dei file in odk.dd.*AgentName*, che è il file *descrittore della distribuzione* di ODA installato con ODA.

Gli adattatori WBI contengono un descrittore principale della distribuzione ODA in \ODA\odk.dd.xml, che specifica i valori predefiniti del file di messaggio e di traccia. Per specificare file di messaggio differenti per differenti istanze ODA, è possibile copiare il file principale descrittore della distribuzione ODA nella directory di ODA e ridenominarlo oda.dd.xml (ad esempio, \ODA\XMLODA\oda.dd.xml). Modificare questo file in modo da cambiare di conseguenza i valori di messagefile, tracefile e tracelevel. Il descrittore principale della distribuzione ODA ha i seguenti formati e valori predefiniti:

```
<odk>
   <startup>
      <messagefile usestandard="true"></messagefile>
   </startup>
```

```
   <diagnostics>
      <tracefile usestandard="true"></tracefile>
      <tracelevel canoverride="true">1</tracelevel>
   </diagnostics>
</odk>
```

Business Object Designer Express presume che si assegni un nome a ciascun file coerentemente con le convenzioni di assegnazione nomi. Ad esempio, se la variabile AGENTNAME specifica XMLODA1, il programma presume che il nome del file di messaggio associato sia XMLODA1Agent.txt. Pertanto, quando Business Object Designer Express fornisce il nome file per verifica come parte della configurazione ODA, il nome file è basato sul nome ODA. Verificare che il file del messaggio predefinito abbia il nome corretto e, se necessario, correggerlo.

**Importante:** L'indicazione non corretta del nome del file di messaggio quando si configura ODA comporta la sua eecuzione senza messaggi. Per ulteriori informazioni sul nome file di messaggi, consultare "Specifica del file di messaggio di ODA" a pagina 84.

Durante la configurazione, specificare:

- Il nome del file in cui ODA scrive le informazioni di errore e di traccia.
- Il livello di traccia, che varia da 0 a 5. Consultare Tabella 17 a pagina 84 per una descrizione di questi valori.

# Parte 2. Sviluppo di un Object Discovery Agent

# Capitolo 5. Sviluppo di in Object Discovery Agent

In questo capitolo vengono fornite le informazioni relative all'utilizzo delle classi definite nelle API ODK (Object Discovery Agent Development Kit) per sviluppare un ODA (Object Discovery Agent). Un ODA opera con la Procedura guidata dell'oggetto Business di Business Object Designer Express per sviluppare definizioni di oggetti business per uno specifico connettore o gestore dati che opera con un'applicazione, un database oppure un file system specifico.

Gli argomenti principali di questo capitolo sono:



## Esecuzione di un ODA

Durante il runtime, l'esecuzione di un ODA coinvolge i seguenti componenti:

- Business Object Designer Express fornisce un'interfaccia grafica sotto forma di procedura guidata per interagire con l'ODA: Procedura guidata dell'oggetto Business. La procedura guidata visualizzerà una serie di caselle di dialogo per ottenere informazioni necessarie all'ODA per generare il contenuto.
- Il *Il runtime ODA* è il componente intermedio tra la Procedura guidata dell'oggetto Business e l'ODA. Utilizza le classi dell'API ODK e la infrastruttura per comunicare con l'ODA. Il runtime ODA viene avviato con lo script di avvio ODA.
- Il *ODA* è il componente che "scopre" i nodi origine nell'origine dati e genera il contenuto. L'ODA riceve informazioni nelle caselle di dialogo della Procedura guidata dell'oggetto Business dal runtime ODA. Invia quindi informazioni (quali, ad esempio, il contenuto generato) al runtime ODA, che lo invia a Procedura guidata dell'oggetto Business.

Figura 54 illustra i componenti dell'architettura del runtime ODA.

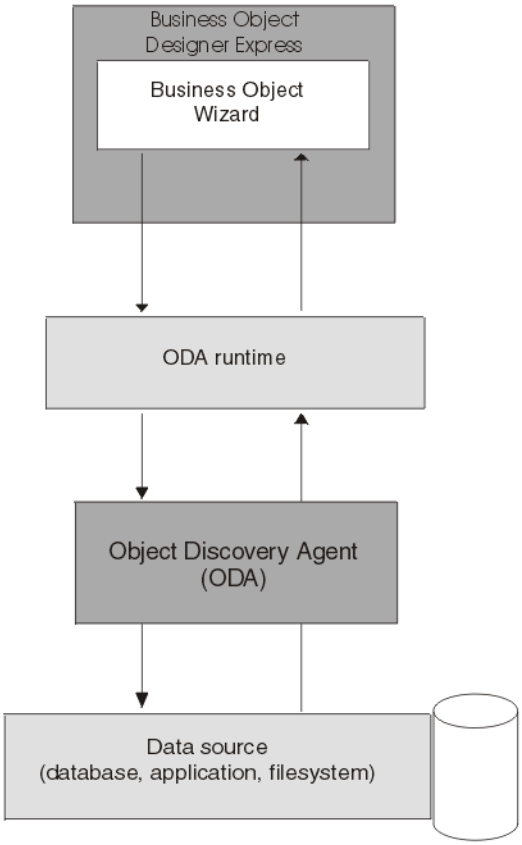



*Figura 54. Architettura dell'Object Discovery Agent*

Per generare le definizioni di oggetti business, l'ODA deve effettuare le seguenti operazioni:

1. Ottenere valori per le proprietà di configurazione ODA (quali, ad esempio, il nome utente e il tipo database) che ODA richiede per collegarsi all'origine dati (quale un'applicazione, un database o un file system).
2. Utilizzare le proprietà di configurazione per collegarsi all'origine dati.
3. Ottenere l'elenco dei nodi origine per cui sono stati create le definizioni di oggetti business.
4. Rilevare i requisiti per l'entità origine dati che è sottostante al nodo di origine (come definito da un'applicazione, una tabella di database, un file system oppure un DTD).
5. Generare definizioni di oggetti business che soddisfino i requisiti del sistema WebSphere Business Integration Server Express e il componente che elabora l'oggetto business, e che restituiscano definizioni di oggetti business.

**Nota:** Oltre alle definizioni di oggetti business, un ODA può anche generare file come contenuto. Per ulteriori informazioni, consultare "Creazione di contenuto" a pagina 100.

Tabella 18 effettua un riepilogo dei passi compiuti nell'esecuzione di un ODA e i passi nella Procedura guidata dell'oggetto Business che li avvia.

*Tabella 18. Esecuzione di Object Discovery Agent*

| Attività | | Passi in rrocedura guidata Business Object | Per ulteriori informazioni |
|---|---|---|---|
| 1. | Selezionare l'ODA desiderata per iniziare | Passo 1: Selezione di un agente | "Selezione di un ODA" |
| 2. | Ottenere le proprietà di configurazione ODA, comprese quelle che descrivono l'origine dati da aprire. | Passo 2: Configurazione di un agente | "Ottenimento delle proprietà di configurazione ODA" |
| 3. | Ottenere i dati di origine per i quali generare il contenuto ODA. | Passo 3: Selezione di un'origine | "Selezione e conferma dei dati di origine" a pagina 99 |
| 4. | Confermare i dati di origine selezionati. | Passo 4: Conferma dei nodi di origine | "Selezione e conferma dei dati di origine" a pagina 99 |
| 5. | Generare le definizioni di oggetti business. | Passo 5: Creazione di oggetti business | "Creazione di contenuto" a pagina 100 |
| | | Proprietà degli oggetti business | "Ottenimento delle proprietà di oggetti business" a pagina 101 |
| 6. | Salvare le definizioni di oggetti business. | Passo 6: Salvataggio degli oggetti business | "Salvataggio del contenuto" a pagina 103 |

## Selezione di un ODA

Quando si seleziona **File > Nuovo utilizzando ODA**, Business Object Designer Express richiama la procedura guidata Business Object per eseguire l'ODA. Passo 1 di la Procedura guidata dell'oggetto Business visualizza la casella di dialogo Seleziona agente, che fornisce accesso grafico a tutti gli ODA disponibili. Da questa casella di dialogo gli utenti selezionano l'ODA da eseguire.

La Procedura guidata dell'oggetto Business si collega a questa ODA effettuando le seguenti operazioni:

- Crea istanze per un oggetto ODA, che è un oggetto della *classe ODA*. La classe ODA è l'estensione della classe di base ODA, `ODKAgentBase2`. Definisce il comportamento dell'ODA.
- Ottiene un identificativo di file per l'oggetto ODA che può essere utilizzato per accedere a questo oggetto quando avviato.

**Nota:** Un ODA deve essere già avviato affinché la Procedura guidata dell'oggetto Business lo elenchi come un ODA disponibile all'esecuzione. Per ulteriori informazioni, consultare "Prima di usare un ODA" a pagina 71.

Per ulteriori informazioni su come creare la classe ODA, consultare "Estensione della classe di base ODA" a pagina 109.

## Ottenimento delle proprietà di configurazione ODA

Passo 2 di la Procedura guidata dell'oggetto Business visualizza la casella di dialogo Configura agente, che visualizza le proprietà di configurazione di ODA. Le *Proprietà di configurazione* sono queste proprietà che occorrono a ODA per poter iniziare l'esecuzione. L'API ODK rappresenta una proprietà di configurazione come un oggetto di proprietà dell'agente (`AgentProperty`). In questo passo, la procedura guidata visualizza le proprietà di configurazione, consente di aggiornarle e quindi scrive le proprietà inizializzate dall'utente nella memoria runtime di ODA.

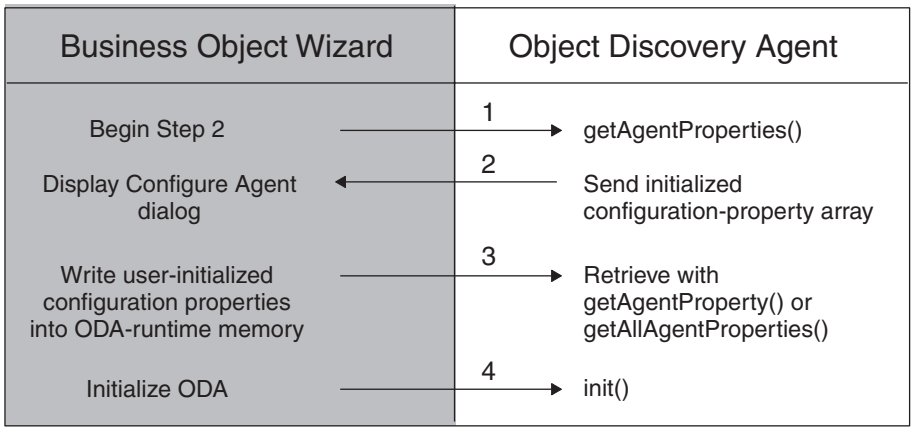



*Figura 55. Configura agente (passo 2) della procedura guidata Business Object*

Come viene illustrato in Figura 55, la Procedura guidata dell'oggetto Business effettua le seguenti azioni:

1. Ottiene le proprietà di configurazione dall'ODA selezionato e li visualizza nella casella di dialogo Configura agente.

   Per ottenere le proprietà di configurazione dall'ODA, la procedura guida richiama il `getAgentProperties()`, definito nella classe di base ODA, `ODKAgentBase2`. Questo metodo è un metodo astratto che lo sviluppatore ODA deve implementare come parte della classe ODA. Vengono restituite le proprietà di configurazione ODA alla Procedura guidata dell'oggetto Business come un array degli oggetti `AgentProperty`. Tali proprietà di configurazione possono includere nomi, tipi, tutti i valori validi, le descrizioni, le limitazioni di immissione e tutti i valori predefiniti.

   Oltre alle proprietà di configurazione fornite da `getAgentProperties()`, la Procedura guidata dell'oggetto Business fornisce sempre un insieme di proprietà di configurazione comuni a tutte le ODA:

   - `MessageFile`
   - `TraceLevel`
   - `TraceFileName`

   Per ulteriori informazioni, consultare "Ottenimento delle proprietà di configurazione" a pagina 111.
2. Dalla casella di dialogo Configura agente, accetta i valori immessi o le modifiche per le proprietà di configurazione. Ls procedura guidata invia le proprietà di configurazione inizializzate dall'utente all'ODA.

   La Procedura guidata dell'oggetto Business salva tali proprietà nella memoria runtime di ODA. All'interno di ODA, è possibile accedere a queste proprietà attraverso un'istanza della classe `ODKUtility`, che fornisce i metodi `getAgentProperty()` e `getAllAgentProperties()` a questo scopo.
3. Inizializza imetadati ODA, che forniscono informazioni su ODA e le relative capacità.

   Dopo aver richiamato `getAgentProperties()`, la Procedura guidata dell'oggetto Business richiama il metodo `getMetaData()` della classe di base ODA, `ODKAgentBase2`. Questo metodo è un metodo astratto che lo sviluppatore ODA deve implementare come parte della classe ODA. Restituisce un oggetto `AgentMetaData` che contiene i metadati ODA.
4. Inizializza l'ODA basato su le proprietà di avvio inizializzate dall'utente.

Per inizializzare l'ODA, la procedura guidata richiama il metodo `init()` della classe di base dell'ODA, `ODKAgentBase2`. Questo metodo è un metodo astratto che lo sviluppatore ODA deve implementare come parte della classe ODA. Vengono eseguite le attività di inizializzazione, quali l'allocazione di risorse e la creazione di una connessione all'origine di dati.

In questo capitolo vengono fornite le seguenti informazioni su come implementare i metodi coinvolti nell'inizializzazione di un ODA:

| Metodo di inizializzazione | Per ulteriori informazioni |
|---|---|
| `getAgentProperties()` | "Ottenimento delle proprietà di configurazione" a pagina 111 |
| `getMetaData()` | "Inizializzazione di metadati ODA" a pagina 113 |
| `init()` | "Inizializzazione dell'avvio di ODA" a pagina 115 |

## Selezione e conferma dei dati di origine

Passo 3 di la Procedura guidata dell'oggetto Business visualizza la casella di dialogo Seleziona origine, che visualizza i *nodi origine* dell'origine di dati. I nodi di origine vengono disposti in una gerarchia di nodi origine. Ciascun nodo origine è il nome di un oggetto "scoperto" da ODA nell'origine di dati. Può essere espansa per visualizzare altri nodi secondari oppure selezionato per le generazione di contenuto. Gli utenti possono espandere questa gerarchia di nodi origine per selezionare oggetti nell'origine di dati per la conversione di contenuto. Per informazioni, consultare "Spostamento tra la gerarchia sorgente-nodo" a pagina 85..

Nel passo 3, la procedura guidata effettua le seguenti azioni:

1. Ottiene la gerarchia di nodi origine dall'ODA selezionato e la visualizza a livello superiore nella casella di dialogo Seleziona origine.

   Per ottenere la gerarchia del nodo di origine, la procedura guidata richiama il `metodo getTreeNodes()` dell'interfaccia `IGeneratesBoDefs`. Lo sviluppatore ODA deve implementare questo metodo come parte di un'implementazione della classe ODA dell'interfaccia `IGeneratesBoDefs`. Effettua la ricerca dell'origine di dati per "scoprire" i nodi origine e restituire questi nodi alla Procedura guidata dell'oggetto Business come un array degli oggetti `TreeNode`. Quando gli utenti espandono un nodo per la prima volta, la procedura guidata richiama `getTreeNodes()` per visualizzare questo particolare livello nella gerarchia di nodi origine. Gli utenti possono attraversare questa gerarchia per selezionare il livello di dettagli. Per ulteriori informazioni, consultare "Spostamento tra la gerarchia sorgente-nodo" a pagina 85.
2. dalla casella di dialogo Seleziona origine, tenere traccia dei nomi dei nodi origine nella gerarchia selezionata per la generazione di contenuto. La procedura guidata genera un array che contiene i nomi dei nodi origine selezionati.

Passo 4 di la Procedura guidata dell'oggetto Business visualizza la casella di dialogo Conferma nodi origine, che visualizza i nodi origine selezionati. Gli utenti possono confermare le selezioni oppure tornare alla casella di dialogo Seleziona origine per selezionare nuovamente i nodi origine. Quando si fa clic sul pulsante **Avanti**, la procedura guidata inizia la generazione di contenuto.

Per informazioni su come implementare il metodo `getTreeNodes()`, consultare "Creazione dei nodi di origine" a pagina 122.

## Creazione di contenuto

È possibile scrivere un ODA per generare uno o entrambi i *tipi di contenuto* riportati in Tabella 19. Il tipo di contenuti determina la struttura dei dati generati da ODA. Perché un ODA supporti un contenuto particolare, è necessario implementare l'*interfaccia di generazione contenuto* per l'ODA. Tabella 19 elenca i tipi di contenuto che un ODA può supportare così come l'interfaccia di generazione associata che deve essere implementata dall'ODA.

*Tabella 19. Tipi di contenuto per un ODA*

| Tipo di contenuto | Descrizione | Interfaccia di generazione contenuto |
|---|---|---|
| Definizioni oggetto business | L'ODA genera definizioni di oggetti business per rappresentare gli oggetti nell'origine di dati. | `IGeneratesBoDefs` |
| File binari | L'ODA genera gli oggetti di file per conservare informazioni sul contenuto generato. | `IGeneratesBinFiles` |

**Nota:** Con questo release, un ODA *deve* supportare la generazione di definizioni di oggetti business come contenuto. Pertanto, *deve* implementare l'interfaccia `IGeneratesBoDefs`. Inoltre, ODA può supportare la generazione di file come contenuto implementando l'interfaccia `IGeneratesBinFiles`.

Dopo aver selezionato e confermato i nodi origine, la Procedura guidata dell'oggetto Business immette i passi 5 della generazione di contenuto. Visualizza il pannello Creazione di Business Objects e passa l'array dei nodi origine selezionati dall'utente (dal passo 4) all'ODA richiamando il metodo di generazione contenuto per le definizioni di oggetti business, `generateBoDefs()`. Questo metodo genera le definizioni di oggetti business corrispondenti per i nodi origine selezionati. Poiché un ODA *deve* supportare la generazione di definizioni di oggetti business nel protocollo di contenuto a richiesta, la Procedura guidata dell'oggetto Business richiama *sempre* il metodo `generateBoDefs()`. Pertanto, lo sviluppatore ODA deve implementare questo metodo come parte di un'implementazione ODA dell'interfaccia IGeneratesBoDefs.

La generazione di contenuto di file da parte di ODA, dipende dall'implementazione dell'interfaccia `IGeneratesBinFiles`. Se la classe ODA implementa questa interfaccia, il metodo, che attualmente fornisce il contenuto generato, dipende dal *protocollo di contenuto* utilizzato da ODA per il tipo di contenuto file, nel modo seguente:

- Se l'ODA utilizza il protocollo del contenuto *su richiesta* per generare il contenuto, la Procedura guidata dell'oggetto Business inizia la generazione di contenuto come parte del passo 5 richiamando il metodo di generazione contenuto, `generatesBinFiles()`. L'array di nodi di origine selezionati dall'utente viene passata a questo metodo. Pertanto, affinché ODA supporti il contenuto di file, lo sviluppatore ODA deve implementare questo metodo come parte di un'implementazione ODA dell'interfaccia IGeneratesBinFiles .
- Se l'ODA utilizza il protocollo del contenuto *callback* per generare contenuto, l'ODA (oppure un altro processo esterno) inizia la generazione di contenuto richiamando un metodo definito dall'utente. Lo sviluppatore ODA deve implementare un meccanismo per generare i file.

Pertanto, se la Procedura guidata dell'oggetto Business richiama il metodo di generazione contenuto per i file, `generateBinFiles()`, dipende dalle seguenti condizioni:

- Se ODA implementa questa interfaccia `IGeneratesBinFiles`
- Se implementa `IGeneratesBinFiles`, il cui protocollo di contenuto viene utilizzato da ODA per generare file

**Nota:** Per ulteriori informazioni sui protocolli di contenuto, consultare "Selezione del protocollo del contenuto ODA" a pagina 118.

Indipendentemente dal protocollo di contenuto utilizzato, la generazione di contenuto richiede i seguenti passi:

1. Per ricevere informazioni aggiuntive, quali i valori di verbo, proprietà di oggetti business.
2. Per generare il contenuto richiesto e salvarlo nella struttura di contenuto generato nella memoria ODA.

Nelle seguenti sezioni viene fornito un riepilogo di questi passi. Per un ulteriore e più dettagliata panoramica sul processo di generazione contenuto, Tabella 20 illustra dove reperire ulteriori informazioni per ciascun tipo di contenuto supportato.

*Tabella 20. Processo di generazione contenuto*

| Tipo di contenuto | Per ulteriori informazioni |
|---|---|
| Definizioni oggetto business | "Creazione di definizioni di oggetti business" a pagina 129 |
| File binari | "Creazione di file" a pagina 147 |

## Ottenimento delle proprietà di oggetti business

Spesso occorrono ulteriori informazioni prima che ODA possa generare le definizioni di oggetti business. L'ODA può richiedere queste informazioni aggiuntive definendo le *proprietà dell'oggetto business*. L'API ODK rappresenta una proprietà di oggetti business come un oggetto di proprietà dell'agente (`AgentProperty`). Per raccogliere le proprietà di oggetti business, ODA può far in modo che la Procedura guidata dell'oggetto Business visualizzi casella di dialogo Proprietà BO. In questa casella di dialogo, la procedura guidata visualizza le proprietà di oggetti business, consente gli aggiornamenti e scrive le proprietà inizializzate dall'utente nella memoria runtime di ODA, come viene illustrato in Figura 53 a pagina 90.

Per visualizzare la finestra Proprietà BO, il metodo di generazione del contenuto dell'ODA richiama il metodo `getBOSpecificProps()` (definito nella classe `ODKUtility`).

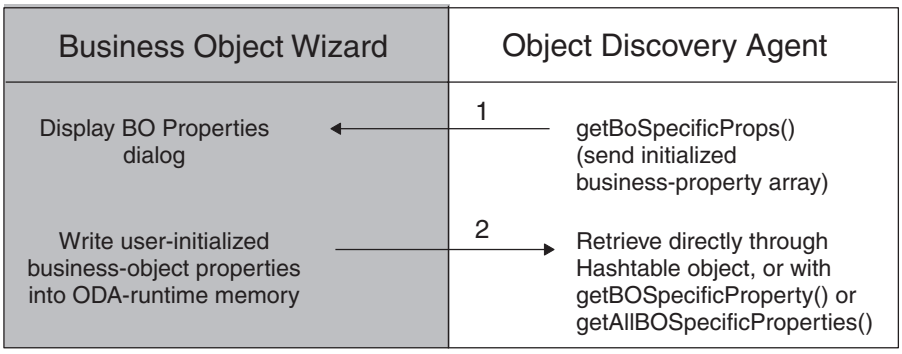



*Figura 56. Ottenimento delle proprietà di oggetti business*

Come viene illustrato in Figura 56, con il metodo `getBOSpecificProps()` vengono effettuate le seguenti operazioni:

1. Invia le proprietà degli oggetti business a Procedura guidata dell'oggetto Business, che le visualizza nella finestra Proprietà BO.

   Per inviare le proprietà di oggetti business, il metodo `getBOSpecificProps()` invia come argomento l'array inizializzata degli oggetti delle proprietà dell'agente `AgentProperty`), un oggetto per ciascuna proprietà di oggetti business da visualizzare.
2. Dalla casella di dialogo Proprietà BO, è possibile aggiungere o modificare valori. Dopo aver fatto clic sul pulsante **Avanti**, la procedura guidata invia le proprietà di oggetti business inizializzati dall'utente di nuovo al metodo `getBOSpecificProps()` nell'ODA.

   È possibile accedere queste proprietà di oggetti business all'interno di ODA attraverso l'oggetto `Hashtable`Java che restituisce `getBOSpecificProps()`. In alternativa, è possibile accedere alle proprietà mediante un'istanza della classe `ODKUtility`, che fornisce i metodi `getBOSpecificProperty()` e `getAllBOSpecificProperties()`.

L'ODA può richiamare ripetutamente `getBOSpecificProps()` per ottenere differenti insiemi di proprietà di oggetti business. Per ulteriori informazioni su come utilizzare il metodo `getBOSpecificProps()`, consultare "Richiesta di proprietà di oggetti business" a pagina 130.

## Aggiunta di contenuto generato

L'ODA fornisce il contenuto generato alla Procedura guidata dell'oggetto Business in due parti:

- I metadati di contenuto

  Un oggetto di metadati del contenuto `ContentMetaData`) contiene le informazioni sul contenuto generato dell'ODA. La Procedura guidata dell'oggetto Business utilizza queste informazioni per determinare quale metodo di richiamo contenuto utilizzare per richiamare il contenuto generato.
- Il contenuto stesso

  L'ODA scrive il contenuto generato in una *struttura di contenuto generato*, che è accessibile dai metodi della classe ODA. Ad esempio, può scrivere il contenuto ad un array che è un membro variabile della classe ODA.

Il metodo che fornisce il contenuto generato dipende dal protocollo di contenuto generato che ODA utilizza per un particolare tipo di contenuto, nel modo seguente:

- Se l'ODA utilizza il protocollo di contenuto a richiesta per generare contenuto, è il metodo di generazione contenuto che popola la struttura di contenuto generato e restituisce un oggetto metadati di contenuto da Procedura guidata dell'oggetto Business. La Procedura guidata dell'oggetto Business richiama il metodo di generazione contenuto in base al tipo di contenuto, nel modo seguente:
  - Per definizioni di oggetti business, `generateBoDefs()` nell'interfaccia `IGeneratesBoDefs`
  - Per i file, `generateBinFiles()` nell'interfaccia `IGenerateBinFiles`
- Se l'ODA utilizza il protocollo di contenuto callback per generare contenuto, è il metodo definito dall'utente che popola la struttura di contenuto generato e invia un oggetto metadati di contenuto da Procedura guidata dell'oggetto Business.

**Nota:** Per ulteriori informazioni sui protocolli di contenuto, consultare "Selezione del protocollo del contenuto ODA" a pagina 118.

Nella seguente tabella viene illustrato dove reperire ulteriori informazioni su come fornire contenuto generato:

| Tipo di contenuto | Per ulteriori informazioni |
|---|---|
| Definizioni oggetto business | "Aggiunta di definizioni di oggetti business generati" a pagina 142 |
| File binari | "Aggiunta di file generati" a pagina 150 |

Per recuperare il contenuto generato, la Procedura guidata dell'oggetto Business richiama il metodo di richiamo contenuto come viene illustrato in Tabella 21.

*Tabella 21. Metodi di richiamo contenuto*

| Tipo di contenuto | Metodo di richiamo contenuto | Per ulteriori informazioni |
|---|---|---|
| Definizioni di oggetto business | `IGeneratesBoDefs.getBoDefs()` | "Accesso alle definizioni di oggetti business generati" a pagina 143 |
| File binari | `IGeneratesBinFiles.getBinFile()` | "Aggiunta di accesso ai file generati" a pagina 152 |

Il metodo di richiamo contenuto accede alla struttura di contenuto generato all'interno dell'oggetto ODA e restituisce il contenuto specificato in un array alla Procedura guidata dell'oggetto Business. La Procedura guidata dell'oggetto Business deve avere accesso al contenuto generato prima che possa elaborare la richiesta per salvare il contenuto nel passo 6. Per ulteriori informazioni, consultare "Salvataggio del contenuto".

## Salvataggio del contenuto

Passo 6 di la Procedura guidata dell'oggetto Business visualizza la casella di dialogo Salva oggetti business, che fornisce opzioni per il salvataggio delle definizioni di oggetto business.Come viene illustrato in Figura 44 a pagina 81, la Procedura guidata dell'oggetto Business consente di salvare il contenuto generato in un progetto ICL o in un file, oppure di aprire ciascuna definizione di oggetto business in Business Object Designer Express. Per salvare le definizioni di oggetti

business generati nel formato specifico, la Procedura guidata dell'oggetto Business deve accedere al contenuto generato. Questo contenuto è stato richiamato nel precedente passo (passo 5), utilizzando il metodo di richiamo contenuto di ODA elencato in Tabella 21.

# Panoramica del processo di sviluppo ODA

In questa sezione vengono fornite le seguenti informazioni relative al processo di sviluppo di un ODA:

- "Strumenti per lo sviluppo ODA"
- "Processo di sviluppo ODA" a pagina 107

## Strumenti per lo sviluppo ODA

Un ODA è uno dei possibili componenti di un adattatore WebSphere Business Integration Server Express. Un *adattatore* comprende i componenti runtime per supportare la comunicazione tra InterChange Server Express e le applicazioni o tecnologie. Uno di questi componenti runtime è ODA, che crea le definizioni di oggetti business per il connettore da utilizzare al runtime. Il *connettore* è il componente runtime che gestisce le comunicazioni tra un'applicazione (o tecnologa) e InterChange Server Express. Inoltre, l'adattatore comprende un *framework dell'adattatore*, che comprende i componenti per la configurazione, il runtime e lo sviluppo di adattatori personalizzati nei casi in cui un adattatore precostituito per una particolare applicazione precedente o specializzata non sia attualmente disponibile come parte del prodotto WebSphere Business Integration Server Express.

Per lo sviluppo di un ODA, il framework dell'adattatore comprende il supporto allo sviluppo elencato in Tabella 22.

*Tabella 22. Supporto del framework dell'adattatore per lo sviluppo di un ODA*

| Componente dell'adattatore | Strumento di configurazione | API |
|---|---|---|
| definizione di oggetti business | Business Object Designer Express | Non applicabile |
| ODA (Object Discovery Agent) | Business Object Designer Express | ODK (Object Discovery Agent Development Kit) |

**Nota:** Inoltre, il framework dell'adattatore fornisce supporto per lo sviluppo di connettori.

Oltre al framework dell'adattatore, l'ADK (Adapter Development Kit) è un toolkit che fornisce esempi di codice di ODA e dei connettori. Per ulteriori informazioni, consultare "ADK (Adapter Development Kit)".

### ADK (Adapter Development Kit)

L'ADK (Adapter Development Kit) fornisce file ed esempi di ausilio allo sviluppo di un adattatore. Fornisce esempi per molti componenti di adattatori, compreso un ODA (Object Discovery Agent), un connettore e un gestore dati. Gli ADK forniscono questi esempi nella subdirectory `DevelopmentKits` della directory del prodotto.

**Nota:** L'ADK è parte del prodotto WebSphere Business Integration Adapters e richiede un'installazione separata. Pertanto, per avere accesso agli esempi di

sviluppo in ADK, occorre avere accesso al prodotto WebSphere Business Integration Adapters e installare ADK. Si noti che ADK è disponibile *solo* per sistemi Windows.

Tabella 23 elenca gli esempi forniti da ADK per lo sviluppo di un ODA così come la subdirectory della directory `DevelopmentKits` in cui essi sono memorizzati.

*Tabella 23. Esempi ADK per lo sviluppo ODA*

| Componente ADK (Adapter Development Kit) | Descrizione | Subdirectory DevelopmentKits |
|---|---|---|
| ODK (Object Discovery Agent Development Kit) | Fornisce esempi ODA | `Odk` |
| Esempio dell'adattatore Twineball | Fornisce un adattatore di esempio, che include un ODA | `Twineball_sample` |

Come viene illustrato in Tabella 23, l'ADK include esempi di ODA. Tali esempi sono presenti nella seguente directory:

```
DevelopmentKits\Odk
```

Per ulteriori informazioni, consultare "Supporto di sviluppo per gli ODA".

**Nota:** Come viene illustrato in Tabella 23, ADK fornisce anche supporto per lo sviluppo di connettori, un altro componente dell'adattatore.

## Supporto di sviluppo per le definizioni di oggetti business

Tabella 24 illustra lo strumento che il prodotto WebSphere Business Integration Server Express fornisce come ausilio allo sviluppo di definizioni business.

*Tabella 24. Strumento per lo sviluppo di definizioni di oggetti business*

| Strumenti di sviluppo | Descrizione |
|---|---|
| Business Object Designer Express | Lo strumento grafico che agevola nella creazione di definizioni di oggetti business, in modo manuale o attraverso ODA. |

Per una breve introduzione sulle definizioni di oggetti business, consultare "Definizioni di oggetti business" a pagina 4.

## Supporto di sviluppo per gli ODA

Tabella 25 illustra gli strumenti che il prodotto WebSphere Business Integration Server Express fornisce come ausilio allo sviluppo di un ODA.

*Tabella 25. Strumenti per lo sviluppo degli ODA*

| Strumenti di sviluppo | Descrizione |
|---|---|
| Business Object Designer Express | Lo strumento grafico che agevola nella creazione di definizioni di oggetti business, in modo manuale o attraverso ODA. |

*Tabella 25. Strumenti per lo sviluppo degli ODA (Continua)*

| Strumenti di sviluppo | Descrizione |
|---|---|
| ODK (Object Discovery Agent Development Kit) | Contiene:<br>• API ODK: un insieme di classi Java con le quali è possibile creare un ODA personalizzato. Per una panoramica di queste classi, consultare Capitolo 7, "Panoramica dell'API ODK", a pagina 181.<br>• Runtime ODA: un insieme di classi Java utilizzate dal runtime ODA per gestire le comunicazioni tra ODA e Business Object Designer Express<br>• Esempi ODA: installati come parte di ADK (Adapter Development Kit). Per ulteriori informazioni, consultare "ADK (Adapter Development Kit)" a pagina 104. |

Come viene illustrato in Tabella 25 a pagina 105, ODK fornisce allo sviluppatore ODK l'API ODK (che è la libreria dei metodi da utilizzare nell'ODA) e gli esempi ODA, che sono reperibili nella seguente subdirectory del prodotto:

```
DevelopmentKits\Odk\Samples
```

ODK comprende i seguenti esempi ODA

*Tabella 26. Esempi ODA*

| Esempio ODA | Descrizione | Subdirectory di DevelopmentKits\Odk |
|---|---|---|
| ODA dell'esercito romano | Converte i nomi dei generali e dei soldati romani provenienti da un file XML in definizioni di oggetti business e fornisce file binari che descrivono la conversione. Questo ODA utilizza API ODK, come descritto in questo capitolo. | Per gli script di avvio: `Esempi`<br>Per file esterni e file .jar: `RomanArmy`<br>Per l'origine Java: `com\ibm\btools\ODK2\RomanArmy` |
| ODA JDBC | Converte dati JDBC (tabelle e schemi) in definizioni di oggetti business. Per questo esempio ODA da eseguire, occorre avere accesso a un database JDBC. Questo esempio viene basato su una versione precedente dell'API ODK, che gestisce *solo* la generazione di definizioni di oggetti business non quella del contenuto di file.<br>**Nota:** Se si sta sviluppando un nuovo ODA, utilizzare *solo* questo come esempio di una creazione di definizioni di oggetti business più complessa. Utilizzare l'esempio ODA dell'esercito romano come esempio di come deve essere strutturato un nuovo ODA. | Per gli script di avvio: `Esempi`<br>Per l'origine Java: `com\crossworlds\JDBC` |

Per una breve introduzione sugli ODA, consultare "Uso di Object Discovery Agent per creare una definizione di oggetto business" a pagina 70. Per istruzioni su come eseguire l'esempio ODA dell'esercito romano, consultare "Uso di un campione ODA" a pagina 72.

## Processo di sviluppo ODA

In questa sezione viene fornita una panoramica del processo di sviluppo ODA, che comprende i seguenti passi di livello avanzato:

1. Installare e impostare il software di WebSphere Business Integration Server Express e installare JDK (Java Development Kit). .
2. Creazione e implementazione di ODA.

### Impostazione dell'ambiente di sviluppo

Prima di avviare il processo di sviluppo, occorre che le seguenti condizioni siano valide:

- Il software WebSphere Business Integration Server Express è installato su una macchina che è possibile accedere.

  Affinché venga eseguito un ODA, occorre poter accedere alla libreria ODA, `CwODA.jar`. Pertanto, questa libreria ODA deve essere installata. Per ulteriori informazioni, consultare le informazioni di installazione del prodotto.

  **WebSphere Business Integration Server Express**

  Se il sistema di integrazione business utilizza InterChange Server Express, il file `CwODA.jar` viene installato come parte del software InterChange Server Express. Fare riferimento alla guida all'installazione di WebSphere Business Integration Server Express per informazioni su come installare e avviare il sistema InterChange Server Express system.

- Il JDK (Java Development Kit) o un prodotto di sviluppo compatibile con JDK viene installato sulla macchina di sviluppo.

  Per informazioni sulla versione richiesta di JDK e su come installarla, fare riferimento alle informazioni di installazione del prodotto. Assicurarsi di aggiornare la variabile d'ambiente PATH per includere la directory Java installata. Riavviare InterChange Server Express dopo aver aggiornato il percorso.

- L'ambiente di sviluppo deve essere in grado di accedere alla directory che contiene il file della libreria ODA, `CwODA.jar`:

  *ProductDir*`\lib`

  Per compilare ODA, il compilatore *deve* essere in grado di accedere a questa directory ODA. Per informazioni sulla compilazione di un ODA, consultare "Compilazione di ODA" a pagina 173.

**Nota:** Per creare un ODA e verificarne il contenuto generato, *non* occorre avere InterChange Server Express o un connettore in esecuzione. Tuttavia, allo stesso punto, il connettore deve essere in esecuzione per verificare che il contenuto generato da ODA descriva gli oggetti business del connettore. Per verificare tutto il sistema WebSphere Business Integration Server Express, InterChange Server Express e il connettore devono essere in grado di comunicare.

## Fasi dello sviluppo ODA

Per sviluppare un ODA, occorre eseguire le operazioni elencate in Tabella 27.

*Tabella 27. Passi nello sviluppo di un ODA*

| | Passo dello sviluppo ODA | Per ulteriori informazioni |
|---|---|---|
| 1. | Estendere la classe di base ODA, `ODKAgentBase2`, per creare la classe ODA. | "Estensione della classe di base ODA" a pagina 109 |
| 2. | Implementare i metodi della classe ODA, che forniscono lo strumento di avvio di ODA. | "Avvio di ODA" a pagina 110 |
| 3. | Creazione e implementazione di contenuto ODA:<br>• Quali tipo di contenuto sono supportati da ODA:<br>– Le definizioni di oggetti business: implementano l'interfaccia `IGeneratesBoDefs` (obbligatorio)<br>– File binari: implementano l'interfaccia `IGeneratesBinFiles` (facoltativo)<br>• Quali protocolli di contenuto sono utilizzati da ODA:<br>– a richiesta (richiesti dalle definizioni di oggetti business)<br>– callback | "Individuazione del contenuto generato ODA" a pagina 117 |
| 4. | Errore di implementazione e gestione dei messaggi per tutti i metodi ODA. Messaggi di traccia di implementazione ai livelli di traccia appropriati. | "Gestione delle eccezioni" a pagina 170 e "Gestione dei messaggi di errore e di traccia" a pagina 163 |
| 5. | Creazione di classi necessarie alla gestione di interazioni di origini di dati, quali:<br>• Gestione connessioni<br>• Analisi e definizione del contenuto | IBM raccomanda di organizzare l'ODA in classi di componenti che gestiscano processi significativi separati. Dettagli che dipendono dall'origine di dati. |
| 6. | Creazione di ODA. | "Compilazione di ODA" a pagina 173 |
| 7. | Creazione di uno script di avvio per il nuovo ODA. | "Avvio di un nuovo ODA" a pagina 174 |
| 8. | Verifica e debug di ODA, ricodifica (se necessario). | |

La scrittura di un codice ODA è solo una parte dell'attività generale per lo sviluppo di oggetti business. Prima di iniziare la scrittura del codice ODA, occorre comprendere in modo chiaro i problemi di creazione degli oggetti business, l'applicazione le cui entità verranno rappresentate dagli oggetti business e il connettore e il gestore dati che elaborerà gli oggetti business. Occorre avere dimestichezza con le operazioni seguite dagli utenti in Business Object Designer Express per creare una definizione di oggetti business utilizzando un ODA.

**Nota:** Per informazioni sulla creazione di oggetti business, fare riferimento a Capitolo 2, "Progettazione di oggetti business", a pagina 17. Per informazioni sull'utilizzo di ODA in Business Object Designer Express, consultare "Uso di Object Discovery Agent per creare una definizione di oggetto business" a pagina 70.

## Estensione della classe di base ODA

Per creare un ODA, è necessario estendere la classe di base ODA, `ODKAgentBase2`, per creare la propria *classe ODA*. La classe `ODKAgentBase2` comprende i metodi per l'inizializzazione, l'impostazione e la chiusura degli ODA. Per implementare l'ODA, occorre estendere questa classe di base ODA per creare la classe ODA.

Per derivare una classe ODA, eseguire le seguenti operazioni:

1. Creare una classe ODA che estende la classe `ODKAgentBase2`. Come nome per questa classe si suggerisce:

   *`ODAname`*`.java`

   dove *ODAname* identifica in modo univoco l'ODA e ha il formato dei dati di origine di ODA con l'estensione `ODA(`*`srcDataName`*`ODA)`. Per informazioni sui nomi dei dati di origine, consultare "Denominazione dell'ODA" a pagina 173. Ad esempio, per creare un ODA per oggetti HTML, creare un file della classe ODA denominato `HTMLODA.java`.
2. Nel file della classe ODA, definire un nome del pacchetto che contenga ODA. Per convenzione, un nome pacchetto ODA ha il seguente formato:

   `com.ibm.oda.`*`srcDataName.ODAname`*

   Nel formato appena citato, *ODAname* è uguale a quello definito nel passo 1 e *srcDataName* è uguale a quello definito nel passo 1 *tranne* che è specificato in lettere minuscole. Ad esempio, il nome pacchetto di un ODA per oggetti HTML può essere definito nella classe ODA nel modo seguente:

   ```
   package com.ibm.oda.html.HTMLODA;
   ```
3. Verificare che il file della classe ODA importi le classi del pacchetto `com.crossworlds.ODK`:

   ```
   import com.crossworlds.ODK.*;
   ```

   Per accedere ai metodi dell'API ODK, la classe ODA deve importare il pacchetto ODK, che è contenuto nel file `CwODK.jar` che si trova nella directory `lib` della directory del prodotto. Se si creano svariati file per mantenere il codice della classe ODA, occorre importare le classi del pacchetto ODK in *ogni* file di origine .
4. Definire la classe ODA e includere nella definizione tutte le interfacce di generazione contenuto utilizzate da ODA. Un ODA deve implementare l'interfaccia di generazione contenuto `IGeneratesBoDefs` per generare il contenuto della definizione di oggetti business. È anche possibile implementare l'interfaccia di generazione contenuto `IGeneratesBinFiles` per generare contenuto di file binari.

   Ad esempio, si supponga che ODA per l'HTML implementi *solo* l'interfaccia `IGeneratesBoDefs` richiesta. La relativa definizione dovrebbe essere come segue:

   ```
   public class HTMLODA extends ODKAgentBase2 implements IGeneratesBoDefs {
   ```

   Per ulteriori informazioni sulle interfacce di generazione contenuto, consultare "Individuazione del contenuto generato ODA" a pagina 117.
5. Implementare i metodi astratti della classe `ODKAgentBase2` per la classe ODA. Tabella 28 fornisce una panoramica di questi metodi, elencandoli nell'ordine in cui vengono richiamati da Procedura guidata dell'oggetto Business. Per ulteriori informazioni su come implementare tali metodi astratti, consultare Tabella 28.

*Tabella 28. Estensione dei metodi astratti per la classe ODKAgentBase2*

| Metodo astratto ODKAgentBase2 | Descrizione | Per ulteriori informazioni |
|---|---|---|
| `getAgentProperties()` | Questo metodo esegue le seguenti attività:<br>• Definire le proprietà di configurazione necessarie all'inizializzazione di ODA; comprese le informazioni necessarie a ODA per collegarsi all'origine di dati.<br>• Inviare le proprietà di configurazione in un array a Procedura guidata dell'oggetto Business. | "Ottenimento delle proprietà di configurazione" a pagina 111 |
| `getMetaData()` | Creare un'istanza dell'oggetto `AgentMetaData` che contiene i metadati di ODA, compresa la capacità di generare contenuto. | "Inizializzazione di metadati ODA" a pagina 113 |
| `init()` | Inizializzare ODA, compresa l'allocazione di risorse e il collegamento all'origine di dati. | "Inizializzazione dell'avvio di ODA" a pagina 115 |
| `terminate()` | Eseguire il cleanup, compreso lo scollagamento dall'origine di dati e il rilascio di risorse utilizzate da ODA. | "Chiusura di ODA" a pagina 162 |

6. Implementazioni dei metodi dell'interfaccia della generazione di contenuto (interfacce) della classe ODA. Tabella 28 elenca i metodi delle interfacce di generazione di contenuto e indica dove individuare ulteriori informazioni su come creare tali metodi.

*Tabella 29. Definizione di metodi dell'interfaccia di generazione contenuto*

| Interfaccia di generazione contenuto | Descrizione | Per ulteriori informazioni |
|---|---|---|
| `IGeneratesBoDefs` | `getContentProtocol()` | "Individuazione del contenuto generato ODA" a pagina 117 |
| | `getTreeNodes()`<br>`generateBoDefs()`<br>`getBoDefs()` | "Creazione di definizioni di oggetti business come contenuto" a pagina 121 |
| `IGeneratesBinFiles` | `getContentProtocol()` | "Individuazione del contenuto generato ODA" a pagina 117 |
| | `generateBinFiles()`<br>`getBinFile()` | "Creazione dei file binari come contenuto" a pagina 144 |

# Avvio di ODA

Quando viene avviato ODA, il runtime ODA crea istanze per la classe ODA asociata (un'estensione di `ODKAgentBase2`) e quindi richiama i metodi della classe in Tabella 30.

*Tabella 30. Avvio di ODA*

| Attività di inizializzazione | Metodo ODKAgentBase2 | Per ulteriori informazioni |
|---|---|---|
| 1. Ottenere le proprietà di configurazione, comprese quelle che descrivono l'origine dati da aprire. | `getAgentProperties()` | "Ottenimento delle proprietà di configurazione" a pagina 111 |
| 2. Inizializzare i metadati ODA affinché la Procedura guidata dell'oggetto Business possa ottenere informazioni su ODA (specialmente sul contenuto supportato). | `getMetaData()` | "Inizializzazione di metadati ODA" a pagina 113 |

*Tabella 30. Avvio di ODA (Continua)*

| Attività di inizializzazione | Metodo ODKAgentBase2 | Per ulteriori informazioni |
|---|---|---|
| 3. Inizializzare ODA per eseguire qualsiasi passo di avvio necessario, quale l'apertura di una connessione con l'origine di dati. | `init()` | "Inizializzazione dell'avvio di ODA" a pagina 115 |

Le seguenti sezioni descrivono ciascun passo in Tabella 30.

## Ottenimento delle proprietà di configurazione

Per inizializzare l'ODA, la Procedura guidata dell'oggetto Business richiama il metodo `getAgentProperties()` della classe ODA. Il metodo `getAgentProperties()` è parte della classe di base ODA di un livello inferiore, `ODKAgentBase`. Viene ereditato dalla classe di base ODA. `ODKAgentBase2`, ereditato volta per volta dalla classe ODA.

**Importante:** Come parte dell'implementazione della classe ODA, *occorre* implementare un metodo `getAgentProperties()`.

Il metodo `getAgentProperties()` esegue le seguenti attività:

- "Ottenimento dell'oggetto ODKUtility"
- "Inizializzazione dell'array delle proprietà di configurazione"

### Ottenimento dell'oggetto ODKUtility

Poiché `getAgentProperties()` è il *primo* metodo ODA richiamato da Procedura guidata dell'oggetto Business, è bene creare un'istanza dell'oggetto `ODKUtility`, che fornisce il codice ODA con l'accesso a quanto segue:

- Oggetti della memoria del runtime di ODA, quali le proprietà di configurazione e le proprietà degli oggetti business
- Metodi di utilità che forniscono la traccia e visualizzano le caselle di dialogo di risposta utente

Per avere accesso ad un oggetto `ODKUtility`, utilizzare il metodo `getODKUtility()`. Questo metodo, definito nella classe `ODKUtility`, restituisce un handle all'oggetto `ODKUtility`.

```
odkUtil = ODKUtility.getODKUtility()
```

Se si dichiara l'identificativo di file all'oggetto `ODKUtility` come globale all'intera classe ODA, *tutti* i metodi all'interno di questa classe possono accedere a questi metodi di utilizzo.

**Nota:** Invece di creare un'istanza dell'oggetto `ODKUtility` nel metodo `getAgentProperties()`, l'esempio dell'esercito romano (Roman Army) ODA fornisce una variabile di membro denominata `m_utility` nella classe ODA e la inizializza nel modo seguente:

```
final ODKUtility m_utility = ODKUtility.getODKUtility();
```

### Inizializzazione dell'array delle proprietà di configurazione

Come descritto in "Ottenimento delle proprietà di configurazione ODA" a pagina 97, la Procedura guidata dell'oggetto Business utilizza l'array di proprietà di configurazione che `getAgentProperties()` restituisce per inizializzare la finestra Configura agente (passo 2). Questa casella di dialogo visualizza tutte le proprietà di configurazione ODA e consente l'immissione o la modifica di questi valori.

L'array delle proprietà di configurazione è un array degli oggetti `AgentProperty`. La classe `AgentProperty` fornisce supporto per la proprietà di configurazione per ottenere le seguenti funzioni:

- Un valore predefinito
- Mantiene solo un valore o più di un valore
- Un elenco di valori validi tra cui l'utente può effettuare una selezione
- Condizioni che limitano il valore che l'utente può immetere

**Nota:** Per ulteriori informazioni, consultare "Operazioni con le proprietà agente" a pagina 153.

Lo scopo di `getAgentProperties()` è di inviare alla Procedura guidata dell'oggetto Business un array di oggetti `AgentProperty` che descrivano le le proprietà ODA. Per inizializzare l'array delle proprietà di configurazione in `getAgentProperties()`, effettuare le seguenti operazioni:

1. Creare un'istanza di un oggetto `AgentProperty` per una proprietà di configurazione, che lo inizializza con le informazioni di proprietà appropriate.

   L'implementazione del metodo `getAgentProperties()` deve creare un'istanza degli oggetti della proprietà agente per ciascuna proprietà di configurazione visualizzata dalla procedura guidata Business Object. Quando si crea un'istanza dell'oggetto di proprietà dell'agente, si inizializzano alcune o tutte le variabili membro (illustrate in Tabella 51 a pagina 153).
2. Memorizzare l'oggetto `AgentProperty` inizializzato nell'array delle proprietà di configurazione.
3. Ritornare all'array delle proprietà di configurazione dal metodo `getAgentProperties()`.

Figura 57 illustra l'implementazione del metodo `getAgentProperties()` (definito nella classe `ArmyAgent2` dell'esercito romano ODA).

```
public AgentProperties[] getAgentProperties()
   throws com.crossworlds.ODK.ODKException
{
   AgentProperty general = new AgentProperty("Army general",
      AgentProperty.TYPE_STRING, true, false, false,
      "A general is a soldier at least 45 years old", true,
      ODKConstant.SINGLE_CARD, m_generals.toArray(), null);

   AgentProperty recAdop = new AgentProperty("Allow adoption",
      AgentProperty.TYPE_BOOLEAN, true, false, false,
      "Select \"yes\" if adopted children can be business objects", true,
      ODKConstant.SINGLE_CARD, new Object[]{"true", "false"}, null);

   AgentProperty minAge = new AgentProperty("Minimum age for drafting",
      AgentProperty.TYPE_INTEGER, true, false, false,
      "Drafted soldiers will be generable nodes", false,
      ODKConstant.SINGLE_CARD, null, new Object[] {"15"});

   AgentProperty maxAge = new AgentProperty("Maximum age for drafting",
      AgentProperty.TYPE_INTEGER, true, false, false,
      "Drafted soldiers will be generable nodes", false,
      ODKConstant.SINGLE_CARD, null, new Object[] {"55"});

   AgentProperty minAdo = new AgentProperty("Minimum age for adopting",
      AgentProperty.TYPE_INTEGER, true, false, true,
      "Drafted soldiers will be generable nodes", false,
      ODKConstant.SINGLE_CARD, null, new Object[] {"" + m_minAdoptionAge});

   AgentProperty[] props = new AgentProperty[]
      {general, minAge, maxAge, recAdop, minAdo});

   return props;
}
```

*Figura 57. Inizializzazione dell'array delle proprietà di configurazione*

Figura 57 inizializza le cinque proprietà di configurazione ODA per l'esempio dell'esercito romano (Roman Army) ODA. Le proprietà attuali definite dipendono dall'origine di dati specifica accedute dall'ODA.

Dopo che i valori vengono specificati per le proprietà di configurazione, la Procedura guidata dell'oggetto Business salva tali proprietà nella memoria del runtime di ODA. ODA può accedere a queste proprietà attraverso i metodi quali il metodo `getAgentProperty()` nella classe `ODKUtility`. Per ulteriori informazioni, consultare "Richiamo delle proprietà della configurazione ODA" a pagina 115.

## Inizializzazione di metadati ODA

Dopo che la Procedura guidata dell'oggetto Business richiama il metodo `getAgentProperties()` ODA, richiama il metodo `getMetaData()` per inizializzare i metadati ODA. Il metodo `getMetaData()` è definito nella classe di base ODA, `ODKAgentBase2`, ereditato dalla classe ODA. Esso restituisce un oggetto `AgentMetaData` che contiene i metadati di ODA, compreso il contenuto generato che supporta.

**Importante:** Il metodo `getMetaData()` è un metodo astratto. Come parte dell'implementazione della classe ODA, *occorre* implementare un metodo `getMetaData()`.

L'oggetto `AgentMetaData` fornisce le informazioni in Tabella 31 al runtime ODA quando occorre ottenere i metadati per l'ODA.

*Tabella 31. Contenuti di un oggetto AgentMetaData*

| variabile membro | Descrizione |
|---|---|
| `agentVersion` | La versione della ODA |
| `searchableNodes`, `searchPatternDesc` | Informazioni per specificare il modello di ricerca ODA, che l'utente può specificare per ridurre il numero di nodi della struttura dall'origine dati che sono visualizzati |
| `supportedContent` | Una descrizione del contenuto generato che ODA può supportare. |

Per inizializzare i metadati ODA, implementare il metodo `getMetaData()`, che implica le seguenti operazioni:

- Creare un'istanza della classe `AgentMetaData`, passando in un riferimento di ODA stesso e una versione ODA facoltativa.

  Utilizzare uno dei due moduli del costruttore `AgentMetaData()`. Entrambi i moduli richiedono che `questo` riferimento passi ad un oggetto ODA (un'istanza della classe ODA). Il costruttore interroga l'oggetto ODA per ricevere informazioni sull'interfaccia (o interfacce) della generazione di contenuto implementata da ODA. Utilizza quindi queste informazioni per inizializzare la variabile di membro `supportedContent` con i protocolli di contenuto supportati da ODA per ciascuno dei tipi di contenuto supportati. Per ulteriori informazioni sul contenuto supportato di ODA, consultare "Individuazione del contenuto generato ODA" a pagina 117.

  È anche possibile fornire una versione della versione ODA come un argomento al costruttore per inizializzare la variabile di membro `agentVersion`.
- Inizializzare altre variabili membro come appropriate per ODA.

  Affinché ODA supporti la funzione del modello di ricerca, occorre inizializzare in modo esplicito le mariabili di membro `searchableNodes` e `searchPatternDesc` dopo aver creato un'istanza per il membro `AgentMetaData`. Per ulteriori informazioni, consultare "Implementazione della funzione del modello di ricerca" a pagina 124.
- Ritornare all'oggetto `AgentMetaData` dal metodo `getMetaData()`.

Figura 58 illustra l'implementazione del metodo `getMetaData()` (definito nella classe `ArmyAgent2` per l'esempio dell'esercito romano ODA).

```
public AgentMetaData getMetaData(){
   odkUtil.trace(TRACELEVEL1, XRD_TRACE, "Entering getMetaData()...");
   AgentMetaData amdObj = new AgentMetaData(this, "Sample ODA v1.0.0");
   //Initialize search-pattern feature for tree nodes
   amd.searchableNodes = true;
   amd.searchPatternDesc = "Enter the first letter to search by. " +
     "For example, \"A\", \"k\", "\Z\". Only names that start " +
     "with this letter will be returned."
   return amd;
}
```

*Figura 58. Inizializzazione dei meta dati ODA*

Poiché il metodo `getMetaData()` in Figura 58 viene ereditato dalla classe `ArmyAgent3` (che implementa l'interfaccia `IGeneratesBoDefs`), la chiamata al costruttore `AgentMetaData()` in questo frammento di codice inizializza il tipo di contenuto e il protocollo di contenuto associato per l'ODA. Dopo che `getMetaData()` viene avviata in `ArmyAgent3`, il tipo di contenuto di ODA viene

inizializzato in ContentType.BusinessObject e il protocollo del contenuto nella "richiesta". Per ulteriori informazioni, consultare "Individuazione del contenuto generato ODA" a pagina 117.

Questo metodo getMetaData() fornisce anche supporto per la funzione di modello di ricerca inizializzando le variabili di membro searchableNodes e searchPatternDesc. La variabile searchPatternDesc contiene il testo che viene visualizzato nella casella di dialogo Immetti il modello di ricerca (consultare Figura 48 a pagina 87).

## Inizializzazione dell'avvio di ODA

Dopo che la Procedura guidata dell'oggetto Business richiama il metodo getMetaData() ODA, richiama il metodo init() per avviare l'inizializzazione di ODAstart. Il metodo init() è parte della classe di base ODA di livello inferiore, ODKAgentBase. Viene ereditato dalla classe di base ODA. ODKAgentBase2, ereditato volta per volta dalla classe ODA. Questo metodo esegue operazioni di inizializzazione per ODA.

**Importante:** Il metodo init() è un metodo astratto. Come parte dell'implementazione della classe ODA, *occorre* implementare un metodo init().

Le attività principali del metodo init() comprendono:

- "Richiamo delle proprietà della configurazione ODA"
- "Stabilimento di una connessione" a pagina 116
- "Verifica di una versione ODA" a pagina 117

### Richiamo delle proprietà della configurazione ODA

Il metodo init() può richiamare qualsiasi proprietà di configurazione inizializzata dall'utente necessarie a completare l'inizializzazione dell'ODA. L'ODA inizializza le proprietà di configurazione nel metodo getAgentProperties(). Gli utenti possono aggiornare tali proprietà, se necessario, nella casella di dialogo Configura agente della Procedura guidata dell'oggetto Business. Dopo aver aggiornato le proprietà di configurazione, la Procedura guidata dell'oggetto Business le scrive nella memoria del runtime di ODA.

L'API ODK fornisce i metodi in Tabella 32 per richiamare il valore di una proprietà di configurazione ODA dalla memoria runtime di ODA.

*Tabella 32. Metodi per richiamare il valore di una proprietà di configurazione ODA*

| Metodo della libreria ODK | Descrizione |
|---|---|
| getAgentProperty() | Richiama il valore di una proprietà di configurazione ODA specificata |
| getAllAgentProperties() | Richiama i valori delle proprietà di configurazione di *tutte* le proprietà di configurazione ODA con un oggetto Hashtable Java. |

Tutti i metodi in Tabella 32 vengono definiti nella classe ODKUtility. Pertanto, occorre ottenere un identificativo di file per il singolo oggetto di questa classe *prima* di poter accedere a qualsiasi proprietà di configurazione. Per ulteriori informazioni, consultare "Ottenimento dell'oggetto ODKUtility" a pagina 111.

Figura 59 illustra l'implementazione del metodo `init()` (definito nella classe `ArmyAgent3` dall'esempio dell'esercito romano ODA).

```
public void init() throws com.crossworlds.ODK.ODKException
{
   Hashtable h = m_utility.getAllAgentProperties();
// Obtain values of ODA configuration properties
   AgentProperty property = (AgentProperty) h.get("Army general");
   m_general = property.allValues[0].toString();
   property = (AgentProperty) h.get("Minimum age for drafting");
   m_minAge = Integer.parseInt(property.allValues[0].toString());
   property = (AgentProperty) h.get("Maximum age for drafting");
   m_maxAge = Integer.parseInt(property.allValues[0].toString());
   property = (AgentProperty) h.get("Allow adoption");
   m_allowAdoption = new Boolean(
      property.allValues[0].toString()).booleanValue();
// Clear the generated-content structure
   m_generatedBOs.clear();
}
```

*Figura 59. Inizializzazione di ODA*

In Figura 59, il metodo `init()` utilizza i seguenti metodi per ottenere i valori di proprietà di configurazione:

- Il metodo `getAllAgentProperties()`, definito nella classe `ODKUtility`, per richiamare tutte le proprietà di configurazione nell'oggetto `Hashtable` Java.
- Il metodo `get()`, definito nella classe `Hashtable` Java, richiamare un elemento per nome dalla tabella hash.
- Il La variabile di membro `allValues`, definita nella classe `AgentProperty`, per richiamare il valore che l'utente ha specificato per ciascuna proprietà di configurazione.

Le proprietà di configurazione ottenute da questo metodo `init()` sono tutte proprietà single-cardinality. Pertanto, la variabile membro `allValues` contiene solo un valore. Per un esempio dell'utilizzo di più proprietà multiple-cardinality, consultare "Creazione di un array di proprietà business" a pagina 131. Questo metodo `init()` inizializza anche la struttura del contenuto generato ODA, un vettore chiamato `m_generatedBOs`. Questo vettore manterrà le definizioni di oggetti business generati.

## Stabilimento di una connessione

L'attività principale del metodo di inizializzazione `init()` deve solitamente stabilire una connessione con l'origine di dati. L'ODA effettua una ricerca dell'origine di dati per "individuare" gli oggetti per le conversioni potenziali alle definizioni di oggetti business. Per stabilire una connessione, il metodo `init()` può eseguire le seguenti attività:

- Ottenere qualsiasi proprietà di configurazione ODA che fornisce informazioni sul collegamento e utilizzarle per collegarsi all'origine dei dati. Se una proprietà di configurazione è vuota, il metodo `init()` può fornire un valore predefinito oppure può generare l'eccezione `ODKInvalidPropException` .

  È possibile utilizzare il metodo `getAgentProperty()` per ottenere il valore di una proprietà di configurazione ODA. Per ulteriori informazioni, consultare "Richiamo delle proprietà della configurazione ODA" a pagina 115.
- Ottenere qualsiasi connessione richiesta o file. Ad esempio, il metodo `init()` di solito stabilisce una connessione con l'origine di dati. Se il metodo ODA *non può*

aprire una connessione, il metodo `init()` deve generare un'eccezione `ODKException` (oppure una delle relative sottoclassi) per indicare il motivo dell'errore.

Il metodo `init()` viene eseguito con esito positivo se ODA apre una connessione con esito positivo e l'ODA è pronto ad iniziare l'elaborazione dei dati nell'origine di dati. Se ODA *non può* aprire una connessione, il metodo `init()` deve generare un'eccezione `ODKException` per indicare la causa dell'errore.

### Verifica di una versione ODA

Il Il metodo `getVersion()` restituisce la versione del runtime ODA. Questo metodo è parte della classe di base ODA ad un livello inferiore, `ODKAgentBase`. Viene ereditato dalla classe di base ODA. `ODKAgentBase2`, ereditato dalla classe ODA. Viene richiamato in entrambi i seguenti contesti:

- Il metodo `init()` deve richiamare `getVersion()` per verificare la versione del runtime ODA.
- Il runtime ODA richiama il metodo `getVersion`() quando occorre per richiamare questa versione.

**Nota:** Il metodo `getVersion()` restituisce la versione del runtime ODA, ma *non* la versione ODA (memorizzata come parte dei metadati ODA).

## Individuazione del contenuto generato ODA

In questa sezione vengono fornite le seguenti informazioni sui problemi che occorre considerare quando si individua il contenuto dell'ODA che è possibile generare:

- "Selezione del tipo di contento ODA"
- "Selezione del protocollo del contenuto ODA" a pagina 118

## Selezione del tipo di contento ODA

L'API ODK identifica i tipi di contenuto validi che ODA può supportare con la classe `ContentType`. Questa classe contiene le variabili membro statico per ciascun tipo di contenuti supportato, come illustrato in Tabella 33.

*Tabella 33. Modi con cui vengono rappresentati i tipi di contenuto*

| Tipo di contenuto | variabile membro ContentType |
|---|---|
| Definizioni oggetto business | `BusinessObject` |
| File binari | `BinaryFile` |

La classe `ContentType` simula un elenco numerato dei tipi di contenuto ODA supportati. Ad esempio, un oggetto di tipo contenuto che rappresenta le definizioni di oggetti business utilizzerebbe solo la variabile membro `BusinessObject` nel modo seguente:

```
ContentType.BusinessObject
```

Per fornire supporto per generare di un particolare tipo di contenuto, un ODA deve implementare l'appropriata l'interfaccia di gestione contenuto, come descritto in Tabella 19 a pagina 100. Ogni ODA deve supportare la generazione di supporto delle definizioni di oggetti business. Inoltre può supportare la generazione di file binari come suo contento. Le interfacce di generazione contenuto contengono i tipi di metodo elencati in Tabella 34. Come parte dell'implementazione dell'interfaccia di generazione contenuto, occorre l'implementazione di questi metodi.

*Tabella 34. Metodi in una interfaccia di generazione contenuto*

| Metodo | Scopo del metodo | IGeneratesBoDefs | IGeneratesBinFiles |
|---|---|---|---|
| Metodo di generazione di nodi di origine | La Procedura guidata dell'oggetto Business richiama questo metodo per ottenere la gerarchia dei nodi di origine che viene visualizzato all'utente (passo 3: Selezione dell'origine). | `getTreeNodes()` | Nessuno |
| Metodo di generazione contenuto | La Procedura guidata dell'oggetto Business richiama questo metodo per iniziare la generazione di contenuto specificato per i dati di origine (passo 5: Creazione di Business Objects). | `generateBoDefs()` | `generateBinFiles()` |
| Metodo di richiamo contenuto | La Procedura guidata dell'oggetto Business richiama questo metodo che richiama il contenuto generato dalla memoria ODA (passo 5 : Creazione di oggetti business). | `getBoDefs()` | `getBinFile()` |

Per determinare il metodo dell'interfaccia di generazione contenuto da richiamare, la Procedura guidata dell'oggetto Business verifica i metadati di ODA. Uno dei componenti di questi metadati è la variabile membro `supportedContent`, che viene inizializzata dal costruttore `AgentMetaData()` richiamato all'interno del metodo `getMetaData()` ODA. Per ulteriori informazioni, consultare "Inizializzazione di metadati ODA" a pagina 113.

Tabella 35 illustra le informazioni che vengono fornite in questo capitolo su come implementare i metodi nell'interfaccia di generazione contenuto.

*Tabella 35. Come sviluppare l'interfaccia di generazione contenuto*

| Interfaccia di generazione contenuto | Per ulteriori informazioni |
|---|---|
| `IGeneratesBoDefs` | "Creazione di definizioni di oggetti business come contenuto" a pagina 121 |
| `IGeneratesBinFiles` | "Creazione dei file binari come contenuto" a pagina 144 |

## Selezione del protocollo del contenuto ODA

Un ODA può generare un particolare tipo di contenuto utilizzando uno dei *protocolli di contenuto* elencati in Tabella 36. Il protocollo di contenuti determina come ODA interagisca con la procedura guidata Object Wizard per generare il contenuto supportato, cioè determina se la procedura guidata Business può iniziare in modo esplicito la generazione da parte di ODA.

*Tabella 36. Protocolli di contenuto per un ODA*

| Protocollo del contenuto | Descrizione | Costanti del protocollo del contenuto |
|---|---|---|
| A richiesta | La procedura guidata Business Object richiede in modo esplicito ODA per generare il contenuto richiamando il metodo di generazione contenuto. Una volta completato questo metodo, il contenuto a richiesta è pronto. La procedura guidata Business Object può richiamare questo contenuto, quando richiesto, con una chiamata al metodo di richiamo contenuto. | `CONTENT_PROTOCOL_ONREQUEST` |
| Callback | ODA genera il contenuto in qualche modo e notifica la Procedura guidata dell'oggetto Business quando il contenuto è pronto. Una volta notificato, la Procedura guidata dell'oggetto Business richiama questo contenuto con una chiamata al metodo di richiamo contenuto. | `CONTENT_PROTOCOL_CALLBACK` |

**Nota:** Un ODA deve supportare la generazione di oggetti business con il protocollo di contenuto a richiesta. Inoltre, ODA può supportare la generazione del contenuto di file in uno dei protocolli di contenuto.

Per supportare i protocolli di contenuto, ODA deve eseguire le seguenti operazioni:

- "Indicazione dei protocolli di contenuto implementato"
- "Implementazione del metodo di generazione contenuto" a pagina 120

## Indicazione dei protocolli di contenuto implementato

Entrambe le interfacce `IGeneratesBoDefs` e `IGeneratesBinFiles` sono estensioni della interfaccia `IGeneratesContent`. Pertanto, esse entrambe ereditano il singolo metodo definito da `IGeneratesContent`, `getContentProtocol()`. Come parte dell'implementazione dell'interfaccia di generazione contenuto di ODA, occorre implementare il metodo `getContentProtocol()` per indicare che i protocolli attuali di ODA utilizzeranno i relativi tipi di contenuto supportati.

**Nota:** Un ODA può supportare *un* protocollo di contenuto per un dato tipo di contenuto.

Il metodo `getContentProtocol()` accetta come argomento un oggetto `ContentType` , che identifica un tipo di contenuto supportato da ODA. Il metodo `getContentProtocol()` restituisce il protocollo di contenuto supportato da ODA per il tipo di contenuto specificato. Viene restituito il protocollo di contenuto supportato come una costante dei protocolli di contenuto (illustrati in Tabella 36). Tali costanti vengono definite nell'interfaccia `ODKConstant`.

**Nota:** In questo release, un ODA deve generare definizioni di oggetti business a richiesta. Pertanto, deve implementare il metodo `getContentProtocol()` per restituire la costante `CONTENT_PROTOCOL_ONREQUEST` per un tipo di contenuto di `ContentType.BusinessObject`. Inoltre, ODA può supportare la generazione di file in entrambi i protocolli e restituire la costante del protocollo di contenuti appropriata al tipo di contenuto di `ContentType.BinaryFile`.

Figura 60 illustra un'implementazione di `getContentProtocol()` che indica che ODA supporta il protocollo di callback per la generazione di file e il protocollo a richiesta per la generazione di definizioni di oggetti business.

```
public long getContentProtocol(ContentType contentType)
{
    if (contentType == ContentType.BinaryFile)
      return ODKConstant.CONTENT_PROTOCOL_CALLBACK;
    else
      return ODKConstant.CONTENT_PROTOCOL_ONREQUEST;
}
```

*Figura 60. Indicazione dei protocolli di contenuto supportati*

## Implementazione del metodo di generazione contenuto

L'implementazione del metodo di generazione contenuto dipende dal protocollo di contenuto che il tipo di contenuto supporta, come viene illustrato in Tabella 37.

*Tabella 37. Protocolli di contenuto e il metodo di generazione contenuto*

| Protocollo del contenuto | Come richiamare il metodo di generazione contenuto | Implementazione del metodo di generazione contenuto |
|---|---|---|
| A richiesta | Procedura guidata dell'oggetto Business richiama esplicitamente il metodo di generazione del contenuto per iniziare le definizioni degli oggetti business pe la generazione del contenuto o dei file). | Il metodo deve generare il contenuto per i nodi di origine trasmessi nel relativo argomento e restituire gli appropriati metadati di contenuto a Procedura guidata dell'oggetto Business. |
| Callback | Procedura guidata dell'oggetto Business *non* richiama mai esplicitamente il metodo di generazione del contenuto in quanto la generazione del contenuto (solo per i file) viene iniziata dall'ODA per il protocollo del contenuto. | Il metodo deve generare un'eccezione poiché non deve essere *mai* richiamato direttamente. La generazione attuale del contenuto viene effettuata esternamente al metodo di generazione contenuto, in un metodo, classe o persino processo differente. |

Tabella 38 illustra le informazioni fornite in questo capitolo su come implementare i metodi di generazione contenuto.

*Tabella 38. Come sviluppare un metodo di generazione contenuto*

| Metodo di generazione contenuto | Per ulteriori informazioni |
|---|---|
| `IGeneratesBoDefs.generateBODefs()` | "Definizioni del metodo generateBoDefs()" a pagina 130 |
| `IGeneratesBinFiles.generateBinFiles()` | "Definizione del metodo generateBinFiles()" a pagina 148 |

## Creazione di definizioni di oggetti business come contenuto

Come discusso in "Definizioni di oggetti business" a pagina 4, una *definizione di oggetti business* rappresenta un modello per i dati in modo da poterli gestire come un'unità collettiva. Lo scopo di un ODA è quello di generare definizioni di oggetti business per gli oggetti in un'origine dati. Affinché un ODA generi il contenuto della dewfinizione degli oggetti business, la classe ODA deve implementare l'interfaccia `IGeneratesBoDefs`.

**Nota:** Poiché un ODA *deve* supportare la generazione di definizioni di oggetti business, la relativa classe ODA deve implementare l'interfaccia `IGeneratesBoDefs`.

Tabella 39 elenca i metodi che la classe ODA deve definire per implementare l'interfaccia `IGeneratesBoDefs`.

*Tabella 39. Metodi nell'interfaccia IGeneratesBoDefs*

| Metodo | metodo IGeneratesBoDefs | Descrizione |
|---|---|---|
| Metodo di generazione di nodi di origine | `getTreeNodes()` | Esegue le seguenti operazioni in modo iterativo:<br>• Individuare i nodi di origine per gli oggetti all'interno dell'origine di dati<br>• Costruire un array di nodi tree che rappresenta la gerarchia di nodi di origine.<br>• Restituire un array di nodi tree a Procedura guidata dell'oggetto Business, che li visualizza agli utenti nella casella di dialogo Seleziona origine. |
| Metodo di generazione contenuto | `generateBoDefs()` | Genera le definizioni di oggetti business per i dati di origine selezionati dall'utente, scrivendoli nella memoria ODA |
| Metodo di richiamo contenuto | `getBoDefs()` | Richiama una definizione di oggetti business specificati oppure tutte le definizioni di oggetti business da una memoria ODA |

**Nota:** Oltre ai metodi descritti in Tabella 39, `IGeneratesBoDefs` comprende anche il metodo `getContentProtocol()` per specificare il protocollo di contenuto supportato da ODA per la generazione di definizioni di oggetti business. Per ulteriori informazioni, consultare "Selezione del protocollo del contenuto ODA" a pagina 118.

Con l'implementazione dell'interfaccia `IGeneratesBoDefs`, la Procedura guidata dell'oggetto Business richiama i metodi illustrati in Tabella 40 per ottenere i nodi di origine così come generare e richiamare contenuto.

*Tabella 40. Procedura guidata Business Object e metodi IGeneratesBoDefs*

| Passi in Business Object Wizard | metodo IGeneratesBoDefs | Per ulteriori informazioni |
|---|---|---|
| Passo 3: Selezione di un'origine | `getTreeNodes()` | "Creazione dei nodi di origine" a pagina 122 |
| Passo 5: Creazione di oggetti business | `generateBoDefs()` | "Creazione di definizioni di oggetti business" a pagina 129 |

*Tabella 40. Procedura guidata Business Object e metodi IGeneratesBoDefs (Continua)*

| Passi in Business Object Wizard | metodo IGeneratesBoDefs | Per ulteriori informazioni |
|---|---|---|
| Passo 5: Creazione di oggetti business | `getBoDefs()` | "Accesso alle definizioni di oggetti business generati" a pagina 143 |

Nelle seguenti sezioni viene illustrata l'implementazione di ciascun metodo in Tabella 40.

## Creazione dei nodi di origine

Procedura guidata dell'oggetto Business richiama il metodo `getTreeNodes()` per individuare i nodi di origine nell'origine di dati ODA e creare la gerarchia di nodi di origine, che Procedura guidata dell'oggetto Business visualizza nella finestra Seleziona origine (passo 3). Il metodo `getTreeNodes()` fa parte dell'interfaccia `IGeneratesBoDefs` che ogni classe ODA deve implementare per supportare la generazione delle definizioni degli oggetti di business.

**Importante:** Come parte dell'implementazione dell'interfaccia `IGeneratesBoDefs`, *occorre* implementare un metodo `getTreeNodes()` per l'ODA.

Come viene illustrato in "Selezione e conferma dei dati di origine" a pagina 99, la Procedura guidata dell'oggetto Business utilizza l'array di nodi tree che `getTreeNodes()` restituisce per inizializzare la casella di dialogo Seleziona origine. In questa casella di dialogo viene visualizzata la gerarchia di nodi di origine, che consente agli utenti di spostarsi tra i nodi di origine ottenuti dall'origine di dati e per selezionare quelli per cui ODA genera le definizioni di oggetti business. Ogni volta che un nodo di origine viene espanso, la Procedura guidata dell'oggetto Business richiama il metodo `getTreeNodes()` , che restituisce un array di nodi tree con i contenuti del nodo di origine espanso.

Ad esempio, il metodo `getTreeNodes()` nell'esempio dell'esercito romano ODA inizializza la casella di dialogo Seleziona origine con il generale dell'esercito di livello superiore, che è stato ottenuto dall'origine di dati dell'esempio, il file `RomanArmy.xml`. Quando si espande un particolare nodo, `getTreeNodes()` ottiene i figli del generale dell'esercito di tale nodo dal file XML e li mette in un array del nodo tree. La Procedura guidata dell'oggetto Business utilizza questo array del nodo tree per visualizzare il nodo di origine espanso.

Pertanto, lo scopo di `getTreeNodes()` è quello di individuare i nodi di origine nell'origine di dati e quindi costruirli e restituire un array di i nodi tree. A tale scopo, `getTreeNodes()` esegue le seguenti attività:

- "Individuazione del percorso del nodo principale"
- "Implementazione della funzione del modello di ricerca" a pagina 124
- "Interrogazione dell'origine di dati" a pagina 125
- "Costruzione dei nodi tree" a pagina 126

### Individuazione del percorso del nodo principale

Quando la Procedura guidata dell'oggetto Business richiama il metodo `getTreeNodes()`, trasmette il valore del *percorso del nodo principale* a questo metodo. Questo percorso identifica il nodo selezionato dall'utente che verrà espanso da `getTreeNodes()`. È un `String` che contiene il percorso completo del nodo,

dall'elemento principale di livello superiore fino al nodo selezionato dall'utente. I nomi di nodi all'interno di questo percorso vengono separati con i due punti (:).

Ad esempio, Figura 61 illustra una casella di dialogo Seleziona origine che visualizza una vista della gerarchia di nodi di origine per l'esempio dell'esercito romano ODA.

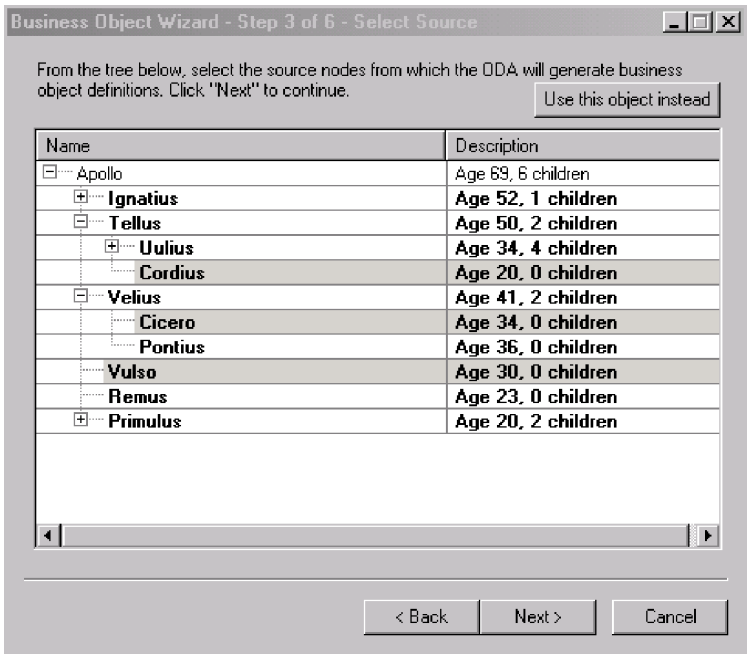



*Figura 61. Gerarchia dell'esempio del nodo di origine*

In Figura 61, il percorso del nodo principale per il nodo di origine Uulius è il seguente:

```
Apollo:Tellus:Uulius
```

Gli utenti possono specificare un nodo principale daespandere in uno dei seguenti modi:

- Facendo clic sul simbolo + a sinistra del nome del nodo principale.

  La Procedura guidata dell'oggetto Business costruisce un percorso del nodo principale per il nodo di origine selezionato e lo trasmette a `getTreeNodes()`.
- Selezionando **Sostituisci con questo oggetto** nella parte superiore della casella di dialogo Seleziona origine e specificando un percorso esplicito del nodo principale nella casella di dialogo Percorso oggetto.

  Occorre specificare il percorso del nodo principale con la stessa sintassi prevista da `getTreeNodes()` per il percorso del nodo principale. Per ulteriori informazioni, consultare "Specifica del percorso dell'oggetto" a pagina 87.

Il metodo `getTreeNode()` utilizza il percorso del nodo principale per determinare il livello della gerarchia del nodo di origine da replicare nel relativo nodo tree. Questo array del nodo tree conterrà tutti i nodi secondari del nodo identificati dal percorso del nodo principale. Per comunicare a `getTreeNodes()` di ritornare al livello superiore della gerarchia dei nodi di origine, la Procedura guidata dell'oggetto Business passa in un percorso di nodi principali "vuoto". Pertanto, il metodo `getTreeNodes()` deve verificare che vi sia un percorso di nodo vuoto come primo passo, come viene illustrato nel seguente frammento di codice:

```
if (parentNodePath = null || parentNodePath.length() == 0)
    //return the top-level of the source-node hierarchy
```

Se il percorso del nodo principale *non* è vuoto, `getTreeNodes()` deve creare i nodi tree per gli elementi secondari del percorso del nodo principale specificato, riportando l'array appropriato degli oggetti `TreeNode` a Procedura guidata dell'oggetto Business.

Figura 62 illustra l'implementazione del metodo `getTreeNodes()` (definito nella classe `ArmyAgent3` dell'esempio dell'esercito romano ODA).

```
public TreeNode[] getTreeNodes(String parentNodePath, String searchPattern)
   throws ODKException
{
   if (parentNodePath == null || parentNodePath.length() == 0)
      return getNodes(m_army, searchPattern);
   return getNodes(findSon(parentNodePath, searchPattern));
}
```

*Figura 62. Creazione di array di nodi tree*

Figura 62 illustra un importante concetto nell'implementazione del metodo `getTreeNodes()`. Questo metodo viene spesso suddiviso in moduli, ponendo la ricerca attuale dell'origine di dati in un metodo separato oppure in una classe separata. Questo metodo `getTreeNodes()` richiama il metodo `getNodes()` per generare attualmente l'array di nodi tree per i dati dell'origine di dati selezionata. Se il percorso del nodo principale è vuoto, `getTreeNodes()` invia a `getNodes()` l'intero contenuto del file XML (nella variabile `m_army`). Altrimenti, `getTreeNodes()` invia a `getNodes()` il risultato del metodo `findSon()`, che effettua l'interrogazione attuale dell'origine di dati.

## Implementazione della funzione del modello di ricerca

Un *modello di ricerca* consente di specificare criteri che i nodi secondari devono soddisfare per essere visualizzati quando viene espanso un nodo principale. La funzione del modello di ricerca viene avviata facendo clic con il pulsante destro del mouse e quindi facendo clic su **Ricerca di voci**. Il Viene visualizzata la finestra di dialogo Immetti un modello di ricerca, dove è possibile specificare i criteri di ricerca.

**Nota:** Per ulteriori informazioni su come utilizzare la funzione del modello di ricerca, consultare "Specifica dei modelli di ricerca" a pagina 87.

Quando la Procedura guidata dell'oggetto Business riceve un modello di ricerca, richiama di nuovo `getTreeNode()` per richiamare un nuovo array del nodo tree per l'origine di dati. La Procedura guidata dell'oggetto Business trasmette il modello di ricerca come un argomento a `getTreeNodes()`. Il modello di ricerca contiene i caratteri globali e altri simboli che viene riconosciuta dall'origine di dati sottostante. Ad esempio, se l'origine di dati è un database, i criteri di ricerca validi potrebbero includere simboli di ricerca SQL, quali un segno percentuale (%) o un punto interrogativo (?).

Il metodo `getTreeNodes()` ricerca nell'origine di dati i nodi secondari che corrispondono ai criteri di ricerca e copia i nodi secondari trovati nell'array del nodo tree, che viene restituito a Procedura guidata dell'oggetto Business. In tal modo, è possibile specificare in modo dinamico le nuove condizioni per i nodi di origine da soddisfare.

**Nota:** Con un modello di ricerca, diversamente che con un filtro, la Procedura guidata dell'oggetto Business richiama di nuovo il metodo `getTreeNodes()`. Con un filtro, la Procedura guidata dell'oggetto Business ricerca i nodi secondari del nodo principale che viene attualmente visualizzato, cioè la Procedura guidata dell'oggetto Business ricerca nodi secondari già contenuti nella gerarchia dei nodi di origine attuali. *Non* richiama `getTreeNodes()` per ricercare nell'origine di dati i nuovi nodi secondari corrispondenti.

Per implementare la funzione del modello di ricerca per ODA, eseguire le seguenti operazioni:

- Abilitare la funzione del modello di ricerca nei metadati di ODA (`AgentMetaData`) impostando la variabile membro `searchableNodes` su `true`.

  È inoltre necessario inizializzare la variabile membro `searchPatternDesc` in una stringa che descriva i criteri di ricerca validi per l'utente. La Procedura guidata dell'oggetto Business visualizza questa stringa come testo per la casella di dialogo Immetti il percorso di ricerca. I metadati di ODA vengono inizializzati nel metodo `getMetaData()`. Per ulteriori informazioni, consultare "Inizializzazione di metadati ODA" a pagina 113.
- Implementare il metodo `getTreeNodes()` per utilizzare il valore dell'argomento *searchPattern* nelle interrogazioni delle origini di dati effettuate.

  Ad esempio, se l'origine di dati è un database, è possibile includere il modello di ricerca nelle istruzioni SQL che interrogano le tabelle del database.

Nell'esempio dell'esercito romano ODA, il modello di ricerca consente all'utente di immettere una lettera come modello di ricerca. Il metodo `getTreeNodes()` richiama il metodo `getNodes()` per gestire la generazione attuale di nodi tree. Il seguente frammento di codice da questo metodo `getNodes()` (definito in `ArmyAgent3`) illustra come il metodo utilizza il modello di ricerca nella ricerca dell'origine di dati:

```
TreeNode[] getNodes(Son parent, String searchPattern)
{
   Vector nodes = new Vector();
   if (searchPattern == null || searchPattern.length() == 0)
      searchPattern = "";
   else
      searchPattern = new String(new char[] {searchPattern.charAt(0)});
```

Quando il metodo `getNodes()` confronta successivamente il nome dell'oggetto nel file XML (l'origine di dati) con il nome attuale nel percorso dei nodi principali, verifica che il nome dell'oggetto inizi con il modello di ricerca specificato, come segue:

```
if (currSon.name.getValue().startsWith(searchPattern))
```

Per il contesto di questo uso del modello di ricerca, consultare Figura 63 a pagina 128.

## Interrogazione dell'origine di dati

Lo scopo principale del metodo `getTreeNodes()` è di interrogare l'origine di dati per individuare i nodi di origine, che sono gli oggetti per i quali ODA può generare contenuto. Il meccanismo di interrogazione dell'origine di dati dipende dal tipo di origine di dati con cui opera ODA. Ad esempio, l'ODA XML (un ODA precostituito che è parte del prodotto WebSphere Business Integration Adapters) interroga i file XML per presentare i nomi degli oggetti all'interno di questi file per possibili generazioni di contenuto. Un altro esempio, l'ODA JDBC (un altro ODA precostituito, parte di WebSphere Business Integration Adapters) interroga un database JDBC per presentare i nomi delle tabelle all'interno del database per possibili generazioni di contenuto.

Come suggerito in Tabella 27 a pagina 108, se la logica necessaria per interrogare l'origine di dati è ragionevolmente complessa, occorre sviluppare classi Java speciali per gestire questa interazione. Il metodo `getTreeNodes()` può creare un'istanza e accedere a queste classi, se necessario. Accertarsi di includere queste classi nel file della libreria ODA. Per ulteriori informazioni, consultare "Compilazione di ODA" a pagina 173.

Per l'esempio dell'esercito romano ODA, il metodo `findSon()` (definito nella classe `ArmyAgent3`) esegue nell'attività di interrogazione dell'origine di dati. Individua un particolare soldato (identificato dal relativo percorso dei nodi principali) nel file XML dell'esercito romano (Roman-Army). Restituisce le informazioni per il nome specificato come un oggetto `Son`. L'esempio definisce la classe `Son` per leggere un oggetto nel file XML.

## Costruzione dei nodi tree

Poiché ODA effettua la ricerca di nodi di origine nell'origine di dati, deve generare i nodi tree associati per rappresentare ciascun nodo origine individuato. L'API ODK rappresenta un nodo della struttura come oggetto della classe `TreeNode` che contiene le informazioni riportate in Tabella 41.

*Tabella 41. Contenuto di un nodo della struttura*

| variabile membro | Descrizione |
|---|---|
| metadati | |
| `nome` | Il nome di questo nodo della struttura, che viene visualizzato nella colonna Nome della finestra Seleziona origine |
| `descrizione` | Una descrizione del nodo della struttura, che viene visualizzato nella colonna Descrizione della finestra Seleziona origine |
| `polymorphicNature` | Indica se la natura del nodo della struttura è "normale" (espandibile oppure un nodo secondario) o "file" (associata a un file) |
| `isExpandable` | Indica se il nodo della struttura è espandibile, ovvero se il nodo contiene altri nodi secondari oppure se è un nodo di fine |
| `isGeneratable` | Se il contenuto può essere generato per questo nodo della struttura |
| Dati | |
| `nodes` | Un array di nodi secondari, se questo nodo è espandibile. |

Per creare un nodo della struttura, utilizzare uno dei moduli del costruttore `TreeNode()`. Per un elenco di questi formati, consultare "TreeNode()" a pagina 294.

**Nodi di natura normale:** Quando un nodo ha una natura "normale", può avere una delle seguenti strutture:

- Un nodo espandibile

  La Procedura guidata dell'oggetto Business visualizza un nodo espandibile con un segno più (+) posto a sinistra del nome nodo (per indicare che l'utente può espandere il nodo) oppure un segno meno
  (-) (per indicare che l'utente può comprimere il nodo). Nella seguente tabella viene illustrata l'inizializzazione richiesta dall'oggetto `TreeNode` per essere visualizzato come un nodo espandibile:

| Variabile di membro TreeNode | Valore |
|---|---|
| isExpandable | true |
| nodes | Un array di nodi secondari |
| isGeneratable | false (solitamente) |

Per informazioni su come spostare la gerarchia dei nodi origine, consultare "Spostamento tra la gerarchia sorgente-nodo" a pagina 85

- Un nodo secondario

  Procedura guidata dell'oggetto Business visualizza un nodo secondario (di fine) copme nome del nodo. Nella seguente tabella viene illustrata l'inizializzazione richiesta dall'oggetto TreeNode per essere visualizzato come un nodo leaf:

| variabile membro Tree-node | Valore |
|---|---|
| isExpandable | false |
| nodes | null |
| isGeneratable | true (solitamente) |

Entrambi i nodi leaf e espandibile sono nodi di natura normale. Pertanto, entrambi avranno la variabile membro polymorphicNature impostata sulla costante della natura del nodo NODE_NATURE_NORMAL. Questa costante viene definita nell'interfaccia ODKConstant (che viene implementata dalla classe TreeNode). Le prime due forme di costruttore TreeNode() *non* specificano la variabile membro polymorphicNature. Pertanto, la variabile membro assume come valore predefinito NODE_NATURE_NORMAL.

Si supponga che ODA generi la gerarchia del nodo origine illustrata in Figura 61 a pagina 123. Se l'utente espande il nodo Uulius getTreeNodes() deve generare un array di nodo tree che contenga i nodi secondari di Uulius. Poiché il nodo principale *non* è vuoto (è Apollo:Tellus:Uulius), il metodo getTreeNodes() effettua la seguente chiamata al metodo getNodes() (consultare Figura 62 a pagina 124):

```
getNodes(findSon(parentNodePath), searchPattern))
```

Questa chiamata a getNodes() utilizza il metodo findSon() per interrogare l'origine di dati per il nodo Uulius e restituire l'oggetto Son che contiene le informazioni dal file XML. Una delle variabili membro in questo oggetto Son è un vettore di oggetti XML (XmlObjectVector) con l'informazioni sugli elementi secondari di Uulius. Figura 63 illustra un frammento di codice dal metodo getNodes() che va in loop attraverso questo vettore di oggetto XML e crea un oggetto TreeNode per ciascun elemento secondario:

```
for (int i=0; i<sons.size(); i++)
   {
   Son currSon = (Son) sons.getAt(i);
   if (currSon.name.getValues().startsWith(searchPattern))
      {
      int age = currSon.age.getIntValue();
      int children = currSon.Son == null ? 0 : currSon.Son.size();
      int nature = TreeNode.NODE_NATURE_NORMAL;
      TreeNode tn = new TreeNode(currSon.name.getValue(), " ",
         canRecruit(currSon), children > 0, null, nature);
      nodes.add(tn);
      }
   }
```

*Figura 63. Costruzione di nodi tree*

Il frammento di codice in Figura 63 inizializza un nuovo oggetto `TreeNode` per ciascun soldato secondario con il nome del soldato, se questo nome è generabile (basato sul fatto che il soldato abbia un'età valida per il reclutamento) e se questo nodo è espandibile (basato sul numero di figli del soldato). Questa chiamata al costruttore `TreeNode()` *non* inizializza il nodo tree con una descrizione ("") e non fornisce alcun nodo secondario. Una volta che viene creata un'istanza per ogni nuovo oggetto `TreeNode` , il codice aggiunge l'oggetto ad un `vettore` Java (`nodes`). Quando `getNodes()` ha generato gli oggetti `TreeNode` per *tutti* i nodi secondari, copia il contenuto di questo vettore in un nodo tree con il seguente codice:

```
TreeNode[] tn = new TreeNode[nodes.size()];
System.arraycopy(nodes.toArray(), 0, tn, 0, nodes.size());
```

Il metodo `getNodes()` restituisce questo array di nodo tree a `getTreeNodes()`, che volta per volta restituisce questo array al programma chiamante, Procedura guidata dell'oggetto Business. La Procedura guidata dell'oggetto Business utilizza questo nuovo array di nodo tree per visualizzare il contenuto espanso del nodo Uulius, come viene illustrato in Figura 64.

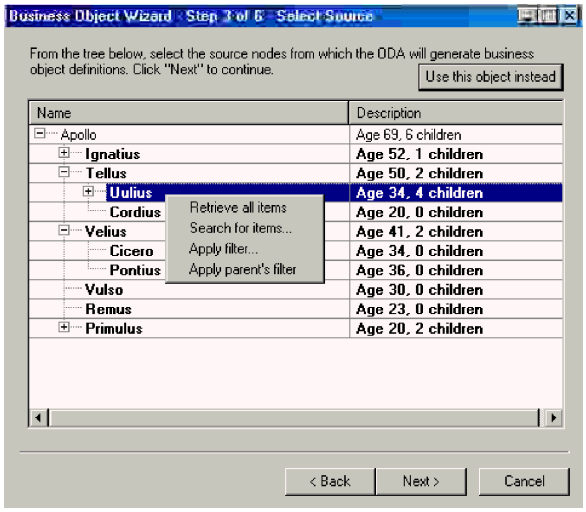



*Figura 64. Espansione del nodo origine Uulius*

**Nodi di natura file:** Quando un nodo ha una natura di tipo "file", l'utente può associare un file del sistema operativo al nodo. La Procedura guidata dell'oggetto Business indica che un nodo ha una natura file attivando la voce di menu Associa file quando l'utente fa doppio clic con il pulsante destro sul nome del nodo. Per

informazioni su come utilizzare questa voce di menu, consultare "Associazione di un file del sistema operativo" a pagina 88.

Un nodo con natura file ha la variabile `polymorphicNature` impostata sulla costante della natura del nodo `NODE_NATURE_FILE`. Questa costante viene definita nell'interfaccia `ODKConstant` (che viene implementata dalla classe `TreeNode`). Nella seguente tabella viene illustrata l'inizializzazione richiesta dall'oggetto `TreeNode` per funzionare come un nodo di natura file:

| Variabile membro TreeNode | Valore |
|---|---|
| `polymorphicNature` | `NODE_NATURE_FILE` |
| `isExpandable` | `false` |
| `nodes` | `null` |
| `isGeneratable` | `false` (solitamente) |

Nell'esempio dell'esercito romano la classe `ArmyAgent4` implementa un metodo `getNode()` che supporta i nodi di natura file. Questo esempio consente all'utente di associare un file con un nodo di origine per ciascun nodo che rappresenta un soldato di almeno 28 anni (l'età minima predefinita) e non ha figli. Il codice per questa versione di `getNode()` è abbastanza identica al codice in Figura 63 a pagina 128. L'unica differenza è nell'assegnazione del valore ala variabile membro `polymorphicNature`. Invece di assegnare la costante `NODE_NATURE_NORMAL` a *tutti* i nodi, la versione `ArmyAgent4` di `getNode()` utilizza la seguente riga di codice per impostare la natura del nodo su `NODE_NATURE_FILE` se il nodo rappresenta un soldato di almeno 38 anni che non ha figli:

```
int nature = m_allowAdoption && canAdopt(currSon) ?
   TreeNode.NODE_NATURE_FILE : TreeNode.NODE_NATURE_NORMAL;
```

## Creazione di definizioni di oggetti business

Dopo che gli utenti hanno selezionato i nodi di origine nella casella di dialogo Seleziona nodi (passo 3), l'ODA è pronta per iniziare la generazione di contenuto. La Procedura guidata dell'oggetto Business richiama il metodo di generazione del contenuto `generateBoDefs()` per generare le definizioni di oggetti business per i nodi di origine selezionati dall'utente. Per ODA la Procedura guidata dell'oggetto Business invia l'elenco dei nodi di origine (selezionato nel passo 3). L'obiettivo del processo di generazione della definizione di oggetti business è di creare una definizione di oggetti business per ciascun nodo di origine selezionato. Mentre viene eseguito il metodo `generateBoDefs()`, la Procedura guidata dell'oggetto Business visualizza il pannello Creazione di oggetti business (passo 5).

**Nota:** Poiché l'ODA genera le definizioni di oggetti business "su richiesta", la Procedura guidata dell'oggetto Business richiama in modo esplicito il metodo `generateBoDefs()` per iniziare la generazione dei file. Pertanto, *occorre* implementare `generateBoDefs()` in modo che gestisca la generazione delle definizioni di oggetti business (oggetti `BusObjDef`), memorizzandoli nella struttura di contenuto generato e restituendo i metadati di contenuto a Procedura guidata dell'oggetto Business.

In questa sezione vengono descritte le seguenti operazioni che devono essere effettuate dal metodo `generateBoDefs()` per generare le definizioni di oggetti business:

1. "Definizioni del metodo generateBoDefs()" a pagina 130
2. "Richiesta di proprietà di oggetti business" a pagina 130

3. “Creazione di definizioni di oggetti business” a pagina 134
4. “Aggiunta di definizioni di oggetti business generati” a pagina 142

## Definizioni del metodo generateBoDefs()

Per fornire la generazione di definizioni di oggetti business, la classe ODA (derivata da `ODKAgentBase2`) deve implementare il metodo `generateBoDefs()`, definito nell'interfaccia `IGeneratesBoDefs`. Il metodo `generateBoDefs()` riceve questi nodi di origine selezionati dall'utente come un argomento, un array dei percorsi dei nodi di origine (oggetti`String`). Il metodo deve generare una definizione di oggetti business per ciascun nodo di origine in questo array. Può utilizzare il percorso per individuare il nodo di origine nell'origine di dati. Come ultima operazione, `generateBoDefs()` restituisce un oggetto di metadati di contenuto (`ContentMetaData`) per descrivere le definizioni di oggetti business che ha generato.

L'esempio dell'esercito romano ODA supporta il protocollo di contenuto a richiesta per la generazione di definizioni di oggetti business (consultare Figura 60 a pagina 120). L'implementazione di questo metodo `generateBoDefs()` nella classe `ArmyAgent3` comprende il frammento di codice in Figura 65, che dichiara la variabile per la struttura (`m_generatedBOs`) e definisce il metodo `generateBoDefs()`.

```
final Vector m_generatedBOs = new Vector();
public ContentMetaData generateBoDefs(String[] nodes)
    throws ODKException
{
```

*Figura 65. Definizione del metodo generateBoDefs()*

## Richiesta di proprietà di oggetti business

Se, durante il processo di generazione contenuto, ODA richiede informazioni aggiuntive, può visualizzare la casella di dialogo Proprietà BO e richiedere i valori per le *proprietà dell'oggetto di business*. Per un'introduzione sulle proprietà degli oggetti business, consultare “Ottenimento delle proprietà di oggetti business” a pagina 101.

Figura 66 illustra un esempio della casella di dialogo Proprietà BO che visualizza le due proprietà degli oggetti business:

- La proprietà degli oggetti business `Verbs` consente agli utenti di specificare quali verbi sono supportati dalle definizioni di oggetti business. Questa proprietà fornisce un elenco a discesa dei verbi validi, tra cui l'utente può selezionare uno o più valori.
- La proprietà di oggetti business `Prefix` consente agli utenti di immettere il prefisso (quali `JDBC`, `SAP`, `LegacyApp`) da aggiungere ai nomi di tutte le definizioni di oggetti business generati. Questa proprietà fornisce solo un campo vuoto in cui gli utenti specificano il prefisso della stringa.

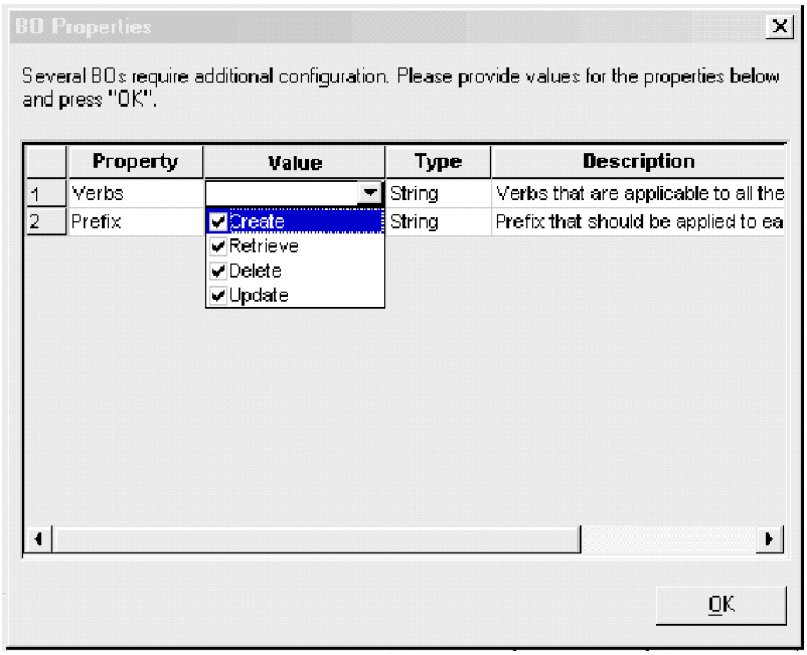



*Figura 66. Informazioni aggiuntive richieste sulle proprietà*

Per fornire le proprietà illustrate in Figura 66, il metodo `generateBoDefs()` effettua le seguenti operazioni:

1. Crea un array di proprietà di business per le proprietà Verbs e Prefix.
2. Richiama il metodo `getBOSpecificProps()` per visualizzare le proprietà di oggetti business.
3. Ottiene i valori inizializzati dall'utente per le proprietà di oggetti business.

**Creazione di un array di proprietà business:** Il metodo `getBOSpecificProps()` richiede un array degli oggetti di proprietà dell'agente come un argomento. Questo argomento è l'*array delle proprietà di oggetti business* e contiene un oggetto di proprietà dell'agente per ciascuna proprietà di oggetti business da visualizzare nella casella di dialogo Proprietà BO. Prima che `generateBoDefs()` richiami `getBOSpecificProps()`, deve effettuare le operazioni necessarie per creare l'array che definisce le proprietà di oggetti business, inizializzare le proprietà dell'oggetto business e salvarle nell'array.

Il primo passo è di definire un'array di proprietà di oggetti business per mantenere le proprietà Verbs e Prefix. Il passaggio successivo consiste nell'inizializzare le proprietà dell'oggetto business mediante il costruttore `AgentProperty()`. Con questo costruttore, vengono specificati i valori per i vari metadati supportati dalla classe `AgentProperty`. La classe `AgentProperty` fornisce supporto per la proprietà di oggetti business per avere le seguenti funzioni:

- Un valore predefinito
- La capacità di limitare i valori a solo un valore oppure a più di un valore
- Un elenco di valori validi tra cui l'utente può effettuare una selezione
- Condizioni che limitano il valore che l'utente può immettere

**Nota:** Per ulteriori informazioni, consultare "Operazioni con le proprietà agente" a pagina 153.

Per inizializzare le proprietà Verbs e Prefix, occorre fornire al costruttore `AgentProperty()` le seguenti informazioni:

- La proprietà Verbs è una proprietà di più cardinalità che fornisce più valori tra cui l'utente può effettuare una scelta. Inoltre, ha dei valori predefiniti.

  Pertanto, per questa proprietà sono necessari i seguenti metadati nel costruttore `AgentProperty()`

| metadati | Variabile membro AgentProperty | Valore |
|---|---|---|
| Molteplici cardinalità | `cardinalità` | `ODKConstant.MULTIPLE_CARD` |
| Consente all'utente di effettuare una scelta tra più valori | `isMultiple` | `true` |
| | `allValidValues` | array `validValues` (che contiene i valori validi da visualizzare) |
| Fornisce i valori predefiniti | `allDefaultValues` | array `defaultValues` (che contiene i valori validi da visualizzare) |
| Non viene richiesto all'utente di immettere un valore | `isRequired` | `false` |

  Prima della chiamata al costruttore `AgentProperty()`, il codice Figura 67 a pagina 133 prima crea e inizializza l'array `validValues` e `defaultValues` in modo che siano disponibili per il costruttore.

- La proprietà Prefix è una proprietà a cardinalità singola che *non* visualizza più valori tra cui l'utente può scegliere. Non ha un valore predefinito.

  Pertanto, per questa proprietà sono necessari i seguenti metadati nel costruttore `AgentProperty()`

| metadati | Variabile membro AgentProperty | Valore |
|---|---|---|
| Cardinalità singola | `cardinalità` | `ODKConstant.SINGLE_CARD` |
| Non consente all'utente di effettuare una scelta tra più valori | `isMultiple` | `false` |
| | `allValidValues` | `null` |
| Non fornisce i valori predefiniti | `allDefaultValues` | `null` |
| Non viene richiesto all'utente di immettere un valore | `isRequired` | `false` |

Il frammento del codice in Figura 67 crea e inizializza l'array delle proprietà di oggetti business:

```
// Create the business-object-property array
   AgentProperty AgtProps[] = new AgentProperty[2];

  // Provide list of valid values for Verbs property
   Object[] validValues = new Object[4];
   validValues[0] = new String("Create");
   validValues[1] = new String("Retrieve");
   validValues[2] = new String("Delete");
   validValues[3] = new String("Update");

 // Provide list of default values for Verbs property
   Object[] defaultValues = new Object[4];
   defaultValues[0] = new String("Create");
   defaultValues[1] = new String("Retrieve");
   defaultValues[2] = new String("Delete");
   defaultValues[3] = new String("Update");

// Instantiate the Verbs property
   AgtProps[0] = new AgentProperty("Verbs", AgentProperty.TYPE_STRING,
         "Verbs that are applicable to all the selected objects",
         false, true, ODKConstant.MULTIPLE_CARD, validValues,
         defaultValues);

// Instantiate the Prefix property
   AgtProps[1] = new AgentProperty("Prefix", AgentProperty.TYPE_STRING,
         "Prefix that should be applied to each business object name",
         false, false, ODKConstant.SINGLE_CARD, null, null);
```

*Figura 67. Creazione di array di oggetti business*

Per ulteriori informazioni sui metadati delle proprietà di oggetti business, consultare "Operazioni con le proprietà agente" a pagina 153.

**Visualizzazione della casella di dialogo BO:** Una volta inizializzata l'array di proprietà dell'oggetto business, l'ODA può richiamare il metodo `getBOSpecificProps()` per inviare l'array alla Procedura guidata dell'oggetto Business per la visualizzazione all'utente nella finestra Proprietà BO. Questo metodo è definito nella classe `ODKUtility` e pertanto deve accedere a un oggetto `ODKUtility`. In genere, viene creata un'istanza di questo oggetto come parte dell'inizializzazione ODA. Per ulteriori informazioni, consultare "Ottenimento dell'oggetto ODKUtility" a pagina 111.

**Nota:** Se vi sono valori delle proprietà non validi, `getBOSpecificProps()` genera l'eccezione `ODKInvalidPropException`.

La chiamata a `getBOSpecificProps()` in Figura 68 invia l'array `AgtProps` (inizializzato in Figura 67) alla Procedura guidata dell'oggetto Business per essere visualizzato nella casella di dialogo Proprietà BO.

```
// Display BO Properties dialog box, initializing it with AgtProps
   Util.getBOSpecificProps(AgtProps, "For all the Tables selected");
```

*Figura 68. Visualizzazione della casella di dialogo Proprietà BO*

**Richiamo di valori specifici dell'utente:** Una volta gli utenti hanno specificato i valori per le proprietà di oggetti business nella casella di dialogo Proprietà BO e fatto clic su **Avanti**, la Procedura guidata dell'oggetto Business invia di nuovo i valori specificati dall'utente all'ODA. L'ODA può richiedere questi valori in entrambi i seguenti dei due modi seguenti:

- La Procedura guidata dell'oggetto Business salva i valori specificati dall'utenti nell'oggetto `Hashtable` Java e invia questo oggetto come il valore di ritorno per `getBOSpecificProps()`. Ciascuna proprietà viene fissata sul nome in questo oggetto `Hashtable`. L'ODA può accedere queste proprietà con i metodi

Hashtable. I valori specificati dall'utente per la proprietà sono nella variabile membro allValues del relativo oggetto proprietà agente AgentProperty).

- La Procedura guidata dell'oggetto Business scrive i valori specificati dall'utente nella memoria runtime di ODA. L'ODA può accedere a questi valori con il metodo getBOSpecificProperty() o getAllBOSpecificProperties() nella classe ODKUtility.

La chiamata getBOSpecificProps() in Figura 68 non salva l'oggetto Hashtable creato dalla Procedura guidata dell'oggetto Business. Pertanto, questo frammento di codice utilizza il metodo getBOSpecificProperty() per richiamare il valore delle proprietà specificare per i verbi e ciascun prefisso di oggetti business:

```
// Get the value of the Verbs and the Prefix properties
AgentProperty propVerb =
      Util.getBOSpecificProperty("Verbs");
AgentProperty propPrefix =
      Util.getBOSpecificProperty("Prefix");
```

## Creazione di definizioni di oggetti business

Il metodo generateBoDefs() deve generare una definizione di oggetto business per ciascun nodo di origine nell'array che riceve come un argomento dalla Procedura guidata dell'oggetto Business. Per generare la definizione di oggetti business, generateBoDefs() effettua le seguenti operazioni:

1. Utilizzare il percorso dei nodi di origine, dall'array che generateBoDefs() riceve dalla Procedura guidata dell'oggetto Business, per individuare l'oggetto associato nell'origine di dati.
2. Reperire tutte le informazioni necessarie a popolare la definizione di oggetti business dall'oggetto associato nell'origine di dati.
3. Creare un oggetto di definizione di oggetti business per rappresentare il nodo di origine.
4. Popolare quest'oggetto di definizione di oggetti business con le informazioni ottenute dall'oggetto associato nell'origine di dati (consultare 2).

L'API ODK rappresenta una definizione di oggetti business come oggetto della definizione (BusObjDef). E' possibile utilizzare il costruttore BusObjDef() per creare un'istanza della nuova definizione di oggetti business e fornirle un nome. È possibile fornire la definizione di oggetti business con le informazioni fornite in Tabella 42.

*Tabella 42. Contenuto di un definizione dell'oggetto business*

| Informazioni sulle definizioni di oggetti business | Descrizione | Metodo del dispositivo di accesso |
|---|---|---|
| metadati | | |
| Nome | Il nome della definizione di oggetti business | getName() |
| Informazioni specifiche sull'applicazione | Informazioni specifiche dell'applicazione del livello di oggetti business, che contengono le informazioni applicabili all'intera definizione di oggetti business | getAppInfo(), setAppInfo() |
| Dati | | |

*Tabella 42. Contenuto di un definizione dell'oggetto business (Continua)*

| Informazioni sulle definizioni di oggetti business | Descrizione | Metodo del dispositivo di accesso |
|---|---|---|
| Elenco attributi | Un elenco degli attributi nella definizione di oggetti business; ciacun attributo è un oggetto `BusObjAttr`. | `getAttributeList()`, `setAttributeList()`, `insertAttribute()`, `removeAttribute()` |
| Elenco verbi | Un elenco dei verbi supportati nella definizione di oggetti business; ciascun verbo è un oggetto `BusObjVerb`. | `getVerbList()`, `setVerbList()`, `insertVerb()`, `removeVerb()` |

Come illustrato in Tabella 42, una definizione di oggetti business contiene metadati e dati. Nella seguente sezione viene descritto come accedere a queste parti della definizione di oggetti business:

- "Definizione dei metadati per la definizione di oggetti business"
- "Creazione di attributi" a pagina 137
- "Assegnazione di verbi supportati" a pagina 140

**Definizione dei metadati per la definizione di oggetti business:** Come viene illustrato in Tabella 42, i metadati di una definizione di oggetti sono costituiti dalle seguenti informazioni:

- Nome della definizione di oggetti business
- Informazioni specifiche dell'applicazione (a livello di definizione di oggetti business)

*Assegnazione di un nome alla definizione di oggetti business:* Il metodo `generateBoDefs()` riceve un elenco di nodi di origine selezionati dall'utente come un argomento. Questo elenco è un array degli oggetti `String` che contengono i percorsi di nodi per i nodi di origine selezionati dall'utente. Per informazioni sui percorsi di nodi, consultare "Individuazione del percorso del nodo principale" a pagina 122. Con quest'array, ODA deve creare il nome appropriato per la definizione di oggetto di business associata con ciascun nodo di origine. Di solito, ODA fa in modo che il nome della definizione di oggetti business (o basata su di essi) corrisponda al nome dell'oggetto dell'origine di dati rappresentato dal nodo di origine. ODA deve analizzare il percorso del nodo di origine per ottenere il nome del nodo di origine, utilizzare il nome per individuare l'oggetto di origine dati associato, quindi ottenere il nome dall'oggetto di origine di dati.

Ad esempio, nell'esempio dell'esercito romano, i nomi degli oggetti dell'origine di dati corrispondono alle definizioni di oggetti business. Pertanto, il codice di esempio richiama il metodo `findSon()` (definito nelle classi `ArmyAgent3` e `ArmyAgent4`) per ottenere l'oggetto di origine di dati che il nodo di origine rappresenta, utilizzando il nodo di origine dall'array di immissione dei nodi di origine (`nodes`), come illustrato qui di seguito:

```
for (int i=0; i<nodes.length; i++)
   {
   Son sonNode = findSon(nodes[i]);
   BusObjDef sonBo = new BusObjDef(sonNode.name.getValue());
   ...
```

**Nota:** Tutti i moduli del costruttore `BusObjDef()` specificano il nome della definizione dell'oggetto business.

Il metodo `findSon()` analizza il percorso del nodo di origine per ottenere il nome dell'ultimo nodo nel percorso.

Come altro esempio, si immagini che l'origine di dati sia un database e i relativi nodi di origine rappresentino delle tabelle. Se i percorsi dei nodi di origine comprendono i nomi degli schemi (*schema:tabella*), per ODA è necessario analizzare i percorsi dei nodi di origine per assegnare solo un nome di tabella alle definizioni di oggetti business corrispondenti. Se ODA supporta un prefisso specifico dell'utente per le definizioni di oggetti business (con una variabile di configurazione), ODA deve preporre questo prefisso *prima*di richiamare il costruttore `BusObjDef()` per creare un oggetto di definizione di oggetti business, come viene illustrato nel seguente frammento di codice:

```
AgentProperty propPrefix = getBOSpecificProperty("Prefix");
for (int i=0; i<names.length; i++)
   {
   strToken = new StringTokenizer(names[i], ":");
   schemaName = strToken.nextToken();
   tableName = strToken.nextToken()
   if (propPrefix.allValues != null && propPrefix.allValues[0] != null)
      boDef = new BusObjDef(propPrefix.allValues[0] + tableName);
   else
      boDef = new BusObjDef(tableName);
   ...
```

Se gli oggetti di dati *non* hanno esattamente i nomi che si desidera assegnare alle definizioni di oggetti business, ODA deve analizzare in qualche modo il formato di questi nomi come necessario.

*Creazione delle informazioni specifiche delle applicazioni di oggetti business:* Come viene descritto in "Informazioni specifiche dell'applicazione dell'oggetto business" a pagina 8, le informazioni specifiche delle applicazioni sono un modo efficace di inserire informazioni di elaborazione specifiche delle applicazioni all'interno di una definizione di oggetti business. Spostando queste informazioni dal programma di elaborazione (come un connettore), il programma di elaborazione può essere guidato dai metadati, cioè può essere scritto in un modo più generico, e ottenere le istruzioni di elaborazione specifiche per le applicazioni dalla definizione di oggetti business. Pertanto, se le definizioni di oggetti business devono essere utilizzate con programmi di elaborazione guidati dai metadati, è importante che essi comprendano le informazioni specifiche delle applicazioni formattate correttamente a livello di oggetto business, attributo e verbo.

**Nota:** Per informazioni sulle informazioni sugli attributi che sono specifiche delle applicazioni, consultare "Creazione di attributi" a pagina 137. Per informazioni sulle informazioni sui verbi specifiche delle applicazioni, consultare "Assegnazione di verbi supportati" a pagina 140.

Le definizioni di oggetti business generate dall'esempio dell'esercito romano non forniscono informazioni specifiche delle applicazioni. Tuttavia, si supponga che l'origine di dati era un database con le tabelle come nodi di origine. L'ODA potrebbe generare le definizioni di oggetti per ciascuna tabella selezionata dall'utente. In queste definizioni di oggetti business, occorre includere il nome della tabella come informazioni specifiche delle applicazioni a livello di oggetti business. Pertanto, questo frammento di codice utilizza il metodo `setAppInfo()`, definito nella classe `BusObjDef`, per creare le coppie di valori di nomi appropriati per queste informazioni specifiche delle applicazioni:

```
boDef.setAppInfo("TN=" + tableName + ";SCN=" + schemaName +";");
```

Questo codice crea le coppie di valori nome TN e SCH per rappresentare rispettivamente i nomi della tabella e dello schema. Concatena il nome della tabella e quello dello schema con il tag utilizzato per denominare l'elemento. Quindi utilizza il metodo per assegnare setAppInfo() questa intera stringa come informazioni specifiche delle applicazioni a livello di oggetti business.

**Creazione di attributi:** Una definizione di oggetti business contiene *attributi*, che descrivono l'oggetto che la definizione di oggetti business rappresenta. La definizione di oggetti business conserva gli attributi nell'*elenco attributi*. L'API ODK rappresenta un attributo come un oggetto di attributo (BusObjAttr). Per creare un'istanza di un oggetto attributo, utilizzare il costruttore BusObjAttr().

Tabella 43 riepiloga le proprietà di un oggetto attributo. Tali proprietà corrispondono ai metadati dell'attributo.

*Tabella 43. Proprietà di un attributo*

| Proprietà attributi | Descrizione | Metodo del dispositivo di accesso |
|---|---|---|
| Nome | Il nome dell'attributo | getName(), setName() |
| Informazioni specifiche sull'applicazione | Informazioni specifiche dell'applicazione del livello di oggetti attributi, che contengono le informazioni applicabili all'attributo | getAppText(), setAppText() |
| Tipo | Il tipo di dati del valore dell'attributo | getAttrType(), getAttrTypeName(), setAttrType() |
| Cardinalità | La cardinalità dell'attributo, che identifica il numero di valori contenuti dall'attributo | getCardinality(), setCardinality() |
| Valore predefinito | Il valore da assegnare all'attributo prima che l'utente immetta un valore | getDefault(), setDefault() |
| Maxlength | La lunghezza massima del valore dell'attributo | getMaxLength(), setMaxLength() |
| Commenti | I commenti opzionali per descrivere lo scopo dell'attributo | getComments(), setComments() |
| Tipo relazione | Una stringa per identificare il tipo di relazione in cui partecipa l'attributo | getRelationType(), setRelationType() |
| Chiave primaria | Se l'attributo è parte di una chiave primaria | isKey(), setIsKey() |
| Chiave esterna | eSe l'attributo è parte di una chiave esterna | isForeignKey(), setIsForeignKey() |
| Chiave richiesta | Se l'attributo è richiesto | isRequiredKey(), setIsRequiredKey() |

**Importante**

Il processo do generazione di definizioni di oggetti business crea automaticamente l' attributo ObjectEventId. Se la Procedura guidata dell'oggetto Business salva la definizione di oggetti business in un file, aggiunge automaticamente la versione del repository all'inizio del file. La versione repository è necessaria per InterChange Server Express.

Nell'esempio dell'esercito romano ODA, ciascuna definizione di oggetti business rappresenta un soldato romano. Il metodo `generatesBoDefs()` crea i seguenti attributi per la definizione di oggetti business:

- L'attributo Età contiene l'età del soldato romano.
- L'attributo ChildNo contiene il numero di figli del soldato romano (adottivi o meno).

Figura 69 contiene un frammento di codice che crea tali oggetti di attributi per la definizione di oggetti business.

```
// 1. Create an attribute object for Age attribute
   BusObjAttr attr = new BusObjAttr("Age", BusObjAttrType.INTEGER,
      BusObjAttrType.AttrTypes[BusObjAttrType.INTEGER]);
//    Set the Age attribute as the business object definition's key
   attr.setIsKey(true);
//    Add the attribute to the business object definition's attribute list
   sonBo.insertAttribute(attr);
// 2. Create an attribute object for ChildNo attribute
   attr = new BusObjAttr("ChildNo", BusObjAttrType.INTEGER,
      BusObjAttrType.AttrTypes[BusObjAttrType.INTEGER]);
//    Set the default value to number of children
   attr.setDefault(sonNode.Son == null ? "0" : "" + sonNode.Son.size());
//    Add the attribute to the business object definition's attribute list
   boDef.insertAttribute(attr);
```

*Figura 69. Creazione di attributi*

Per creare un attributo Age, il frammento di codice in Figura 69 eseguire le seguenti operazioni:

1. Utilizzare Costruttore `BusObjAttr()` per creare l'oggetto attributi Age (`attr`). Utilizza il formato di questo costruttore che inizializza l'oggetto di attributi con questo nome, tipo e nome del tipo.

   Per inizializzare il tipo, il codice specifica la costante del tipo attributo per Integer ( `BusObjAttrType.INTEGER`). Per inizializzare il nome del tipo, utilizza la variabile membro `AttrTypes` nell'interfaccia `BusObjAttrType`. Questa variabile membro statico fornisce i nomi del tipo per tutti i tipi di attributi supportati e può essere indicizzato dalle costanti del tipo di attributo. In tal modo, è possibile assegnare il nome tipo senza proteggere la stringa del nome tipo.
2. Utilizzare Il metodo `setIsKey()` per impostare in modo esplicito la proprietà di chiave primaria su true.

   Poiché questo formato del costruttore `BusObjAttr()` specifica solo tre proprietà degli attributi, tutte le altre proprietà degli attributi assumono come valore predefinito "undefined". Pertanto, dopo la chiamata `BusObjAttr()`, la proprietà dell'attributo della chiave primaria è false. Per indicare che l'attributo Age *è* l'attributo chiave, il campione di codice chiama `setIsKey()`.
3. Utilizzare `insertAttribute()`, definito nella classe `BusObjDef`per aggiungere l'attributo Age all'elenco di attributi della definizione di oggetti business.

Il frammento del codice in Figura 69 ripete queste operazioni di base per generare l'attributo ChildNo. La differenza principale è che, poiché ChildNo *non* è l'attributo chiave, non occorre nessuna chiamata `setIsKey()`. Tuttavia, il frammento di codice fornisce un valore predefinito per questo attributo richiamando il metodo `setDefault()`.

Le definizioni di oggetti business generati dall'esempio dell'esercito romano sono molto semplici. e . Esistono solo due attributi nella definizione di oggetti business e i relativi nomi sono noti al momento della compilazione. Inoltre, devono essere impostate solo poche proprietà di attributi. Per un esempio più complesso, si immagini che l'origine di dati sia un database, con le tabelle come nodi di origine e come nomi delle definizioni di oggetti business. In questo caso, le colonne del database corrisponderebbero gli attributi della definizione di oggetti business. Sarebbe necessario impostare molti altre proprietà degli attributi per questi attributi.

Il seguente frammento di codice crea attributi per le colonne in una tabella di database:

```
Vector Attributes;
// 1. Retrieve columns from database table into 'rst' result set
try{
   ResultSet rst = null;
   // Retrieve columns from database
   rst = db.dbmd.getColumns(null, schemaName, tableName, "%");

   String colName = null;
   String colType = null;
   int cType = 0;
   int colSize = 0;
   // Obtain next column from result set
   rst.next();
   do{
      // Get column name & type
      colName = rst.getString(4);
      colType = rst.getString("DATA_TYPE");

      // Convert database types to supported types.
      // Load converted types into the cType variable
      // (steps not shown)
// 2. Create an attribute object for each column in the result set.
      Attributes = new Vector(1, 10);
      try
        {
         // Create attribute object for column
         BusObjAttr attrib = new BusObjAttr(colName, cType);
         // Set the cardinality and maxLength attribute properties
         attrib.setCardinality(BusObjAttr.CARD_SINGLE);
         colSize = rst.getInt("COLUMN_SIZE");
         attrib.setMaxLength(colSize);
         // Determine whether it is a primary key in the table: compare
         // column name against earlier retrieve of table's primary keys
         // (stored in pKeys -- code not included here)
         if (pKeys.contains(colName))== true {
            attrib.setIsKey(true);
         }else
            attrib.setIsKey(false);
         // Determine whether it is a foreign key in the table: compare
         // column name against earlier retrieve of table's primary keys
         // (stored in fKeys -- code not included here)
         if (fKeys.contains(colName))== true {
            attrib.setIsForeignKey(true);
         }else
            attrib.setIsForeignKey(false);
```

```
        // Set the isRequired property
        if ((rst.getString("IS_NULLABLE").equals("NO")) &&
              (attrib.isKey() != true)){
           attrib.setIsRequiredKey(true);
        }
       // Create attribute application-specific information:
       // CN tag provides column name
        String asi = "CN="+colName;
        attrib.setAppText(asi);
        attrib.setDefault("");
       // Add attribute object to Attributes vector
        Attributes.add(attrib);
        ...
// 3. Save the attribute vector as the business object
      //    definition's attribute list
       boDef.setAttributeList(Attributes);
```

In questo processo le operazioni da eseguire sono le seguenti:

1. Utilizzare il costruttore `BusObjAttr()` per creare un semplice attributo di oggetti business dalle informazioni delle colonne. Questo modulo del costruttore specifica solo il nome e il tipo dell'attributo.
2. Impostare le proprietà degli attributi cardinalità e maxLength, basati su questi valori dalla colonna nel database.

   **Nota:** Per creare un attributo che rappresenti un oggetto business secondario oppure un array di oggetti business secondari, specificare il nome dell'oggetto business secondario come tipo e impostare la cardinalità su 1 oppure `n`, quando necessario. Ad esempio, per creare un attributo denominato `LineItems` che rappresenta un array di oggetti business OrderLineItems, utilizzare il seguente codice:

   ```
   BusObjAttr attrib = new BusObjAttr(LineItems, OrderLineItems);
   attrib.setCardinality(BusObjAttr.CARD_MULTIPLE);
   ```
3. Richiama la chiave primaria e le informazioni sulla chiave esterna per impostare gli attributi che rappresentano la chiave primaria o esterna. Il frammento di codice confronta il nome della colonna corrente con i nomi degli array esistenti che contengono colonne delle chiave primarie (`pKey`) e colonne chiave esterne (`fKey`) selezionate dal database. Il codice che seleziona le colonne di chiavi primarie ed esterne non viene illustrato qui.
4. Impostare la proprietà dell'attributo "chiave richiesta", in base al fatto che l'attributo sia una chiave primaria.
5. Impostare le informazioni specifiche dell'applicazione di livello attributo.

   Per le definizioni di oggetti business generate dalle tabelle di database, è possibile includere il nome della colonna come informazioni specifiche dell'applicazione del livello di attributo per ogni attributo nella definizione di un oggetto business. Questo frammento di codice utilizza il metodo `setAppText()`, definito nella classe `BusObjAttr`, per creare le coppie nome-valore `CN` appropriati per queste informazioni specifiche delle applicazioni. Il codice concatena il nome della colonna con la tag `CN`. Quindi utilizza il metodo `setAppText()` per assegnare questa intera stringa come informazioni specifiche dell'applicazione dell'attributo.
6. Utilizzare il metodo `setAttributeList()`, definito nella classe `BusObjDef`, per assegnare il vettore di attributi generati (`Attributes`) come elenco di attributi della definizione di oggetti business.

**Assegnazione di verbi supportati:** Una definizione di oggetto business contiene *verbi* supportati, che descrivono le operazioni che possono essere eseguite sugli

oggetti business di questa definizione di oggetti business. La definizione di oggetti business contiene i verbi supportati nell'*elenco di verbi*. L'API ODK rappresenta un verbo come un oggetto di verbo (`BusObjVerb`). Per creare un'istanza di un oggetto verbo, utilizzare il costruttore `BusObjVerb()`.

Tabella 44 riepiloga i metadati di un oggetto verbo.

*Tabella 44. metadati per un verbo*

| Matadati verbo | Descrizione | Metodo del dispositivo di accesso |
|---|---|---|
| Nome | Il nome del verbo supportato (quale Create, Retrieve, Update oppure Delete) | `getName()`, `setName()` |
| Informazioni specifiche sull'applicazione | Le informazioni specifiche dell'applicazione di livello verbo, che contengono le informazioni applicabili solo al verbo | `getAppInfo()`, `setAppInfo()` |

Nell'esempio dell'esercito romano, il metodo `generatesBoDefs()` assegna a ciascuna definizione di oggetti business un verbo Create supportato. Pertanto, questo frammento di codice utilizza il metodo `insertVerb()` definito nella classe `BusObjDef` per aggiungere il verbo Create all'elenco di verbi della definizione dell'oggetto business:

```
sonBo.insertVerb("Create", null);
```

Le definizioni di oggetti business generate dall'esempio dell'esercito romano non forniscono informazioni specifiche delle applicazioni. Pertanto, il secondo argomento a questa chiamata `insertVerb()`, che fornisce le informazioni dei verbi specifiche dell'applicazione, è `null`.

ODA può utilizzare la casella di dialogo Proprietà BO affinché il verbo supporti le definizioni di oggetti business. Definendo una proprietà di oggetti business denominata Verbs e consentendo agli utenti di selezionare i verbi supportati, ODA può ottenere un supporto dei verbi più personalizzato. Per ulteriori informazioni sull'utilizzo della casella di dialogo Proprietà BO, consultare "Richiesta di proprietà di oggetti business" a pagina 130.

Nel seguente frammento di codice si presuppone che ODA abbia ottenuto valori specificati dall'utente per una proprietà di oggetti business denominata Verbs ed utilizza questa proprietà per ottenere i verbi dall'elenco di verbi della definizione di oggetti business.

```
Vector Verbs;
AgentProperty propVerbs = getBOSpecificProperty("Verbs");

if (propVerbs.allValues[0] != null)
   {
   int len = propVerbs.allValues.length;
   BusObjVerb verb;

   for(int i=0; i<len; i++)
      {
      if(propVerbs.allValues[i] != null)
         {
         try {
            verb = new BusObjVerb(propVerbs.allValues[i].toString(), "");
            Verbs.add(verb);
         }
```

```
      }
   }
...
boDef.setVerbList(Verbs);
```

Questo frammento di codice utilizza il costruttore `BusObjVerb()` per copiare un verbo nella variabile `verb` del tipo `BusObjVerb`. Carica quindi una versione `String` di questo oggetto verbo nel vettore `Verbs`. Il codice non specifica le informazioni dei verbi specifiche delle applicazioni. Infine, questo frammento di codice utilizza il metodo `setVerbList()` definito nella classe `BusObjDef` per assegnare il vettore dei verbi generati (`Verbs`) come elenco di verbi della definizione degli oggetti business.

## Aggiunta di definizioni di oggetti business generati

Come discusso in "Aggiunta di contenuto generato" a pagina 102, ODA deve restituire il contenuto generato alla Procedura guidata dell'oggetto Business in due parti. Pertanto, se ODA genera le definizioni di oggetti business come contenuto, deve restituire quanto segue:

- Una struttura del contenuto generato, che contiene le definizioni degli oggetti business generati
- Un oggetto di metadati del contenuto (`ContentMetaData`) che descrive le definizioni di oggetti business in una struttura del contenuto generato

Poiché l'ODA deve generare le definizioni di oggetti business "a richiesta", il metodo `generateBoDefs()` fornisce le informazioni di questo contenuto, nel modo seguente:

- Popola la struttura di contenuto generato. Questa struttura deve in qualche modo essere globale per la classe ODA così che `generateBoDefs()` e `getBoDefs()` possano accederla.
- Restituisce un oggetto di metadati di contenuto (`ContentMetaDatache` descrive gli oggetti di business generato al chiamante, Procedura guidata dell'oggetto Business.

  Una volta che la Procedura guidata dell'oggetto Business riceve questo oggetto di metadati di contenuto, può accedere alle definizioni di oggetti business generati (all'interno della struttura di contenuto generata) come necessario con il metodo `getBoDefs()`.

**Nota:** Per ulteriori informazioni su `getBoDefs()`, consultare "Accesso alle definizioni di oggetti business generati" a pagina 143.

Il campione di codice in Figura 70 illustra l'ultima parte del metodo `generateBoDefs()` dell'esempio dell'esercito romano ODA.

```
            m_generatedBOs.add(sonBo);
        } // this for loop terminates when all bus obj defs are generated
        return new ContentMetaData(ContentType.BusinessObject, -1,
           m_generatedBOs.size());
} // end of generateBoDefs()
```

*Figura 70. Aggiunta di definizioni di oggetti business generati*

Figura 70 gestisce il contenuto generato nel modo seguente:

- Il metodo `generatedBoDefs()` salva la definizione di oggetti business nella struttura di contenuto generato, `m_generatedBOs`.

  Come illustrato in Figura 65 a pagina 130, l'esempio dell'esercito romano utilizza un vettore Java denominato `m_generatedBOs` come struttura di contenuto generato. Questa struttura è globale per i metodi della classe ODA dell'esercito

romano. Per salvare la definizione di oggetti business che ha generato, `generateBoDefs()` lo salva nel vettore `m_generatedBOs`. Questa operazione avviene in un loop che determina quando ODA ha generato le definizioni di oggetti business per *tutti* i nodi di origine degli utent selezionati. Quando occorre che la Procedura guidata dell'oggetto Business acceda alla struttura di contenuto generato, richiama il metodo di richiamo contenuto `getBoDefs()`.

- Come operazione finale, `generateBoDefs()` restituisce i metadati di contenuto descritti nel contenuto generale.

  Il metodo `generateBoDefs()` crea un'istanza di un oggetto `ContentMetaData` e in questo costruttore trasferisce le informazioni illustrate in Tabella 45.

*Tabella 45. Inizializzazione dei metadati di contenuto per la generazione della definizione di oggetti business*

| Informazioni ContentMetaData | Codice | Descrizione |
|---|---|---|
| Tipo di contenuto | `ContentType.BusinessObject` | Indica che il tipo di contenuto siano le definizioni di oggetti business |
| Dimensione del contenuto generato | -1 | Indica che la dimensione totale *non* è richiesta. Il valore della lunghezza *non* è necessario nell'implementazione corrente di un oggetto `ContentMetaData`. |
| Conteggio del contenuto generato | `m_generatedBOs.size()` | Il metodo `size()` restituisce il numero di elementi attualmente nel vettore. |

## Accesso alle definizioni di oggetti business generati

Il metodo `generateBoDefs()` *non* restituisce le effettive definizioni di oggetti business generate. Affinché la Procedura guidata dell'oggetto Business possa accedere al contenuto generato, la classe ODA deve implementare il metodo di richiamo contenuto per le definizioni di oggetti business. La Procedura guidata dell'oggetto Business utilizza le informazioni nell'oggetto di metadati di contenuto (che `generateBoDefs()` *non* restituisce) per determinare se chiamare il metodo di richiamo contenuto appropriato. Se `generateBoDefs()` ha generato gli oggetti business generato, la Procedura guidata dell'oggetto Business richiama il metodo `getBoDefs()` per richiamare le definizioni dell'oggetto business generato.

**Nota:** In questo rilascio, la Procedura guidata dell'oggetto Business richiama sempre il metodo `generateBoDefs()` per iniziare la generazione di definizioni di oggetti business poiché ODA deve supportare protocollo di contenuto a richiesta. L'ODA *non* deve supportare il il protocollo del contenuto callback per la generazione delle definizione degli oggetti business. Per ulteriori informazioni sui protocolli di contenuto, consultare "Selezione del protocollo del contenuto ODA" a pagina 118.

Per fornire le definizioni di oggetti business, la classe ODA deve implementare il metodo `getBoDefs()`. Questo metodo è definito come parte dell'interfaccia `IGeneratesBoDefs`. Il metodo accetta come argomento un indice, che identifica il numero delle definizioni di oggetti business da restituire. Accede queste definizioni di oggetti business nella struttura di contenuto generato e restituisce un array degli oggetti (`BusObjDef`) della definizione di oggetti business. Il numero di definizioni di oggetti business in questo array dipende dall'argomento index, come viene illustrato in Tabella 46.

*Tabella 46. Richiamo di definizioni di oggetti business*

| Valore dell'indice | Descrizione | Numero di elementi in un array restituiti da getBoDefs() |
|---|---|---|
| Nell'intervallo da 0 a *numero* - 1, dove *numero* è il numero totale di definizioni di oggetti business nella struttura di contenuto generato | Specifica la posizione dell'indice nella struttura di contenuto generato della definizione di oggetti business da richiamare | Una definizione di oggetti business |
| `ODKConstant.GET_ALL_OBJECTS` | Costante speciale per indicare la restituzione di *tutte* le definizioni di oggetti business nella struttura di contenuto generato | Tutte le definizioni di oggetti business nella struttura di contenuto generato (*numero*) |

Per l'esempio dell'esercito romano ODA, il metodo `generateBoDefs()` (definito nella classe `ArmyAgent3`) popola il vettore `m_generatedBOs` con le definizioni aziendadi oggetti business generate. Pertanto, il metodo `getBoDefs()` (anche definito in `ArmyAgent3`) richiama il numero specificato delle definizioni di oggetti business da questo vettore e le copia nell'array di ritorno. Il seguente codice illustra il metodo `getBoDefs()` per l'esempio romano ODA:

```
public BusObjDef[] getBoDefs(long index) throws ODKException
{
   BusObjDef[] bos = null;
   if (index == ODKConstant.GET_ALL_OBJECTS)
      {
      bos = new BusObjDef[m_generatedBOs.size()]
      System.arraycopy(m_generatedBOs.toArray(), 0, bos, 0,
         m_generatedBOs.size());
      }
   else
      bos = new BusObjDef[] {(BusObjDef)m_generatedBOs.get((int)index)};
   return bos;
}
```

## Creazione dei file binari come contenuto

Un *file binario* è un file del sistema operativo che è rappresentato come oggetto Java `File`. Affinché un ODA generi il contenuto di file binari, la classe ODA deve implementare l'interfaccia `IGeneratesBinFiles`. Tabella 47 elenca i metodi che la classe ODA deve definire per implementare l'interfaccia `IGeneratesBinFiles`.

*Tabella 47. Metodi nell'interfaccia IGeneratesBinFiles*

| Metodo | Metodo IGeneratesBinFiles | Descrizione |
|---|---|---|
| Metodo di generazione di nodi di origine | Nessuno | Generazione di nodi di origine devono essere effettuati dal metodo `getTreeNodes()` dell'interfaccia `IGeneratesBoDefs`. Per ulteriori informazioni, consultare "Utilizzo di file" a pagina 145. |
| Metodo di generazione contenuto | `generateBinFiles()` | Genera i file binari, scrivendoli nella memoria ODA |
| Metodo di richiamo contenuto | `getBinFile()` | Richiama un file binario specificato oppure tutti i file binari dalla memoria ODA |

**Nota:** Oltre ai metodi in Tabella 47, `IGeneratesBinFiles` comprende anche il metodo `getContentProtocol()` per specificare il protocollo di contenuto supportato da ODA per la generazione di file. Per ulteriori informazioni, consultare "Selezione del protocollo del contenuto ODA" a pagina 118.

La Procedura guidata dell'oggetto Business genera e richiama il contenuto mentre visualizza la casella di dialogo Creazione di Business Objects (passo 5). Con l'interfaccia `IGeneratesBinFiles` implementata, la Procedura guidata dell'oggetto Business richiama i metodi illustrati in Tabella 48 per generare e richiamare contenuto.

*Tabella 48. Procedura guidata Business Object e metodi IGeneratesBinFiles*

| Utilizzo dei metodi | Metodo IGeneratesBinFiles | Per ulteriori informazioni |
|---|---|---|
| Creare file come contenuto | `generateBinFiles()` | "Creazione di file" a pagina 147 |
| Richiamare i file generati | `getBinFile()` | "Aggiunta di accesso ai file generati" a pagina 152 |

Nelle seguenti sezioni viene illustrata l'implementazione di ciascun metodo in Tabella 48.

## Utilizzo di file

Quando un ODA che implementa l'interfaccia `IGeneratesBinFiles`, può supportare l'utilizzo dei file del sistema operativo nel seguenti contesti:

- ODA può *creare* nuovi file per supportare la generazione del contenuto di file.
- ODA può *leggere* i file esistenti per supportare l'associazione di file con i nodi di origine.

### Creazione dei file per il contenuto di file

Quando un ODA implementa l'interfaccia `IGeneratesBinFiles`, supporta la creazione dei file come contenuto. I file creati da ODA conservano le informazioni raccolte da oDA dal processo di generazione delle definizioni di oggetti business e altrove. Se al processo di generazione di file occorre l'array di nodi di origine selezionati dall'utente (che la Procedura guidata dell'oggetto Business crea come risultato del passo 3m Seleziona origine), ODA può ricevere questo array da Procedura guidata dell'oggetto Business. Per informazioni su come implementare il metodo che genera i file, consultare "Creazione di file" a pagina 147.

Tuttavia, l'interfaccia `IGeneratesBinFiles` *non* definisce un metodo di generazione di nodi di origine, che rileva i nodi di origine e genera l'array dei nodi di origine per la Procedura guidata dell'oggetto Business per visualizzarli nella casella di dialogo Seleziona origine. Se ODA supporta il contenuto di file e questa generazione di file richiede un array di nodi di origine selezionati dall'utente, ODA deve utilizzare il metodo di generazione dei nodi di origine nell'interfaccia `IGeneratesBoDefs`, `getTreeNodes()`. Questo metodo interroga l'origine di dati per i nodi secondari del nodo principale specificato e costruisce i nodi tree associati, come descritto in "Creazione dei nodi di origine" a pagina 122.

**Nota:** In questo release, ad ogni ODA viene richiesto di supportare la generazione di definizioni di oggetti business. Pertanto, occorre implementare l'interfaccia `IGeneratesBoDefs` e tutti i relativi metodi (compreso il metodo `getTreeNodes()`).

Se un ODA supporta *solo* la creazione di nuovi file (generazione di file), può utilizzare il metodo `getTreeNodes()` come definito in `IGeneratesBoDefs`. Questo metodo interroga l'origine di dati per i nodi secondari del nodo principale specificato e costruisce i nodi tree associati, come descritto in "Creazione dei nodi di origine" a pagina 122.

## Lettura di file per i dati di origine

Quando un ODA implementa l'interfaccia `IGeneratesBinFiles`, supporta la lettura dei file del sistema operativo che vengono associati ai nodo di origine (per informazioni su come associare un file a un nodo, fare riferimento a "Associazione di un file del sistema operativo" a pagina 88.). I file letti da ODA conservano i dati di origine, di cui ODA deve effettuare la ricerca per gli oggetti che vengono rappresentati come nodi di origine. Per supportare l'associazione di file con i nodi, ODA deve effettuare le seguenti operazioni:

- Impostare la natura del nodo di ciascun nodo tree cui è associato un file associato su "file" (la variabile membro `polymorphicNature` impostata su `ODKConstant.NODE_NATURE_FILE`). Per ulteriori informazioni, consultare "Nodi di natura file" a pagina 128.
- Implementare i metodi che devono accedere oggetti rappresentati dai nodi di origine affinché l'interroghino non solo l'origine di dati ODA, ma qualsiasi file associato con un nodo di origine. Per richiamare il contenuto di un file del sistema operativo specificato dal percorso del nodo di origine, utilizzare il metodo `getClientFile()` definito nella classe `ODKUtility`.

**Importante**

Un ODA deve implementare l'interfaccia `IGeneratesBinFiles` per il metodo `getClientFile()` per richiamare con esito positivo un file del sistema operativo specificato. Se ODA implementa solo l'interfaccia `IGeneratesBoDefs`, `getClientFile()` genera un'eccezione `UnsupportedContentException`.

Il metodo `getClientFile()` prevede come argomento il percorso del nodo di origine del file da richiamare. Questo percorso del nodo di origine ha il seguente formato:

*`fileNodePath`*`:`*`filePath`*

dove *fileNodePath* è il percorso del nodo (nomi del nodo separati da due punti (:)) del nodo che ha un file associato e *filePath* è il percorso del sistema operativo del file associato. Quando gli utenti espandono o selezionano un nodo che è un file associato, la Procedura guidata dell'oggetto Business crea questo percorso per il nodo.

Ad esempio, la classe `ArmyAgent5` dell'esempio dell'esercito romano ODA supporta entrambi l'interfaccia `IGeneratesBinFiles` e l'associazioni di file con i nodi. Si supponga il file `Flavius.xml` (nella directory `C:\IBM\XMLFiles`) per il nodo di origine Vulso, come viene illustrato in Figura 51 a pagina 89. Se si seleziona il nodo `Flavius.xml` (consultare Figura 52 a pagina 89) dalla gerarchia dei nodi di origine, la Procedura guidata dell'oggetto Business mette il seguente percorso dei nodi nell'array dei nodi di origine:

```
Apollo:Vulso:Flavius.xml:C:\IBM\XMLFiles\Flavius.xml
```

Questo ODA fornisce il metodo `findSon()` per analizzare un percorso del nodo di origine e individuare l'oggetto associato che rappresenta il nodo di origine. La versione di `findSon()` nella classe `ArmyAgent3` interroga solo l'origine di dati ODA

(un file XML denominato RomanArmy.xml) per l'oggetto associato con il nodo di origine specificato. Una versione revisionata nella classe ArmyAgent4 aggiunge la capacità di interrogare un file associato fornendo il metodo remoteSon(), che utilizza getClientFile() per ottenere il contenuto del file specificato e restituire il contenuto come Son.

**Nota:** La classe ArmyAgent4, che implementa il metodo remoteSon(), *non* supporta l'interfaccia IGeneratesBinFiles. Pertanto, il metodo remoteSon() individua l'eccezione UnsupportedContentException generata da getClientFile()e crea un oggetto Son "fittizio" (consultare Figura 78 a pagina 171) La classe ArmyAgent5, che estende ArmyAgent4, *implementa* IGeneratesBinFiles. Pertanto, questa versione di ODA può supportare pienamente l'accesso a file associati con i nodi di origine con getClientFile().

Se un nodo di origine può avere un file associato, la capacità di interpretare il percorso dei nodi di origine del file e di leggerne il contenuto durante la generazione, il metodo, che genera il contenuto, deve poter accedere alle informazioni in nodi che sono in un file. Implementano il metodo che genera contenuto in modo da utilizzare getClientFile() per richiamare un file del sistema operativo associato con un nodo. Il metodo che fornisce tale supporto è il seguente:

- Il metodo generateBoDefs() genera le definizioni di oggetti business. Il metodo getClientFile() fornisce i contenuti del file specificato in modo da poter ottenere generateBoDefs() le informazioni necessarie a creare una definizione di oggetti business. Se il metodo generateBoDefs() è già stato implementato per ottenere informazioni sui nodi di origine per l'origine di dati ODA, deve essere migliorato per ottenere informazioni da un file associato.

  **Nota:** Il metodo getClientFile() *non può* richiamare contenuto di un file specificato quando chiamato da generateBoDefs() finché ODA non implementi anche l'interfaccia IGeneratesBinFiles.
- Il metodo che genera i file dipende dal protocollo di controllo supportato da ODA. Per una generazione su richiesta, il generateBinFiles() genera i file. Per la generazione callback, un metodo definito dall'utente genera i file. In entrambi i casi, il metodo getClientFile() fornisce il contenuto del file specificato affinché il metodo possa ottenere le informazioni necessarie per creare un file.

Per ulteriori informazioni su come generare il contenuto di un file, consultare "Creazione di file".

## Creazione di file

Dopo che l'utente ha selezionato i nodi di origine nella casella di dialogo Seleziona nodi, ODA è pronto per iniziare la generazione di contenuti. L'obiettivo del processo di generazione file è di creare un file (o vari file) necessario a ODA o ad altri processi. Il passo che inizia la generazione di file dipende dal protocollo di contenuto associato con il tipo di contenuto di file (ContentType.BinaryFile), nel modo seguente:

- Se i file devono essere generati *su richiesta*, Procedura guidata dell'oggetto Business inizia la generazione di contenuto richiamando il metodo di generazione contenuto, generateBinFiles(). Questo metodo è parte dell'interfaccia IGeneratesBinFiles.
- Se i file devono essere generati attraverso i *callback*, ODA inizia la generazione in un modo definito dall'utente. La Procedura guidata dell'oggetto Business *non*

chiama generateBinFiles() ma attende un segnale di "generazione di contenuto completa" da PDA prima di accedere al contenuto generato.

In questa sezione vengono descritte le seguenti operazioni che devono essere effettuate dal metodo generateBinFiles() per generare files:

1. "Definizione del metodo generateBinFiles()"
2. "Richiesta di proprietà per le informazioni di file" a pagina 149
3. "Creazione dei file" a pagina 149
4. "Aggiunta di file generati" a pagina 150

## Definizione del metodo generateBinFiles()

Il metodo generateBinFiles() viene definito nell'interfaccia IGeneratesBinFiles. Pertanto, la classe ODA (derivata da ODKAgentBase2) deve implementare questo metodo quando implementa l'interfaccia IGeneratesBinFiles. La scopo del metodo generateBinFiles() dipende dal protocollo di contenuto utilizzato dall'ODA per la generazione del contenuto del file ( ContentType.BinaryFile) come riportato di seguito:

- Se ODA genera i file "a richiesta", la Procedura guidata dell'oggetto Business richiama in modo esplicito il metodo generateBinFiles() per generare i file.
- Se ODA genera i file attraverso i callback, la Procedura guidata dell'oggetto Business non richiama *mai* in modo esplicito il metodo generateBinFiles(). L'ODA utilizza altri modi per generare i file, che possono essere acceduti dalla Procedura guidata dell'oggetto Business.

**Creazione di file a richiesta:** Se ODA crea i file "a richiesta", la Procedura guidata dell'oggetto Business richiama in modo esplicito il metodo generateBinFiles() per iniziare la generazione dei file. Pertanto, occorre implementare generateBinFiles() in modo che gestisca la generazione degli oggetti file, memorizzandoli nella struttura di contenuto generato e restituendo i metadati di contenuto alla Procedura guidata dell'oggetto Business.

Mentre viene eseguito il metodo generateBinFiles(), la Procedura guidata dell'oggetto Business visualizza il pannello Creazione di Business Objects (passo 5). Come ultima operazione, generateBinFiles() restituisce un oggetto di metadati di contenuto (ContentMetaData) per descrivere i file generati (sebbene *non* contenga i file generati attualmente).

**Creazione di file attraverso i callback:** Se ODA genera i file attraverso i callback, la Procedura guidata dell'oggetto Business non richiama *mai* in modo esplicito il metodo generateBinFiles(). L'ODA utilizza altri modi per generare i file in "modo spontaneo". Occorre sviluppare un metodo per gestire la generazione di file, memorizzandoli nella struttura di contenuto generato e notificando alla Procedura guidata dell'oggetto Business che la generazione di contenuto è completa. Tuttavia, l'interfaccia IGeneratesBinFiles richiede che si definisca il metodo generateBinFiles(). Pertanto, occorre implementare generateBinFiles() affinché il programma di chiamata non venga mai richiamato.

L'esempio dell'esercito romano ODA supporta il protocollo di contenuto callback per la generazione di file (consultare Figura 60 a pagina 120). Definisce il metodo generateBinDefs() nella classe ArmyAgent5. Questa implementazione del metodo comprende il codice in Figura 71, che definisce il metodo generateBinFiles() affinché generi un'eccezione se viene richiamato.

```
public ContentMetaData generateBinFiles(String[] nodes)
    throws ODKException
{
   throw new ODKException(
      "Files are produced as callbacks. Do not call for file generation.");
}
```

*Figura 71. Definizione del metodo generateBinFiles()*

Come alternativa alla generazione di un'eccezione, il metodo `generateBinFiles()` può utilizzare il metodo `contentUnavailable()` (definito in `ContentMetaData`) per restituire i metadati di contenuto alla Procedura guidata dell'oggetto Business, nel modo seguente:

```
return (ContentMetaData.contentUnavailable(ContentType.BinaryFile));
```

## Richiesta di proprietà per le informazioni di file

Se, durante il processo di generazione contenuto, ODA richiede informazioni aggiuntive, può visualizzare la finestra Proprietà BO in cui gli utenti possono fornire i valori per le proprietà degli oggetti business. Anche se tali proprietà sono dette proprietà di oggetti business, è possibile utilizzare il metodo `getBOSpecificProps()` per visualizzare le informazioni che potrebbero essere richieste dal processo di geenrazione file. Per ulteriori informazioni sull'utilizzo della casella di dialogo Proprietà BO, consultare "Richiesta di proprietà di oggetti business" a pagina 130.

## Creazione dei file

L'API ODK non fornisce una classe speciale per rappresentare un file binario poiché Java già fornisce la classe `File` nel pacchetto `java.io`. Questo pacchetto contiene molte classi di immissione/emissione che possono essere utili nella generazione e nell'accesso di file. Per ciascun file generato da ODA, occorre eseguire le seguenti operazioni:

- Creare un nuovo oggetto `File` con il nome file appropriato.
- Scrivere il contenuto in questo file, chiudendo il file una volta completata la scrittura.

La generazione di file attuale effettuata da ODA dipende dalla progettazione di ODA. Implementare la generazione di file nel modo più adatto ai requisiti di ODA e dei componenti che richiedono i file.

La classe `ArmyAgent5` dell'esempio dell'esercito romano ODA definisce una classe separata `FileCreator`, per gestire la generazione attuale dei file. Per simulare la generazione di file in "modo spontaneo", l'esempio richiama il costruttore `FileCreator()` dal metodo `generateBoDefs()`, come viene illustrato nel seguente frammento di codice:

```
public ContentMetaData generateBoDefs(String[] nodes) throws ODKException
{
   ContentMetaData cmd = super.generateBoDefs(nodes);
   new FileCreator(this, nodes).start();
   return cmd;
}
```

Il costruttore `FileCreator()` crea un thread per generare i file. Riceve come argomento un riferimento all'oggetto ODA attuale (`this`) e all'array con i percorsi nodo dei nodi dell'origine selezionata. Crea quindi i seguenti file:

- Il file `stats.zip`, che contiene il numero di definizioni di oggetti business generate da ODA.

- Il file `adopted.txt`, se tutti i nodi di origine selezionati dall'utente sono figli adottati

## Aggiunta di file generati

Come discusso in "Aggiunta di contenuto generato" a pagina 102, ODA deve restituire il contenuto generato alla Procedura guidata dell'oggetto Business in due parti. Pertanto, se ODA genera i files come contenuto, deve restituire quanto segue:

- Una struttura del contenuto generato, che contiene i file generati
- Un oggetto (`ContentMetaData` ) di metadati di contenuto che descrive i file nella struttura generata di contenuto.

Questo metodo, che fornisce le informazioni, dipende dal protocollo di contenuto utilizzato da ODA per generare file, nel modo seguente:

- Se ODA genera i file "a richiesta", il metodo `generateBinFiles()` fornisce queste informazioni di contenuto.
- Se ODA genera i file attraverso i callback, un metodo definto dall'utente deve fornire queste informazioni di contenuto.

**Aggiunta di contenuto per file a richiesta:** Se ODA genera i file "a richiesta", la Procedura guidata dell'oggetto Business richiama il metodo `generateBinFiles()` per gestire la generazione di file. Tuttavia, `generateBinFiles()` fornisce il contenuto generato nel modo seguente:

- Popola la struttura di contenuto generato. Questa struttura deve essere in qualche modo visibile per `generateBinFiles()` e `getBinFile()`, così che entrambi possano accederla.
- Restituisce un oggetto di metadati di contenuto (`ContentMetaData`) che descrive i file generati al programma di chiamata, la Procedura guidata dell'oggetto Business.

  Una volta che la Procedura guidata dell'oggetto Business riceve questo oggetto di metadati di contenuto, può accedere ai file generati (all'interno della struttura di contenuto) come necessario nel metodo `getBinFile()`.

Per ulteriori informazioni su `getBinFile()`, consultare "Aggiunta di accesso ai file generati" a pagina 152.

**Aggiunta di contenuto per i file generati di callback:** Se ODA genera i file attraverso i callback, la Procedura guidata dell'oggetto Business *non* richiama il metodo `generateBinFiles()` per gestire la generazione di file. L'ODA utilizza alcuni metodi definiti dall'utente per generare i file in "modo spontaneo". Questo metodo potrebbe far parte della classe ODA oppure essere in una classe all'interno del pacchetto ODA. Tuttavia, deve fornire il contenuto generato nel modo seguente:

- Popola la struttura di contenuto generato. Questa struttura deve essere in qualche modo visibile per il metodo definito dall'utente che genera i file *e* per `getBinFile()` (che la classe ODA può implementare) affinché entrambi i metodi possano accederla.
- Invia un oggetto di metadati di contenuto (`ContentMetaData`) che descrive i file generati alla Procedura guidata dell'oggetto Business.

  Il metodo definito dall'utente che genera i file non può restituire direttamente i metadati di contenuto alla Procedura guidata dell'oggetto Business poiché la Procedura guidata dell'oggetto Business non ha richiamato questo metodo. Il metodo deve inviare un segnale di "generazione di contenuto completa" alla Procedura guidata dell'oggetto Business richiamando il metodo `contentComplete()` (definito nella classe `ODKUtility`). Questo metodo accetta un

oggetto di metadati di contenuto come argomento. Consultare Tabella 49 per le informazioni che l'oggetto dei metadati di contenuto dovrebbe contenere. Invia questi metadati di contenuto alla Procedura guidata dell'oggetto Business. Una volta che la Procedura guidata dell'oggetto Business riceve l'oggetto dei metadati di contenuto, può utilizzare il metodo `getBinFile()` per accedere ai file generati (all'interno della struttura di contenuto generato).

**Nota:** Per ulteriori informazioni su `getBinFiles()`, consultare "Aggiunta di accesso ai file generati" a pagina 152.

Nella classe `ArmyAgent5` dell'esempio dell'esercito romano ODA, la struttura del contenuto generato viene definita un array degli oggetti `File` denominati `m_files`, nel modo seguente:

```
File[] m_files = null;
```

Il frammento di codice in Figura 72 illustra l'ultima parte del metodo `FileCreator.run()` (definito nel file `ArmyAgent5.java`)

```
            for (int i=0; i<fileV.size(); i++)
               m_agent.m_files[i] = (File) fileV.get(i);
      }
      ODKUtility.getODKUtility.contentComplete(
         new ContentMetaData(ContentType.BinaryFile, 0,
         m_agent.m_files.length);
} // end of run() in FileCreator class
```

*Figura 72. Aggiunta di contenuto di file*

Figura 72 gestisce il contenuto generato nel modo seguente:

- Il metodo `FileCreator.run()` salva i file generati nella struttura di contenuto generato, `m_files`.

  L'esempio dell'esercito romano ODA utilizza l'array `m_files` come struttura di contenuto generata. Per salvare i file generati, `run()` li salva in questo array `m_files`. Questa operazione si verifica dopo che `run()` ha generato *tutti* i file. La Procedura guidata dell'oggetto Business può accedere all'array `m_files` attraverso una chiamata al metodo di richiamo contenuto, `getBinFile()`.
- Come operazione finale, `FileCreator.run()` invia l'oggetto di metadati di contenuto che descrive il contenuto generato alla Procedura guidata dell'oggetto Business.

  Il metodo `run()` richiama il metodo `contentComplete()`, passandolo a un nuovo oggetto `ContentMetaData`. In questo costruttore `ContentMetaData()`, `run()` trasferisce le informazioni visualizzate in Tabella 49.

*Tabella 49. Inizializzazione dei metadati di contenuto per la generazione di file*

| Informazioni ContentMetaData | Codice | Descrizione |
|---|---|---|
| Tipo di contenuto | `ContentType.BinaryFile` | Indica che il tipo di contenuto è files |
| Dimensione del contenuto generato | 0 | Indica che la dimensione totale *non* è richiesta. Il valore della lunghezza non è necessario nell'implementazione corrente di un oggetto `ContentMetaData`. |
| Conteggio del contenuto generato | `m_files.length` | La variabile membro `length()` restituisce il numero di elementi attualmente nell'array. |

## Aggiunta di accesso ai file generati

Il metodo `generateBinFiles()` *non* restituisce le effettive definizioni di oggetti business generate. Affinché la Procedura guidata dell'oggetto Business possa accedere al contenuto generato, la classe ODA deve implementare il metodo di richiamo contenuto per i file. La Procedura guidata dell'oggetto Business utilizza le informazioni nell'oggetto dei metadati di contenuto (`generateBinFiles()` *non* restituisce) per determinare quale metodo di richiamo contenuto richiamare. Per il contenuto del file, la Procedura guidata dell'oggetto Business richiama il metodo `getBinFile()` per richiamare le definizioni dell'oggetto business generato.

**Nota:** La Procedura guidata dell'oggetto Business richiama il metodo `generateBinFile()` per la generazione di file se ODA supporta il protocollo di contenuto a richiesta. Se l'ODA supporta il il protocollo di contenuto callback per la generazione di file, un metodo definito dall'utente genera i file. Tuttavia, questo metodo non restituisce l'effettivo contenuto generato. Pertanto, la Procedura guidata dell'oggetto Business *ancora* richiede il metodo `getBinFile()` per accedere ai file generati.

Indipendentemente dal protocollo di contenuto supportato da ODA per la generazione di file, la classe ODA deve implementare il metodo `getBinFile()`. Questo metodo è definito come parte dell'interfaccia `IGeneratesBinFiles`. Il metodo accetta come argomento un indice, che identifica il numero di file da restituire. Accede questi file nella struttura di contenuto generato e restituisce un array degli oggetti (`File`) del file richiamato. Il numero di file in questo array dipende dall'argomento index, come viene illustrato in Tabella 50.

*Tabella 50. Richiamo di file*

| Valore dell'indice | Descrizione | Numero di elementi nell'array che restituisce getBinFile() |
|---|---|---|
| Nell'intervallo da 0 a *numero* - 1, dove *numero* è il numero totale di file nella struttura di contenuto generato | Specifica la posizione dell'indice nella struttura di contenuto generato del file da richiamare | Un oggetto di file |
| `ODKConstant.GET_ALL_OBJECTS` | Costante speciale per indicare la restituzione di tutti i file nella struttura di contenuto generato | Tutti i file nella struttura di contenuto generato (*numero*) |

Per l'esempio dell'esercito romano ODA, il metodo `FileCreator.run()` (definito nella classe `ArmyAgent5`) popola l'array `m_files` con i file generati. Pertanto, il metodo `getBinFile()` (anche definito in `ArmyAgent5`) richiama il numero specificato dei file da questo array. Il seguente codice illustra il metodo `getBinFile()` per l'esempio romano ODA:

```
public File[] getBinFile(long index) throws ODKException
{
   if (index == ODKConstant.GET_ALL_OBJECTS)
      return m_files;
   else
      return new File[] {m_files[(int)index]};
}
```

## Operazioni con le proprietà agente

Vi sono due situazioni in cui un ODA fornisce proprietà dell'agente alla Procedura guidata dell'oggetto Business:

- Per fornire le proprietà di configurazione ODA inizializzate (nella finestra Configura agente)
- Per fornire le proprietà dell'oggetto business inizializzate (nella finestra di dialogo Proprietà BO)

Per rappresentare una proprietà dell'agente, l'API ODK definisce un'oggetto di proprietà dell'agente, che è una creazione di un'istanza della classe `AgentProperty`. Quando si crea un'istanza dell'oggetto di proprietà dell'agente, si inizializzano alcune o tutte le variabili membro, illustrate in Tabella 51.

*Tabella 51. Contenuto di un oggetto delle proprietà agente*

| variabile membro | Descrizione |
|---|---|
| `propName` | Il nome della proprietà dell'agente |
| `descrizione` | Una stringa di testo che descrive lo scopo della proprietà dell'agente |
| `type` | Il tipo di dati della proprietà dell'agente, come rappresentato dalla costante di tipo proprietà |
| `cardinalità` | La cardinalità della proprietà dell'agente, cioè se la proprietà può avere uno o più valori |
| `isHidden` | Determina se Procedura guidata dell'oggetto Business visualizza il valore della proprietà come testo normale o in formato crittografato. |
| `isMultiple` | Determina se la Procedura guidata dell'oggetto Business visualizza un elenco a discesa dei valori validi per proprietà dell'agente, per la scelta degli utenti |
| `isReadOnly` | Determina se il valore della proprietà dell'agente è di sola lettura, cioè se gli utenti possono modificare il valore visualizzato |
| `isRequired` | Determina se il valore della proprietà dell'agente, cioè se gli utenti devono specificare un valore |
| `allDefaultValues` | Un array di valori predefiniti per la proprietà dell'agente |
| `allDependencies` | Un array di condizioni per la proprietà dell'agente |
| `allValidValues` | Un array di valori validi per la proprietà dell'agente |
| `allValues` | Un array di valori inizializzati dall'utente per la proprietà dell'agente |

Per creare un'istanza di un oggetto proprietà agente, utilizzare uno dei moduli del costruttore `AgentProperty()`:

- Il primo modulo definisce un oggetto agent-property e lo inizializza con il nome di una proprità *only*.
- Il secondo modulo definisce un nuovo oggetto agent-property e lo inizializza con le variabili membro *all*.
- I terzo formato definisce un nuovo oggetto agent-property e lo inizializza con tutte le variabili membro *tranne* `isHiddene isReadOnly`.

## Definizione della proprietà dell'agente

Tabella 52 illustra le informazioni di base relative alla proprietà di un agente che l'oggetto di proprietà dell'agente contiene.

*Tabella 52. Informazioni di base per una proprietà dell'agente*

| Informazioni delle proprietà dell'agente | Variabile membro AgentProperty | Descrizione |
|---|---|---|
| Nome | `propName` | Identifica la proprietà dell'agente.<br><br>La Procedura guidata dell'oggetto Business visualizza questo valore nella colonna Proprietà della casella di dialogo Configura agente (proprietà di configurazione) oppure la casella di dialogo Proprietà BO (proprietà dell'oggetto business). È possibile inizializzare il nome della proprietà dell'agente con uno dei moduli del costruttore `AgentProperty()`. |
| Descrizione (facoltativo) | `descrizione` | Fornisce le informazioni aggiuntive per descrivere lo scopo della proprietà dell'agente.<br><br>La Procedura guidata dell'oggetto Business visualizza questo valore nella colonna Descrizione della casella di dialogo Configura agente (proprietà di configurazione) oppure la casella di dialogo Proprietà BO (proprietà dell'oggetto business). Occorre utilizzare il secondo o terzo modulo del costruttore `AgentProperty()` per inizializzare la descrizione di proprietà dell'agente. |
| Tipo di dati | `tipo` | Definisce il tipo di dati dei valori mantenuti dalla proprietà dell'agente<br><br>La Procedura guidata dell'oggetto Business visualizza questo valore nella colonna Tipo della casella di dialogo Configura agente (proprietà di configurazione) oppure la casella di dialogo Proprietà BO (proprietà dell'oggetto business). Se si utilizza il primo modulo del costruttore `AgentProperty()` per inizializzare la proprietà dell'agente (che specifica solo il nome della proprietà), i valori predefiniti del tipo della proprietà dell'agente. Per specificare un tipo, utilizzare il secondo o terzo modulo del costruttore `AgentProperty()` per inizializzare la proprietà dell'agente. Rappresenta il tipo della proprietà dell'agente con una delle costanti del tipo di proprietà in Tabella 69 a pagina 189. |

## Definizione del valore della proprietà

La Procedura guidata dell'oggetto Business visualizza il valore di proprietà dell'agente nella colonna Valore della casella di dialogo Configura agente (proprietà di configurazione) oppure la casella di dialogo Proprietà BO. Come parte del processo di inizializzazione di una proprietà dell'agente, occorre indirizzare le seguenti attività:

## Selezione del tipo di controllo di visualizzazione

La Procedura guidata dell'oggetto Business utilizza i seguenti metadati `AgentProperty` per determinare il tipo di controllo per la visualizzazione del valore delle proprietà:

- Il Il parametro `isMultiple` determina se il controllo per il valore della proprietà visualizza più valori in un elenco a discesa. Per inizializzare l'elenco a discesa con i valori, è possibile specificare i valori nella array `allValidValues`.
- Il parametro `cardinality` determina se il controllo consente agli utenti di specificare uno o più valori per la proprietà:

| Cardinalità | Descrizione | Costanti di cardinalità |
|---|---|---|
| Singolo | La proprietà può contenere solo un valore. Pertanto, gli utenti possono specificare *solo un* valore per la proprietà. | `ODKConstant.SINGLE_CARD` |
| Molteplici (n) | La proprietà può contenere uno o più valori. Pertanto gli utenti possono specificare i *più* valori per la proprietà. | `ODKConstant.MULTIPLE_CARD` |

Tabella 53 illustra le combinazioni possibili per la visualizzazione del controllo del valore della proprietà.

*Tabella 53. Possibili tipi del controllo del valore della proprietà*

| Cardinalità | Visualizza più valori (isMultiple) | Sono valori validi (tutti ValidValues) forniti? | Spiegazione |
|---|---|---|---|
| 1 | `Falso` | No | Il valore della proprietà viene visualizzato come un controllo di modifica, cioè una semplice casella in cui è possibile immettere e modificare un valore. |
| 1 | `Vero` | Sì | Il valore della proprietà viene visualizzato come un elenco a discesa che visualizza i valori validi specificati (consultare Figura 73 a pagina 156). Da questo elenco gli utenti possono selezionare solo un valore. |
| n | `Vero` | Sì | Il valore della proprietà viene visualizzato come un elenco a discesa che visualizza i valori validi specificati. Ciascun valore di questo elenco viene visualizzato con una casella di spunta, che, se selezionato, consente di includere il valore nell'insieme dei valori della proprietà (consultare Figura 73 a pagina 156). |

*Tabella 53. Possibili tipi del controllo del valore della proprietà (Continua)*

| Cardinalità | Visualizza più valori (isMultiple) | Sono valori validi (tutti ValidValues) forniti? | Spiegazione |
|---|---|---|---|
| n | Vero | No | Il valore della proprietà viene visualizzato come controllo della griglia che contiene valori che non vengono visualizzati. Inizialmente, questa griglia visualizza una griglia secondaria con una riga vuota. Se l'utente immette testo in tale griglia, la Procedura guidata dell'oggetto Business inserisce un altra riga vuota. Questo processo continua fino a che gli utenti finiscono l'immissione di nuove righe. Per eliminare un valore, gli utenti eliminano il testo del valore. La Procedura guidata dell'oggetto Business include solo righe non vuote nell'insieme dei valori delle proprietà. |

Quando la Procedura guidata dell'oggetto Business visualizza una proprietà di cardinalità singola che *non* ha valori validi, lascia il campo Valore della proprietà vuoto. È possibile, tuttavia, definire un valore predefinito per la proprietà. In questo caso, la Procedura guidata dell'oggetto Business visualizza il valore predefinito nel campo valore. Per ulteriori informazioni, consultare "Specifica di valori predefiniti".

Figura 73 illustra i due controlli che visualizzano più valori (`isMultiple = true`) nella Procedura guidata dell'oggetto Business.

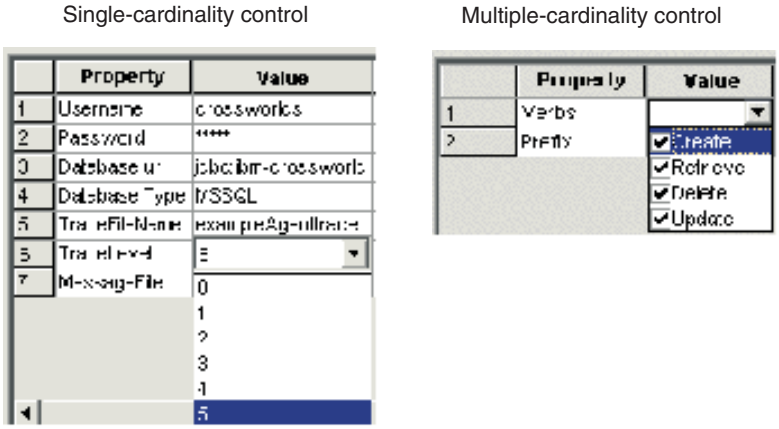



*Figura 73. Controlli a cardinalità singola o multipla per le proprietà con valori multipli.*

Figura 73 illustra i controlli a cardinalità singola e multipla, che entrambi visualizzano molteplici valori in un elenco a discesa.

- Il controllo a cardinalità singola (a sinistra in Figura 73) visualizza più livelli di traccia in un elenco a discesa, ma consente di selezionare *solo un* valore (`cardinality = ODKConstant.SINGLE_CARD`) da questo elenco.
- Il controllo a cardinalità multipla (a destra in Figura 73) visualizza più verbi in un elenco a discesa e consente di selezionarne *un numero qualsiasi* (`cardinality = ODKConstant.MULTIPLE_CARD`) da questo elenco.

## Specifica di valori predefiniti

Per specificare un valore predefinito per una proprietà dell'agente, occorre fornire il valore o i valori predefiniti nella relativa variabile membro `allDefaultValues`.

Questa variabile membro è un array di valori `Object`. Il numero di elementi in questo array deve corrispondere alla cardinalità della proprietà nel seguente modo:

- Per una proprietà a cardinalità singola, l'array `allDefaultValues` deve contenere solo un elemento.
- Per una proprietà a cardinalità multipla, l'array `allDefaultValues` può contenere uno o più elementi.

La Procedura guidata dell'oggetto Business assegna il valore predefinito alla proprietà prima di visualizzare la proprietà. Se gli utenti non sostituiscono questo valore predefinito specificando un valore della proprietà, questo valore predefinito rimane il valore della proprietà.

**Nota:** Tutti i valori validi specificati per la proprietà *non* sono automaticamente i valori predefiniti. Occorre specificare esplicitamente i valori predefiniti.

Tabella 54 riepiloga il comportamento dei valori predefiniti.

*Tabella 54. Valori predefiniti per le proprietà agente*

| Cardinalità | Contenuto di allDefaultValues | Visualizza |
|---|---|---|
| Singolo | Un elemento | Con valori validi (`isMultiple=true`): il valore valido viene visualizzato come voce "selezionata" nell'elenco a discesa dei valori validi.<br><br>Senza valori validi (`isMultiple=false`): il valore valido viene visualizzato nel campo Valore della proprietà |
| Multiplo | Uno o più elementi | I valori predefiniti vengono visualizzati come voci "selezionate" nell'elenco a discesa dei valori validi. |

## Inizializzazione di una proprietà a cardinalità singola

Per inizializzare una proprietà dell'agente a cardinalità singola, effettuare le seguenti operazioni:

- Limitare il numero di valori, che possono essere specificati dall'utente, a uno. Impostare la variabile membro della `cardinalità` della proprietà su `ODKConstant.SINGLE_CARD`.
- Determinare se fornire un elenco di valori validi da cui selezionare il valore singolo della proprietà. Se si fornisce un elenco di valori validi:
  - Impostare la variabile `isMultiple` su `true`.
  - Inizializzare l'array (`allValidValues`) dei valori validi con un elenco di valori validi.

  Se *non* si fornisce un elenco di valori validi, impostare la variabile `isMultiple` su `false` e non passare in un array di valori validi.
- Inizializzare, se necessario, l'array `allDefaultValues` per contenere un `Object` con un singolo valore predefinito.

Il seguente frammento di codice inizializza una proprietà dell'agente a singola cardinalità, che non fornisce un elenco di valori tra cui scegliere e che ha come valore predefinito 256:

```
defaultVal[0] = 256;
AgentProperty("Property1", AgentProperty.TYPE_INTEGER,
"Description of property", false, false,
ODKConstant.SINGLE_CARD, null, defaultVal);
```

### Inizializzazione di una proprietà a cardinalità multipla

Per inizializzare una proprietà dell'agente a cardinalità multipla, effettuare le seguenti operazioni:

- Indicare gli utenti che possono specificare un numero di valori per la proprietà. Impostare la variabile membro della `cardinalità` della proprietà su `ODKConstant.MULTIPLE_CARD`.
- Indicare che la Procedura guidata dell'oggetto Business deve gestire immissioni di più valori per la proprietà. Impostare la variabile membro `isMultiple` su `true`.
- Determinare se fornire un elenco di valori validi tra cui l'utente può scegliere. Se si fornisce un elenco di valori, inizializzare l'elenco di valori validi nell'array `allValidValues`. Tali valori inizializzano l'elenco a discesa della proprietà.

  Se *non* si fornisce un elenco di valori validi, la Procedura guidata dell'oggetto Business fornisce una griglia secondaria agli utenti per specificare ciascun valore della proprietà.
- Inizializzare, se necessario, l'array `allDefaultValues` per contenere un `Object` con ciascun valore predefinito.

Il frammento di codice in Figura 67 a pagina 133 inizializza la proprietà dell'agente a cardinalità multipla, denominato Verbs, che ha un elenco di valori validi e predefiniti.

## Impostazione delle condizioni sul valore della proprietà

La classe `AgentProperty` fornisce la capacità di definire condizioni su una proprietà dell'agente. Una *condizione* può limitare i valori di una proprietà dell'agente, denominata una *proprietà dipendente*, in base al valore di un'altra proprietà dell'agente. Una condizione ha due parti, ciascuna parte un particolare tipo di condizioni secondarie, come viene illustrato in Tabella 55.

*Tabella 55. Parti di una condizione di proprietà dell'agente*

| Condizione secondaria | Descrizione | Classe API ODK |
|---|---|---|
| Condizione di immissione | Definisce una condizione sul valore della proprietà dell'agente corrente | `InputCondition` |
| Condizione dipendente | Definisce una condizione che deve essere soddisfatta nella proprietà dipendente quando la condizione di immissione associata restituisce true | `DependentCondition` |

### Definizione della condizione completa

Per rappresentare una condizione, l'API ODK definisce un oggetto di condizione completa, che è una installazione della classe `CompleteCondition`. Tabella 56 illustra le variabili membro contenute da un oggetto di condizione completa.

*Tabella 56. Contenuto di un oggetto di condizione completa*

| variabile membro | Descrizione |
|---|---|
| `allInputConditions` | Un array di oggetti di condizioni di immissione (`InputCondition`), dove ciascun oggetto definisce una condizione sul valore della proprietà dell'agente |
| `allDependentConditions` | Un array di oggetti di condizioni di immissione (`DependentCondition`), dove ciascun oggetto definisce una limitazione sul valore della proprietà dipendente. Questa limitazione applica il valore della proprietà dipendente quando le condizioni di immissione associate (nell'array `allInputConditions` array) restituisce true. |

Un oggetto di condizione completa contiene le informazioni che descrivono una condizione *singola* su una proprietà dell'agente. Una proprietà dell'agente può avere più condizioni definite. Ogni oggetto della condizione è memorizzato nella variabile membro `allDependencies` dell'oggetto `AgentProperty` della proprietà dell'agente.

Per creare una condizione su una proprietà dell'agente, effettuare le seguenti operazioni:

1. Creare un'istanza di un oggetto `CompleteCondition` per conservare le informazioni di condizione.
2. Creare un'istanza degli oggetti `InputCondition` appropriati per descrivere le condizioni di immissione per la proprietà dell'agente. Salvare ciascun oggetto `InputCondition` nell'array delle condizioni di immissione (variabile membro `allInputConditions`) dell'oggetto di condizioni complete. Per ulteriori informazioni sulle condizioni di immissione, consultare "Definizione delle condizioni di immissione".
3. Creare un'istanza degli oggetti `DependentCondition` appropriati per descrivere le condizioni dipendenti per la proprietà dell'agente. Salvare ciascun oggetto `DependentCondition` nell'array delle condizioni dipendenti (variabile membro `allDependentConditions`) dell'oggetto di condizioni complete. Per ulteriori informazioni, consultare "Definizione delle condizioni dipendenti" a pagina 160.
4. Salvare l'oggetto di condizioni complete nell'array di condizioni della proprietà dell'agente. La variabile membro `allDependencies` dell'oggetto delle proprietà dell'agente contiene quest'array di condizione.

## Definizione delle condizioni di immissione

Il La classe `InputCondition` rappresenta una *condizione di immissione*, che descrive una condizione sul valore della proprietà dell'agente corrente. Quando una condizione di immissione restituisce true, le condizioni dipendenti associate vengono applicate alla proprietà dell'agente dipendente. Tabella 57 illustra le informazioni necessarie a definire una condizione di immissione.

*Tabella 57. Informazioni per una condizione di immissione*

| Informazioni di condizione di immissione | Descrizione | InputCondition variabile membro |
|---|---|---|
| Operatore | Il tipo di confronto da effettuare sul valore della proprietà dell'agente. Un confronto viene indicato come un operatore relazionale e viene specificato come una delle costanti dell'operatore nella classe `CompleteCondition`. | `operatorType` |
| Valore specifico | Il valore con cui confrontare il valore delle proprietà dell'agente. Questo valore può essere una costante o il nome di un'altra proprietà dell'agente. | `specificValue`, `typeOfSpecificValue` |
| Se il confronto del valore della proprietà dell'agente viene eseguita dinamicamente | Un valore booleano per indicare se confrontare dinamicamente il valore della proprietà dell'agente con un altro valore della proprietà. Il confronto che riguarda le costanti non richiede confronti dinamici. | `isDynamic` |

Per creare una condizione di immissione, utilizzare uno dei moduli del costruttore `InputCondition()`. Il frammento del codice in Figura 76 a pagina 162 crea condizioni di immissione che confrontano il valore della proprietà dell'agente con due valori stringa delle costanti, "optionA" e "optionB". È anche possibile confrontare il valore della proprietà dell'agente con altri valori della proprietà. Il frammento di codice in Figura 74 crea una condizione di immissione per confrontare un valore di proprietà dell'agente con il valore attualmente nella proprietà dell'agente `Property2`.

```
// Instantiate a complete-condition object
condition1 = new CompleteCondition();

// Input condition to compare property value with
       // Property2's value
condition1.allInputConditions[0] = new InputCondition(
   CompleteCondition.OP_NOT_EQUAL, true,
   AgentProperty.TYPE_INTEGER, "Property2");
```

*Figura 74. Condizione di immissione per confrontare un valore della proprietà con un altro valore della proprietà*

In Figura 74, la variabile membro `isDynamic` è impostata su `true` così che la Procedura guidata dell'oggetto Business sappia di ottenere prima il valore corrente della proprietà `Property2`, *prima* di confrontare il valore specificato dall'utente con questo valore. Inoltre, `specificValue` è impostato su "`Property2`", il nome della proprietà con cui viene effettuato il confronto. Come risultato di questa condizione di immissione, le condizioni dipendenti per la proprietà si applicano *solo* se questo valore della proprietà *non è lo stesso* del valore di `Property2`.

## Definizione delle condizioni dipendenti

Il La classe `DependentCondition` rappresenta una *condizione dipendente*, che descrive una limitazione sul valore di una particolare proprietà dipendente. Una proprietà dipendente è una proprietà il cui valore dipende in qualche modo dal valore della proprietà corrente. Quando la condizione di immissione associata (o le condizioni) restituisce true, il valore della proprietà dipendente deve soddisfare le limitazioni specificate dalla condizione dipendente. Tabella 58 illustra le informazioni necessarie per definire una condizione dipendente.

*Tabella 58. Informazioni per una condizione dipendente*

| Informazioni per una condizione dipendente | Descrizione | DependentCondition variabile membro |
|---|---|---|
| Nome | Il nome della proprietà dipendente a cui viene applicata la condizione dipendente se la condizione di immissione associata (o condizioni) restituisce true. | `propertyName` |
| Operatore | Il tipo di confronto da effettuare sul valore della proprietà dipendente. Un confronto viene indicato come un operatore relazionale e viene specificato come una delle costanti dell'operatore nella classe `CompleteCondition`. | `operatorType` |
| Valore specifico | Il valore con cui confrontare il valore delle proprietà dipendente. Questo valore può essere una costante o il nome di un'altra proprietà dell'agente. | `specificValue`, `typeOfSpecificValue` |
| Se il confronto del valore specificata dall'utente viene eseguita dinamicamente | Un valore booleano per indicare se confrontare dinamicamente il valore della proprietà dipendente con un altro valore della proprietà. Il confronto che riguarda le costanti non richiede confronti dinamici. | `isDynamic` |

Per creare una condizione dipendente, utilizzare uno dei moduli del costruttore `DependentCondition()`. Il frammento del codice in Figura 76 a pagina 162 crea e inizializza le seguenti condizioni dipendenti:

- Quattro condizioni dipendenti per la condizione di immissione "`optionA`" specificano quattro valori possibili per la proprietà dipendente `DepProperty1` quando quella corrente ha un valore "`optionA`".
- Due condizioni dipendenti per la condizione di immissione "`optionB`" specificano una scala di valori possibili per la proprietà dipendente `DepProperty2` quando quella corrente ha un valore "`optionB`".

È anche possibile confrontare il valore della proprietà dipendente con altri valori della proprietà. Il frammento di codice in Figura 75 crea una condizione dipendente per confrontare un valore di proprietà dipendente con il valore attualmente nella proprietà dell'agente `Property2`.

```
// Dependent condition to compare property value
// with Property2's value
condition1.allDependentConditions[0] = new DependentCondition(
   CompleteCondition.OP_EQUAL, true,
   AgentProperty.TYPE_INTEGER, "Property2");
```

*Figura 75. Condizione dipendente per confrontare un valore della proprietà con un altro valore della proprietà*

In Figura 75, la variabile membro `isDynamic` è impostata su `true` così che la Procedura guidata dell'oggetto Business sappia di ottenere prima il valore corrente della proprietà `Property2`, prima di confrontare il valore della proprietà dipendente con questo valore. Inoltre, `specificValue` è impostato su "`Property2`", il nome della proprietà con cui viene effettuato il confronto.

### Definizione di una condizione semplice

Si supponga che si desideri definire le condizioni su una proprietà dell'agente ( `Property1`) che specifica le limitazioni su due proprietà dipendenti basate sui valori di `Property1`, nel modo seguente:

- La prima condizione limita il valore della proprietà dipendente `DepProperty1` ad uno dei quattro valori interi (0, 1, 256, o 512) se `Property1` ha il valore "`optionA`".
- La seconda condizione limita il valore della proprietà dipendente `DepProperty2` in modo che sia compreso in una scala da 1 a 5 (incluso) se `Property1` ha il valore "`optionB`".

Figura 76illustra il codice che implementa queste due condizioni

```
// 1. Instantiate the complete-condition object
condition1 = new CompleteCondition();
// 2. Create the condition on the "optionA" value
//   a) Instantiate the input condition on "optionA"
   condition1.allInputConditions[0] = new InputCondition(
      CompleteCondition.OP_EQUAL, false, AgentProperty.TYPE_STRING,
      "optionA");
//   b) Instantiate the dependent conditions for DepProperty1
   condition1.allDependentConditions[0] = new DependentCondition(
      "DepProperty1", CompleteCondition.OP_EQUAL, false,
      AgentProperty.TYPE_INTEGER, "0");
   condition1.allDependentConditions[1] = new DependentCondition(
      "DepProperty1", CompleteCondition.OP_EQUAL, false,
      AgentProperty.TYPE_INTEGER, "1");
   condition1.allDependentConditions[2] = new DependentCondition(
      "DepProperty1", CompleteCondition.OP_EQUAL, false,
      AgentProperty.TYPE_INTEGER, "256");
   condition1.allDependentConditions[3] = new DependentCondition(
      "DepProperty1", CompleteCondition.OP_EQUAL, false,
      AgentProperty.TYPE_INTEGER, "512");
// 3. Instantiate the next complete-condition object
condition2 = new CompleteCondition();
// 4. Create the condition on the "optionB" value
//   a) Instantiate the input condition on "optionB"
   condition2.allInputConditions[0] = new InputCondition(
      CompleteCondition.OP_EQUAL, false, AgentProperty.TYPE_STRING,
      "optionB");
//   b) Instantiate the dependent conditions for DepProperty2
   condition2.allDependentConditions[0] = new DependentCondition(
      "DepProperty2", CompleteCondition.OP_GREATER_THAN_EQUAL, false,
      AgentProperty.TYPE_INTEGER, "1");
   condition2.allDependentConditions[1] = new DependentCondition(
      "DepProperty2", CompleteCondition.OP_LESS_THAN_EQUAL, false,
      AgentProperty.TYPE_INTEGER, "5");
// Save conditions in the agent-property object
agentProp.allDependencies[0] = condition1;
agentProp.allDependencies[1] = condition2;
```

*Figura 76. Definizione di condizioni di proprietà dell'agente*

## Chiusura di ODA

Dopo che ODA ha generato il contenuto appropriato, la Procedura guidata dell'oggetto Business visualizza il passo 6, casella di dialogo Salva oggetti business (Passo 6). Questa casella di dialogo consente di specificare come salvare il contenuto generato. Come parte di questo passo, la Procedura guidata dell'oggetto

Business chiude ODA. Il runtime ODA richiama il metodo `terminate()` per effettuare le attività di clean-up e di rilasciare risorse dall'ODA. Ad esempio, se l'ODA è connesso a una origine dati nel metodo `init()`, deve scollegarsi da questa origine nel relativo metodo `terminate()`. Nell'API ODK, il metodo `terminate()` per un ODA è parte della classe base di livello inferiore ODA, `ODKAgentBase`. Viene ereditato dalla classe di base ODA, `ODKAgentBase2`, e quindi dalla classe ODA.

Figura 77 illustra un esempio di metodo `terminate()` per un ODA che chiude una connessione di database ed esegue il clean-up su oggetti che hanno acceduto al database.

```
public void terminate()
{
   specList = null;
   //close connection
   if(db != null)
      db.disconnect();
   if(dbAnalizer != null)
      dbAnalizer.cleanup();
 }
```

*Figura 77. Metodo terminate() ODA di esempio*

## Gestione dei messaggi di errore e di traccia

Un *messaggio* è una stringa di informazioni che ODA può inviare ad un log ODA esterno, dove può essere controllato da un amministratore di sistema oppure dallo sviluppatore per fornire informazioni sullo stato del runtime di ODA. Esistono due differenti categorie di messaggi che possono essere inviate da ODA al log ODA:

- Messaggi di errore o informativi
- Messaggi di traccia

I messaggi possono essere generati all'interno del codice ODA oppure ottenuti da un file di messaggi. L'API ODK fornisce un metodo `trace()`, definito nella classe `ODKUtility` per registrate i messaggi di traccia o di errore. Questa sezione fornisce le seguenti informazioni:

- "Indicazione della destinazione del log"
- "Invio di un messaggio al file di traccia" a pagina 164
- "File di messaggi" a pagina 166

### Indicazione della destinazione del log

ODA invia i messaggi nella destinazione del log. Il *log* è una destinazione esterna, pronta per essere visualizzata da chi deve controllare lo stato di avvio di ODA. La destinazione del log è definita durante la configurazione dell'ODA mediante la prorpietà di configurazione `TraceFileName` come un percorso assoluto di un file esterno, che deve trovarsi sulla stessa macchina del processo di ODA.

**Nota:** Poiché l'API ODK fornisce un metodo per registrare i messaggi di traccia e di errore, ODA ha solo un file per conservare entrambi questi tipi di messaggi. Pertanto, nonostante questo file sia denominato *file di traccia*, contiene anche eventuali messaggi di errore generati da ODA.

Per informazioni sul formato del nome del file di traccia, consultare "Specifica del file di traccia" a pagina 83.

## Invio di un messaggio al file di traccia

L'API ODK fornisce un metodo `trace()`, definito nella classe `ODKUtility` per registrate i messaggi di traccia o di errore. Il tipo di messaggio che il meccanismo di tracciamento ODA invia dipende dal livello di traccia del messaggio, nel modo seguente:

*Tabella 59. Livello di traccia e tipo di messaggio*

| Livello di traccia del messaggio | Descrizione |
|---|---|
| zero (0) | Il meccanismo di traccia di ODA consente di registrare i *messaggi di errore* sul file di traccia. |
| Qualsiasi livello tra 1 e 5 | Il meccanismo di traccia di ODA consente di registrare i *messaggi di traccia* sul file di traccia. I messaggi di traccia sono per informazioni quali i messaggi di stato, i valori delle proprietà e i nomi di oggetti business. |

**Nota:** il runtime ODA gestisce implicitamente il meccanismo di traccia di ODA. Questo meccanismo *non* ha effetto finché il file di traccia non viene impostato nella casella di dialogo Configura agente della Procedura guidata dell'oggetto Business. Per ulteriori informazioni, consultare "Avvio di ODA" a pagina 110.

Nella chiamata a `trace()`, è necessario specificare il livello di traccia come argomento. L'API ODK fornisce le costanti del livello di traccia per questo scopo. Per ulteriori informazioni su come generare un messaggio, consultare "Creazione di una stringa di messaggio" a pagina 168. Per informazioni sull'impostazione del livello di traccia, consultare "Specifica del file di traccia e del livello di traccia" a pagina 83.

Il meccanismo di traccia di ODA genera i file nello stesso formato di quelli in Connector Development Kit e InterChange Server Express.

### Messaggi di errore e informativi

Quando il livello di traccia è impostato a zero (0), un ODA può inviare informazioni sullo stato alla destinazione del log. La creazione di un record di errori e dello stato viene spesso chiamata *log*. Nel log sono consigliati i seguenti tipi di informazioni:

- Errori e errori irreversibili dal codice al file di log
- Avvertenze che richiedono l'attenzione dell'amministratore di sistema, dal codice al file di log.
- Messaggi informativi quali:
  - Avvio di ODA e terminazione messaggi
  - Messaggi importanti provenienti dall'applicazione

Sebbene un ODA possa inviare messaggi informativi o di errore, a questo processo di registrazione si fa riferimento come *registrazione degli errori*.

**Importante:** È preferibile che per ogni eccezione venga generata l'eccezione in modo tale che venga visualizzata nella Procedura guidata dell'oggetto Business e venga scritto un messaggio di errore che descrive l'eccezione nel file di traccia. Registrando tutte le eccezioni nel file di traccia, è ancora possibile individuarle nel caso in cui ODA o Business Object Designer Express dovessero avessero un problema.

La registrazione degli errori viene attivata quando il livello di traccia è zero (0). Per impostazione predefinita, la registrazione di un ODA viene disattivata poiché il livello di traccia predefinito è 5. Impostare il livello di traccia con la proprietà di configurazione `TraceLevel`. È possibile impostare `TraceLevel` ad un valore pari a 0 per indicare che un messaggio è un messaggio di errore.

Per inviare un messaggio di errore al log, utilizzare il metodo `trace()`. Tabella 60 riepiloga le informazioni di traccia per `trace()` per inviare un messaggio di errore.

*Tabella 60. Informazioni di traccia per messaggi di errore*

| Informazioni di traccia | Descrizione | Costante ODKConstant |
|---|---|---|
| Livello di traccia | 0 | `TRACELEVEL0` |
| Tipo di messaggio | Errori | `XRD_FATAL, XRD_ERROR` |
| | Avvertenze | `XRD_URGENTWARNING, XRD_WARNING` |
| | Informativo | `XRD_INFO` |

Oltre alle informazioni in Tabella 60, il metodo `trace()` richiede anche il contenuto del messaggio di errore. È possibile ottenere il contenuto dei messaggi come testo di messaggio in uno dei seguenti modi:

- Una stringa di messaggio (un valore `String`)

  ```
  Util.trace(ODKConstant.TRACELEVEL0, ODKConstant.XRD_ERROR,
        "Invalid property value");
  ```

  È anche possibile richiamare la stringa di messaggi da un'eccezione con il metodo `getMsg()` della classe `ODKException` nel modo seguente:

  ```
  try
     {
     boDef.setAttributeList(Attributes);
     boDef.setVerbList(Verbs);
     defList[i] = boDef;
     }
  catch(BusObjInvalidAttrException e)
     {
     Util.trace(ODKConstant.TRACELEVEL0,
        ODKConstant.XRD_ERROR, e.getMsg());
     }
  ```

- Un messaggio che viene richiamato da un file di messaggi

  ```
  Util.trace(ODKConstant.TRACELEVEL0, ODKConstant.XRD_WARNING, 1009);
  ```

  Per ulteriori informazioni sull'utilizzo dei file di messaggi, consultare "File di messaggi" a pagina 166.

## Messaggi di traccia

La traccia è una funzione opzionale di risoluzione dei problemi e di debug che può essere attivata per gli ODA. Quando la traccia è attiva, gli amministratori di sistema possono seguire la generazione di contenuto mentre ODA esegue le proprie attività. La traccia consente a tutti gli utenti del codice ODA di controllare il comportamento di ODA. La traccia può anche essere disattivata quando vengono richiamati metodi specifici ODA.

La traccia viene attivata quando il livello di traccia è compreso tra 1 e 5. Per impostazione predefinita, la registrazione di un ODA viene attivata poiché il livello di traccia predefinito è 5. Impostare il livello di traccia con la proprietà di configurazione `TraceLevel`. È possibile impostare `TraceLevel` ad un valore da 1 a 5 per ottenere il livello appropriato di dettagli. La traccia di livello 5 registra i

messaggi di traccia di *tutti* i livelli di traccia più bassi. Si è responsabile per la definizione di quali informazioni restituisce ODA ad ogni di livello di traccia. Tabella 17 a pagina 84 illustra i componenti consigliati dai messaggi di traccia ODA. Per ulteriori informazioni, consultare "Impostazione del livello di traccia" a pagina 83.

Per inviare un messaggio di traccia, utilizzare il metodo `trace()`. Tabella 60 riepiloga le informazioni di traccia per `trace()` per inviare un messaggio di traccia.

*Tabella 61. Informazioni di traccia per i messaggi di traccia*

| Informazioni di traccia | Descrizione | Costante ODKConstant |
|---|---|---|
| Livello di traccia | 1 | `TRACELEVEL1` |
| | 2 | `TRACELEVEL2` |
| | 3 | `TRACELEVEL3` |
| | 4 | `TRACELEVEL4` |
| | 5 | `TRACELEVEL5` |
| Tipo di messaggio | Traccia | `XRD_TRACE` |

Oltre alle informazioni in Tabella 61, il metodo `trace()` richiede anche il contenuto del messaggio di traccia. È possibile ottenere il contenuto dei messaggi in uno dei seguenti modi:

- Come messaggio di testo:
  - Una stringa di messaggio (un valore `String`)

    ```
    Util.trace(ODKConstant.TRACELEVEL1, ODKConstant.XRD_TRACE,
          "Entering method getProperties");
    ```

  - Un messaggio che viene richiamato da un file di messaggi

    ```
    Util.trace(ODKConstant.TRACELEVEL1, ODKConstant.XRD_TRACE, 1009);
    ```

    Per ulteriori informazioni sull'utilizzo dei file di messaggi, consultare "File di messaggi".
- Come una definizione di oggetti business (un oggetto `BusObjDef`)

  In questo caso, `trace()` formatta i contenuti della definizione di oggetti business specificata.

  ```
  BusObjDef boDef = new BusObjDef();
  // code that populates business object definition
  ...
  // write out the business object definition
  ODKUtility.getODKUtility().trace(ODKConstant.TRACELEVEL5,
     ODKConstant.XRD_TRACE, boDef);
  ```
- Come un'array della proprietà dell'agente (oggetti `AgentProperty`)

  In questo caso, `trace()` formatta l'elenco delle proprietà dell'agente, precedendolo con una stringa che è possibile specificare.

  ```
  AgentProperties[] propArray;
  // code that populates agent-property array
  ...
  // write out the agent-property array
  ODKUtility.getODKUtility().trace(ODKConstant.TRACELEVEL2,
     ODKConstant.XRD_TRACE, propArray,
     "List of configuration properties:");
  ```

## File di messaggi

In entrambi i messaggi di errore e di traccia è possibile fornire il contenuto del messaggio come una stringa protetta o come una stringa richiamata da un *file di messaggi*. Un file di messaggi è un file di testo che contiene numeri di messaggio e

testo di messaggio associato. Il testo del messaggio può contenere i parametri della posizione per trasmettere i dati di runtime al di fuori di ODA. È possibile fornire un file di messaggi creando un file e definendo il messaggio richiesto.

In questa sezione vengono fornite le seguenti informazioni sui file di messaggi:

- "Formato del messaggio"
- "Nome e ubicazione di un file di messaggi"
- "Creazione di una stringa di messaggio" a pagina 168
- "Conservazione del file di messaggi" a pagina 170

## Formato del messaggio

Con un file di messaggi, i messaggi hanno il seguente formato:

```
Messaggio MessageNum
[EXPL]
Spiegazione
```

Il *MessageNum* è un numero intero che identifica in modo univoco il messaggio. Questo numero di messaggio deve essere visualizzato su una riga. Il testo del *Messaggio* può essere su più righe, con un ritorno carrello alla fine di ciascuna riga. Il testo della *Spiegazione* è una spiegazione più dettagliata della condizione che provoca la creazione del messaggio. Non inserire una riga vuota dopo l'ultima riga del testo esplicativo. Il numero del successivo messaggio deve essere visualizzato sulla riga immediatamente dopo la spiegazione. Modificare il file di messaggi con un editor di testo, quale Blocco Note.

Ad esempio, il numero di messaggio 1005 potrebbe essere simile al seguente esempio:

```
1005
La generazione di contenuto ODA è completa.
[EXPL]
Questo è un messaggio di log che indica il completamento dell'ODA.
```

I messaggi possono contenere parametri i cui valori sono sostituiti al runtime dai valori provenienti dal programma. Tali parametri sono posizionali e vengono indicati nel messaggio da un numero tra parentesi. Ad esempio, il seguente messaggio ha tre parametri per specificare i nomi della proprietà dell'agente:

```
1003
Le proprietà di configurazione dell'agente sono {1}, {2}, {3}.
[EXPL]
Questo è un messaggio di traccia che fornisce le proprietà di avvio.
```

Per ulteriori informazioni su come fornire parametri di un messaggio, consultare "Utilizzo dei valori dei parametri" a pagina 169.

## Nome e ubicazione di un file di messaggi

Un ODA può ottenere i messaggi da uno dei due file di messaggi:

- Un file di messaggi ODA è denominato

  *ODAname*`Agent.txt`

  dove *ODAname* è il nome che identifica ODA in modo univoco. Per ulteriori informazioni, consultare "Denominazione dell'ODA" a pagina 173. Inserire i messaggi specifici di ODA in questo file di messaggi. Ad esempio, se si crea un ODA denominato `LegacyApp` , il nome del file di messaggi è `LegacyAppAgent.txt`.

**Nota:** La Procedura guidata dell'oggetto Business include automaticamente `MessageFile` nell'elenco delle proprietà di configurazione con il nome del file di messaggi nel formato *ODAname*`Agent.txt`. Quando si configura ODA, è possibile modificare questo file di messaggi per puntare ad un file esistente. Il file di messaggi specificato *deve* esistere per ODA per proseguire l'esecuzione. Per informazioni su come specificare il file di messaggi, consultare "Specifica del file di messaggio di ODA" a pagina 84.

- Il file di messaggi globali ODA viene denominato `useragentmessages.txt`.

  Se si creano messaggi che sono globali per *tutti* gli ODA (Object Discovery Agents) aggiungere i messaggi al file di messaggi globali.

Entrambi questi file di messaggi devono trovarsi nella seguente sottodirectory della directory del prodotto:

*ProductDir*`\ODA\messages`

## Creazione di una stringa di messaggio

I metodi nel richiamare un messaggio predefinito da file di messaggi

*Tabella 62. Metodi che creano una stringa di messaggio*

| Metodo del messaggio | Descrizione |
|---|---|
| `getMsg()` | Genera un messaggi della gravità specificata da un file di messaggi |
| `trace()` | Genera un messaggio della gravità specificata da un file di messaggi e lo invia al file di traccia. |

I metodi di generazione messaggi in Tabella 62 sono definito nella classe `ODKUtility`. Tali metodi richiedono le seguenti informazioni:

- "Specifica di un numero di messaggio"
- "Specifica di un tipo di messaggio"
- "Utilizzo dei valori dei parametri" a pagina 169

**Specifica di un numero di messaggio:** I metodi di generazione dei messaggi riportati in Tabella 62 richiedono un numero di messaggio come argomento. Questo argomento specifica il numero di messaggi da ottenere da un file di messaggi. Come descritto in "Formato del messaggio" a pagina 167, ciascun messaggio in un file di messaggi deve avere un numero di messaggio intero univoco associato ad esso. Questi metodi di generazione messaggi effettuano la ricerca nel file di messaggi per il numero del messaggio specificato ed estraggono il testo del messaggio associato.

Tali metodi effettuano la ricerca dei file di messaggi ODA per il numero di messaggio nel seguente ordine:

1. Il file di messaggi specifici ODA, il cui nome predefinito è *ODAname* `Agent.txt`
2. Il file di messaggi globali ODK, `useragentmessages.txt`

**Specifica di un tipo di messaggio:** I metodi di generazione dei messaggi riportati in Tabella 62 richiedono anche un tipo di messaggio come argomento. Questo argomento indica la *gravità* del messaggio. Tabella 63 riporta i tipi di messaggi validi e i relativi tipi associati costanti.

*Tabella 63. Tipi di messaggi*

| Costante del tipo di messaggio | Livello di gravità | Descrizione |
|---|---|---|
| XRD_FATAL | Errore irreversibile | Indica un errore che arresta il programma in esecuzione |
| XRD_ERROR | Errore | Indica un errore che occorre approfondire |
| XRD_URGENTWARNING | Avvertenza urgente | Indica una condizione che rappresenta un problema e probabilmente non deve essere ignorata |
| XRD_WARNING | Attenzione | Indica una condizione che potrebbe rappresentare un problema, che però potrebbe essere ignorato |
| XRD_INFO | Informativo | Solo messaggio informativo, non è richiesta alcuna azione |
| XRD_TRACE | ---- | Utilizzare per i messaggi di traccia |

Per specificare un tipo di messaggio da associare ad un messaggio, utilizzare una delle costanti dei tipi di messaggio in Tabella 63, nel modo seguente:

- Per un messaggio di errore, utilizzare una costante di tipo di messaggio che indica la gravità di un messaggio (in ordine decrescente di gravità): XRD_FATAL, XRD_ERROR, XRD_URGENTWARNING, XRD_WARNING oppure XRD_INFO.
- Per un messaggio di traccia, utilizzare la costante XRD_TRACE.

Le costanti dei tipi di messaggio vengono definite nella classe ODKConstant.

**Utilizzo dei valori dei parametri:** Non è necessario scrivere messaggi separati per ciascuna situazione possibile. Utilizzare i parametri che rappresentano i valori che cambiano al runtime. L'utilizzo dei parametri consente ad ogni messaggio di essere utilizzato in più situazioni e aiuta a limitare le dimensioni del file di messaggi.

Un paramento viene sempre visualizzato come un numero compreso tra parentesi graffe: { *numero*}. Per ciascun parametro occorre aggiungere il messaggio, inserire il numero tra parentesi graffe all'interno del testo del messaggio, nel modo seguente:

```
testo messaggio {numero} altro testo messaggio.
```

Con i metodi di generazione messaggi in Tabella 62 a pagina 168, è possibile specificare un numero opzionale di valori per i parametri di messaggi. Il numero di valori dei parametri nella chiamata del metodo deve corrispondere al numero di parametri definiti nel testo del messaggio. Il metodo di generazione messaggi deve fornire un valore per ogni parametro. Ad esempio, consideriamo di nuovo il messaggio 1003:

```
1003
Le proprietà di configurazione dell'agente sono {1}, {2}, {3}.
[EXPL]
Questo è un messaggio di traccia che fornisce le proprietà di avvio.
```

Nel codice che invia questo messaggio, potrebbe essere visualizzata la seguente riga:

```
Vector params = new Vector(3);
for(int i=0; i<3; i++)
   params.add(agtProperties[i].propName);
Util.trace(ODKConstant.TRACELEVEL2, 1003, ODKConstant.XRD_TRACE,
   params);
```

Il metodo `trace()` combina questi valori dei parametri con il testo del messaggio nel file di messaggi e crea il messaggio. Prima di scrivere il messaggio al file di traccia, `trace()` sostituisce i parametri del messaggio con i valori della variabile `params`.

Il messaggio 1003 potrebbe essere visualizzato nel file di traccia nel modo seguente:

```
Le proprietà di configurazione dell'agente sono Nomeutente, Password e Url.
```

Poiché il messaggio di testo utilizza parametri per specificare i tipi di proprietà specifici, invece di includerle come stringhe protette, è possibile utilizzare lo stesso messaggio per ciascun insieme di proprietà mancanti.

### Conservazione del file di messaggi

Un amministratore potrebbe impostare una procedura per il filtro dei messaggi ODA presso una postazione utente e utilizzare e-mail o il cerca persone per inviare una notifica a chi può risolvere il problema. A tale scopo è importante che il numero di errori e il significato associato con i numeri resti lo stesso dopo il primo release di un ODA (Object Discovery Agent). È possibile modificare il testo associato a dei numeri di errore, ma non occorre modificare il testo o riassegnare i numeri dell'errore.

Tuttavia, se si modifica il significato associato ai numeri di errore, accertarsi cdi tenere traccia della modifica e inviare una notifica agli utenti di ODA.

È possibile modificare un file di messaggi ODA mentre ODA è in esecuzione. Tuttavia, le modifiche non sono attive finché non viene riavviato ODA e il file di messaggio pronto in memoria. Se InterChange Server Express presenta un problema mentre un ODA è in esecuzione, il server legge automaticamente nella memoria i file di messaggi di tutti gli ODA che erano in esecuzione.

## Gestione delle eccezioni

I metodi delle API ODK possono generare eccezioni per indicare certe condizioni predefinite. In questa sezione vengono fornite le seguenti informazioni su come gestire le eccezioni in un connettore Java:

- "Cos'è un'eccezione ODK?"
- "Eccezioni dalla libreria API ODK" a pagina 171

**Nota:** È anche possibile utilizzare il log degli errori e dei messaggi per gestire le condizioni di errore e i messaggi nel connettore. Per ulteriori informazioni, consultare "Gestione dei messaggi di errore e di traccia" a pagina 163.

### Cos'è un'eccezione ODK?

Quando un metodo dell'API ODK emette un'eccezione, questo *oggetto di eccezione* fa parte della classe `ODKException` o di una delle clkassi secondarie, che è una estensione della classe Java `Exception`. Per creare una eccezione ODK, utilizzare il costruttore `ODKException()`. Tabella 64 illustra i metodi di accesso che la classe `ODKException` fornisce per ottenere informazioni nell'oggetto di eccezione.

*Tabella 64. Informazioni contenute nell'oggetto eccezione*

| Membro | Metodo del dispositivo di accesso |
|---|---|
| Testo del messaggio | `getMsg()` |

**Nota:** Per ulteriori informazioni sui metodi nella classe `ODKException`, consultare Capitolo 24, “Classe ODKException”, a pagina 275.

La classe `ODKException` fornisce alcune classi secondarie per indicare le condizioni di errore specificje, come viene illustrato in Tabella 108 a pagina 276.

## Eccezioni dalla libreria API ODK

Quando si scrive il codice per un ODA, è possibile includere istruzioni `try` e `catch` Javaper gestire specifiche eccezioni generate ai metodi delle API ODK. La descrizione del riferimento per la maggior parte dei metodi API ODK ha una sezione deniminata Eccezioni, che elenca le eccezioni generate da questo metodo.

Figura 78 illustra un frammento di codice dall’esempio dell’esercito romano ODA (nella classe `ArmyAgent4`) che rileva le eccezioni generate dal metodo `getClientFile()`.

```
try
   {
   remotefile = ODKUtility.getODKUtility().getClientFile(filePath, this);
   }
catch (IOException ex)            //file was not found
   {
    return null;
   }
//agent doesn't implement IGeneratesBinFiles, so "getClientFile" failed.
catch (UnsupportedContentException ex)
   {                              //We'll return a random Son instance for now.
    return new Son("X" + ("" + new Date().hashCode()).substring(1),
       new Date().hashCode() % 10  + 2);
   }
```

*Figura 78. Rilevazione delle eccezioni da getClientFile()*

Quando un metodo API ODK genera un’eccezione, non fornisce di solito un messaggio e informazioni sullo stato nell’oggetto di eccezione. Tuttavia, è possibile decidere di inserire un messaggio nell’oggetto di eccezione, se desiderato.

# Capitolo 6. Aggiunta di un ODA (Object Discovery Agent) per il sistema di integrazione business

Per il sistema WebSphere Business Integration Server Express per essere capaci di accedere ad un ODA (Object Discovery Agent) sviluppato, occorre eseguire le seguenti operazioni:

1. Stabilire il nome ODA e le relative convenzioni di denominazione.
2. Compilare la classe ODA in un file jar.
3. Creare lo script di avvio ODA.

## Denominazione dell'ODA

Questo capitolo fornisce convenzioni di denominazione suggerite per i file e le directory utilizzate nello sviluppo ODA. Le convenzioni di denominazione forniscono un modo per rendere il codice di ODA più facile da localizzare e identificare. Tabella 65 riepiloga le convenzioni di denominazione suggerite per un ODA.

*Tabella 65. Convenzioni di denominazioni suggerite per un ODA*

| nome ODA | pacchetto e nome classe di ODA | script di avvio di ODA | file di libreria di ODA | directory di esecuzione di ODA |
|---|---|---|---|---|
| *srcDataName*ODA | com.ibm.oda.*srcDataName*.*ODAname* | start_*ODAname* | *ODAname*.jar | ODA*srcDataName* |

Ciascun ODA deve avere un nome che lo identifica in modo univoco nel sistema WebSphere Business Integration Server Express. Per convenzione, un nome ODA *ODAname*) assume il seguente formato:

*srcDataName*ODA

dove *srcDataName* è una stringa univoca che identifica i dati di origine convertiti da ODA. Ad esempio, se un ODA converte oggetti HTML in definizioni di oggetti business, i relativi dati di origine sono in formato HTML. Pertanto, il relativo nome ODA è HTMLODA. In alternativa, questo nome ODA può identificare l'adattatore con cui viene associato ODA. Ad esempio, l'ODA che genera definizioni di oggetti business per l'adattatore per PeopleSoft ha come nome ODA PeopleSoftODA.

## Compilazione di ODA

Per compilare un ODA, effettuare le seguenti operazioni:

- Utilizzare un ambiente di sviluppo JDK. Per informazioni su come installare JDK, consultare "Impostazione dell'ambiente di sviluppo" a pagina 107.
- Accertarsi che i file di libreria per l'API ODK (Object Discovery Agent Development Kit) sono nella sottodirectory lib della directory del prodotto. Il file di libreria dell'API ODK viene denominata come segue:

  CwODK.jar

  I file di libreria ODK aggiuntivi sono:

  xrmi.jar, xerces.jar

- Compilare i file di origine ODA (.java) nei file di classe (.class) con il compilatore Java.

  Questi file includono l'origine della classe ODA (che è un'estensione della classe `ODAAgentBase2`) così come qualsiasi altra classe utilizzata da ODA. Per informazioni sulla denominazione dei file di classe ODA, consultare "Estensione della classe di base ODA" a pagina 109.
- Creare il file di libreria ODA, che è un file di archivio (jar) Java che contiene il codice Java compilato.

  Per convenzione, il nome del file jar assume il seguente formato:

  *srcDataName*`ODA.jar`

  dove *srcDataName* identifica in modo univoco i dati di origine (o l'adattatore) per l'ODA. Per ulteriori informazioni sul nome ODA, consultare "Denominazione dell'ODA" a pagina 173.

  Ad esempio, per un ODA che opera con `HTML`, il relativo nome ODA sarebbe `HTMLODA`. Pertanto, assegnare al relativo file jar il seguente nome:

  `HTMLODA.jar`

## Avvio di un nuovo ODA

Per avviare ODA, è neecssario eseguire uno *script di avvio ODA*. Questo script avvia il runtime ODA. Questo script di avvio è un file batch che avvia il runtime di ODA. Per convenzione, il nome dello script di avvio assume il seguente formato:

`start_`*`ODAName`*`.bat`

dove ODAname è il nome univoco dell'ODA (il relativo nome di dati origine) con la stringa "`ODA`" aggiunta. Ad esempio, se un ODA ha dati di origine nel formato HTML, il relativo nome ODA sarà `HTMLODA`. Pertanto, assegnare al relativo script di avvio il seguente nome:

`start_HTMLODA.bat`

Prima di avviare un ODA che è stato sviluppato, occorre accertarsi che uno script di avvio esista per supportare un nuovo ODA. Per abilitare il script di avvio per avviare il proprio ODA, è necessario eseguire le seguenti operazioni:

1. Preparare una directory di runtime ODA per il proprio ODA.
2. Creare uno script di avvio per l'ODA. Ad esempio, creare un collegamento per l'avvio di ODA.
3. Impostare uno script di avvio come un servizio (facoltativo).

Nelle seguenti sezioni viene descritta ciascuna delle seguenti operazioni.

### Preparazione di una directory di runtime ODA

La *directory di runtime ODA* contiene i file di runtime per ODA. Per preparare la directory di runtime ODA, eseguire queste operazioni:

1. Creare una directory di runtime ODA per un nuovo ODA sotto la sottodirectory `ODA` della directory del prodotto.

   *ProductDir*`\ODA\`*srcDataName*

   Per convenzione, il nome della directory corrisponde al nome dei dati di origine ODA (*srcDataName*). Il nome dei dati di origine è una stringa che identifica in modo univoco i dati di origine (oppure l'adattatore) che operano con ODA. Per ulteriori informazioni, consultare "Denominazione dell'ODA" a pagina 173.

2. Spostare il file della libreria ODA in questa directory di runtime ODA.

   Il file di libreria ODA è un file di archivio (jar) Java. Questo file jar viene creato quando si compila ODA. Per ulteriori informazioni, consultare “Compilazione di ODA” a pagina 173.

## Creazione di script di avvio

Come descrive “File di avvio del sistema” a pagina 71, ODA richiede uno script di avvio ODA che sia in grado di effettuare l'avvio. ODA richiede uno script di avvio affinché l'amministratore di sistema possa avviare il processo runtime di ODA. Quando il programma di installazione di WebSphere Business Integration Adapters installa gli adattatori su un sistema Windows, eseguire le seguenti operazioni per gli ODA:

- Installare lo script di avvio `start_`*`ODAname`*`.bat` nella sottodirectory `ODA\`*`srcDataName`* della directory del prodotto.
- Creare le opzioni di menu per ciascun ODA al di sotto di `Programmi > IBM WebSphere Business Integration Adapters > Adattatori > menu ODA (Object Discovery Agent)`. Ciascuna voce di menu è un collegamento che richiama lo script di avvio di Windows `start_`*`ODAname`*`.bat` per ciascun ODA.

Per fornire la capacità di avviare il proprio ODA; occorre generare lo script di avvio e fornire i collegamenti che richiamano lo script di avvio.

### Creazione dello script di avvio

In questo file `start_`*`ODAname`*`.bat`, accertarsi di effettuare le seguenti operazioni:

- Impostare le seguenti variabili all'interno dello script di avvio:

| Nome variabile | Valore |
|---|---|
| PATH | Aggiungere il percorso della directory di runtime ODA all'inizio della variabile PATH (affinché runtime possa individuare il JRE ODA):<br>`PATH="%CROSSWORLDS%"\ODA\`*`ODAruntimeDir`*`;%PATH%`<br>dove *ODAruntimeDir* è la directory di runtime ODA, che ha il formato *srcDataName*. Per ulteriori informazioni, consultare “Preparazione di una directory di runtime ODA” a pagina 174. |
| AGENTNAME | Specifica il nome ODA per l'ODA (*ODAname*), che ha il seguente formato:<br>*srcDataName*`ODA`<br>dove *srcDataName* è il nome dei dati di origine. Per ulteriori informazioni, consultare “Denominazione dell'ODA” a pagina 173. |

| Nome variabile | Valore |
| --- | --- |
| AGENT | Specificare il percorso completo del file di librerie ODA, il file jar che contiene la classe ODA. Questo nome percorso ha il seguente formato:<br>`"%CROSSWORLDS%"\ODA\`*`ODAruntimeDir`*`\`*`ODAlibrary`*`.jar`<br>dove:<br>• *ODAruntimeDir* è la directory di runtime ODA, che ha il formato *srcDataName*. Per ulteriori informazioni, consultare "Preparazione di una directory di runtime ODA" a pagina 174.<br>• *ODAlibrary* è il file di libreria ODA, che ha il formato *ODAname*`.jar`. Per ulteriori informazioni, consultare "Compilazione di ODA" a pagina 173. |
| AGENTCLASS | Specificare il nome del pacchetto e della classe ODA che ha il seguente formato:<br>`com.ibm.oda.`*`srcDataName`*`.`*`ODAname`*<br>dove:<br>• *srcDataName* è il nome dei dati di origine ODA in lettere minuscole. Per ulteriori informazioni, consultare "Denominazione dell'ODA" a pagina 173.<br>• *ODAname* è il nome della classe ODA (l'estensione della classe base `ODKAgentBase2`) |
| JCLASSES | Aggiungere i file jar specifici di ODA a questa variabile. I file jar son separate da un punto e virgola (;). Questa variabile deve essere impostata almeno per includere le seguenti classi:<br>• I file di libreria ODK: `CwODK.jar`, `xrmi.jar`, `xerces.jar`<br>• Il file di libreria ODA, che viene memorizzato nella variabile d'ambiente AGENT (vedi sopra) |

- Definire e impostare qualsiasi variabile aggiuntiva specifica di ODA che occorre allo script di avvio.

  Definire le variabili per le informazioni che possono cambiare da release a release. È possibile impostare la variabile su un valore appropriato per questo release e quindi includere la variabile nella riga comandi appropriata dello script di avvio. Se le informazioni cambiano in seguito, occorre modificare solo il valore della variabile. Non occorre individuare tutte le righe di comandi che utilizzano queste informazioni.
- Includere i parametri di avvio appropriati sulla riga che richiama il runtime ODA (l'ultima ruga dello script di avvio) compreso quanto segue:
  - Tutti i parametri di avvio richiesti: -l e -c
  - Tutti i parametri di avvio opzionali che si applicano a tutti i richiami di ODA:

    -v: segue questo parametro con la versione di ODA
- La riga che richiama il runtime ODA deve avere il seguente formato:

```
"%CROSSWORLDS%\bin\java" -Duser.home="%CROSSWORLDS%" -mx128m
-classpath %JCLASSES% com.crossworlds.ODKInfrastructure.XRmiAgent
-l%AGENTNAME% -c%AGENTCLASS%
```

  **Nota:** Accertarsi che la riga per richiamare il runtime ODA sia tutta *su una riga* nello script di avvio, cioè che non vi siano ritorni a capo all'interruzione riga illustrata nella riga di avvio di esempio.

## Creazione di collegamenti

Un collegamento consente di avviare ODA da una voce di menu contenuta in `Programmi > IBM WebSphere Business Integration Adapters > Adattatori > Object Discovery Agents`. Un modo semplice per creare un collegamento per avviare un ODA in esecuzione su Windows è di copiare il collegamento ODA esistente e modificarne le proprietà per cambiare il nome del connettore o aggiungere qualsiasi altro parametro di avvio.

# Parte 3. Riferimento della classe ODK

# Capitolo 7. Panoramica dell'API ODK

L'API (Application Programming Interface) ODK (Object Discovery Agent Development Kit) include le librerie delle classi che occorre utilizzare quando si sviluppa un ODA. Tale API ODK contiene classi predefinite per gli ODA. Utilizzare queste librerie di classi per derivare classi e metodi ODA. L'API ODA fornisce utility, quali metodi per implementate servizi di traccia e di log.

IBM fornisce un file jar Java (file di archivio Java), `CwODK.jar`, che contiene le classi predefinite e le interfacce dell'API ODK. Questo file jar si trova nella sottodirectory `lib` della directory del prodotto.

**Nota:** Per istruzioni sulla creazione di un ODA da eseguire su Windows 2003, consultare "Compilazione di ODA" a pagina 173.

## Classi e interfacce

Le classi e le interfacce dell' API ODK appartengono al seguente pacchetto:
`com.crossworlds.ODK`

Tabella 66 elenca le classi e le interfacce nell'API ODK.

*Tabella 66. Classi e interfacce del'API ODK*

| Classe o interfaccia | Descrizione | Pagina |
|---|---|---|
| `AgentMetaData` | Rappresenta un oggetto di proprietà dell'agente, che può rappresentare una proprietà di avvio o una proprietà di oggetti business. | 183 |
| `AgentProperty` | Definisce le costanti per il tipo attributo | 189 |
| `BusObjAttr` | Rappresenta un attributo all'interno della definizione dell'oggetto business | 199 |
| `BusObjAttrType` | L'interfaccia che definisce le costanti del tipo attributo | |
| `BusObjDef` | Rappresenta una definizione di oggetto business che descrive un oggetto business | 215 |
| `BusObjVerb` | Rappresenta un verbo di oggetto business, che descrive un'azione o un'operazione valida sull'oggett o business | 227 |
| `CompleteCondition` | | 231 |
| `ContentMetaData` | Rappresenta i metadati del contenuto di ODA, che descrivono il contenuto generato da ODA | 235 |
| `ContentType` | Rappresenta il tipo di contenuto supportato da ODA | 239 |
| `DependentCondition` | | 247 |
| `IGeneratesBinFiles` | L'interfaccia che deve essere implementata da ODA per fornire supporto per la generazione di file binari dai dati origine | 251 |
| `IGeneratesBoDefs` | L'interfaccia che deve essere implementata da ODA per fornire supporto per la generazione di definizioni di oggetti business dai dati origine | 255 |

*Tabella 66. Classi e interfacce del'API ODK (Continua)*

| Classe o interfaccia | Descrizione | Pagina |
|---|---|---|
| `IGeneratesContent` | È la classe base per le due interfacce di generazione contenuto, `IGeneratesBinFiles` e `IGeneratesBoDefs`. Definisce il metodo `getContentProtocol()`.<br>**Nota:** In questo manuale *non* viene fornito un capitolo separato per questa interfaccia. Per informazioni su `getContentProtocol()`, consultare la descrizione dell'interfaccia `IGeneratesBinFiles` o `IGeneratesBoDefs` | Nessuno |
| `InputCondition` | | 261 |
| `ODKAgentBase` | È la classe base per la classe base ODA, `ODKAgentBase2`. Definisce svariati metodi che eredita `ODKAgentBase2`.<br>**Nota:** In questo manuale *non* viene fornito un capitolo separato per questa classe. Per informazioni consultare la descrizione della classe `ODKAgentBase2`. | Nessuno |
| `ODKAgentBase2` | Rappresenta la classe base per un ODA Questa classe viene estesa per definire la classe ODA e implementa i metodi richiesti. | 265 |
| `ODKConstant` | L'interfaccia che definisce le costanti da utilizzare con l'API ODK:<br>• costanti dello stato di risultato<br>• costanti verbo | 269 |
| `ODKException` | Rappresenta un oggetto eccezione per l'API ODK | 275 |
| `ODKUtility` | Fornisce una serie di metodi di utility per utilizzarli in un ODA. Tali metodi di utility fanno parte delle seguenti categorie generali:<br>• Metodi statici per la generazione e il log dei messaggi<br>• Metodi statici per la creazione di oggetti business<br>• Metodi statici per ottenere proprietà di configurazione connettori<br>• Metodi per ottenere informazioni locali | 277 |
| `TreeNode` | | 291 |

# Capitolo 8. Classe AgentMetaData

L' Object Discovery Agent Development Kit (ODK) API fornisce la `AgentMetaData` classe per contenere i metadati per Object Discovery Agent (ODA). Le variabili dei membri di questa classe rappresentano i metadati ODA. Procedura guidata dell'oggetto Business può accedere ai metadati ODA's richiamando il metodo `getMetaData()` nella classe ODA'.

La classe `AgentMetaData` definisce quanto segue:

- "Variabili del membro"
- "Metodi" a pagina 185

La classe `AgentMetaData` implementa l'interfaccia `ODKConstant`. Pertanto, tutte le costanti definite in `ODKConstant` sono disponibili in un oggetto `AgentMetaData`. Per un elenco delle costanti che l'interfaccia `ODKConstant` definisce, consultare Capitolo 23, "Interfaccia ODKConstant", a pagina 269.

## Variabili del membro

Tabella 67 riepiloga le variabili dei membri della classe `AgentMetaData`.

*Tabella 67. Variabili dei membri della classe AgentMetaData.*

| Variabile del membro | Descrizione | Pagina |
|---|---|---|
| `agentVersion` | Specifica la versione per ODA. | 190 |
| `searchableNodes` | Determina se è possibile effettuare una ricerca per modello specificato dall'utente di elementi secondari di nodi espansibili (nel nodo radice). | 190 |
| `searchPatternDesc` | Specifica la descrizione da visualizzare agli utenti per spiegare un valido criterio del modello di ricerca. | 191 |
| `supportedContent` | Memorizza una descrizione del protocollo del contenuto che ODA supporta per ciascuno dei suoi tipi di contenuto supportato. | 191 |

### agentVersion

Specifica la versione per ODA.

#### Tipo

```
Stringa pubblica agentVersion
```

#### Note

Il secondo formato del costruttore `AgentMetaData()` può inizializzare la variabile del membro `agentVersion`. Se non si inizializza `agentVersion`, il suo valore predefinito prevede una stringa vuota. ODA dovrebbe inizializzare la sua versione ODA come parte del metodo `getMetaData()`, che inizializza i metadati di ODA'.

### searchableNodes

Indica se è possibile effettuare una ricerca per nodi specificati dall'utente di elementi secondari di nodi espandibili (nel nodo tree). modello di ricerca.

### Tipo

```
searchableNodes booleano pubblico
```

### Note

La variabile del membro `searchableNodes` contiene un valore booleano che determina se all'utente è consentito ricercare elementi secondari di un nodo espandibile nel nodo tree (nella finestra di dialogo per la selezione del sorgente di Procedura guidata dell'oggetto Business):

- Se la variabile è `true`, Procedura guidata dell'oggetto Business abilita la voce del menu **Ricerca per voci** quando l'utente fa clic con il tasto destro sul nome del nodo espandibile. L'utente può fare clic su questa voce del menu per visualizzare la finestra di dialogo Immettere un modello di ricerca. In essa, l'utente può specificare un modello di ricerca.

  Procedura guidata dell'oggetto Business chiama il metodo `getTreeNodes()` per ricercare il nodo principale, superando il modello di ricerca specificato dall'utente. Il metodo di ricerca `getTreeNodes()` ricerca il sorgente dei dati per quegli elementi secondari i cui nomi corrispondono con i modelli di ricerca, restituendo solo quelli che corrispondono. Procedura guidata dell'oggetto Business visualizza all'utente questi elementi secondari quando visualizza il nodo principale espanso.
- Se questa variabile è `false`, la voce del menu **Ricerca per voci** non è disponibile quando l'utente fa clic con il tasto destro sul nome del nodo espandibile. In questo caso, il metodo `getTreeNodes()` non ha bisogno di gestire un modello di ricerca specificato.

il costruttore di `AgentMetaData()` *non* inizializza la variabile del membro `searchableNodes`. Se non si inizializza `searchableNodes`, il suo valore predefinito viene impostato su `false`. Se ODA supporta la funzione di modello di ricerca, deve inizializzare la variabile del membro `searchableNodes` come parte del metodo `getMetaData()` nella classe ODA. Per ulteriori informazioni, consultare "Implementazione della funzione del modello di ricerca" a pagina 124.

## searchPatternDesc

Specifica la descrizione da visualizzare agli utenti per spiegare un valido criterio di modello di ricerca.

### Tipo

```
Stringa pubblica searchPatternDesc
```

### Note

La variabile del membro `searchPatternDesc` memorizza la descrizione del modello di ricerca visualizzata nella casella di dialogo Immetti un modello di ricerca. Procedura guidata dell'oggetto Business visualizza questa casella di dialogo quando l'utente fa clic con il tasto destro sul nodo sorgente e fa clic su **Ricerca per voci**. Questa descrizione fornisce informazioni sulla semantica che l'utente deve usare per specificare i criteri di ricerca; cioè, descrive quali sono i criteri di ricerca che ODA implementa. La variabile di questo membro contiene un valore valido *only* quando la variabile del membro `searchableNodes` è `true`. Se ODA supporta la funzione del modello di ricerca, deve inizializzare la variabile del membro `searchPatternDesc` come parte del metodo `getMetaData()` nella classe ODA. Per ulteriori informazioni, consultare "Implementazione della funzione del modello di ricerca" a pagina 124.

## supportedContent

Contiene un vettore che descrive quale protocollo del contenuto è supportato da ODA per ciascuno dei suoi tipi di contenuto supportato.

### Tipo

```
Vettore pubblico supportedContent
```

### Note

La variabile del membro `supportedContent` memorizza unJava `java.util.Vector` di oggetti `ContentProtocol` che descrivono qual è il contenuto supportato da ODA. Ciascun oggetto `ContentProtocol` contiene le seguenti informazioni:

| **informazioni sulla generazione del contenuto** | **Descrizione** |
|---|---|
| Tipo di contenuto | Un oggetto di `ContentType`, che elenca uno dei tipi di contenuto supportato:<br>• `BusinessObject`<br>• `BinaryFile` |
| Protocollo del contenuto | Una maschera delle costanti del protocollo del contenuto, per indicare i protocolli del contenuto supportato per il tipo di contenuto specificato:<br>• `CONTENT_PROTOCOL_ONREQUEST`<br>• `CONTENT_PROTOCOL_CALLBACK`<br>Le costanti del protocollo del contenuto sono definite nell'interfaccia `ODKConstant`. |

**Nota:** La classe `ContentProtocol` è parte del pacchetto `ODAInfrastructure`, che contiene le classi utilizzate da ODA runtime e da Procedura guidata dell'oggetto Business. Questo pacchetto *non* è reso disponibile agli sviluppatori ODA. Tutti gli accessi agli oggetti `ContentProtocol` sono gestiti da ODA runtime o da Procedura guidata dell'oggetto Business. ODA *non* accede agli oggetti di questa classe direttamente.

Il costruttore `AgentMetaData()` inizializza la variabile del membro `supportedContent` facendo una query dell'oggetto ODA che riceve come argomento. *Non* si deve inizializzare esplicitamente la variabile di questo membro.

# Metodi

Tabella 68 riepiloga i metodi della classe `AgentMetaData`.

*Tabella 68. Metodi del membro classe AgentMetaData*

| **Metodi del membro** | **Descrizione** | **Pagina** |
|---|---|---|
| AgentMetaData() | Crea un oggetto dell'agent-metadata. | 186 |
| toXml() | Copia la proprietà specificata nell'oggetto corrente AgentProperty. | 187 |

## AgentMetaData()

Crea un oggetto dell'agent-metadata.

### Sintassi

```
AgentMetaData pubblico (ODKAgentBase2 ODAobject);
AgentProperty pubblico(ODKAgentBase2 ODAobject, Stringa versione);
```

### Parametri

*ODAobject* E' un riferimento agli oggetti ODA che rappresentano ODA. Il costruttore richiede che questo oggetto inizializzi la variabile del membro `supportedContent` dell'oggetto `AgentMetaData` ("supportedContent" a pagina 185).

*versione* Specifica la versione di ODA; il valore di questo parametro inizializza la variabile de membro `agentVersion` dell'oggetto `AgentMetaData` ("agentVersion" a pagina 183).

### Valori di ritorno

Un oggetto `AgentMetaData` di cui è stata appena creata un'istanza.

### Note

Il metodo `AgentMetaData()` richiede *ODAobject* ODA per il suo contenuto supportato. Questo costruttore fornisce i seguenti modelli per creare l'istanza di un nuovo oggetto `AgentMetaData`:

- Il primo modulo definisce un nuovo oggetto `AgentMetaData` e inizializza solo il suo contenuto supportato. Questo formato presume che ODA *non* abbia una versione.
- Il secondo formato definisce un nuovo oggetto `AgentMetaData` e lo inizializza sia con suoi contenuti che con le versioni supportate.

Entrambi i formati del costruttore usano il riferimento a *ODAobject* per richiedere ODA per il suo contenuto supportato. Usando questa informazione, il costruttore inizializza la variabile del membro `supportedContent` .

**Nota:** Il costruttore `AgentMetaData()` *non* inizializza le variabili del membro che supporta la funzione del modello di ricerca. Perché ODA supporti i modelli di ricerca, bisogna esplicitamente inizializzare le variabili dei membri `searchableNodes` e `searchPatternDesc` dopo aver creato l'istanza dell'oggetto `AgentMetaData`. Se non si inizializza `searchableNodes`, il suo valore predefinito viene impostato sul valore `false`.

## toXml()

Converte i metadati ODA in un formato XML.

### Sintassi

```
Stringa pubblica toXml();
```

### Parametri

Nessuno.

### Valori di ritorno

Una stringa che contiene il formato XML per l'oggetto corrente `AgentMetaData`.

# Capitolo 9. classe AgentProperty

L' Object Discovery Agent Development Kit (ODK) API fornisce la classe `AgentProperty` per rappresentare un *oggetto agent-property*. Ciascun oggetto agent-property contiene informazioni sulle proprietà richieste per Object Discovery Agent (ODA), come ad esempio:

- *Proprietà di configurazione*, che fornisce valori di cui ODA ha bisogno per l'inizializzazione.
- *Proprietà Business-object*, che fornisce informazioni aggiuntive di cui ODA ha bisogno per la generazione delle definizioni degli oggetti di business.

La classe `AgentProperty` definisce quanto segue:



## Costanti di tipo Proprietà

La classe `AgentProperty` definisce variabili statiche del membro per rappresentare costanti property-type. Tabella 69 riepiloga queste costanti property-type che rappresentano valori validi per il tipo di dati della proprietà dell'agente tipo. Tutte le costanti di property-type sono degli interi (`int`).

*Tabella 69. Costanti property-type della classe AgentProperty*

| costante property-type | Descrizione |
|---|---|
| `TYPE_BOOLEAN` | Indica che il tipo di proprietà è `Booleano`. |
| `TYPE_DOUBLE` | Indica che il tipo di proprietà è `Doppio`. |
| `TYPE_FLOAT` | Indica che il tipo di proprietà è `Mobile`. |
| `TYPE_INTEGER` | Indica che il tipo di proprietà è `Intero`. |
| `TYPE_STRING` | Indica che il tipo di proprietà è `Stringa`. |

## Variabili del membro

Tabella 70 riepiloga le variabili del membro della classe `AgentProperty`.

*Tabella 70. Variabili del membro della classe AgentProperty.*

| Variabile del membro | Descrizione | Pagina |
|---|---|---|
| `allDefaultValues` | Specifica i valori predefiniti da visualizzare per la proprietà dell'agente. | 190 |
| `allDependencies` | Specifica le condizioni che descrivono le dipendenze tra la proprietà di questo agente e le altre proprietà dipendenti. | 190 |
| `allValidValues` | Specifica i valori da visualizzare per la proprietà dell'agente. | 191 |
| `allValues` | Memorizza i valori che l'utente seleziona per la proprietà dell'agente. | 191 |
| `cardinalità` | Specifica se la proprietà dell'agente può contenere uno o più valori multipli. | 191 |

*Tabella 70. Variabili del membro della classe AgentProperty. (Continua)*

| Variabile del membro | Descrizione | Pagina |
|---|---|---|
| `descrizione` | Fornisce una spiegazione di testo delle proprietà dell'agente e può contenere altre informazioni rilevanti. | 192 |
| `isHidden` | Determina se il valore delle proprietà dell'agente deve essere visualizzata come codificata. | 192 |
| `isMultiple` | Determina se Procedura guidata dell'oggetto Business fornisce un meccanismo per l'immissione utente di valori multipli per il valore agent-property. | 193 |
| `isReadOnly` | Determina se un utente può specificare un valore per la proprietà dell'agente o se può solo visualizzare il valore della proprietà. | 194 |
| `isRequired` | Determina se il valore deve essere sempre specificato per la proprietà dell'agente. | 194 |
| `propName` | Specifica il nome della proprietà dell'agente. | 194 |
| `tipo` | Specifica il tipo di dati della proprietà dell'agente. | 195 |

## allDefaultValues

Specifica i valori predefiniti da visualizzare per la proprietà dell'agente.

### Tipo

```
public java.lang.Object[] allDefaultValues
```

### Note

La variabile del membro `allDefaultValues` contiene un array di valori predefiniti per la proprietà dell'agente. Il numero di elementi `Oggetto` in questo array deve corrispondere alla cardinalità con la proprietà nel seguente modo:

- Per una proprietà single-cardinality (`ODKConstant.SINGLE_CARD`), il array `allDefaultValues` deve contenere *solo un* elemento.
- Per una proprietà multiple-cardinality (`ODKConstant.MULTI_CARD`), il array `allDefaultValues` può contenere *uno o più* elementi.

Per ulteriori informazioni, consultare "Specifica di valori predefiniti" a pagina 156.

## allDependencies

Specifica un elenco delle condizioni che descrivono le dipendenze tra questa proprietà dell'agente e le altre proprietà dipendenti.

### Tipo

```
public CompleteCondition[] allDependencies
```

### Note

La variabile del membro `allDependencies` contiene un elenco delle condizioni nell'array della condizione, che è un array dell'oggetto `CompleteCondition`. Ciascun oggetto `CompleteCondition` contiene una condizione sul valore della proprietà dell'agente. Una condizione contiene condizioni di immissione e di dipendenza. Per ulteriori informazioni, consultare "Impostazione delle condizioni sul valore della proprietà" a pagina 158.

## allValidValues

Specifica i valori validi da visualizzare per la proprietà dell'agente.

### Tipo

```
public java.lang.Object[] allValidValues
```

### Note

La variabile del membro `allValidValues` contiene un elenco di valori con i quali inizializzare l'elenco a scorrimento delle proprietà dell'agente. Da questo elenco a scorrimento, l'utente può scegliere uno (cardinalità singola) o più (cardinalità multipla) valori per la proprietà.

Se `allValidValues` specifica un elenco di valori, Procedura guidata dell'oggetto Business visualizza tali valori nell'elenco a scorrimento per qualunque proprietà dell'agente la cui variabile del membro `isMultiple` sia `true`. Se `isHidden` è `true` e `allValidValues` è nullo, Procedura guidata dell'oggetto Business visualizza una sotto-griglia per gli utenti per specificare i valori.

**Nota:** Se la variabile del membro `isMultiple` è `false`, la variabile del membro `allValidValues` dovrebbe essere `nulla`.

Per ulteriori informazioni, consultare "Selezione del tipo di controllo di visualizzazione" a pagina 155.

## allValues

Memorizza i valori che l'utente fornisce per la proprietà dell'agente.

### Tipo

```
public java.lang.Object[] allValues
```

### Note

La variabile del membro `allValues` è una variabile in uscita; è cioè riempita da Procedura guidata dell'oggetto Business *dopo* il completamento dell'immissione utente. Contiene i valori che l'utente seleziona dalla colonna Valore nel passo Configurazione agente di Procedura guidata dell'oggetto Business. Questa variabile è la *sola* variabile del membro che *non* richiede l'inizializzazione prima che la proprietà dell'agente venga visualizzata all'utente.

il numero di valori nell'array `allValues` è determinato dalla cardinalità della proprietà dell'agente.

- Se la proprità dell'agente ha una cardinalità singola (la sua variabile di `cardinalità` è `ODKConstant.SINGLE_CARD`), l'array `allValues` contiene un valore.
- Se la proprietà dell'agente ha una cardinalità multipla (la sua variabile di `cardinalità` è `ODKConstant.MULTI_CARD`), l'array `allValues` contiene valori multipli, uno per ogni valore specificato dall'utente.

## cardinalità

Specifica se la proprietà dell'agente può contenere uno o più valori.

### Tipo

```
public java.lang.String cardinality
```

### Note

La variabile del membro `cardinalità` determina se il valore della proprietà dell'agente consiste di un valore o valori multipli. Pertanto, determina quanti valori l'utente può specificare per la proprietà.

| Cardinalità | Numero di valori di proprietà dell'agente che l'utente può specificare | Valore della variabile del membro della cardinalità |
|---|---|---|
| Singolo | Uno | `ODKConstant.SINGLE_CARD` |
| Multiplo | Molte | `ODKConstant.MULTIPLE_CARD` |

La proprietà della cardinalità ha un effetto sul tipo di controllo che Procedura guidata dell'oggetto Business visualizza per la proprietà. Per ulteriori informazioni, consultare "Selezione del tipo di controllo di visualizzazione" a pagina 155.

Per inizializzare la cardinalità della proprietà dell'agente, la seguente chiamata al terzo modulo del costruttore di `AgentProperty()` specifica un valore di descrizione della stringa come sesto argomento:

```
AgentProperty agt = new AgentProperty("Username",
        AgentProperty.TYPE_STRING,
        "User Id for logging into the database", true, false,
        ODKConstant.SINGLE_CARD, null, null);
```

**Nota:** E' inoltre possibile specificare un valore per la cardinalità della proprietà dell'agente con il secondo formato del costruttore di `AgentProperty()`, usando i suoi otto argomenti.

## descrizione

Fornisce una spiegazione di testo della proprietà dell'agente e può contenere altre informazioni rilevanti.

### Tipo

```
public java.lang.String description;
```

### Note

La variabile del membro `descrizione` visualizza nella colonna Descrizione nel passo della configurazione dell'agente di Procedura guidata dell'oggetto Business. Per inizializzare la descrizione della proprietà di un agente, la seguente chiamata al terzo modulo del costruttore `AgentProperty()` specifica il valore della descrizione di una stringa come terzo argomento:

```
AgentProperty agt = new AgentProperty("Username",
        AgentProperty.TYPE_STRING,
        "User Id for logging into the database", true, false,
        ODKConstant.SINGLE_CARD, null, null);
```

**Nota:** E' inoltre possibile specificare un valore per la descrizione della proprietà dell'agente con il secondo modulo del costruttore di `AgentProperty()`, usando i suoi sei argomenti.

## isHidden

Determina se il valore della proprietà dell'agente deve essere visualizzato codificato.

### Tipo

```
public boolean isHidden;
```

### Note

La variabile del membro `isHidden` è un valore booleano che determina se un il valore della proprietà dell'agente viene visualizzato in Procedura guidata dell'oggetto Business. Se `isHidden` è true, il valore della proprietà dell'agente viene codificato durante la visualizzazione; cioè, il valore appare come una stringa di asterischi (*). Per indicare se il valore della proprietà dell'agente è codificato, specificare un valore booleano come quarto argomento nel secondo modulo del costruttore `AgentProperty()`:

```
AgentProperty agt = new AgentProperty("Username",
         AgentProperty.TYPE_STRING, true, false, true,
         "User Id for logging into the database", true,
         ODKConstant.SINGLE_CARD, null, null);
```

## isMultiple

Determina se Procedura guidata dell'oggetto Business fornisce un metodo per immettere valori multipli per una proprietà dell'agente.

### Tipo

```
public boolean isMultiple;
```

### Note

La variabile del membro The `isMultiple` è un valore booleano che determina se Procedura guidata dell'oggetto Business deve fornire un meccanismo per consentire l'immissione da parte dell'utente di valori multipli per una proprietà dell'agente:

- Se `isMultiple` è `true`, Procedura guidata dell'oggetto Business visualizza un elenco a discesa con l'elenco dei valori che contiene il membro della variabile `allValidValues`. Da questo elenco, l'utente fa clic sul valore per eseguire l'assegnazione alla proprietà dell'agente. Il valore di questa variabile del membro `cardinality` determina quanti di questi valori l'utente può scegliere dall'elenco a discesa. Se non viene fornito alcun array `allValidValues`, Procedura guidata dell'oggetto Business fornisce una sotto-griglia di righe che l'utente può usare per immettere ciascun valore.
- Se `isMultiple` è `false`, Procedura guidata dell'oggetto Business *non* consente all'utente l'immissione di valori multipli. Visualizza invece un campo vuoto o il valore predefinito (se ne è specificato uno). In questo campo, l'utente immette il valore agent-property. Il valore della variabile del membro della cardinalità deve essere `ODKConstant.SINGLE_CARD`.

**Nota:** Per ulteriori informazioni, consultare "Selezione del tipo di controllo di visualizzazione" a pagina 155.

Per inizializzare la proprietà dell'agente con un elenco di valori multipli da cui l'utente può effettuare una scelta, la seguente chiamata al terzo modulo del costruttore `AgentProperty()` specifica il valore booleano `true` come quarto argomento (il valore della variabile `isMultiple`):

```
AgentProperty agt = new AgentProperty("Username",
         AgentProperty.TYPE_STRING,
         "User Id for logging into the database", true, true,
         ODKConstant.SINGLE_CARD, null, null);
```

**Nota:** E' inoltre possibile specificare un valore per `isMultiple` con il secondo modulo del costruttore `AgentProperty()`, usando il suo settimo argomento.

## isReadOnly

Determina se l'utente può specificare un valore nella proprietà dell'agente o se può solo visualizzare il valore.

### Tipo

```
public boolean isReadOnly;
```

### Note

La variabile de membro `isReadOnly` è un valore booleano che determina se il valore di proprietà dell'agente può essere modificato dall'utente quando la proprietà viene visualizzata in Procedura guidata dell'oggetto Business. Per indicare se è richiesto un valore di proprietà dell'agente, specificare un valore booleano some quinto argomento nel secondo modulo del costruttore `AgentProperty()`:

```
AgentProperty agt = new AgentProperty("Username",
         AgentProperty.TYPE_STRING, true, false, true,
         "User Id for logging into the database", true,
         ODKConstant.SINGLE_CARD, null, null);
```

## isRequired

Determina se è richiesto un valore per la proprietà dell'agente.

### Tipo

```
public boolean isRequired;
```

### Note

La variabile del membro `isRequired` è un valore booleano che determina se deve essere sempre specificato un valore per la proprietà dell'agente o se l'utente può lasciare vuoto il valore della proprietà. Se `isRequired` è `true`, l'utente deve fornire un valore per questa proprietà. Per indicare che è richiesto un valore per la proprietà dell'agente, la seguente chiamata al terzo modulo del costruttore `AgentProperty()` specifica un valore booleano `true` come quarto argomento:

```
AgentProperty agt = new AgentProperty("Username",
         AgentProperty.TYPE_STRING,
         "User Id for logging into the database", true, false,
         ODKConstant.SINGLE_CARD, null, null);
```

**Nota:** E' inoltre possibile specificare un valore per `isRequired` con il secondo modulo del costruttore `AgentProperty()`, usando il suo terzo argomento.

## propName

Specifica il nome della proprietà dell'agente.

### Tipo

```
public java.lang.String propName;
```

### Note

La variabile del membro `propName` contiene una stringa con il nome della proprietà dell'agente — ad esempio: Username, Password, DatabaseUrl. Viene visualizzata la variabile del membro `propName` nella colonna Proprietà nel passo della

configurazione dell'agente di Procedura guidata dell'oggetto Business. Per inizializzare il nome della proprietà dell'agente, la seguente chiamata al terzo modulo del costruttore `AgentProperty()` specifica un nome come primo argomento:

```
AgentProperty agt = new AgentProperty("Username",
         AgentProperty.TYPE_STRING,
         "User Id for logging into the database", true, false,
         ODKConstant.SINGLE_CARD, null, null);
```

**Nota:** Tutti i moduli del costruttore `AgentProperty()` richiedono che si indichi il nome di una proprietà per inizializzare la variabile del membro `propName`.

## tipo

Specifica il tipo della proprietà dell'agente.

### Tipo

```
public int type;
```

### Note

La variabile del membro `type` contiene un valore intero che rappresenta il tipo di dati della proprietà dell'agente. Tabella 69 a pagina 189 elenca le costanti del tipo di proprietà da usare per rappresentare tipi di proprietà validi. La rappresentazione di una stringa del valore della variabile de membro `type` viene visualizzata nella colonna Tipo nel passo per la configurazione dell'agente di Procedura guidata dell'oggetto Business. Per inizializzare il tipo di un dato di proprietà dell'agente, specificare una costante property-type come secondo argomento del costruttore `AgentProperty()`:

```
AgentProperty agt = new AgentProperty("Username",
         AgentProperty.TYPE_STRING,
         "User Id for logging into the database", true, false,
         ODKConstant.SINGLE_CARD, null, null);
```

# Metodi

Tabella 71 riepiloga i metodi della classe `AgentProperty`.

*Tabella 71. Metodo del membro della classe AgentProperty*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| AgentProperty() | Crea un oggetto agent-property. | 195 |
| copy() | Copia la proprietà specificata nell'oggetto corrente AgentProperty. | 197 |

## AgentProperty()

Crea un oggetto agent-property.

### Sintassi

```
public AgentProperty(String name);

public AgentProperty(String name, int type, boolean isReqd, boolean isHid,
   boolean isRdOnly, String desc, boolean isMult, String cardinality,
   Object[] validValues, Object[] defaultValues);

public AgentProperty(String name, int type, String desc, boolean isReqd,
   boolean isMult, String cardinality, Object[] validValues,
   Object[] defaultValues);
```

## Parametri

*cardinalità* Specifica se la proprietà può contenere valori multipli; il valore di questo parametro inizializza la variabile del membro della `cardinalità` dell'oggetto agent-property ("cardinalità" a pagina 191).

*defaultValues* Specifica i valori predefiniti per la proprietà; il valore di questo parametro inizializza la variabile del membro `allDefaultValues` dell'oggetto agent-property ("allDefaultValues" a pagina 190).

*desc* Fornisce una descrizione della proprietà; il valore di questo parametro inizializza la variabile del membro della `descrizione` dell'oggetto agent-property ("descrizione" a pagina 192).

*isHid* Specifica se il valore della proprietà deve essere codificato; il valore di questo parametro inizializza la variabile del membro `isHidden` dell'oggetto agent-property object ("isHidden" a pagina 192).

*isMult* Specifica se la proprietà può fornire valori multipli da cui l'utente può scegliere; il valore di questo parametro inizializza la variabile del membro `isMultiple` dell'oggetto agent-property ("isMultiple" a pagina 193).

*isRdOnly* Specifica se un utente può immettere il valore della proprietà o se può solo visualizzarlo; il valore di questo parametro inizializza la variabile del membro `isReadOnly` dell'oggetto agent-property ("isReadOnly" a pagina 194).

*isReqd* Specifica se è richiesto un valore per la proprietà; il valore di questo parametro inizializza la variabile del membro `isRequired` dell'oggetto agent-property ("isRequired" a pagina 194).

*name* Specifica il nome della proprietà; il valore di questo parametro inizializza la variabile del membro `propName` dell'oggetto agent-property ("propName" a pagina 194).

*type* Specifica il tipo di proprietà; il valore di questo parametro inizializza la variabile del membro `type` dell'oggetto agent-property ("tipo" a pagina 195).

*validValues* Specifica i valori validi per la proprietà; il valore di questo parametro inizializza la variabile del membro `allValidValues` dell'oggetto agent-property object ("allValidValues" a pagina 191).

## Valori di ritorno

Un oggetto `AgentProperty` di cui è stata appena creata un'istanza.

## Eccezioni

**`IllegalArgumentException`**
Viene restituito se il valore del parametro *name* è `null` o se il parametro *type* non è una costante property-type valida (consultare Tabella 69 a pagina 189).

## Note

Il metodo `AgentProperty()` fornisce i seguenti moduli per creare l'istanza di un nuovo oggetto agent-property:

- Il primo modulo definisce un oggetto agent-property e lo inizializza con il nome di una proprietà *only*. Il valore predefinito del tipo di questa proprietà dell'agente è `String`. La proprietà è una proprietà single-cardinality che *non* visualizza valori multipli all'utente.

- Il secondo modulo definisce un nuovo oggetto agent-property e lo inizializza con le variabili del membro *all*. E' possibile personalizzare i metadati della proprietà specificando i valori appropriati per le variabili del membro.
- I terzo formato definisce un nuovo oggetto agent-property e lo inizializza con tutte le variabili del membro *tranne* `isHiddene isReadOnly`. In questo caso, le variabili `isHidden` e `isReadOnly` vengono impostate come valore predefinito su `false`.

## copy()

Copia la proprietà specificata nell'oggetto corrente AgentProperty.

### Sintassi

```
public void copy(AgentProperty prop);
```

### Parametri

*prop* Specifica il nome della proprietà da copiare.

# Capitolo 10. Classe BusObjAttr

L' Oggetto API ODK (Discovery Agent Development Kit) fornisce la classe `BusObjAttr` per rappresentare gli attributi nella definizione di un oggetto business. Un'istanza `BusObjAttr` rappresenta un *oggetto dell'attributo*. Questa classe definisce quanto segue:

- "Costanti dell'attributo"
- "Metodi"

**Nota:** Una definizione di oggetto business (`BusObjDef` oggetto) automaticamente definisce un oggetto dell'attributo per l' `attributo ObjectEventId`. Questo attributo viene automaticamente contrassegnato con la costante `BusObjAttr.OBJECT_EVENT_ID` per indicare il suo scopo particolare.

## Costanti dell'attributo

La classe `BusObjAttr` definisce le variabile statiche del membro per rappresentare le costanti dell'attributo. Tabella 72 riepiloga le costanti dell'attributo. Tutte le costanti dell'attributo sono di tipo intero (`int`).

*Tabella 72. Costanti dell'attributo della classe BusObjAttr.*

| Costante attributo | Descrizione |
|---|---|
| Costanti cardinalità | |
| `CARD_MULTIPLE` | Indica che l'attributo rappresenta un array di oggetti business secondari; cioè, l'attributo ha cardinalità multiple. |
| `CARD_SINGLE` | Indica che l'attributo rappresenta un valore o un oggetto business secondario; cioè, l'attributo ha cardinalità singole. |
| costante ObjectEventId | |
| `OBJECT_EVENT_ID` | Indica che l'attributo è ObjectEventId. |

## Metodi

Tabella 73 riepiloga i metodi del membro della classe `BusObjAttr`.

*Tabella 73. Metodi del membro della classe BusObjAttr*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `BusObjAttr()` | Crea un oggetto business-object-attribute. | 201 |
| `getAppText()` | Richiama informazioni specifiche sull'applicazione di un attributo. | 202 |
| `getAttrType()` | Richiama il tipo di un attributo semplice. | 202 |
| `getAttrTypeName()` | Richiama il tipo di un oggetto business secondario come tipo di attributo, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari. | 203 |

*Tabella 73. Metodi del membro della classe BusObjAttr (Continua)*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `getBOVersion()` | Richiama il numero della versione della definizione dell'oggetto business, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari. | 203 |
| `getCardinality()` | Richiama la cardinalità dell'attributo, per un attributo che rappresenta un oggetto business secondario o un array di un oggetto business secondario. | 203 |
| `getComments()` | Richiama commenti associati all'attributo. | 204 |
| `getDefault()` | Richiama il valore predefinito per un attributo. | 204 |
| `getMaxLength()` | Richiama la lunghezza massima per questo attributo. | 204 |
| `getName()` | Richiama il nome dell'attributo. | 205 |
| `getRelationType()` | Richiama il tipo di relazione dell'attributo, che è il contenimento per un attributo che rappresenta un oggetto business secondario o un array di un oggetto business secondario. | 205 |
| `isForeignKey()` | Determina se questo attributo è parte della chiave esterna dell'oggetto business. | 205 |
| `isKey()` | Determina se questo attributo è parte della chiave dell'oggetto business. | 206 |
| `isRequiredKey()` | Determina se questo attributo è parte della chiave richiesta dell'oggetto business. | 206 |
| `isRequiredServerBound()` | Determina se un attributo è richiesto quando l'oggetto business rappresenta un evento di triggering. | 206 |
| `isSimpleType()` | Determina se un attributo è di tipo semplice (come una Stringa, un Intero o Mobile) o se rappresenta un oggetto business secondario o un array di oggetti business secondari. | 207 |
| `setAppText()` | Imposta le informazioni di un attributo specifiche dell'applicazione. | 207 |
| `setAttrType()` | Imposta il tipo di attributo. | 207 |
| `setBOVersion()` | Imposta la versione dell'oggetto business secondario o dell'oggetto rappresentato da un attributo, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari. | 208 |
| `setCardinality()` | Imposta a cardinalità di un attributo, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari. | 208 |
| `setComments()` | Imposta i commenti associati all'attributo. | 209 |
| `setDefault()` | Imposta il valore predefinito per un attributo. | 209 |
| `setIsForeignKey()` | Imposta l'attributo su un valore booleano che indica se l'attributo è parte di una chiave esterna. | 210 |
| `setIsKey()` | Imposta l'attributo su un valore booleano che indica se l'attributo è parte di una chiave. | 210 |
| `setIsRequiredKey()` | Imposta l'attributo su un valore booleano che indica se l'attributo è parte di una chiave richiesta da un oggetto business. | 210 |
| `setMaxLength()` | Imposta la lunghezza massima per un attributo. | 210 |
| `setName()` | Imposta il nome dell'attributo. | 211 |

*Tabella 73. Metodi del membro della classe BusObjAttr (Continua)*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `setRelationType()` | Imposta il tipo di relazione di un attributo, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari. | 211 |

# BusObjAttr()

Crea un nuovo oggetto business-object-attribute.

## Sintassi

```
public BusObjAttr(String name, int type);

public BusObjAttr(String name, int type, String typeName);

public BusObjAttr(String name, int type,
   String typeName, boolean isKey, boolean isForeignKey,
   boolean isReqd, String appSpecInfo, int maxLen,
   String defaultValue, String BOversion,
   String cardinality, String relType,
   boolean isReqdServerBound, String comments);
```

## Parametri

*appSpecInfo* Specifica le informazioni specifiche dell'applicazione per l'attributo.

*BOversion* Specifica la versione dell'oggetto business secondario o degli oggetti, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari.

*cardinalità* Specifica la cardinalità di un attributo, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari.

*commenti* Specifica i commenti facoltativi da associare all'attributo.

*defaultValue* Specifica il valore predefinito per un attributo.

*isForeignKey* Specifica se l'attributo è parte della chiave esterna dell'oggetto business.

*isKey* Specifica se l'attributo è parte della chiave esterna dell'oggetto business.

*isReqd* Specifica se è richiesto un valore per l'attributo.

*isReqdServerBound*
Specifica se è richiesto un valore per l'attributo quando l'oggetto business rappresenta un evento di triggering.

*maxLen* Specifica la lunghezza massima del valore dell'attributo.

*name* Specifica il nome dell'attributo.

*relType* Specifica il tipo di relazione, per un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari.

*type* Specifica il tipo dell'attributo.

*typeName* Specifica il tipo di oggetto business come tipo di attributo, per un attributo che rappresenta un oggetto business secondaio o un array di oggetti business secondari.

### Valori di ritorno
L'oggetto `BusObjAttr` di cui è stata appena creata un'istanza.

## getAppText()

Richiama informazioni specifiche sull'applicazione di un attributo.

### Sintassi
```
public String getAppText();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `Stringa` che contiene informazioni specifiche sull'applicazione di un attributo.

### Vedere anche
setAppText()

## getAttrType()

Richiama il tipo di un attributo.

### Sintassi
```
public int getAttrType();
```

### Parametri
Nessuno.

### Valori di ritorno
Un intero che rappresenta il tipo di un attributo. Confrontare questo valore intero con quello delle costanti del tipo di attributo.

`BusObjAttrType.BOOLEAN`
: L'attributo ha un tipo di dati Booleano.

`BusObjAttrType.CIPHERTEXT`
: L'attributo ha il tipo di dati Cipher Text.

`BusObjAttrType.DATE`
: L'attributo ha il tipo di dati Date.

`BusObjAttrType.DOUBLE`
: L'attributo ha il tipo di dati Double.

`BusObjAttrType.FLOAT`
: L'attributo ha il tipo di dati Float.

`BusObjAttrType.INTEGER`
: L'attributo ha il tipo di dati Integer.

`BusObjAttrType.INVALID_TYPE`
: L'attributo ha un tipo di dati non valido.

`BusObjAttrType.LONGTEXT`
: L'attributo ha il tipo di dati Long Text.

`BusObjAttrType.OBJECT`
: L'attributo ha il tipo di dati Object (contiene un altro oggetto business).

`BusObjAttrType.STRING`
L'attributo ha il tipo di dati String.

### Vedere anche
getAttrTypeName(), setAttrType()

## getAttrTypeName()

Richiama il nome del tipo di dati dell'attributo .

### Sintassi
```
public String getAttrTypeName();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `stringa` che contiene il nome della definizione dell'oggetto business che è il tipo di oggetto business secondario (quando l'attributo contiene un oggetto business secondario).

### Note
Il metodo `getAttrTypeName()` richiama il nome del tipo di attributo per un oggetto business secondario. Quando un attributo rappresenta un oggetto business secondario (o un array di oggetti business secondari), il suo tipo di attributo è `BusObjAttrType.OBJECT` ed il nome del tipo di attributo è il nome della definizione dell'oggetto business per la definizione dell'oggetto business secondario.

### Vedere anche
getAttrType(), setAttrType()

## getBOVersion()

Richiama il numero della versione della definizione dell'oggetto business per un attributo che rappresenta un oggetto business secondario o array di oggetti business secondari.

### Sintassi
```
public String getBOVersion();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `Stringa` che contiene il numero della versione della definizione dell'oggetto business secondario rappresentato dall'attributo.

### Vedere anche
setBOVersion()

## getCardinality()

Richiama la cardinalità dell' attributo, per un attributo che rappresenta un oggetto business secondario o array di oggetti business secondari.

### Sintassi
```
public String getCardinality();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `Stringa` che contiene la cardinalità di un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari. Confrontare il valore di questa stringa con le seguenti costanti di cardinalità:

**`BusObjAttr.CARD_SINGLE`**
L'attributo ha una singola cardinalità.

**`BusObjAttr.CARD_MULTIPLE`**
L'attributo ha multiple cardinalità.

### Vedere anche
setCardinality()

## getComments()

Richiama i commenti associati all' attributo.

### Sintassi
```
public String getComments();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `Stringa` che contiene i commenti per un attributo.

## getDefault()

Richiama il valore predefinito per un attributo.

### Sintassi
```
public String getDefault();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `Stringa` che contiene il valore predefinito per un attributo.

### Vedere anche
setDefault()

## getMaxLength()

Richiama la lunghezza massima per questo attributo.

### Sintassi
```
public int getMaxLength();
```

### Parametri
Nessuno.

### Valori di ritorno
Un intero che rappresenta la lunghezza massima del valore di un attributo.

### Vedere anche
setMaxLength()

## getName()

Richiama il nome di un attributo.

### Sintassi
```
public String getName();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `Stringa` che contiene il nome di un attributo.

### Vedere anche
setName()

## getRelationType()

Richiama il tipo di relazione, che è il contenimento per un attributo che rappresenta un oggetto business secondario o array di oggetti business secondari.

### Sintassi
```
public String getRelationType();
```

### Parametri
Nessuno.

### Valori di ritorno
Una `Stringa` che contiene il tipo di relazione (`"contenimento"`) di un attributo che rappresenta un oggetto business secondario o un array di oggetti business secondari.

### Vedere anche
setRelationType()

## isForeignKey()

Determina se questo attributo è parte della chiave esterna dell'oggetto business.

### Sintassi
```
public boolean isForeignKey();
```

### Parametri
Nessuno.

### Valori di ritorno
Restituisce `true`, se l'attributo è una chiave esterna o parte della chiave esterna, altrimenti restituisce `false`.

### Vedere anche
setIsForeignKey()

## isKey()

Determina se questo attributo è parte della chiave primaria dell'oggetto busines.

### Sintassi
```
public boolean isKey();
```

### Parametri
Nessuno.

### Valori di ritorno
Restituisce un valore `true`, se l'attributo è una chiave o una parte della chiave; altrimenti restituisce `false`.

### Vedere anche
setIsKey()

## isRequiredKey()

Determina se questo attributo è parte della chiave richiesta dell'oggetto business.

### Sintassi
```
public boolean isRequiredKey();
```

### Parametri
Nessuno.

### Valori di ritorno
Restituisce `true`, se l'attributo è una chiave richiesta parte di una chiave, altrimenti restituisce `falso`.

### Vedere anche
setIsRequiredKey()

## isRequiredServerBound()

Determina se un attributo è richiesto quando l'oggetto business rappresenta un evento di triggering.

### Sintassi
```
public boolean isRequiredServerBound();
```

### Parametri
Nessuno.

### Valori di ritorno
Restituisce `true`, se l'attributo è richiesto quando l'oggetto business rappresenta una richiesta di collaborazione dell'oggetto; altrimenti restituisce `false`.

## isSimpleType()

Determina se un attributo è di tipo semplice (come Stringa, Intero o Mobile) o se rappresenta un oggetto business secondario o un array di oggetti business secondari.

### Sintassi

```
public boolean isSimpleType();
```

### Parametri

Nessuno.

### Valori di ritorno

Restituisce `true`, se l'attributo è di tipo semplice; altrimenti restituisce `false`.

### Vedere anche

getAttrType(), setAttrType()

## setAppText()

Imposta le informazioni specifiche dell'applicazione di un attributo.

### Sintassi

```
public void setAppText(String appInfo);
```

### Parametri

*appInfo* E' l'informazione specifica dell'applicazione da assegnare all'attributo.

### Valori di ritorno

Nessuno.

### Vedere anche

getAppText()

## setAttrType()

Imposta il tipo di attributo.

### Sintassi

```
public void setAttrType(int type);
public void setAttrType(int type, String typeName);
```

### Parametri

*type* E' il tipo di attributo, rappresentato come una delle costanti del tipo di attributo:

```
BusObjAttrType.BOOLEAN
BusObjAttrType.CIPHERTEXT
BusObjAttrType.DATE
BusObjAttrType.DOUBLE
BusObjAttrType.FLOAT
BusObjAttrType.INTEGER
BusObjAttrType.LONGTEXT
BusObjAttrType.OBJECT
BusObjAttrType.STRING
```

*typeName* E' il nome dell'oggetto business per un attributo che rappresenta

un oggetto business secondario o un array di oggetti business secondari; in questo caso, il tipo di attributo è lo stesso del tipo di oggetto business secondario e il `tipo` di valore è `OBJECT`.

### Valori di ritorno
Nessuno.

### Eccezioni
**`BusObjInvalidAttrException`**

Vene restituito se *type* non è valido; cioè, non è uno dei valori rappresentati dalle costanti del tipo di attributo.

### Note
Il metodo `setAttrType()` fornisce i seguenti moduli:

- Il primo modulo consente di impostare il tipo di attributo per un attributo semplice, specificato come una costante del tipo di attributo che è definita nella classe `BusObjAttrType`.
- Il secondo modulo consente di impostare il tipo di attributo per un oggetto business secondario o un array di oggetti business secondari. Questo modulo consente di specificare il tipo di attributo (come costante del tipo di attributo `BusObjAttrType.OBJECT`) e il nome della definizione di un oggetto business per un oggetto business secondario.

### Vedere anche
getAttrType(), getAttrTypeName()

Per informazioni correlate, consultare Capitolo 11, “Interfaccia BusObjAttrType”, a pagina 213 e Capitolo 24, “Classe ODKException”, a pagina 275.

## setBOVersion()

Imposta il numero della versione di definizione di oggetto business, per un attributo che rappresenta un oggetto business secondario o array di oggetti business secondari.

### Sintassi
```
public void setBOVersion(String version);
```

### Parametri
*`versione`* E’ la versione della definizione dell’oggetto business per l’oggetto business secondario o per gli oggetti che questo attributo rappresenta.

### Valori di ritorno
Nessuno.

### Vedere anche
getBOVersion()

## setCardinality()

Imposta la cardinalità dell’ attributo, per un attributo che rappresenta un oggetto business secondario o array di oggetti business secondari.

### Sintassi

```
public void setCardinality(String cardinality);
```

### Parametri

*cardinalità* E' la cardinalità da assegnare a questo attributo. La cardinalità è rappresentata da una delle seguenti costanti di cardinalità:

```
BusObjAttr.CARD_SINGLE
BusObjAttr.CARD_MULTIPLE
```

### Valori di ritorno

Nessuno.

### Eccezioni

**BusObjInvalidAttrException**

Viene restituito se la *cardinalità* non è valida; cioè se non contiene una costante di cardinalità valida.

### Vedere anche

getCardinality()

## setComments()

Imposta i commenti associati con un attributo.

### Sintassi

```
public void setComments(String comment);
```

### Parametri

*commento* E' la stringa del commento per fornire informazioni aggiuntive per l'attributo.

### Valori di ritorno

Nessuno.

### Vedere anche

getComments()

## setDefault()

Imposta il valore predefinito per un attributo.

### Sintassi

```
public void setDefault(String defaultValue);
```

### Parametri

*defaultValue* E' il valore predefinito da assegnare all'attributo.

### Valori di ritorno

Nessuno.

### Vedere anche

getDefault()

## setIsForeignKey()

Imposta la proprietà dell'attributo che indica se l' attributo è parte della chiave esterna.

### Sintassi

```
public void setIsForeignKey(boolean fKey);
```

### Parametri

*fKey* Indica se questo attributo è parte di una chiave esterna.

### Valori di ritorno

Nessuno.

### Vedere anche

isForeignKey()

## setIsKey()

Imposta la proprietà dell'attributo che indica se l' attributo è parte della chiave primaria.

### Sintassi

```
public void setIsKey(boolean key);
```

### Parametri

*key* Indica se questo attributo è parte di una chiave.

### Valori di ritorno

Nessuno.

### Vedere anche

isKey()

## setIsRequiredKey()

Imposta l'attributo su un valore booleano che indica se l'attributo è parte della chiave richiesta dell'oggetto business.

### Sintassi

```
public void setIsRequiredKey(boolean isReqd);
```

### Parametri

*isReqd* Indica se questo attributo è una chiave richiesta.

### Valori di ritorno

Nessuno.

### Vedere anche

isRequiredKey()

## setMaxLength()

Imposta la lunghezza massima per un attributo.

### Sintassi
```
public void setMaxLength(int maxLength);
```

### Parametri
*maxLength* E' la lunghezza massima da assegnare all'attributo.

### Valori di ritorno
Nessuno.

### Eccezioni
**BusObjInvalidAttrException**

Viene restituito se la lunghezza massima è `maxLength < 0` o `maxLength > 2^31-1`

### Vedere anche
getMaxLength()

## setName()

Imposta il nome di un attributo.

### Sintassi
```
public void setName(String name);
```

### Parametri
*name* E' il nome da assegnare all'attributo.

### Valori di ritorno
Nessuno.

### Vedere anche
getName()

## setRelationType()

Imposta il tipo di relazione di un attributo per il contenimento, per un attributo che rappresenta un oggetto business secondario o array di oggetti business secondari.

### Sintassi
```
public void setRelationType(String relType);
```

### Parametri
*relType* E' il tipo di relazione da assegnare a questo attributo.

### Valori di ritorno
Nessuno.

### Vedere anche
getRelationType()

# Capitolo 11. Interfaccia BusObjAttrType

L' Object Discovery Agent Development Kit (ODK) API fornisce la classe `BusObjAttrType` per rappresentare i tipi di dati validi per attributi nella definizione di un oggetto business. Qualsiasi classe che implementa l'interfaccia `BusObjAttrType` può accedere direttamente alle sue costanti definite. Per esempio, se la classe `ODKAgentBase2` implementa l'interfaccia `BusObjAttrType`, il suo metodo può accedere alla costante `BOOLEAN` nel modo seguente:

```
int bool_type = BOOLEAN;
```

La classe `BusObjAttrType` definisce quanto segue:

- "Costanti del tipo di attributo"
- "Variabile del membro statico"

## Costanti del tipo di attributo

La classe `BusObjAttrType` definisce le variabili statiche del membro per rappresentare le costanti dei tipi di attributo. Tabella 74 riepiloga le costanti di questi tipo di attributi. Tutte le costanti del tipo di proprietà sono di tipo interi (`int`).

*Tabella 74. Costanti del tipo di attributo della casse BusObjAttrType.*

| Costante del tipo di attributo | Descrizione |
|---|---|
| `BOOLEAN` | Rappresenta un tipo di attributo di Booleano. |
| `CIPHERTEXT` | Rappresenta il tipo di attributo di CipherText. |
| `DATE` | Rappresenta il tipo di attributo di Date. |
| `DOUBLE` | Rappresenta il tipo di attributo di Double. |
| `FLOAT` | Rappresenta un tipo di attributo di Float. |
| `INTEGER` | Rappresenta un tipo di attributo di Integer. |
| `INVALID_TYPE` | Rappresenta un tipo di attributo non valido. |
| `LONGTEXT` | Rappresenta un tipo di attributo di Long Text. |
| `OBJECT` | Rappresenta un tipo di attributo di Object. |
| `STRING` | Rappresenta il tipo di attributo di String. |

## Variabile del membro statico

In aggiunta alle costanti del tipo di attributo (che sono definite come variabili del membro statico), la classe `BusObjAttrType` definisce la variabile del membro statico in Tabella 75.

*Tabella 75. Variabile del membro statico della classe BusObjAttrType.*

| Variabili del membro statico | Descrizione |
|---|---|
| `AttrTypes` | Un array `stringa` che contiene i nomi per i diversi tipi di attributo. Questo array può essere indicizzato per tipo di attributo; ad esempio, il seguente codice richiama il nome tipo per il tipo di attributo Integer:<br>`BusObjAttrType.AttrTypes[BusObjAttrType.INTEGER]` |

# Capitolo 12. Classe BusObjDef

L' Object Discovery Agent Development Kit (ODK) API fornisce la `BusObjDef`classe per rappresentare una definizione di oggetto business generata da ODA (Object Discovery Agent). Tabella 76 riepiloga i metodi nella classe `BusObjDef`.

*Tabella 76. Metodi del membro della BusObjDef class.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `BusObjDef()` | Crea un oggetto business-object-definition. | 216 |
| `addDefaultVerbs()` | Aggiunge le istruzioni predefinite (Crea, Richiama, Aggiorna e Cancella) nell'elenco delle istruzioni supportate. | 216 |
| `getAppInfo()` | Richiama le informazioni specifiche dell'applicazione per la definizione di oggetto business. | 216 |
| `getAttrCount()` | Richiama il numero di attributi, compreso ObjectEventId, nell'elenco degli attributi della definizione dell'oggetto business. | 217 |
| `getAttribute()` | Richiama l'attributo dal nome o dalla posizione specificata nella definizione di oggetto business. | 217 |
| `getAttributeIndex()` | Richiama la posizione ordinale dell'attributo nella definizione di oggetto business, dato il suo nome dell'attributo. | 218 |
| `getAttributeList()` | Richiama un vettore che contiene l'elenco degli attributi nella definizione di oggetto business. | 218 |
| `getName()` | Richiama il nome della definizione di oggetto business. | 219 |
| `getVerb()` | Richiama l'oggetto dell'istruzione per il nome dell'istruzione specificata. | 219 |
| `getVerbCount()` | Richiama il numero di istruzioni nell'elenco delle istruzioni. | 220 |
| `getVerbList()` | Richiama un vettore che contiene l'elenco delle istruzioni nella definizione dell'oggetto business. | 220 |
| `getVersion()` | Richiama la versione della definizione dell'oggetto business. | 221 |
| `insertAttribute()` | Inserisce l'attributo specificato nell'elenco degli attributi dell'oggetto business. | 221 |
| `insertVerb()` | Inserisce l'istruzione specificata nell'elenco delle istruzioni dell'oggetto business. | 222 |
| `removeAttribute()` | Rimuove l'attributo nella posizione specificata nell'elenco degli attributi. | 222 |
| `removeVerb()` | Rimuove l'istruzione con il nome specificato nell'elenco delle istruzioni. | 223 |
| `setAppInfo()` | Imposta le informazioni specifiche dell'applicazione per la definizione di oggetto business. | 224 |
| `setAttributeList()` | Imposta l'elenco degli attributi per la definizione di un oggetto business. | 224 |
| `setVerbList()` | Imposta l'elenco delle istruzioni per la definizione di un oggetto business. | 225 |

## BusObjDef()

Crea un oggetto business-object-definition.

### Sintassi

```
public BusObjDef(String name);
public BusObjDef(String name, Vector attrList, String[] verbNames,
   String appSpecInfo);
public BusObjDef(String name, Vector attrList, Vector verbList,
   String appSpecInfo);
```

### Parametri

*appSpecInfo* Specifica le informazioni specifiche dell'applicazione di business-object-level.

*attrList* Specifica un vettore Java che contiene l'elenco degli attributi di una definizione di oggetto business.

*name* Specifica il nome della definizione di oggetto business.

*verbList* Specifica un array delle istruzioni di oggetto business.

*verbNames* Specifica un array di stringa dei nomi delle istruzioni dell'oggetto business.

### Valori di ritorno

Un oggetto `BusObjDef` di cui è stata appena creata un'istanza.

### Eccezioni

**BusObjInvalidDefException**
Definizione

**BusObjInvalidVerbException**
Definizione

## addDefaultVerbs()

Aggiunge il valore predefinito delle istruzioni (Crea, Richiama, Aggiorna e Cancella) all' elenco delle istruzioni della definizione di oggetto business.

### Sintassi

```
public void addDefaultVerbs();
```

### Parametri

Nessuno.

### Valori di ritorno

Nessuno.

## getAppInfo()

Richiama le informazioni specifiche dell'applicazione per definizione di oggetto business

### Sintassi

```
public String getAppInfo();
```

### Parametri

Nessuno.

### Valori di ritorno

Una `stringa` che contiene informazioni specifiche dell'applicazione di business-object-level.

### Vedere anche

setAppInfo()

---

## getAttrCount()

Richiama il numero di attributi nell'elenco degli attributi di definizione di oggetto business.

### Sintassi

```
public int getAttrCount();
```

### Parametri

Nessuno.

### Valori di ritorno

Il numero di attributi nella definizione di oggetto business (incluso l' ObjectEventId attribute).

---

## getAttribute()

Richiama l' l'attributo per il suo nome o attraverso la sua posizione specificata nell'elenco della definizione di oggetto business.

### Sintassi

```
public BusObjAttr getAttribute(String attrName);
public BusObjAttr getAttribute(int pos);
```

### Parametri

*attrName* E' il nome dell'attributo da richiamare dall'elenco dell'attributo della definizione di oggetto business.

*pos* E' un intero che specifica la posizione ordinale dell'attributo nell'elenco degli attributi della definizione di oggetto business.

### Valori di ritorno

L'oggetto dell'attributo (`BusObjAttr`) per l'attributo specificato nella definizione di oggetto business.

## Eccezioni

**BusObjNoSuchAttrException**
Restituito se l'attributo specificato non esiste o se la posizione all'interno dell'elenco dell'attributo non è valido.

## Note

il metodo `getAttribute()` richiama un attributo dall'elenco degli attributi della definizione di oggetto business. restituisce questo attributo come un oggetto dell'attributo (`BusObjAttr`). E' possibile usare metodi di classe `BusObjAttr` per ottenere un'informazione sull'attributo.

## Vedere anche

getAttributeIndex(), getAttributeList()

---

# getAttributeIndex()

Richiama la posizione ordinale di attributo nella definizione di oggetto business, dato il suo nome dell'attributo.

## Sintassi

```
public int getAttributeIndex(String attrName);
```

## Parametri

*attrName* E' il nome dell'attributo di cui viene richiamata la sua posizione ordinale.

## Valori di ritorno

La posizione intera dell'attributo nell'elenco degli attributi della definizione di oggetto business.

## Eccezioni

**BusObjNoSuchAttrException**
Restituito se l'attributo specificato non esiste nella definizione di oggetto business.

## Vedere anche

getAttribute()

---

# getAttributeList()

Richiama l'elenco degli attributi nella definizione di oggetto business.

## Sintassi

```
public Vector getAttributeList();
```

## Parametri

Nessuno.

## Valori di ritorno

Un oggetto `java.util.Vector` che contiene un oggetto (`BusObjAttr`) dell'attributo per ciascun attributo nella definizione di oggetto business.

## Note

Il metodo `getAttributeList()` restituisce l'elenco degli attributi della definizione di oggetto business come un array Java degli oggetti dell'attributo. Si possono usare metodi di classe `java.util.Vector` per richiamare gli oggetti dell'attributo da questo oggetto `array`. E' possibile usare metodi di classe `BusObjAttr` per ottenere informazioni dall'oggetto dell'attributo.

## Vedere anche

setAttributeList()

---

# getName()

Richiama il nome della definizione di oggetto business.

## Sintassi

```
public String getName();
```

## Parametri

Nessuno.

## Valori di ritorno

La `stringa` che contiene il nome della definizione di oggetto business.

---

# getVerb()

Richiama l' istruzione specificata dall' elenco istruzioni della definizione di oggetto business.

## Sintassi

```
public BusObjVerb getVerb(String verb);
```

## Parametri

*verb* E' il nome dell'istruzione da richiamare dall'elenco definizioni della definizione di oggetto business.

## Valori di ritorno

L'oggetto dell'istruzione (`BusObjVerb`) per l'istruzione specificata nell'elenco istruzioni della definizione di oggetto business.

## Eccezioni

**`BusObjNoSuchVerbException`**
Viene restituito se l'istruzione specificata non esiste.

### Note

Il metodo `getVerb()` richiama un'istruzione dall'elenco istruzioni della definizione di oggetto business. Restituisce questa istruzione come un oggetto dell'istruzione (`BusObjVerb`). E' possibile usare i metodi di classe `BusObjVerb` per ottenere informazioni sull'istruzione.

### Vedere anche

getVerbCount(), getVerbList()

---

## getVerbCount()

Richiama il numero di istruzioni nell' elenco istruzioni della definizione di oggetto business.

### Sintassi

```
public int getVerbCount();
```

### Parametri

Nessuno.

### Valori di ritorno

Il numero intero di istruzioni nell'elenco istruzioni della definizione di oggetto business.

### Vedere anche

getVerb()

---

## getVerbList()

Richiama l'elenco di istruzioni nella definizione di oggetto business.

### Sintassi

```
public Vector getVerbList();
```

### Parametri

Nessuno.

### Valori di ritorno

Un oggetto `java.util.Vector` che contiene un oggetto istruzione (`BusObjVerb`) per ciascuna istruzione supportata nella definizione di oggetto business.

### Note

Il metodo `getVerbList()` restituisce l'elenco delle istruzioni della definizione di oggetto business come un Vettore Java degli oggetti delle istruzioni. E' possibile usare metodi di classe `java.util.Vector` per richiamare gli oggetti delle istruzioni da questo oggetto `Istruzione`. E' possibile usare metodi di classe `BusObjVerb` per ottenere informazioni dall'oggetto dell'istruzione.

### Vedere anche

setVerbList()

## getVersion()

Richiama la versione della definizione dell'oggetto business.

### Sintassi

```
public String getVersion();
```

### Parametri

Nessuno.

### Valori di ritorno

LA `stringa` che contiene la versione della definizione di oggetto business.

## insertAttribute()

Inserisce l' attributo specificato in elenco della definizione di oggetto business.

### Sintassi

```
public void insertAttribute(BusObjAttr attrObj);
public void insertAttribute(BusObjAttr attrObj, int pos);
```

### Parametri

| | |
|---|---|
| *attrObj* | E' l'oggetto dell'attributo da aggiungere all'elenco dell'attributo della definizione di oggetto business. |
| *pos* | E' la posizione ordinale cui l'attributo deve essere aggiunto nell'elenco degli attributi. |

### Valori di ritorno

Nessuno.

### Eccezioni

**`BusObjInvalidAttrException`**
Viene restituito se l'attributo descritto da un oggetto dell'attributo non è valido.

### Note

Il metodo `insertAttribute()` fornisce i seguenti moduli:

- Il primo modulo specifica l'attributo da aggiungere per il nome del suo attributo. Quando si usa questo formato, `insertAttribute()` inserisce l'attributo specificato nella posizione immediatamente *sopra* l'attributo`ObjectEventId` nell'elenco dell'attributo dell'oggetto business.
- Il secondo modulo specifica l'attributo da aggiungere e la posizione ordinale all'interno dell'elenco dell'attributo a cui aggiungere tale attributo. Quando si specifica una posizione ordinale, `insertAttribute()` inserisce l'attributo

specificato nella posizione *pos* specificata nell'elenco dell'attributo della definizione di oggetto business e sposta ogni attributo che segue nell'elenco di una posizione verso il basso.

**Importante:** Se si specifica una posizione ordinale, accertarsi che la posizione specificata sia *sopra* l'attributo `ObjectEventId`.

### Vedere anche

removeAttribute()

## insertVerb()

Inserisce l' istruzione nell' elenco istruzioni della definizione di oggetto business.

### Sintassi

```
public void insertVerb(BusObjVerb verbObj);
public void insertVerb(String verbStrng, String appSpecInfo);
```

### Parametri

*appSpecInfo* E' l'informazione specifica dell'applicazione per l'istruzione da aggiungere all'elenco delle istruzioni.

*verbObj* E' l'oggetto dell'istruzione da aggiungere all'elenco delle istruzioni.

*verbStrng* E' il nome dell'istruzione da aggiungere all'elenco delle istruzioni.

### Eccezioni

**`BusObjInvalidVerbException`**
Viene restituita se l'istruzione che l'oggetto istruzione descrive è un duplicato.

### Note

Il metodo `insertVerb()` fornisce i seguenti moduli per inserire un oggetto istruzione nell'elenco delle istruzioni della definizione di oggetto business in uno dei seguenti modi:

- Il primo modulo specifica l'istruzione da aggiungere come un oggetto istruzione inizializzata (un'istanza `BusObjVerb`). E' possibile usare metodi di classe `BusObjVerb` per inizializzare l'oggetto istruzione.
- Il secondo modulo specifica le informazioni delle istruzioni, incluso il nome e le informazioni specifiche dell'applicazione per l'istruzione.

### Vedere anche

removeVerb()

## removeAttribute()

Rimuove l'attributo specificato dall' elenco della definizione di oggetto business.

### Sintassi

```
public BusObjAttr removeAttribute(int pos);
public BusObjAttr removeAttribute(String attrName);
```

## Parametri

*attrName* E' il nome dell'attributo da eliminare dall'elenco degli attributi della definizione di oggetto business.

*pos* E' la posizione ordinale da cui rimuovere l'attributo.

## Valori di ritorno

Un oggetto attributo (`BusObjAttr`) che contiene l'attributo rimosso.

## Eccezioni

**`BusObjNoSuchAttrException`**
Restituito se l'attributo specificato non esiste.

**`BusObjInvalidAttrException`**
Viene restituito se l'attributo da eliminare è uno che non può essere rimosso, come l'attributo `ObjectEventId`.

## Note

Il metodo `removeAttribute()` fornisce i seguenti moduli:

- Il primo modulo specifica l'attributo da rimuovere tramite la sua posizione ordinale all'interno dell'elenco attributi nella definizione di oggetto business.
- Il secondo modulo specifica l'attributo da rimuovere per il suo nome e posizione ordinale all'interno dell'elenco degli attributi cui aggiungere questo attributo.

**Importante:** Se si specifica una posizione ordinale, accertarsi che la posizione specificata *non* sia l'attributo `ObjectEventId`.

## Vedere anche

insertAttribute()

---

# removeVerb()

Rimuove l'istruzione specificata dall' elenco istruzioni della definizione di oggetto business.

## Sintassi

```
public BusObjVerb removeVerb(String verb);
```

## Parametri

*verb* E' il nome dell'istruzione il cui oggetto istruzione deve essere rimosso dall'elenco della definizione di oggetto business.

## Valori di ritorno

Un oggetto istruzione che contiene l'istruzione rimossa (`BusObjVerb`).

## Eccezioni

**`BusObjNoSuchVerbException`**
Viene restituito se l'istruzione specificata non esiste.

## Vedere anche

insertVerb()

---

# setAppInfo()

Imposta le informazioni specifiche dell'applicazione per definizione di oggetto business.

## Sintassi

```
public void setAppInfo(String appSpecInfo);
```

## Parametri

*appSpecInfo* E' l'informazione specifica dell'applicazione business-object-level.

## Valori di ritorno

Nessuno.

## Vedere anche

getAppInfo()

---

# setAttributeList()

Imposta l'elenco degli attributi per la definizione di un oggetto business.

## Sintassi

```
public void setAttributeList(Vector attrList);
```

## Parametri

*attrList* E' un ogetto `java.util.Vector` che contiene gli oggetti dell'attributo da memorizzare nell'elenco degli attributi della definizione di oggetto business.

## Eccezioni

**`BusObjInvalidAttrException`**
Viene restituito se un oggetto attributo in *attrList* contiene un attributo duplicato o nullo.

## Note

Il metodo `setAttributeList()` passa l'elenco degli attributi *attrList* come un Vettore Java di oggetti dell'attributo. E' possibile usare metodi di classe `BusObjAttr` per memorizzare informazioni nell'oggetto attributo. E' possibile usare metodi della classe `java.util.Vector` per memorizzare gli oggetti dell'attributo in questo oggetto `Vector`.

## Vedere anche

getAttributeList()

## setVerbList()

Imposta l'elenco di istruzioni per la definizione di oggetto business.

### Sintassi

```
public void setVerbList(Vector verbList);
```

### Parametri

*verbList* E' un oggetto `java.util.Vector` che contiene gli oggetti delle istruzioni da memorizzare nell'elenco delle istruzioni della definizione di oggetto business.

### Valori di ritorno

Nessuno.

### Eccezioni

**`BusObjInvalidVerbException`**
Viene restituito se un oggetto attributo in *verbList* contiene un'istruzione duplicata o nulla.

### Note

Il metodo `setVerbList()` passa l'elenco delle istruzioni *verbList* come un Vettore Java degli oggetti delle istruzioni. E' possibile usare metodi della classe `BusObjVerb` per memorizzare informazioni nell'oggetto dell'istruzione. E' possibile usare metodi di classe `java.util.Vector` per memorizzare oggetti di istruzioni nell'oggetto `Vector`.

### Vedere anche

getVerbList()

# Capitolo 13. classe BusObjVerb

L' Object Discovery Agent Development Kit (ODK) API fornisce la classe `BusObjVerb` per rappresentare le istruzioni in una definizione di oggetto business. Un'istanza `BusObjVerb` rappresenta un *oggetto dell'istruzione*. Tabella 77 riepiloga i metodi di classe `BusObjVerb`.

*Tabella 77. Metodi del membro della classe BusObjVerb.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `BusObjVerb()` | Crea oggetti business-object-verb. | 227 |
| `clone()` | Clona un oggetto dell'istruzione. | 227 |
| `getAppInfo()` | Richiama le informazioni dell'istruzione specifiche dell'applicazione. | 228 |
| `getName()` | Richiama il nome dell'istruzione. | 228 |
| `setAppInfo()` | Richiama le informazioni dell'istruzione specifiche dell'applicazione. | 228 |
| `setName()` | Richiama il nome dell'istruzione. | 229 |

## BusObjVerb()

Crea un oggetto business-object-verb.

### Sintassi

```
public BusObjVerb(String verb, String appSpecInfo);
```

### Parametri

*appSpecInfo* Specifica le informazioni dell'istruzione specifiche dell'applicazione.

*verb* Specifica un'istruzione che è supportata dalla definizione di oggetto business.

### Valori di ritorno

L'oggetto `BusObjVerb` di cui è stata appena creata una nuova istanza.

### Eccezioni

**`BusObjInvalidVerbException`**
Viene restituito se l'istruzione specificata non è valida.

## clone()

Clona un oggetto dell'istruzione.

### Sintassi

```
public Object clone();
```

### Parametri

Nessuno.

### Valori di ritorno

Nessuno.

### Note

Questo metodo `clone()` sovrappone il metodo `clone()` nella classe `java.lang.Object`.

## getAppInfo()

Richiama le informazioni specifiche dell'applicazione dell'istruzione.

### Sintassi

```
public String getAppInfo();
```

### Parametri

Nessuno.

### Valori di ritorno

Una `Stringa` che contiene informazioni dell'istruzione specifiche dell'applicazione.
verb

### Vedere anche

setAppInfo()

## getName()

Richiama il nome della istruzione.

### Sintassi

```
public String getName();
```

### Parametri

Nessuno.

### Valori di ritorno

Una `Stringa` che contiene il nome dell'istruzione.

### Vedere anche

setName()

## setAppInfo()

Imposta le informazioni specifiche dell'applicazione dell'istruzione.

## Sintassi

```
public void setAppInfo(String appSpecInfo);
```

## Parametri

**appSpecInfo**
E' l'informazione verb-level specifica dell'applicazione per memorizzare l'oggetto dell'istruzione.

## Valori di ritorno

Nessuno.

## Vedere anche

getAppInfo()

---

# setName()

Imposta il nome della istruzione.

## Sintassi

```
public void setName(String verb);
```

## Parametri

**verb** E' il nome dell'istruzione da memorizzare nell'oggetto dell'istruzione.

## Valori di ritorno

Nessuno.

## Eccezioni

**BusObjInvalidVerbException**
Viene restituito se l'istruzione specificata non è valida.

## Vedere anche

getName()

# Capitolo 14. Classe CompleteCondition

L' Object Discovery Agent Development Kit (ODK) API fornisce la classe `CompleteCondition` per rappresentare una condizione sul valore di una proprietà dell'agente (rappresentata da un oggetto `AgentProperty`). Una condizione consiste di due tipi di sotto-condizioni, condizioni di immissione e dipendenti. Una proprietà agente memorizza tutte le sue condizioni nelle sue variabili del membro `allDependencies`.

**Nota:** Per informazioni sulle condizioni di immissione, consultare Capitolo 21, "Classe InputCondition", a pagina 261. Per informazioni sulle condizioni dipendenti, consultare Capitolo 18, "Classe DependentCondition", a pagina 247.

La classe `CompleteCondition` definisce quanto segue:



## costanti di Operator

La classe The `CompleteCondition` definisce le variabili statiche del membro da rappresentare. costanti di operator. Tabella 78 riepiloga quelle costanti di operator, che rappresentano operatori validi da usare in certe condizioni. Tutte le costanti di operator sono di tipo `stringa`.

*Tabella 78. Costanti operator della classe CompleteCondition.*

| Costante Operator | Descrizione |
|---|---|
| `OP_EQUAL` | Contiene una `Stringa` che rappresenta l'operatore Equals (=). |
| `OP_EXISTS` | Contiene una `Stringa` che rappresenta l'operatore Exists. |
| `OP_GREATER_THAN` | Contiene una `Stringa` che rappresenta l'operatore Greater Than (>). |
| `OP_GREATER_THAN_EQUAL` | Contiene una `Stringa` che rappresenta l'operatore Maggiore di oppure Uguale a (>=). |
| `OP_LESS_THAN` | Contiene una `Stringa` che rappresenta l'operatore Minore di (<). |
| `OP_LESS_THAN_EQUAL` | Contiene una `Stringa` che rappresenta l'operatore Minore d oppure Uguale a (<=). |
| `OP_NOT_EQUAL` | Contiene una `Stringa` che rappresenta l'operatore Non uguale (!=). |

## Variabili dei membri

Tabella 79 riepiloga le variabili del membro nella classe `CompleteCondition`.

*Tabella 79. Variabili del membro della classe CompleteCondition.*

| variabile del membro | Descrizione | Pagina |
|---|---|---|
| `allDependentConditions` | Specifica tutte le condizioni dipendenti per la proprietà. | 232 |
| `allInputConditions` | Specifica tutte le condizioni di immissione per la proprietà. | 232 |

### allDependentConditions

Specifica un array di tutte le condizioni dipendenti nella corrente condizione completa.

#### Tipo

```
public DependentCondition[] allDependentConditions
```

#### Note

La variabile del membro `allDependentConditions` contiene un elenco delle condizioni dipendenti nell'array dependent-condition, che è un array di oggetti `DependentCondition`. Ciascun oggetto `DependentCondition` contiene una condizione dipendente, che limita il valore della proprietà dipendente quando condizioni di immissione associate sono valutate su true. Per ulteriori informazioni, consultare "Impostazione delle condizioni sul valore della proprietà" a pagina 158.

### allInputConditions

Specifica un array di tutte le condizioni di immissione nella condizione completa corrente.

#### Tipo

```
public InputCondition[] allInputConditions
```

#### Note

La variabile del membro `allInputConditions` contiene un elenco delle condizioni del vettore input-condition, che è un array di oggetti `InputCondition`. Ciascun oggetto `InputCondition` contiene una condizione di immissione, che specifica un confronto da fare sul valore corrente della proprietà dell'agente. Per ulteriori informazioni, consultare "Impostazione delle condizioni sul valore della proprietà" a pagina 158.

## Metodi

Tabella 80 riepiloga i metodi nella classe `CompleteCondition`.

*Tabella 80. Metodi del membro della classe CompleteCondition.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `CompleteCondition()` | Crea un oggetto complete-condition. | 233 |
| `copy()` | Copia la condizione completa corrente nell'oggetto specificato complete-condition. | 233 |

## CompleteCondition()

Crea un oggetto complete-condition.

### Sintassi

```
public CompleteCondition();
public CompleteCondition(InputCondition[] allInputConds,
   DependentCondition[] allDepConds);
```

### Parametri

*allDepConds* Specifica un array di condizioni dipendenti; il valore di questo parametro inizializza la variabile del membro `allDependentConditions` (“allDependentConditions” a pagina 232).

*allInputConds* Specifica un array di condizioni di immissione; il valore di questi parametri inizializza la variabile del membro `allInputConditions` (“allInputConditions” a pagina 232).

### Valori di ritorno

Un oggetto `CompleteCondition` di cui è stata appena creata un’istanza.

## copy()

Copia la condizione completa corrente in un oggetto complete-condition specificato.

### Sintassi

```
public void copy(CompleteCondition completeCond);
```

### Parametri

*completeCond* Specifica il nome dell’oggetto complete-condition in cui viene copiata la condizione completa corrente.

### Valori di ritorno

Nessuno.

# Capitolo 15. Classe ContentMetaData

Il Object Discovery Agent Development Kit (ODK) API fornisce la `ContentMetaData` classe per contenere i metadati per il contenuto generato di Object Discovery Agent (ODA). Le variabili dei membri di questa classe rappresentano i metadati ODA. Quando l'ODA genera il suo contenuto, deve restituire un oggetto metadati del contenuto per descrivere il contenuto generato. Il metodo che restituisce i metadati del contenuto dipende dal protocollo del contenuto supportato dall'ODA nel seguente modo:

- Se l'ODA supporta un protocollo su richiesta per un determinato tipo di contenuto (definizioni dell'oggetto business o file), il metodo di generazione contenuto appropriato restituisce i metadati del contenuto al Procedura guidata dell'oggetto Business.
- Se l'ODA supporta un protocollo callback (solo per contenuto file), un metodo definito dall'utente restituisce il i metadati del contenuto al Procedura guidata dell'oggetto Business tramite il metodo `ODKUtility.contentComplete()`.

**Nota:** Per ulteriori informazioni, consultare "Aggiunta di contenuto generato" a pagina 102.

Il Business Object Designer Express utilizza l'oggetto metadati del contenuto per ottenere informazioni sul contenuto generato per ciascun tipo di contenuto supportato dall'ODA. Per determinare i protocolli di generazione supportati, il Business Object Designer Express chiama il metodo ODA `getContentProtocol()` (dalla propria classe `IGeneratesContent`).

La classe `ContentMetaData` definisce quanto segue:

- "Variabili dei membri"
- "Metodi" a pagina 236

## Variabili dei membri

Tabella 81 riepiloga le variabili dei membri della classe `ContentMetaData`.

*Tabella 81. Variabili dei membri della classe ContentMetaData.*

| variabile del membro | Descrizione | Pagina |
|---|---|---|
| `contentType` | Indica il tipo di contenuto del contenuto generato. | 235 |
| `conteggio` | Definisce il numero totale di elementi del contenuto nel contenuto richiesto. | 236 |
| `lunghezza` | Definisce la lunghezza totale, in byte, del contenuto richiesto. | 236 |

### contentType

Indica il tipo di contenuto del contenuto generato.

#### Tipo

```
public ContentType contentType
```

### Note

La variabile del membro `contentType` è un oggetto `ContentType` che indica il tipo di contenuto del contenuto generato descritto da questo metadato del contenuto. Deve essere impostato sul tipo di contenuto appropriato del contenuto generato, come mostra Tabella 82.

*Tabella 82. Valori del tipo di contenuto*

| Tipo di contenuto | Valore della variabile del membro contentType |
|---|---|
| Definizioni oggetto business | `ContentType.BusinessObject` |
| File Binari | `ContentType.BinaryFile` |

Ad esempio, quando un'ODA completa la generazione del contenuto, deve restituire un oggetto metadato del contenuto il cui membro `contentType` corrisponda al tipo di contenuto generato.

## conteggio

Definisce il numero totale di elementi del contenuto nel contenuto richiesto. Il valore deve essere maggiore di zero (0).

### Tipo

```
public long count
```

## lunghezza

Definisce la dimensione totale del contenuto richiesto, in byte. Se la lunghezza del contenuto non è nota, assegnare un lunghezza pari a zero (0).

**Importante**

Il Procedura guidata dell'oggetto Business attualmente non utilizza la variabile del membro `length`. Questa variabile viene quindi inizializzata con un valore "nullo", come zero (0) o -1.

### Tipo

```
public long length
```

# Metodi

Tabella 83 riepiloga i metodi della classe `ContentMetaData`.

*Tabella 83. Metodi del membro della classe ContentMetaData*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `ContentMetaData()` | Crea un oggetto metadati del contenuto. | 237 |
| `badContent()` | Restituisce un oggetto metadati del contenuto che indica che l'ODA non è in grado di generare il tipo di contenuto specificato. | 237 |
| `contentNotReady()` | Restituisce un oggetto metadati del contenuto che indica che l'ODA non ha ancora terminato la generazione del contenuto. | 237 |

*Tabella 83. Metodi del membro della classe ContentMetaData (Continua)*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `contentUnavailable()` | Restituisce un oggetto metadati del contenuto che indica che l'ODA non sta generando il contenuto specificato, sebbene implementi l'interfaccia corrispondente. | 238 |

# ContentMetaData()

Crea un oggetto metadati del contenuto.

## Sintassi

```
public ContentMetaData(ContentType contentType, long length, long count);
```

## Parametri

*contentType* È un oggetto `ContentType` che indica il tipo di contenuto del contenuto generato descritto dall'oggetto metadato del contenuto. Il valore di questo parametro inizializza la variabile del membro `contentType` dell'oggetto metadato del contenuto ("contentType" a pagina 235).

*count* Definisce il numero di elementi del contenuto nel contenuto richiesto; il valore di questo parametro inizializza la variabile del membro `count` nell'oggetto del metadato del contenuto ("conteggio" a pagina 236).

*length* Definisce la dimensione totale del contenuto richiesto, in byte. Il valore di questo parametro inizializza la variabile del membro `length` nell'oggetto del metadato del contenuto ("lunghezza" a pagina 236). Il Procedura guidata dell'oggetto Business attualmente *non* utilizza la variabile del membro length.

## Valori di ritorno

Un oggetto `ContentMetaData` di cui è stata appena creata un'istanza.

# badContent()

Informa il Procedura guidata dell'oggetto Business che il contenuto generato dall'ODA non è completo o ha qualche errore.

## Sintassi

```
public static ContentMetaData badContent(ContentType contentType);
```

## Parametri

*contentType* È l'oggetto `ContentType` che identifica il tipo di contenuto del contenuto generato in modo errato.

## Valori di ritorno

Un oggetto `ContentMetaData` che descrive il contenuto generato con esito negativo.

# contentNotReady()

Informa il Procedura guidata dell'oggetto Business che l'ODA non ha ancora terminato la generazione del contenuto specificato.

### Sintassi

```
public static ContentMetaData contentNotReady(ContentType contentType);
```

### Parametri

*contentType* È l'oggetto `ContentType` che identifica il tipo di contenuto del contenuto generato in modo incompleto.

### Valori di ritorno

Un oggetto `ContentMetaData` che descrive il contenuto generato con modo incompleto.

## contentUnavailable()

Informa il Procedura guidata dell'oggetto Business che l'ODA non supporta la generazione del contenuto specificato, sebbene implementi l'interfaccia corrispondente.

### Sintassi

```
public static ContentMetaData contentUnavailable(ContentType contentType);
```

### Parametri

*contentType* È l'oggetto `ContentType` che identifica il tipo di contenuto del contenuto generato non disponibile.

### Valori di ritorno

Un oggetto `ContentMetaData` che descrive il contenuto generato non disponibile.

### Note

Il metodo `contentUnavailable()` indica che l'ODA non genera contenuti del tipo di contenuto *contentType* content type. Ad esempio, se un'ODA supporta *solo* solo un protocollo di contenuto callback per un determinato tipo di contenuto, il Procedura guidata dell'oggetto Business non chiamerà mai il metodo di generazione di contenuto (`generateBoDefs()` per contenuto definizione dell'oggetto business o `generateBinFiles()` per contenuto file binario). Quindi, il metodo di generazione del contenuto può chiamare `contentUnavailable()` come proprio valore da restituire al Procedura guidata dell'oggetto Business.

# Capitolo 16. Classe ContentType

Il Object Discovery Agent Development Kit (ODK) API fornisce la classe `ContentType` per rappresentare i tipi di dati validi che Object Discovery Agent (ODA) può generare. La classe `ContentType` definisce quanto segue:

- "Variabili dei membri"
- "Metodi" a pagina 240

## Variabili dei membri

Tabella 84 riepiloga le variabili dei membri della classe `ContentType`.

*Tabella 84. Variabili dei membri della classe ContentType.*

| variabile del membro | Descrizione | Pagina |
|---|---|---|
| `BinaryFile` | Indica che l'ODA genera file binari come suo contenuto. | 239 |
| `Oggetto business` | Indica che l'ODA genera definizioni di oggetti business come suo contenuto. | 239 |

### BinaryFile

Indica che l'ODA genera dei file binari come contenuto.

#### Tipo

```
public static final ContentType BinaryFile
```

#### Note

La variabile del membro `contentType` indica che l'ODA supporta la generazione di file binari come contenuto. Pertanto, l'ODA implementa l'interfaccia `IGeneratesBinFiles`. I contenuti del file possono essere generati utilizzando uno dei protocolli di contenuto:

- Il protocollo di contenuto su richiesta richiede che l'ODA implementi il metodo `generateBinFiles()` per la gestione della generazione dei file.
- Il protocollo di contenuto callback richiede che l'ODA implementi un metodo definito dall'utente per la gestione della generazione dei file.

Per ulteriori informazioni, consultare "Creazione dei file binari come contenuto" a pagina 144.

### Oggetto business

Indica che l'ODA genera delle definizioni degli oggetti business come contenuto.

#### Tipo

```
public static final ContentType BusinessObject
```

#### Note

La variabile del membro `contentType` indica che l'ODA supporta la generazione di definizioni di oggetti di business come contenuto. Pertanto, l'ODA implementa l'interfaccia `IGeneratesBoDefs`. Il contenuto della definizione dell'oggetto business

deve essere generato utilizzando il protocollo di contenuto su richiesta, che richiede che l'ODA implementi il metodo `generateBoDefs()` per gestire la generazione della definizione dell'oggetto business. Per ulteriori informazioni, consultare "Creazione di definizioni di oggetti business come contenuto" a pagina 121.

## Metodi

Tabella 85 riepiloga i metodi della classe `ContentType`.

*Tabella 85. Metodi del membro della classe ContentType*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `ContentType()` | Crea un oggetto tipo di contenuto. | 240 |
| `equals()` | Confronta due oggetti tipo di contenuto. | 240 |
| `from_int()` | Genera un oggetto tipo contenuto per un valore ordinale specificato. | 241 |
| `toString()` | Restituisce una rappresentazione letterale dell'oggetto tipo di contenuto corrente. | 241 |
| `value()` | Restituisce un valore ordinale per il tipo di contenuto corrente. | 241 |
| `xmlObject()` | Genera un oggetto XML che rappresenta l'oggetto tipo di contenuto corrente. | 241 |

## ContentType()

Crea un oggetto tipo di contenuto.

### Sintassi

```
public ContentType(int contTypeOrdValue);
```

### Parametri

*contTypeOrdValue*
È il valore ordinale che rappresenta il tipo di contenuto.

### Valori di ritorno

Un oggetto `ContentType` di cui è stata appena creata un'istanza.

## equals()

Confronta due oggetti tipo di contenuto.

### Sintassi

```
public boolean equals(Object contentTypeObj);
```

### Parametri

*contentTypeObj*
È una referenza all'oggetto `ContentType` da confrontare con l'oggetto corrente `ContentType`.

### Valori di ritorno

Un valore `booleano` che indica se due oggetti tipo di contenuto sono uguali.

### Note

Il metodo `equals()` sovrascrive il metodo `equals()` nella classe `java.lang.Object`.

## from_int()

Genera un oggetto tipo contenuto per il valore ordinale specificato.

### Sintassi

```
public static ContentMetaData from_int(int contTypeOrdValue);
```

### Parametri

*contTypeOrdValue*
È il valore ordinale che rappresenta il tipo di contenuto corrente.

### Valori di ritorno

Un oggetto `ContentMetaData` che rappresenta il tipo di contenuto del valore orinale specificato.

### Vedere anche

value()

## toString()

Restituisce una rappresentazione letterale dell'oggetto tipo di contenuto corrente.

### Sintassi

```
public String toString();
```

### Parametri

Nessuno.

### Valori di ritorno

Un oggetto stringa `String` che contiene la rappresentazione letterale dell'oggetto tipo di contenuto corrente.

### Note

Il metodo `toString()` sovrascrive il metodo `toString()` nella classe `java.lang.Object`.

## value()

Restituisce un valore ordinale per il tipo di contenuto corrente.

### Sintassi

```
public int value();
```

### Parametri

Nessuno.

### Valori di ritorno

È il valore ordinale intero che rappresenta il tipo di contenuto corrente.

### Vedere anche

from_int()

## xmlObject()

Genera un oggetto XML che rappresenta l'oggetto tipo di contenuto corrente.

### Sintassi

```
public XMLObject xmlObject();
```

### Parametri
Nessuno.

### Valori di ritorno
Un oggetto `com.crossworlds.ODK.XMLObject` che rappresenta l'oggetto tipo di contenuto corrente.

# Capitolo 17. Classe CxBiDiEngine

La classe CxBidiEngine fornisce metodi per il trasferimento degli oggetti business e stringhe da un formato bidirezionale all'altro.

Tabella 86 riepiloga i metodi nella classe CxBidiEngine.

*Tabella 86. CwBidiEngine riepilogo metodo*

| Metodo | Descrizione | Pagina |
|---|---|---|
| `BiDiBOTransformation()` | Trasforma gli oggetti business di tipo BusinessObject da un formato bidirezionale all'altro. | 243 |
| `BiDiBusObjTransformation()` | Trasforma gli oggetti business di tipo BusObj da un formato bidirezionale ad un altro formato. | 244 |
| `BiDiStringTransformation()` | Trasforma le stringhe da un formato bidirezionale all'altro. | 245 |

## BiDiBOTransformation()

Il metodo BiDiTransformation() trasforma gli oggetti business di tipo BusinessObject da un formato bidirezionale in un altro formato. Usare questo metodo quando si sviluppano controllori, connettori e mappe.

### Sintassi

```
BusinessObject BiDiBOTransformation(BusinessObject boIn, String formatIn,
   String formatOut, boolean replace)
```

### Parametri

*boIn* L'oggetto business da trasformare. L'oggetto deve essere di tipo BusinessObject.

*formatIn* Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di immissione. Consultare Tabella 87 a pagina 244 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

*formatOut* Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare Tabella 87 a pagina 244 per i valori validi della stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

*replace* Un valore che indica se l'oggetto business di immissione deve essere sostituito. Il valore valido è `true` o `false`.

### Valori di ritorno

Il valore di ritorno è un oggetto business trasformato. Se il metodo non riesce, restituisce un valore nullo.

### Eccezioni

Nessuno.

## Esempi

Consultare l'esempio in "BiDiStringTransformation()" a pagina 245.

# BiDiBusObjTransformation()

Il metodo BiDiBusObjTransformation() trasforma gli oggetti business di tipo BusObj da un formato bidirezionale all'altro. Usare questo metodo all'interno della collaborazione.

## Sintassi

```
BusObj BiDiBusObjTransformation(BusObj busObjIn, String formatIn,
   String formatOut, boolean replace)
```

## Parametri

*busObjIn* L'oggetto business da trasformare. L'oggetto deve essere di tipo BusObj.

*formatIn* Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di immissione. Consultare Tabella 87 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

*formatOut* Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare Tabella 87 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

*replace* Un valore che indica se l'oggetto business di immissione deve essere sostituito. Il valore valido è `true` o `false`.

*Tabella 87. Valori per le stringhe di formato*

| Posizione lettera | Scopo | Valori | Descrizione | Predefinito |
|---|---|---|---|---|
| 1 | Tipo | I | Implicito (Logico) | I |
| | | V | Visivo | |
| 2 | Direzione | S | Sinistra a destra | S |
| | | R | Destra a sinistra | |
| 3 | Inversione simmetrica | Y | Inversione simmetrica attiva | Y |
| | | N | Inversione simmetrica non attiva | |
| 4 | Struttura | Y | Il testo è strutturato | N |
| | | N | Testo non strutturato | |
| 5 | Struttura numerica | H | Hindi | N |
| | | C | Contestuale | |
| | | N | Nominale | |

## Valori di ritorno

Il valore di ritorno è un oggetto business trasformato. Se il metodo non riesce, restituisce un valore nullo.

## Eccezioni

Nessuno.

## Esempi

Questo esempio trasforma InputBOBusObj dal formato bidirezionale standard di Windows al formato bidirezionale visivo.

```
BusObj dummyBusObj = null;
dummyBusObj = CwBidiEngine.BiDiBusObjTransformation(
                                        InputBOBusObj,
                                    "ILYNN",
                                    "VLYNN",true);
```

# BiDiStringTransformation()

Il metodo BiDiStringTransformation() trasforma le stringhe da un formato bidirezionale all'altro.

## Sintassi

```
BiDiStringTransformation(String strIn, String formatIn, String formatOut
```

## Parametri

*strIn* La stringa da trasformare.

*formatIn* Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di immissione. Consultare Tabella 88 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

*formatOut* Una stringa che rappresenta il formato bidirezionale del contenuto dell'oggetto business di emissione. Consultare Tabella 88 per i valori validi di questa stringa. Se questo parametro è nullo, il metodo predefinito si imposta sul formato standard bidirezionale di Windows.

*Tabella 88. Valori per le stringhe di formato*

| Posizione lettera | Scopo | Valori | Descrizione | Predefinito |
|---|---|---|---|---|
| 1 | Tipo | I | Implicito (Logico) | I |
| | | V | Visivo | |
| 2 | Direzione | S | Sinistra a destra | S |
| | | R | Destra a sinistra | |
| 3 | Inversione simmetrica | Y | Inversione simmetrica attiva | Y |
| | | N | Inversione simmetrica non attiva | |
| 4 | Struttura | Y | Il testo è strutturato | N |
| | | N | Testo non strutturato | |
| 5 | Struttura numerica | H | Hindi | N |
| | | C | Contestuale | |
| | | N | Nominale | |

## Valori di ritorno

Il valore di ritorno è un oggetto stringa trasformato.

## Eccezioni

Nessuno.

## Esempi

Il seguente esempio applica il metodo BiDiStringTransformation() ai valori dell'attributo di un oggetto business.

```
for (int i = 0; i < bo.getAttrCount();i++) {
 intAttrType = bo.getAttributeType(i);
 Object attrValue = bo.getAttrValue(i);
 String attrName = bo.getAttrName(i);

 if (attrValue != null {
    // Si gestisce solo l'attributo Stringa o Testo lungo e non
    // l'attributo ObjectEventId
    if (((attrType == CxObjectAttrType.STRING)
           || (attrType == CxObjectAttrType.LONGTEXT))
           && (!(attrName.equals(OBJECT_EVENT_ID)))) {
             String strOut = BidiStringTransformation(attrValue.toString(),
             bo.setAttrValue(i, strOut);
    } else if (attrType == CxObjectAttrType.OBJECT) {
        CxObjectAttr attrDesc = bo.getAttrDesc(i);
        if (attrDesc.getCardinality().equals(CxObjectAttr.CARD_Single)) {
           BiDiTransformation((BusinessObject) attrValue, "ILYNN",
                                                          "VLYNN",
                                                          true);
       } else {
           // multiple cardinality
           CxObjectContainer cont = (CxObjectContainer) attrValue;
           int objCount = cont.getObjectCount();
           for (int j = 0; j < objCount; j++) {
               BiDiBOTransformation((BusinessObject) (cont.getObject(j)),
                                       "ILYNN",
                                       "VLYNN",
                                       true);
       }
    }
 }
```

# Capitolo 18. Classe DependentCondition

Il Object Discovery Agent Development Kit (ODK) API fornisce la classe `DependentCondition` per rappresentare una condizione sul valore di una . Quando la condizione di input associata è true, la condizione di dipendenza viene applicata alla proprietà dipendente. Le condizioni di dipendenza e la loro condizione di input associata (o condizioni) sono archiviate in un oggetto condizione completa (`CompleteCondition`).

**Nota:** Per informazioni sulle condizioni complete, consultare Capitolo 14, "Classe CompleteCondition", a pagina 231.

La classe `DependentCondition` definisce quanto segue:



## Variabili dei membri

Tabella 89 riepiloga le variabili del membro nella classe `DependentCondition`.

*Tabella 89. Variabili del membro della classe DependentCondition.*

| variabile del membro | Descrizione | Pagina |
|---|---|---|
| `isDynamic` | Specifica se il Procedura guidata dell'oggetto Business deve controllare il valore della proprietà valore specifica prima di effettuare il confronto della condizione di dipendenza | 247 |
| `operatorType` | Definisce il tipo di operatore per la condizione di dipendenza. | 248 |
| `propertyName` | Definisce il nome della proprietà dipendente da visualizzare. | 248 |
| `specificValue` | Definisce il valore da confrontare con il valore della proprietà dipendente. | 248 |
| `typeOfSpecificValue` | Definisce il tipo di dati del valore specifico della condizione di dipendenza. | 249 |

### isDynamic

Specifica se il Procedura guidata dell'oggetto Business deve controllare il valore della proprietà valore specifica prima di effettuare il confronto della condizione di dipendenza.

#### Tipo

```
public boolean isDynamic
```

#### Note

Quando la variabile membro `isDynamic` è `true`, il Procedura guidata dell'oggetto Business ottiene il valore della proprietà che il valore membro `specificValue` specifica *prima* di eseguire il confronto con il valore della proprietà dipendente. Se `specificValue` contiene una costante, `isDynamic` deve essere impostato come `false`.

## operatorType

Definisce il tipo di operatore per la condizione di dipendenza.

### Tipo

```
public String operatorType
```

### Note

La variabile di membro `operatorType` definisce il tipo di confronto che il Procedura guidata dell'oggetto Business effettua tra il valore della proprietà dipendente (che viene specificata dalla variabile membro `propertyName`) e `specificValue`. I valori validi per la variabile `operatorType` sono le costanti dell'operatore definite nella classe `CompleteCondition`. Per ulteriori informazioni, consultare Tabella 78 a pagina 231.

## propertyName

Specifica il nome della proprietà dipendente.

### Tipo

```
public String propertyName
```

### Note

La variabile membro `propertyName` contiene il nome della proprietà dipendente. È il valore della proprietà dipendente che è ristretto dalla condizione di dipendenza (quando la condizione di input associata è true).

## specificValue

Definisce il valore da confrontare con il valore della proprietà dipendente.

### Tipo

```
public String specificValue
```

### Note

La variabile di membro `specificValue` contiene il valore della condizione di dipendenza che il Procedura guidata dell'oggetto Business confronta con il valore della proprietà dipendente (che la variabile di membro `propertyName` specifica). Il tipo di confronto è determinato dalla variabile `operatorType`. Il valore specifico può essere uno dei seguenti:

- Una costante (dello stesso tipo della proprietà dipendente)

  Ad esempio, se la condizione di dipendenza ha specificato l'operatore Minore di (`CompleteCondition.OP_LESS_THAN`) come suo `operatorType` ed ha un valore di 5 come `specificValue`, il valore della proprietà dipendente deve essere minore di 5 quando le condizioni di input associate sono true.
- Il nome di un'altra proprietà dell'agente

  Ad esempio, se la condizione di dipendenza ha specificato l'operatore Maggiore di (`CompleteCondition.OP_GREATER_THAN`) come suo `operatorType` e specifica il nome della proprietà "`Property1`" come suo `specificValue`, il valore della proprietà dipendente deve essere maggiore del valore della proprietà dell'agente `Property1` quando le condizioni di input associate sono true.

La variabile `specificValue` viene dichiarata di tipo `String` così che possa contenere qualsiasi tipo di valore. Per effettuare un confronto correttamente, il Procedura guidata dell'oggetto Business ha bisogno di sapere il tipo di dati effettivo del valore specifico, che è contenuto nella variabile membro `typeOfSpecificValue`.

## typeOfSpecificValue

Definisce il tipo di dati del valore specifico della condizione di dipendenza.

### Tipo

```
public int typeOfSpecificValue
```

### Note

La variabile di membro `typeOfSpecificValue` contiene il tipo di dati del valore specifico della condizione di dipendenza. La variabile `specificValue` viene dichiarata di tipo `String` così che possa contenere qualsiasi tipo di valore. Per effettuare un confronto correttamente, il Procedura guidata dell'oggetto Business ha bisogno di sapere il tipo di dati effettivo del valore specifico. I valori validi per la variabile `typeOfSpecificValue` sono le costanti del tipo di proprietà, definite nella classe `AgentProperty`. Per ulteriori informazioni, consultare Tabella 69 a pagina 189.

Ad esempio, se il valore specifico della condizione di dipendenza è il numero intero costante 5:

- La variabile `specificValue` contiene la stringa "5".
- La variabile `typeOfSpecificValue` contiene la costante tipo proprietà `AgentProperty.TYPE_INTEGER`.

# Metodi

Tabella 90 riepiloga i metodi nella classe `DependentCondition`.

*Tabella 90. Metodi del membro della classe DependentCondition.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `DependentCondition()` | Crea un oggetto condizione di dipendenza. | 249 |
| `copy()` | Copia la condizione di dipendenza corrente nell'oggetto `DependentCondition` specificato. | 250 |

## DependentCondition()

Crea un oggetto condizione di dipendenza.

### Sintassi

```
public DependentCondition();
public DependentCondition(String name, String op,
   boolean isDyn, int type, String specificVal);
```

### Parametri

*isDyn* Specifica se ottenere il valore della proprietà valore specificato in modo dinamico. Il valore di questo parametro inizializza la variabile del membro `isDynamic` ("isDynamic" a pagina 247).

*name* È il nome della proprietà dipendente. Il valore di questo parametro inizializza la variabile del membro `propertyName` ("propertyName" a pagina 248)

*op* È l'operatore che determina il tipo di confronto da effettuare. Il valore di questo parametro inizializza la variabile del membro `operatorType` ("operatorType" a pagina 248).

*specificVal* Definisce il valore specifico della condizione di dipendenza. Il valore di questo parametro inizializza la variabile del membro `specificValue` (“specificValue” a pagina 248)

*type* Definisce il il tipo di dati del valore specifico. Il valore di questo parametro inizializza la variabile del membro `typeOfSpecificValue` (“typeOfSpecificValue” a pagina 249)

### Valori di ritorno

Un nuovo oggetto istanziato `DependentCondition` object.

## copy()

Copia la condizione di dipendenza attuale nell’oggetto condizione di dipendenza specificato.

### Sintassi

```
public void copy(DependentCondition depCond);
```

### Parametri

*depCond* È un riferimento ad un oggetto condizione di dipendenza all’interno del quale viene copiata la condizione di dipendenza attuale.

### Valori di ritorno

Nessuno.

# Capitolo 19. Interfaccia IGeneratesBinFiles

Il L'API ODK (Object Discovery Agent Development Kit) utilizza l'interfaccia `IGeneratesBinFiles` per fornire costanti generali all'ODA (Object Discovery Agent) per la generazione dei file come contenuto. Questa interfaccia definisce l'insieme di metodi che lo sviluppatore ODA implementa per permettere all'ODA di generare file binari. Procedura guidata dell'oggetto Business richiama i metodi dell'interfaccia `IGeneratesBinFiles` per generare un contenuto di accesso che consiste in un oggetto file. Un oggetto file è un oggetto `File`Java,che rappresenta un file di sistema operativo binario.

**Nota:** Un'ODA deve inoltre supportare la generazione di definizioni di oggetti business come contenuto. Per abilitare un'ODA alla generazione di definizioni di oggetti business da dati sorgente, è necessario implementare l' interfaccia `IGeneratesBoDefs`. Per ulteriori informazioni, consultare Capitolo 20, "Interfaccia IGeneratesBoDefs", a pagina 255.

Per fornire all'ODA la capacità di generare oggetti file, lo sviluppatore ODA deve procedere come segue:

- Nella definizione della classe ODA (che è un'estensione della classe `ODKAgentBase2`), includere `IGeneratesBinFiles` come interfaccia implementata dall'ODA.
- All'interno della classe ODA, implementare i metodi dell'interfaccia `IGeneratesBinFiles`. Siccome `IGeneratesBinFiles` è un'interfaccia, gli sviluppatori ODA devono implementare *tutti* i metodi in Tabella 91..

*Tabella 91. Metodi del membro della Interfaccia IGeneratesBinFiles*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `generateBinFiles()` | Genera oggetti file per i nodi origine scelti dall'origine dati. | 251 |
| `getBinFile()` | Recupera oggetti file generati. | 252 |
| `getContentProtocol()` | Indica il protocollo di contenuto supportato per questo tipo di contenuto del file binario. | 253 |

## generateBinFiles()

Genera oggetti di file.

### Sintassi

```
public ContentMetaData generateBinDefs(String[] strNames);
```

### Parametri

*strNames [ ]* È un array di oggetti Stringa. Questo argomento non è attualmente utilizzato.

### Valori di ritorno

Un oggetto `ContentMetaData`, che descrive l'oggetto file generato.

## Eccezioni

**ODKException** Restituito se la generazione dei file binari ha esito negativo.

## Note

La scopo del metodo `generateBinFiles()` dipende dal protocollo di contenuto utilizzato dall'ODA per la generazione del contenuto del file ( `ContentType.BinaryFile`) come riportato di seguito:

- Se l'ODA genera file "su richiesta", il Procedura guidata dell'oggetto Business chiama il metodo `generateBinFiles()` per generare file.
- Se l'ODA genera file tramite richiamate, il Procedura guidata dell'oggetto Business non chiama *mai* il metodo `generateBinFiles()`. In alternativa, l'ODA utilizza altri modi per generare i file, che il Procedura guidata dell'oggetto Business potrà poi accedere.

Se l'ODA genera file "su richiesta", il metodo `generateBinFiles()` è il metodo di generazione contenuto dell'interfaccia `IGeneratesBinFiles`. Può creare oggetti file che contengono informazioni sul processo di generazione delle definizioni dell'oggetto di business. Il Procedura guidata dell'oggetto Business chiama il metodo `generateBinFiles()` per generare il contenuto (se l'ODA supporta la generazione del contenuto del file). questo metodo viene chiamato al passo 5, Generazione di oggetti business.

Per il protocollo su richiesta, questo metodo non restituisce il contenuto generato. In alternativa, restituisce un oggetto metadato del contenuto (`ContentMetaData`), che contiene informazioni che descrivono il contenuto generato. Da questo oggetto metadato contenuto restituito, il Procedura guidata dell'oggetto Business può determinare se il processo di generazione contenuto è stato completato. Quando la generazione è stata completata, il Procedura guidata dell'oggetto Business ottiene gli oggetti file generati con il metodo `getBinFiles()`. Per ulteriori informazioni su come implementare `generateBinFiles()`, consultare "Creazione di file" a pagina 147.

## Vedere anche

generateBoDefs(), getBinFile()

---

# getBinFile()

Richiama gli oggetti file generati dalla struttura del contenuto generato.

## Sintassi

```
public File[] getBinFile(long index);
```

## Parametri

*index* Specifica l'oggetto file da richiamare dalla struttura del contenuto generato.

## Eccezioni

**ODKException** Restituita se il Procedura guidata dell'oggetto Business riscontra un problema nella ricezione degli oggetti file generati dalla struttura del contenuto generato.

## Note

Il metodo `getBinFile()` è un metodo di richiamo del contenuto dell'interfaccia `IGeneratesBinFiles`. Richiama gli oggetti file generati dalla struttura contenuto generato dell'ODA, che è la struttura che l'ODA ha riempito con gli oggetti file generati. Il metodo che ha riempito la struttura del contenuto generato dipende dal protocollo del contenuto supportato dall'ODA per la generazione dei file nel seguente modo:

- Se l'ODA genera file ″su richiesta″, il metodo `generateBinFiles()` riempie la struttura del contenuto generato.
- Se l'ODA genera file tramite richiamate, alcuni metodi definiti da utente riempiono la struttura del contenuto generato.

Il valore dell'argomento *indice* determina se `getBinFile()` restituisce uno o tutti gli oggetti file generati, come mostra Tabella 92.

*Tabella 92. Specifica degli oggetti file da restituire*

| **Valore dell'argomento** *indice* | **Azioni di getBinFile()** |
|---|---|
| Nell'intervallo: 0 a *conteggio*<br>(dove *conteggio* è la variabile del membro nell'oggetto del metadato del contenuto che specifica il numero di oggetti file nella struttura del contenuto generato) | Restituisce un array che contiene *un* oggetto file (`file`Java), l'oggetto `File` nella posizione *indice* specificata nella struttura del contenuto generato. |
| `ODKConstant.GET_ALL_OBJECTS` | Restituisce un array di *tutti* gli oggetti file generati nella struttura contenuto generato. |

Per ulteriori informazioni su come implementare `getBinFile()`, consultare "Aggiunta di accesso ai file generati" a pagina 152.

## Vedere anche

generateBinFiles(), getBoDefs()

# getContentProtocol()

Indica il protocollo del contenuto che l'ODA supporta per un tipo di contenuto specificato.

## Sintassi

```
public long getContentProtocol(ContentType contentType);
```

## Parametri

*contentType* Indica il tipo di contenuto per cui il metodo ottiene il protocollo di contenuto supportato.

## Valori di ritorno

Un valore intero lungo (`lungo`) che indica il protocollo di contenuto implementato dall'ODA. Confronta questo valore `lungo` con le seguenti costanti del protocollo di contenuti:

**`ODKConstant.CONTENT_PROTOCOL_CALLBACK`**
Indica che l'ODA supporta un protocollo di richiamata; che significa che

l'ODK avvia la generazione di un contenuto specificato e informa il Procedura guidata dell'oggetto Business quando la generazione è completa.

**ODKConstant.CONTENT_PROTOCOL_ONREQUEST**
Indica che l'ODA supporta un protocollo su richiesta; che significa che il Procedura guidata dell'oggetto Business avvia la generazione del tipo di contenuto specificato.

## Note

Il metodo `getContentProtocol()` è un metodo singolo definito nell'interfaccia `IGeneratesContent`, che viene esteso dall'interfaccia `IGeneratesBoDefs`. Il Procedura guidata dell'oggetto Business richiama `getContentProtocol()` per determinare il protocollo di contenuto supportato dall'ODA per il tipo di contenuto *contentType*. Per ulteriori informazioni, consultare "Indicazione dei protocolli di contenuto implementato" a pagina 119.

# Capitolo 20. Interfaccia IGeneratesBoDefs

Il L'API ODK (Object Discovery Agent Development Kit) utilizza l'interfaccia `IGeneratesBoDefs` per fornire costanti generali all'ODA (Object Discovery Agent) per la generazione delle definizioni degli oggetti business come contenuto. Questa interfaccia definisce l'insieme di metodi che lo sviluppatore ODA implementa per permettere all'ODA di generare le definizioni dell'oggetto business partendo dai dati sorgente. Procedura guidata dell'oggetto Business richiama i metodi dell'interfaccia `IGeneratesBoDefs` per ottenere i nodi sorgenti, ed anche per generare ed accedere le definizioni dell'oggetto business.

**Nota:** Un ODA supporta inoltre la generazione di un oggetto file come contenuto. Per abilitare un'ODA alla generazione di file binari da dati sorgente, è necessario implementare l'interfaccia `IGeneratesBinFiles`. Per ulteriori informazioni, consultare Capitolo 19, "Interfaccia IGeneratesBinFiles", a pagina 251.

Per fornire all'ODA la capacità di generare definizioni di oggetti business partendo da dati sorgente, lo sviluppatore ODA deve procedere come segue:

- Nella definizione della classe ODA (che è un'estensione della classe `ODKAgentBase2`), includere `IGeneratesBoDefs` come interfaccia implementata dall'ODA.
- All'interno della classe ODA, implementare i metodi dell'interfaccia `IGeneratesBoDefs`. Siccome `IGeneratesBoDefs` è un'interfaccia, gli sviluppatori ODA devono implementare *tutti* i metodi in Tabella 93..

*Tabella 93. Metodi del membro della Interfaccia IGeneratesBoDefs*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `generateBoDefs()` | Genera definizioni dell'oggetto business per i nodi origine specificati partendo dall'origine dei dati. | 255 |
| `getBoDefs()` | Recupera le definizioni dell'oggetto business generato. | 256 |
| `getContentProtocol()` | Indica il protocollo di contenuto supportato per questo tipo di contenuto della definizione dell'oggetto business. | 257 |
| `getTreeNodes()` | Costruisce un array di nodi ad albero che rappresentano la gerarchia dei nodi origine. | 258 |

## generateBoDefs()

Genera definizioni dell'oggetto business per i nodi origine specificati.

### Sintassi

```
public ContentMetaData generateBoDefs(String[] srcNodeNames);
```

## Parametri

*srcNodeNames [ ]*
: È un array che contiene i nomi dei nodi origine selezionati dall'utente.

## Valori di ritorno

Un oggetto `ContentMetaData`, che descrive le definizioni dell'oggetto business generate per i nodi origine nominati nell'argomento *srcNodeNames*.

## Eccezioni

**`ODKException`**
: Restituito se la generazione delle definizioni dell'oggetto business hanno avuto esito negativo.

## Note

Il metodo `generateBoDefs()` è il metodo di generazione contenuti dell'interfaccia `IGeneratesBoDefs`. Crea definizioni di oggetti business per ciascuno dei nodi origine nominati nell'array *srcNodeNames*. L'utente ha selezionato questi nodi origine nella finestra Selezione origine del Procedura guidata dell'oggetto Business. Una volta finita la selezione dei nodi origine da parte dell'utente, Procedura guidata dell'oggetto Business chiama il metodo `generateBinFiles()` per generare il contenuto. Tale metodo viene richiamato al passo 5, Generazione di oggetti business, all'avvio.

**Nota:** Procedura guidata dell'oggetto Business richiama sempre `generateBoDefs()` perché l'ODA deve supportare un protocollo di contenuti su richiesta per la generazione di definizioni dell'oggetto business. Per ulteriori informazioni sui protocolli di contenuto, consultare "Selezione del protocollo del contenuto ODA" a pagina 118.

Lo scopo del metodo `generateBoDefs()` è quello di generare una definizione dell'oggetto business (oggetto `BusObjDef`) per ogni nodo origine selezionato dall'utente, archiviarlo nella struttura contenuto generato, e restituire un oggetto metadato del contenuto (`ContentMetaData`) che descrive il contenuto generato. Questo metodo *non* restituisce effettivamente il contenuto generato al Procedura guidata dell'oggetto Business. Da questo oggetto metadato contenuto restituito, il Procedura guidata dell'oggetto Business può determinare se il processo di generazione contenuto è stato completato. Quando la generazione è stata completata, Procedura guidata dell'oggetto Business ottiene le definizioni dell'oggetto business generate con il metodo `getBoDefs()`. Per ulteriori informazioni su come implementare `generateBoDefs()`, consultare "Creazione di definizioni di oggetti business" a pagina 129.

## Vedere anche

generateBinFiles(), getBoDefs()

---

# getBoDefs()

Richiama le definizioni di oggetto business.

## Sintassi

```
public BusObjDef[] getBoDefs(long index);
```

## Parametri

*indice* Specifica la definizione dell'oggetto business da richiamare dalla struttura contenuto generato.

## Eccezioni

**ODKException** Restituita se il Procedura guidata dell'oggetto Business riscontra un problema nella ricezione delle definizioni dell'oggetto business dalla struttura del contenuto generato.

## Note

Il metodo `getBoDefs()` è un metodo di richiamo del contenuto dell'interfaccia `IGeneratesBoDefs`. Richiama le definizioni dell'oggetto business generate dalla struttura contenuto generato dell'ODA, che è la struttura che il metodo `generateBoDefs()` ha riempito con le definizioni dell'oggetto business generate. Il valore dell'argomento *indice* determina se `getBoDefs()` restituisce una o tutte le definizioni dell'oggetto business generate, come mostra Tabella 94.

*Tabella 94. Specifica delle definizioni dell'oggetto business da restituire*

| **Valore dell'argomento** *indice* | **Azioni di getBoDefs()** |
|---|---|
| Nell'intervallo: 0 a *conteggio*<br>(dove *conteggio* è la variabile del membro nell'oggetto del metadato del contenuto che specifica il numero di definizioni nella struttura del contenuto generato) | Restituisce un array che contiene *una* definizione dell'oggetto business (oggetto `BusObjDef`), l'oggetto `BusObjDef` nella posizione *indice* specificata nella struttura del contenuto generato. |
| `ODKConstant.GET_ALL_OBJECTS` | Restituisce un array di *tutte* le definizioni dell'oggetto business generate nella struttura contenuto generato. |

Per ulteriori informazioni su come implementare `getBoDefs()`, consultare "Accesso alle definizioni di oggetti business generati" a pagina 143.

## Vedere anche

generateBoDefs(), getBinFile()

# getContentProtocol()

Indica il protocollo del contenuto che l'ODA supporta per un tipo di contenuto specificato.

## Sintassi

```
public long getContentProtocol(ContentType contentType);
```

## Parametri

*contentType* Indica il tipo di contenuto per cui il metodo determina il protocollo di contenuto supportato.

## Valori di ritorno

Un valore intero lungo (`lungo`) che indica il protocollo di contenuto implementato dall'ODA. Confronta questo valore `lungo` con le seguenti costanti del protocollo di contenuti:

**ODKConstant.CONTENT_PROTOCOL_CALLBACK**
Indica che l'ODA supporta un protocollo di richiamata; che significa che l'ODK avvia la generazione di un contenuto specificato e informa il Procedura guidata dell'oggetto Business quando la generazione è completa.

**ODKConstant.CONTENT_PROTOCOL_ONREQUEST**
Indica che l'ODA supporta un protocollo su richiesta; che significa che il Procedura guidata dell'oggetto Business avvia la generazione del tipo di contenuto specificato.

## Note

Il metodo `getContentProtocol()` è un metodo singolo definito nell'interfaccia `IGeneratesContent`, che viene esteso dall'interfaccia `IGeneratesBoDefs`. Il Procedura guidata dell'oggetto Business richiama `getContentProtocol()` per determinare il protocollo di contenuto supportato dall'ODA per il tipo di contenuto *contentType*. Per ulteriori informazioni, consultare "Indicazione dei protocolli di contenuto implementato" a pagina 119.

---

# getTreeNodes()

Crea un'array di nodi della struttura che rappresenta un livello nella gerarchia dei nodi origine.

## Sintassi

```
public TreeNode[] getTreeNodes(String parentNodePath, String searchPattern);
```

## Parametri

*parentNodePath*
È un percorso completo dal nodo di livello superiore al nodo origine i cui secondari devono essere restituiti al Procedura guidata dell'oggetto Business. Ogni nodo nel percorso è separato da due punti (`:`).

*searchPattern* È il percorso di ricerca specificato dell'utente per i nodi secondari nel nodo espandibile *parentNodePath*.

## Valori di ritorno

Un array di oggetti `TreeNode`, in cui l'oggetto nodo tree è un nodo secondario nella gerarchia di oggetti specificati.

## Eccezioni

**ODKException** Restituito se Object Discovery Agent incontra un problema nella ricezione di nodi ad albero.

## Note

Il metodo `getTreeNodes()` è un metodo a generazione di nodo origine per l'interfaccia `IGeneratesBoDefs`. Procedura guidata dell'oggetto Business richiama `getTreeNodes()` per ottenere l'array di nodi della struttura che inizializza la finestra Seleziona origine (Passo 3). Da questa finestra di dialogo, l'utente seleziona nodi origine specifici per la generazione di definizioni dell'oggetto business. All'interno del metodo `getTreeNodes()`, si devono costruire nodi ad albero per rappresentare

la gerarchia dei nodi origine nell'origine dati. Il metodo `getTreeNode()` restituisce questa gerarchia di nodi origine come un array di oggetti `TreeNode` al Procedura guidata dell'oggetto Business.

Il vetore di nodi tree restituito da `getTreeNodes()` fornisce i nodi tree ad un livello determinato della gerarchia del nodo origine. A qualsiasi livello, alcuno nodi origine potrebbero essere espandibili (avere nodi secondari) ed altri potrebbero essere nodi foglia (finali). L'utente può spostarsi nella gerarchia espandendo qualsiasi nodo origine che mostri un segno (+) alla propria sinistra. Quando un utente espande un nodo, il Procedura guidata dell'oggetto Business richiama nuovamente `getTreeNodes()`, fornendo come argomento *parentNodePath* il nome del nodo che l'utente vuole espandere. Il nome del nodo è costituito dai nomi dei singoli nodi nel percorso, separati da due punti (:).

**Nota:** Un'ODA utilizza i due punti piuttosto che la barra o la barra rovesciata per tenere indipendente il percorso del sistema operativo.

Il metodo `getTreeNodes()` effettua le seguenti attività di base per generare nodi origine:

1. Analizza il *parentNodePath* per identificare l'oggetto principale da ricercare nell'origine dati.
2. Scopre gli oggetti secondari dell'oggetto principale origine dati specificato.

   Se il Procedura guidata dell'oggetto Business fornisce un argomento *searchPattern* a `getTreeNodes()`, l'utente ha specificato un criterio di ricerca. Quindi, `getTreeNodes()` deve restituire solo i nodi secondari del nodo *parentNodePath* che corrispondono ai criteri di ricerca *searchPattern*. La capacità di applicare un percorso di ricerca ai nodi origine richiede che le seguenti condizioni siano vere:

   - Il *searchPattern* specificato dall'utente deve corrispondere ai criteri di ricerca supportati da ODA.

     La variabile del membro `searchPatternDesc` nell'oggetto metadato dell'ODA (`AgentMetaData`) fornisce una descrizione del criterio di ricerca supportato all'utente. Comunque, il metodo `getTreeNodes()` deve analizzare il *searchPattern* specificato dall'utente per assicurarsi che corrisponda ai criteri di ricerca supportati.
   - L'ODA supporta percorsi di ricerca.

     La variabile del membro `searchableNodes` nell'oggetto metadato dell'ODA (`AgentMetaData`) sia `true`. Se `searchableNodes` è `false`, la voce di menu **Ricerca voci** (che avvia il criterio di ricerca dell'utente) non sarà disponibile. Quindi l'utente non potrà immettere criteri di ricerca.
3. Costruisce nodi tree per gli oggetti secondari ed inserisce questi nodi nell'array dei nodi ad albero.

Per ulteriori informazioni su come implementare `getTreeNodes()`, consultare "Creazione dei nodi di origine" a pagina 122.

## Vedere anche

Per informazioni correlate, consultare Capitolo 26, "Classe TreeNode", a pagina 291.

# Capitolo 21. Classe InputCondition

Il Object Discovery Agent Development Kit (ODK) API fornisce la classe `InputCondition` per rappresentare una condizione sul valore di una proprietà dell'agente. Quando la condizione di input è true, la condizione di dipendenza associata viene applicata alla proprietà dipendente. Le condizioni di input e la loro condizione di dipendenza associata (o condizioni) sono archiviate in un oggetto condizione completa (`CompleteCondition`).

**Nota:** Per informazioni sulle condizioni complete, consultare Capitolo 14, "Classe CompleteCondition", a pagina 231.

La classe `InputCondition` definisce quanto segue:

- "Variabili dei membri"
- "Metodi" a pagina 263

## Variabili dei membri

Tabella 95 riepiloga le variabili del membro nella classe `InputCondition`.

*Tabella 95. Variabili del membro della classe InputCondition.*

| variabile del membro | Descrizione | Pagina |
|---|---|---|
| `isDynamic` | Specifica se il Procedura guidata dell'oggetto Business deve controllare il valore della proprietà valore specifica prima di effettuare il confronto della condizione di input. | 261 |
| `operatorType` | Definisce il tipo di operatore per la condizione di input. | 261 |
| `specificValue` | Definisce il valore da confrontare con il valore della proprietà dell'agente. | 262 |
| `typeOfSpecificValue` | Definisce il tipo di dati del valore specifico della condizione di input. | 262 |

### isDynamic

Specifica se il Procedura guidata dell'oggetto Business deve controllare il valore della proprietà valore specifica prima di effettuare il confronto della condizione di input.

#### Tipo

```
public boolean isDynamic
```

#### Note

Quando la variabile membro `isDynamic` è `true`, il Procedura guidata dell'oggetto Business ottiene il valore della proprietà che il valore membro `specificValue` specifica *prima* di eseguire il confronto con il valore della proprietà dell'agente. Se `specificValue` contiene una costante, `isDynamic` deve essere impostato come `false`.

### operatorType

Definisce il tipo di operatore per la condizione di input.

### Tipo

```
public String operatorType
```

### Note

La variabile di membro `operatorType` definisce il tipo di confronto che il Procedura guidata dell'oggetto Business effettua tra il valore della proprietà dell'agente e `specificValue`. I valori validi per la variabile `operatorType` sono le costanti dell'operatore definite nella classe `CompleteCondition`. Per ulteriori informazioni, consultare Tabella 78 a pagina 231.

## specificValue

Definisce il valore da confrontare con il valore della proprietà dell'agente.

### Tipo

```
public String specificValue
```

### Note

La variabile di membro `specificValue` contiene il valore della condizione di input che il Procedura guidata dell'oggetto Business confronta con il valore della proprietà dell'agente. Il tipo di confronto è determinato dalla variabile `operatorType`. Il valore specifico può essere uno dei seguenti:

- Una costante (dello stesso tipo della proprietà dell'agente)

  Ad esempio, se la condizione di input ha specificato l'operatore Minore di (`CompleteCondition.OP_LESS_THAN`) come suo `operatorType` ed ha un valore di 5 come `specificValue`, la condizione di dipendenza associata si applica quando il valore della proprietà dell'agente è minore di 5.
- Il nome di un'altra proprietà dell'agente

  Ad esempio, se la condizione di input ha specificato l'operatore Maggiore di (`CompleteCondition.OP_GREATER_THAN`) come suo `operatorType` e specifica il nome della proprietà "`Property1`" come suo `specificValue`, la condizione di dipendenza associata si applica quando il valore della proprietà dell'agente è superiore al valore della proprietà dell'agente `Property1`.

La variabile `specificValue` viene dichiarata di tipo `String` così che possa contenere qualsiasi tipo di valore. Per effettuare un confronto correttamente, il Procedura guidata dell'oggetto Business ha bisogno di sapere il tipo di dati effettivo del valore specifico, che è contenuto nella variabile membro `typeOfSpecificValue`.

## typeOfSpecificValue

Definisce il tipo di dati del valore specifico della condizione di input.

### Tipo

```
public int typeOfSpecificValue
```

### Note

La variabile di membro `typeOfSpecificValue` contiene il tipo di dati del valore specifico della condizione di input. La variabile `specificValue` viene dichiarata di tipo `String` così che possa contenere qualsiasi tipo di valore. Per effettuare un confronto correttamente, il Procedura guidata dell'oggetto Business ha bisogno di sapere il tipo di dati effettivo del valore specifico. I valori validi per la variabile `typeOfSpecificValue` sono le costanti del tipo di proprietà, definite nella classe `AgentProperty`. Per ulteriori informazioni, consultare Tabella 69 a pagina 189.

Ad esempio, se il valore specifico della condizione di input è il numero intero costante 5:

- La variabile `specificValue` contiene la stringa "5".
- La variabile `typeOfSpecificValue` contiene la costante tipo proprietà `AgentProperty.TYPE_INTEGER`.

## Metodi

Tabella 96 riepiloga i metodi nella classe `InputCondition`.

*Tabella 96. Metodi del membro della classe InputCondition.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `InputCondition()` | Crea un oggetto condizione di input. | 263 |
| `copy()` | Copia la condizione di input corrente nell'oggetto condizione di input specificato. | 263 |

## InputCondition()

Crea un oggetto condizione di input.

### Sintassi

```
public InputCondition();
public InputCondition(String operator, boolean isDyn, int type,
   String specificVal);
```

### Parametri

*isDyn* Specifica se ottenere il valore della proprietà valore specificato in modo dinamico. Il valore di questo parametro inizializza la variabile del membro `isDynamic` ("isDynamic" a pagina 261).

*operator* È l'operatore che determina il tipo di confronto da effettuare. Il valore di questo parametro inizializza la variabile del membro `operatorType` ("operatorType" a pagina 248).

*specificVal* È il valore specifico della condizione di input. Il valore di questo parametro inizializza la variabile del membro `specificValue` ("specificValue" a pagina 262).

*type* Definisce il il tipo di dati del valore specifico. Il valore di questo parametro inizializza la variabile del membro `typeOfSpecificValue` ("typeOfSpecificValue" a pagina 262)

### Valori di ritorno

Un oggetto creato come nuova istanza `InputCondition`.

## copy()

Copia la condizione di input corrente nell'oggetto condizione di input specificato.

### Sintassi

```
public void copy(InputCondition inputCond);
```

## Parametri

*inputCond* È un riferimento ad un oggetto `condizione di input` all'interno del quale viene copiata la condizione di input attuale.

## Valori di ritorno

Nessuno.

# Capitolo 22. Classe ODKAgentBase2

Il L'API ODK (Object Discovery Agent Development Kit) fornisce la classe ODKAgentBase2 come classe base per un ODA (Object Discovery Agent). Per questa classe uno sviluppatore ODA deve derivare una *classe ODA* e implementare i metodi astratti per l'ODA.

**Nota:** La classe ODKAgentBase2 estende la classe ODKAgentBase della libreria ODA di livello inferiore. Eredita i metodi getAgentProperties(), getVersion(), init() e terminate() di questa classe. Inoltre "disabilita" i metodi getTreeNodes() e generateDefs() di questa classe poiché ora sono sostituiti da funzionalità definite nei metodi getTreeNodes() e generateBoDefs() dell'interfaccia IGeneratesBoDefs.

**Importante**
Tutti gli ODA *devono* estendere questa classe di base ODA e fornire le implementazioni per tutti i relativi metodi *tranne* getVersion().

Tabella 97 riepiloga i metodi della classe ODKAgentBase2.

*Tabella 97. Metodi del membro della Classe ODKAgentBase2*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| getAgentProperties() | Invia un array di proprietà di configurazioni ODA a Procedura guidata dell'oggetto Business. | 265 |
| getMetaData() | Invia i metadati ODA a Procedura guidata dell'oggetto Business. | 266 |
| getVersion() | Richiama la versione di ODA. | 267 |
| init() | Inizializza l'ODA. | 267 |
| terminate() | Termina l'ODA, eseguendo qualsiasi attività di cleanup richiesta. | 268 |

## getAgentProperties()

Invia un array di ODA configurazione delle proprietà a Procedura guidata dell'oggetto Business.

### Sintassi

```
public abstract AgentProperty[] getAgentProperties();
```

### Parametri

Nessuno.

### Valori di ritorno

Un array di oggetti AgentProperty un oggetto per ciascuna proprietà di configurazione ODA.

## Eccezioni

**ODKException** Viene generato se ODA non riesce a richiamare le proprietà di configurazione.

## Note

La Procedura guidata dell'oggetto Business invoca il metodo `getAgentProperties()` per richiamare un array delle proprietà di configurazione ODA che inizializza la finestra Configura agente (passo 2). Da questa casella di dialogo, è possibile immettere o modificare tali valori delle proprietà.

**Importante:** Il metodo `getAgentProperties()` è un metodo astratto che non ha implementazioni predefinite. Pertanto la classe ODA *deve* implementare questo metodo.

All'interno del metodo `getAgentProperties()` occorre eseguire l'istanza e inizializzare delle proprietà agente (`AgentProperty`) per ciascuna ODA e memorizzare ciascuna proprietà nell'array delle proprietà di configurazione. Il metodo `getAgentProperties()` restituisce quest'array delle proprietà di configurazione al relativo programma di chiamata, Procedura guidata dell'oggetto Business. Una volta che l'utente ha impostato le proprietà di configurazione nella casella di dialogo Configura agente, la Procedura guidata dell'oggetto Business legge queste proprietà inizializzate dall'utente nella memoria runtime di ODA. È possibile ottenere la proprietà inizializzata dall'utente con il metodo `getAgentProperty()` o `getAllAgentProperties()` nella classe `ODKUtility`. Per ulteriori informazioni su come implementare `getAgentProperties()`, consultare "Ottenimento delle proprietà di configurazione" a pagina 111.

## Vedere anche

getAgentProperty(), getAllAgentProperties()

---

# getMetaData()

Invia i metadati ODA alla Procedura guidata dell'oggetto Business.

## Sintassi

```
public abstract AgentMetaData getMetaData();
```

## Parametri

Nessuno.

## Valori di ritorno

Un oggetto `AgentMetaData` che contiene i metadati per ODA.

## Note

La Procedura guidata dell'oggetto Business richiama il metodo `getMetaData()` per richiamare i metadati di ODA. Richiama `getMetaData()` dopo il completamento della chiamata al metodo `getAgentProperties()`. Il metodo `getMetaData()` restituisce l'oggetto `AgentMetaData` per ODA. L'oggetto `AgentMetaData` contiene metadati per l'ODA, quali la versione e il contenuto generato supportati. All'interno di questo metodo, richiamare il costruttore `AgentMetaData()` e ritornare all'oggetto di metadati di cui si è creata l'istanza.

**Importante:** Il metodo `getMetaData()` è il metodo astratto che non ha implementazioni predefinite. Pertanto, la classe ODA *deve* implementare questo metodo.

Per ulteriori informazioni su come implementare `getMetaData()`, consultare "Inizializzazione di metadati ODA" a pagina 113.

## getVersion()

Richiama la versione del runtime ODA.

### Sintassi

```
public String getVersion();
```

### Parametri

Nessuno.

### Valori di ritorno

`String` che contiene la versione del runtime ODA.

## init()

Esegue le attività di inizializzazione ODA.

### Sintassi

```
public abstract void init();
```

### Parametri

Nessuno.

### Valori di ritorno

Nessuno.

### Eccezioni

**`ODKException`** Viene generato se l'inizializzazione ODA non ha esito positivo.

### Note

Business Object Designer Express richiama il metodo `init()` per inizializzare l'ODA (Object Discovery Agent). Richiama `init()` dopo il completamento della chiamata ai metodi `getAgentProperties()` e `getMetaData()`. Tipicamente, l'inizializzazione ODA comprende il recupero dei valori per le proprietà di configurazione ODA, stabilendo una collegamento con l'origine dati (che può essere un'applicazione, un database, un file XML e un'altra origine di oggetti business) e assegnando qualsiasi risorsa che richiede ODA.

**Importante:** Il metodo `init()` è un metodo astratto che non ha implementazioni predefinite. Pertanto la classe ODA *deve* implementare questo metodo.

Per ulteriori informazioni su come implementare `init()`, consultare "Inizializzazione dell'avvio di ODA" a pagina 115.

## terminate()

Termina l'ODA, eseguendo le attività di clean-up richieste.

### Sintassi

```
public abstract void terminate();
```

### Parametri

Nessuno.

### Valori di ritorno

Nessuno.

### Note

Business Object Designer Express richiama il metodo `terminate()` quando viene arrestato ODA. Nell'implementazione di questo metodo, è consigliabile liberare tutta la memoria e scollegarsi dall'origine dati.

**Importante:** Il metodo `terminate()` è un metodo astratto che non ha implementazioni predefinite. Pertanto, la classe ODA *deve* implementare questo metodo.

## Metodi non approvati

Alcuni metodi nella classe `ODKAgentBase2` erano supportati nelle versioni precedenti ma non sono più supportati. Tali *metodi non approvati* non genereranno errori, ma IBM ne sconsiglia l'utilizzo e consiglia di migrare il codice esistente ai nuovi metodi. I metodi non approvati potrebbero essere eliminati in un futuro release.

Elencare i metodi non approvati per la classe `ODKAgentBase2`. Se si sta scrivendo un nuovo ODA (non modificando l'ODA esistente), è possibile ignorare questa sezione.

*Tabella 98. Metodi non approvati per la classe ODKAgentBase2*

| Metodo obsoleto | Metodo sostitutivo |
|---|---|
| `getTreeNodes()`<br>(ereditato da `ODKAgentBase`) | `getTreeNodes()` nell'interfaccia `IGeneratesBoDefs` |
| `generateDefs()`<br>(ereditato da `ODKAgentBase`) | `generateBoDefs()` nell'interfaccia `IGeneratesBoDefs` (per generare definizioni di oggetti business)<br>**Nota:** È anche possibile generare file con il metodo `generateBinFiles()` nell'interfaccia `IGeneratesBinFiles`. |

# Capitolo 23. Interfaccia ODKConstant

Il L'API ODK (Object Discovery Agent Development Kit) utilizza l'interfaccia `ODKConstant` per fornire costanti generali all'ODA (Object Discovery Agent). Qualsiasi classe che implementa l'interfaccia `ODKConstant` può accedere direttamente alle relative costanti definite. Ad esempio, se la classe `TreeNode` implementa l'interfaccia `ODKConstant`, i relativi metodi possono accedere alla costante `MSG_QUESTION` nel modo seguente:

```
int message_icon = MSG_QUESTION;
```

L'interfaccia `ODKConstant` definisce le variabili statiche del membro per rappresentare i seguenti tipi di costanti:



## Costanti valore stringa

Tabella 99 riepiloga le costanti del valore stringa nell'interfaccia `ODKConstant`. Tali costanti rappresentano i valori di attributi speciali di Vuoto e Ignora. Tutte le costanti valore stringa sono di tipo `String`.

*Tabella 99. Costanti valore stringa dell'interfaccia ODKConstant.*

| Costante valore stringa | Descrizione |
|---|---|
| `CW_EMPTY_STRING` | Specifica la costante definita per un valore String vuoto (`""`) |
| `CW_NULL_STRING` | Specifica la costante definita per un valore `null` |

## Costanti della casella di dialogo di risposta utente

Il Il metodo `sendMsg()`, definito nella classe `ODKUtility`, fornisce allo sviluppatore ODA uno strumento per visualizzare una casella di dialogo di risposta utente. Per fornire supporto per `sendMsg()`, l'interfaccia `ODKConstant` fornisce le costanti delle caselle di dialogo di risposta utente riportate in Tabella 100. Tutte le costanti della casella di dialogo di risposta utente (`int`).

*Tabella 100. Costanti della casella di dialogo di risposta utente dell'interfaccia ODKConstant.*

| Costante della casella di dialogo di risposta utente | Descrizione |
|---|---|
| costanti dei pulsanti della casella di dialogo | |
| `MSG_OK` | Specifica la visualizzazione del pulsante OK nella casella di dialogo di risposta utente |

*Tabella 100. Costanti della casella di dialogo di risposta utente dell'interfaccia ODKConstant. (Continua)*

| Costante della casella di dialogo di risposta utente | Descrizione |
|---|---|
| MSG_OKCANCEL | Specifica la visualizzazione dei pulsanti OK e Annulla nella casella di dialogo di risposta utente |
| MSG_RETRYCANCEL | Specifica la visualizzazione dei pulsanti Riprova e Annulla nella casella di dialogo di risposta utente |
| MSG_ABORTRETRYIGNORE | Specifica la visualizzazione dei pulsanti Riprova, Ignora e Annulla nella casella di dialogo di risposta utente |
| MSG_YESNO | Specifica la visualizzazione dei pulsanti Sì e No nella casella di dialogo di risposta utente |
| MSG_YESNOCANCEL | Specifica la visualizzazione dei pulsanti Sì, No e Annulla nella casella di dialogo di risposta utente |
| costanti delle icone della casella di dialogo | |
| MSG_ERROR | Specifica che la casella di dialogo di risposta utente visualizzi un'icona di errore. |
| MSG_CRITIALERROR | Specifica che la casella di dialogo di risposta utente visualizzi un'icona di errore critico. |
| MSG_WARNING | Specifica che la casella di dialogo di risposta utente visualizzi un'icona di avvertenza. |
| MSG_INFORMATION | Specifica che la casella di dialogo di risposta utente visualizzi un'icona informativa. |
| MSG_QUESTION | Specifica che la casella di dialogo di risposta utente visualizzi un'icona con punto interrogativo. |
| costanti di risposta utente | |
| ODK_OK | Specifica il pulsante OK nella casella di dialogo di risposta utente |
| ODK_CANCEL | Specifica il pulsante Annulla nella casella di dialogo di risposta utente |
| ODK_RETRY | Specifica il pulsante Riprova nella casella di dialogo di risposta utente |
| ODK_IGNORE | Specifica il pulsante Ignora nella casella di dialogo di risposta utente |
| ODK_ABORT | Specifica il pulsante Interrompi nella casella di dialogo di risposta utente |
| ODK_YES | Specifica il pulsante Sì nella casella di dialogo di risposta utente |
| ODK_NO | Specifica il pulsante No nella casella di dialogo di risposta utente |
| ODK_CLOSE | Specifica il pulsante Chiudi nella casella di dialogo di risposta utente |
| ODK_HELP | Specifica il pulsante ? nella casella di dialogo di risposta utente |

## Costanti di cardinalità

Tabella 101 riepiloga le costanti di cardinalità nell'interfaccia `ODKConstant`. Queste costanti rappresentano i valori validi per la cardinalità di una proprietà agente. Tutte le costanti di cardinalità sono di tipo `String`.

*Tabella 101. Costanti di cardinalità dell'interfaccia ODKConstant.*

| Costanti di cardinalità | Descrizione |
|---|---|
| `MULTIPLE_CARD` | Specifica che una proprietà agente può avere *più* valori, cioè l'utente può specificare più valori per la proprietà. |
| `xSINGLE_CARD` | Specifica che una proprietà agente può avere *solo un* valore, cioè l'utente può specificare solo un valore per la proprietà. |

## Costanti del livello di traccia

Tabella 102 riepiloga le costanti del livello di traccia nell'interfaccia `ODKConstant`. Tali costanti rappresentano livelli di traccia validi per il metodo di traccia `trace()` (definito nella classe `ODKUtility`). Tutte le costanti del livello di traccia sono di tipo intero (`int` ).

*Tabella 102. Costanti di livello di traccia dell'interfaccia ODKConstant.*

| Costante dei livelli di traccia | Descrizione |
|---|---|
| `TRACELEVEL0` | Rappresenta un livello di traccia 0 (errore di log; traccia disattiva) |
| `TRACELEVEL1` | Rappresenta un livello di traccia 1 |
| `TRACELEVEL2` | Rappresenta un livello di traccia 2 |
| `TRACELEVEL3` | Rappresenta un livello di traccia 3 |
| `TRACELEVEL4` | Rappresenta un livello di traccia 4 |
| `TRACELEVEL5` | Rappresenta un livello di traccia 5 |

Per una descrizione di un contenuto previsto a livello di traccia, consultare Tabella 17 a pagina 84.

## Costanti del tipo di messaggio

Tabella 103 riepiloga le costanti dei tipi di messaggi nell'interfaccia `ODKConstant`. Queste costanti indicano il livello di severità per un messagio emesso dal metodo `trace()` (definito nella classe `ODKUtility`). Tutte le costanti dei tipi di messaggio sono di tipo intero (`int`).

*Tabella 103. Costanti di tipi di messaggio dell'interfaccia ODKConstant.*

| Costante del tipo di messaggio | Descrizione |
|---|---|
| `XRD_FATAL` | Rappresenta un errore irreversibile |
| `XRD_ERROR` | Rappresenta un errore |
| `XRD_URGENTWARNING` | Rappresenta una avvertenza |
| `XRD_WARNING` | Rappresenta un'avvertenza |
| `XRD_INFO` | Rappresenta un messaggio informativo |
| `XRD_TRACE` | Rappresenta un messaggio di traccia |

**Importante:** L'interfaccia ODKConstant fornisce anche le costanti dei tipi di messaggio del tipo XRD_INT_*messageType*. Inoltre, definisce la costante XRD_UNKNOWN per rappresentare un tipo di messaggio non definito. Queste costanti del tipo di messaggi sono solo per uso interno. *Non* utilizzare queste costanti del tipo di messaggi nell'ODA.

## Costanti di natura nodo

Tabella 104 riepiloga le costanti nodo nell'interfaccia ODKConstant. Tali costanti indicano le azioni che l'utente può eseguire su un nodo della struttura quando viene visualizzato all'interno della finestra di dialogo Seleziona origine della Procedura guidata dell'oggetto Business. Tutte le costanti di natura nodo sono di tipo intero int).

*Tabella 104. Costanti di natura nodo dell'interfaccia ODKConstant.*

| Costante della natura del nodo | Descrizione |
|---|---|
| NODE_NATURE_NORMAL | Specifica che il nodo tree è "normale", cioè l'utente può espandere il nodo o selezionare il nodo (se è un nodo leaf). |
| NODE_NATURE_FILE | Specifica che il nodo tree può essere associato ad un file, cioè la Procedura guidata dell'oggetto Business abilita la voce di menu Associa file sul menu contestuale del nodo nome per consentire all'utente per individuare il file da associare al nodo. |

## Costanti del protocollo del contenuto

Tabella 105 riepiloga le costanti dei protocollo del contenuto nell'interfaccia ODKConstant. Tali costanti rappresentano il protocollo del contenuto per l' ODA. Tutte le costanti di protocollo del contenuto sono di tipo byte.

*Tabella 105. Costanti del protocollo del contenuto dell'interfaccia ODKConstant.*

| Costanti del protocollo del contenuto | Descrizione |
|---|---|
| CONTENT_PROTOCOL_ONREQUEST | Specifica che l'ODA genera il contenuto "su richiesta"; ovvero Procedura guidata dell'oggetto Business richiede esplicitamente a ODA la generazione di contenuto. Quando tale contenuto è pronto, il runtime ODA lo notifica a Procedura guidata dell'oggetto Business, che può quindi richiamare il contenuto quando necessario. |
| CONTENT_PROTOCOL_CALLBACK | Specifica che l'ODA genera il contenuto "spontaneamente"; ovvero, non può predire o garantire quando il contenuto verrà generato. Quando tale contenuto è pronto, ODA lo deve notificare a Procedura guidata dell'oggetto Business, che può quindi richiamare il contenuto quando necessario. |

## Costante dell'indice di contenuto

Tabella 106 riepiloga le costanti dell'indice del contenuto nell'interfaccia ODKConstant. Questa costante rappresenta un valore speciale per i metodi di richiamo contenuto, che indica di restituire tutto il contenuto generato. La costante di indice di contenuto è di tipo long.

*Tabella 106. Costante di indice del contenuto dell'interfaccia ODKConstant.*

| COstanti indice-contenuto | Descrizione |
|---|---|
| GET_ALL_OBJECTS | Passato come un argomento al metodo di richiamo contenuto dell'ODA. Specifica che il metodo di richiamo contenuto deve restituire *tutto* il contenuto generato. |

Per ulteriori informazioni consultare "Accesso alle definizioni di oggetti business generati" a pagina 143 e "Aggiunta di accesso ai file generati" a pagina 152.

# Capitolo 24. Classe ODKException

La classe ODKException è la classe di base per le eccezioni nell'API Object Discovery Agent Development Kit (ODK). L'API ODK estende la classe Exception Java per creare la classe di eccezione chiamata:

```
com.crossworlds.ODK.ODKException
```

Questa classe rappresenta un *oggetto di eccezione*, che i metodi dell'API ODK possono richiamare.

**Nota:** La descrizione di riferimento per ciascun metodo API ODK elenca le eccezioni generate da questo metodo.

La classe ODKException definisce i seguenti valori:

- "Metodi"
- "Classi secondarie di eccezione" a pagina 276

## Metodi

Tabella 107 riepiloga i metodi della classe ODKException.

*Tabella 107. Metodi del membro della classe ODKException.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| ODKException() | Crea un oggetto eccezione ODK. | 275 |
| getMsg() | Richiama un messaggio di eccezione dall'oggetto eccezione. | 275 |

### ODKException()

Crea un oggetto eccezione.

#### Sintassi

```
public ODKException(String msg);
```

#### Parametri

*msg* è un messaggio di eccezione per l'oggetto eccezione.

#### Valori di ritorno

Un oggetto eccezione ODKException di cui si è creata una nuova istanza.

### getMsg()

Richiama un messaggio di eccezione dall'oggetto eccezione.

#### Sintassi

```
public String getMsg();
```

#### Parametri

Nessuno.

### Valori di ritorno

`String` che contiene il messaggio di eccezione.

## Classi secondarie di eccezione

All'interno della classe `ODKException` vi sono classi secondarie che identificano particolare eccezioni possibili nei metodi di API ODK. Tabella 108 elenca le eccezioni delle classi secondarie.

*Tabella 108. Classi secondarie ODKException.*

| Classi secondarie di eccezioni | Definizione |
|---|---|
| `BusObjInvalidAttrException` | Generato quando un attributo non è valido. |
| `BusObjInvalidDefException` | Generato quando una definizione di oggetti business non è valida. |
| `BusObjInvalidVerbException` | Generato quando un verbo non è valido. |
| `BusObjNoSuchAttrException` | Generato quando un attributo non esiste nella definizione di oggetti business. |
| `BusObjNoSuchVerbException` | Generato quando il verbo non viene supportato dalla definizione di oggetti business. |
| `ODKInvalidNodeException` | Generato per indicare un'eccezione di nodo tree. |
| `ODKInvalidPropException` | Generato per indicare un'eccezione causata da una proprietà non valida. |
| `UnsupportedContentException` | Viene generato se l'ODA non può supportare il contenuto generato richiesto. |

# Capitolo 25. Classe ODKUtility

Il L'API ODK (Object Discovery Agent Development Kit) fornisce la classe `ODKUtility` per fornire un ODA (Object Discovery Agent) con vari metodi di utility. Ottenendo un handle per un oggetto singleton `ODKUtility`, l'ODA deve poter accedere a:

- Informazioni nella memoria del runtime di ODA:
  - Valori specificati dall'utente per le proprietà di configurazione ODA
  - Valori specificati dall'utente per le proprietà di oggetti business
- Metodi di utility che forniscono le seguenti funzioni:
  - Capacità di inviare messaggi che richiedono immissioni utente
  - Capacità di inviare messaggi di stato non bloccanti
  - Capacità di fornire la traccia

**Nota:** Utilizzare il metodo `getODKUtility()` in questa classe per ottenere un identificativo di file per l'oggetto `ODKUtility`.

Tabella 109 riepilogaa i metodi nella classe `ODKUtility`.

*Tabella 109. Metodi del membro della classe ODKUtility.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| `contentComplete()` | Notifica la Procedura guidata dell'oggetto Business che l'ODA ha completato la generazione del contenuto durante l'utilizzo del protocollo callback. | 278 |
| `getAgentProperty()` | Richiama la proprietà di configurazione ODA specificata. | 278 |
| `getAllAgentProperties()` | Richiama tutte le proprietà di configurazione ODA. | 279 |
| `getAllBOSpecificProperties()` | Richiama tutte le proprietà di oggetti business. | 279 |
| `getBOSpecificProperty()` | Richiama le proprietà di oggetti business specificati. | 280 |
| `getBOSpecificProps()` | Invia le proprietà di oggetti business specificati alla casella di dialogo Proprietà per l'immissione utente. | 281 |
| `getClientFile()` | Richiede che la Procedura guidata dell'oggetto Business richiami un fule specifico. | 282 |
| `getMsg()` | Restituisce un messaggio dal file di messaggi ODA. | 283 |
| `getODKUtility()` | restituisce un identificativo di file da un oggetto ODKUtility. | 284 |
| `sendMsg()` | Visualizza una casella di dialogo di risposta utente, che include un messaggio con un pulsante e richiede una risposta da parte dell'utente. | 284 |
| `sendStatusMsg()` | Visualizza un messaggio all'utente. | 286 |
| `trace()` | Scrive un messaggio nel file di traccia. | 286 |

## contentComplete()

Notifica la Procedura guidata dell'oggetto Business che l'ODA ha completato la generazione del contenuto per il protocollo di contenuto callback.

### Sintassi

```
public void contentComplete(ContentMetaData contentMetaData);
```

### Parametri

*contentMetaData*
: Si tratta di un oggetto di metadati di contenuto che descrive lo stato corrente del contenuto generato.

### Valori di ritorno

Nessuno.

### Note

Il metodo `contentComplete()` indica che l'ODA ha completato la generazione del relativo contenuto. Nel protocollo callback, la Procedura guidata dell'oggetto Business *non* inizia la generazione del contenuto richiamando il metodo di generazione contenuto appropriata. L'ODA inizia la generazione del contenuto e la Procedura guidata dell'oggetto Business attende che l'ODA notifichi quando la generazione del contenuto è completa. L'ODA effettua questa notifica richiamando `contentComplete()`. Una volta che la Procedura guidata dell'oggetto Business è stata notificata, richiama il metodo di richiamo contenuto appropriato per ottenere il contenuto generato.

**Importante**

Se l'ODA utilizza il protocollo callback per una generazione di contenuto particolare, *deve* richiamare il metodo `contentComplete()` per notificare alla Procedura guidata dell'oggetto Business che il contenuto è disponibile. Altrimenti, la Procedura guidata dell'oggetto Business non sa che il contenuto generato è disponibile per essere recuperato.

L'oggetto *contentMetaData* deve indicare il tipo di contenuto generato così come il numero di voci nel contenuto generato.

## getAgentProperty()

Richiama l' proprietà di configurazione ODA.

### Sintassi

```
public AgentProperty getAgentProperty(String propName);
```

### Parametri

*propName* È il nome della proprietà di configurazione da richiamare.

## Valori di ritorno

Un oggetto `AgentProperty` che contiene la proprietà di configurazione specificata oppure `null` se non esiste alcuna proprietà di configurazione con quel nome.

## Note

Il metodo `getAgentProperty()` richiama la proprietà di configurazione *propName* dalla memoria runtime di ODA. La Procedura guidata dell'oggetto Business legge le proprietà di configurazione nella memoria runtime di ODA dopo che l'utente specifica i valori della proprietà di configurazione nella casella di dialogo Configura agente. Questo metodo restituisce la proprietà di configurazione specificata come un oggetto `AgentProperty`. È possibile ottenere informazioni sulla proprietà accedendo alle variabili membro dell'oggetto.

## Vedere anche

getAgentProperties(), getAllAgentProperties()

---

# getAllAgentProperties()

Richiama tutte le proprietà ODA.

## Sintassi

```
public Hashtable getAllAgentProperties();
```

## Parametri

Nessuno.

## Valori di ritorno

Un riferimento all'oggetto `java.util.Hashtable` che contiene le proprietà di configurazione ODA (rappresentate come oggetti `AgentProperty`) fissate sul nome della proprietà.

## Note

Il metodo `getAllAgentProperties()` richiama tutte le proprietà di configurazione ODA dalla memoria runtime di ODA. La Procedura guidata dell'oggetto Business legge le proprietà di configurazione nella memoria runtime di ODA dopo che l'utente specifica i valori della proprietà di configurazione nella casella di dialogo Configura agente. Questo metodo richiama le proprietà di configurazione come un oggetto `Hashtable` che esegue la mappatura di chiavi con i valori. Le chiavi sono dei nomi delle proprietà e dei valori che sono associati ai valori delle proprietà. Utilizzare i metodi della classe `Hashtable` (quali `keys()` e `elements()`) per ottenere le informazioni da questa struttura.

## Vedere anche

getAgentProperties(), getAgentProperty()

---

# getAllBOSpecificProperties()

Richiama tutte le proprietà degli oggetti business dalla casella di dialogo Proprietà BO.

## Sintassi

```
public Hashtable getAllBOSpecificProperties();
```

## Parametri

Nessuno.

## Valori di ritorno

Un riferimento ad un oggetto `java.util.Hashtable` che contiene le proprietà di oggetti business (rappresentate come oggetto `AgentProperty`), fissate sul nome della proprietà.

## Note

Il metodo `getAllBOSpecificProperties()` richiama tutte le proprietà degli oggetti business dalla memoria di runtime di ODA. La Procedura guidata dell'oggetto Business salva tali proprietà nella memoria dopo che l'utente specifica i relativi valori nella casella di dialogo Proprietà BO (parte del passo 5). Questo metodo restituisce le proprietà di oggetti business come un oggetto `Hashtable` che esegue la mappatura delle chiavi con i valori. Le chiavi sono dei nomi delle proprietà degli oggetti business e dei valori che sono associati ai valori delle proprietà. Utilizzare i metodi della classe `Hashtable` (quali `keys()` e `elements()`) per ottenere le informazioni da questa struttura.

## Vedere anche

getBOSpecificProperty(), getBOSpecificProps()

---

# getBOSpecificProperty()

Richiama l' proprietà dell'oggetto business.

## Sintassi

```
public AgentProperty getBOSpecificProperty(String propName);
```

## Parametri

*propName* È il nome della proprietà di oggetti business da richiamare.

## Valori di ritorno

Un oggetto `AgentProperty` che contiene la proprietà di oggetti business specificato oppure `null` se non esiste alcuna proprietà di oggetti business con quel nome.

## Note

Il metodo `getBOSpecificProperty()` richiama la proprietà di oggetti business *propName*dalla memoria runtime di ODA. La Procedura guidata dell'oggetto Business salva tali proprietà nella memoria dopo che l'utente specifica i relativi valori nella casella di dialogo Proprietà BO (parte del passo 5). Questo metodo restituisce la proprietà di oggetti business come un oggetto `AgentProperty`. È possibile ottenere informazioni sulla proprietà accedendo alle variabili membro dell'oggetto.

## Vedere anche

getAllBOSpecificProperties(), getBOSpecificProps()

## getBOSpecificProps()

Invia le proprietà di oggetti business specificati alla casella di dialogo Proprietà per l'immissione utente.

### Sintassi

```
public Hashtable getBOSpecificProps(AgentProperty[] properties,
   String titleBarText);
public Hashtable getBOSpecificProps(AgentProperty[] properties,
   String titleBarText, String propGridText);
```

### Parametri

*proprietà* È un array delle proprietà di oggetti business, ciascuna proprietà in un oggetto`AgentProperty`.

*titleBarText* È il testo da visualizzare nella barra del titolo della casella di dialogo Proprietà BO.

*propGridText* È il testo da visualizzare nell'area di testo al di sopra della griglia delle proprietà della casella di dialogo Proprietà BO.

### Valori di ritorno

Un oggetto `Hashtable` Java delle proprietà degli oggetti business (quali oggetti `AgentProperty`) fissati sul nome della proprietà.

### Eccezioni

**`ODKInvalidPropException`**
Viene riportato se la proprietà non è valida— ad esempio, se non dispone di un nome.

**`XMLException`** Viene generato se la conversione XML delle proprietà non ha esito positivo.

### Note

Il metodo `getBOSpecificProps()` invia l'array delle *proprietà* degli oggetti business a Procedura guidata dell'oggetto Business, che le visualizza nella casella di dialogo Proprietà BO. Da questa casella di dialogo l'utente può immettere o modificare questi valori delle proprietà. Prima di richiamare il metodo `getBOSpecificProps()` occorre eseguire l'istanza e inizializzare delle proprietà agente (`AgentProperty`) per ciascuna proprietà di oggetti business e memorizzare ciascuna proprietà nell'array delle proprietà degli oggetti business *proprietà*. Il metodo `getBOSpecificProps()` passa quest'array delle proprietà degli oggetti business al relativo programma di chiamata, Procedura guidata dell'oggetto Business.

Una volta che l'utente ha impostato le proprietà degli oggetti business dalla casella di dialogo Proprietà BO, la Procedura guidata dell'oggetto Business salva tali proprietà specifiche dell'utente in un oggetto `java.util.Hashtable` e la memoria runtime ODA. All'interno di ODA, è possibile ottenere le proprietà inizializzate dall'utente in uno dei seguenti modi:

- Dalla memoria runtime di ODA

  Utilizzare il metodo `getBOSpecificProperty()` oppure `getAllBOSpecificProperties()` nella classe `ODKUtility`. I valori inizializzati dall'utente per la proprietà sono nella variabile membro `allValues` del relativo oggetto proprietà agente `AgentProperty`).

- Dall'oggetto `Hashtable` restituito da `getBOSpecificProps()`
  Utilizzare il metodo dell'oggetto `Hashtable` per ottenere le proprietà dell'agente.

Per ulteriori informazioni su come utilizzare `getBOSpecificProps()`, consultare "Richiesta di proprietà di oggetti business" a pagina 130.

## Vedere anche

getAllBOSpecificProperties(), getBOSpecificProperty()

---

# getClientFile()

Richiede che Procedura guidata dell'oggetto Business richiami il file specificato.

## Sintassi

```
public byte[] getClientFile(String srcNodePath, ODKAgentBase2 ODAobj);
```

## Parametri

*srcNodePath* È il percorso del nodo di origine del file da richiedere da Procedura guidata dell'oggetto Business.

*ODAobj* È l'oggetto ODA (`ODKAgentBase2`), che viene utilizzato per verificare che ODA è autorizzato per effettuare l'operazione, cioè che l'ODA genera il contenuto del file.

## Valori di ritorno

I contenuti del file del sistema operativo specificato, come un array di byte.

## Eccezioni

**`UnsupportedContentException`**
Viene riportato se ODA *non* supporta la generazione del contenuto di file, cioè non implementa l'interfaccia `IGeneratesBinFiles`.

**`Java.io.IOException`**
Viene riportato se si verifica un errore durante il richiamo di file, ad esempio il file non è stato trovato.

## Note

Il metodo `getClientFile()` richiede che la Procedura guidata dell'oggetto Business restituisca i contenuti del file del sistema operativo identificato da *srcNodePath*. Questo percorso *srcNodePath* assume il seguente formato:

*`fileNodePath`*`:`*`fileLocation`*

dove:

- *fileNodePath* è l'elenco separato dai due punti (:) dei nomi di nodi di origine per il nodo associato con il file. Ad esempio, `Apollo:Vulso:Flavius.xml`
- *fileLocation* è il percorso completo del sistema operativo al file. Ad esempio, `C:\temp\XMLFiles\Flavius.xml`

Utilizzare il metodo `getClientFile()` per accedere ad un file associato agli oggetti rappresentati dai nodi di origine. Se un nodo di origine può essere associato ad un file, la capacità di interpretare il percorso del nodo di origine del file e di leggerne il contenuto è necessaria in *entrambi* i seguenti punti:

- Durante la generazione di nodi di origine, il metodo `getTreeNodes()` deve essere capace di "scoprire" un nodo secondario che si trova in un file.
- Durante la generazione del contenuto il metodo che genera il contenuto deve essere capace di accedere a informazioni nei nodi che sono in un file.

Per ulteriori informazioni, consultare "Lettura di file per i dati di origine" a pagina 146.

# getMsg()

Richiama un messaggio dal file di messaggi ODA.

## Sintassi

```
public String getMsg(int msgNum, int msgType);
public String getMsg(int msgNum, int msgType, msgParameters);
public String getMsg(int msgNum, int msgType, Vector paramArray);
```

## Parametri

*msgNum* Specifica il numero di messaggio dal file di messaggio.

*msgParameters* È un elenco opzionale di massimo tre valori di parametro `String` , ciascuno corrispondente ad un parametro nell'elenco di messaggi.

*msgType* È il tipo di messaggio specificato come una delle seguenti costanti del tipo di messaggio:

```
ODKConstant.XRD_FATAL
ODKConstant.XRD_ERROR
ODKConstant.XRD_URGENTWARNING
ODKConstant.XRD_WARNING
ODKConstant.XRD_INFO
```

*paramArray* È un elenco opzionale di parametri come un vettore Java `Vector` da inserire nei parametri del messaggio.

## Eccezioni

**`IllegalArgumentException`**
Viene riportato se l'argomento *msgType* non è valido.

## Valori di ritorno

`String` che contiene il testo associato con il numero di messaggio specifico. Se sono stati forniti i parametri del messaggio, tali valori sono stati inseriti come appropriati nel messaggio. Se *msgNum* non è valido, il metodo restituisce `null`.

## Note

Il metodo `getMsg()` richiama un messaggio da un file di messaggi. Identifica il nome di questo file dalla proprietà di avvio `MessageFile` che ODK include automaticamente con le proprietà di avvio ODA. Il metodo `getMsg()` fornisce i seguenti moduli:

- Il primo modulo richiama un messaggio con un numero specificato di messaggio *msgNum*) dal file di messaggi ODA.
- Il secondo modulo richiama anche il messaggio con il numero di messaggio specificato (*msgNum*) dal file di messaggi ODA. Fornisce anche la capacità di specificare fino a tre parametri di messaggi `String` (*msgParameters*) da inserire nel messaggio prima di richiamarlo.

- Il terzo modulo invia anche un messaggio dal file di messaggi ODA e fornisce parametri di messaggi. Tuttavia, con questo modulo è possibile inviare i parametri di messaggi come elementi in un vettore Java , *paramArray*.

Per informazioni sui file di messaggi ODA; consultare "File di messaggi" a pagina 166. Per informazioni sui parametri dei messaggi, consultare "Utilizzo dei valori dei parametri" a pagina 169.

### Vedere anche

trace()

## getODKUtility()

Restituisce un identificativo di file da un oggetto singleton `ODKUtility`.

### Sintassi

```
public static ODKUtility getODKUtility();
```

### Parametri

Nessuno.

### Valori di ritorno

Un identificativo di file ad un oggetto `ODKUtility`.

### Note

Il metodo `getODKUtility()` fornisce accesso all'interno del codice ODA alle utility nella classe `ODKUtility`. Occorre utilizzare `getODKUtility()` per ottenere un identificativo di file all'oggetto singleton di questa classe *prima* di accedere ai metodi `ODKUtility`.

**Nota:** La chiamata a `getODKUtility()` viene effettuata spesso nel metodo `getAgentProperties()`. Per ulteriori informazioni, consultare "Ottenimento dell'oggetto ODKUtility" a pagina 111.

### Vedere anche

getAgentProperties()

## sendMsg()

Visualizza una casella di dialogo di risposta utente che comprende un messaggi con un pulsante e richiede una risposta da parte dell'utente.

### Sintassi

```
public int sendMsg(String msg, int dialogFlags);
```

### Parametri

| | |
|---|---|
| *msg* | È il messaggio da visualizzare nella casella di dialogo di risposta utente. |
| *dialogFlags* | È un insieme di indicatori che segnalano i pulsanti e le icone da visualizzare come parte della casella di dialogo di risposta utente. |

Indica tali pulsanti e icone come una maschera dei contenuti del dialogo di risposta utente illustrati in Tabella 110.

## Valori di ritorno

Un numero intero che segnala il pulsante su cui l'utente ha fatto clic per terminare la casella di dialogo di risposta utente. Confrontare questo valore intero con le seguenti costanti di risposta utente:

`ODKConstant.ODK_OK`
L'utente ha selezionato il pulsante OK.

`ODKConstant.ODK_CANCEL`
L'utente ha selezionato il pulsante Annulla.

`ODKConstant.ODK_RETRY`
L'utente ha selezionato il pulsante Riprova.

`ODKConstant.ODK_IGNORE`
L'utente ha selezionato il pulsante Ignora.

`ODKConstant.ODK_ABORT`
L'utente ha selezionato il pulsante Interrompi.

`ODKConstant.ODK_YES`
L'utente ha selezionato il pulsante Sì.

`ODKConstant.ODK_NO`
L'utente ha selezionato il pulsante No.

`ODKConstant.ODK_CLOSE`
L'utente ha selezionato il pulsante Chiudi.

`ODKConstant.ODK_HELP`
L'utente ha selezionato il pulsante ?.

## Note

Il metodo `sendMsg()` invia una richiesta alla Procedura guidata dell'oggetto Business per visualizzare una casella di dialogo di risposta utente all'utente. Specificare i seguenti componenti della casella di dialogo di risposta utente:

- La stringa *msg* contiene il testo per indicare la condizione, la domanda o le informazioni che occorrono che devono essere visualizzate all'utente.
- Una maschera *dialogFlags* contiene le funzioni che descrivono l'aspetto della casella di dialogo di risposta utente nel modo seguente:
  - Il pulsante da visualizzare

    L'utente fa clic su uno di questi pulsanti per terminare la casella di dialogo di risposta utente. Specificare questi pulsanti con le costanti del pulsante di dialogo, nella sezione "Pulsanti da visualizzare" di Tabella 110.
  - Le icone da visualizzare

    L'icona determina il tipo di casella di dialogo di risposta utente da visualizzare. Specificare il tipo di casella di dialogo con una delle costanti delle icone di dialogo, nella sezione "icone della casella di dialogo da visualizzare" di Tabella 110.

*Tabella 110. Visualizza l'aspetto della casella di dialogo di risposta utente*

| Aspetto della casella di dialogo di risposta utente | costante della casella di dialogo di risposta utente ODKConstant |
|---|---|
| Pulsanti da visualizzare: | |

*Tabella 110. Visualizza l'aspetto della casella di dialogo di risposta utente (Continua)*

| Aspetto della casella di dialogo di risposta utente | costante della casella di dialogo di risposta utente ODKConstant |
|---|---|
| OK | MSG_OK |
| OK, CANCEL | MSG_OKCANCEL |
| RETRY, CANCEL | MSG_RETRYCANCEL |
| RETRY, IGNORE, ABORT | MSG_ABORTRETRYIGNORE |
| YES, NO | MSG_YESNO |
| YES, NO, CANCEL | MSG_YESNOCANCEL |
| icona della casella di dialogo da visualizzare: | |
| Icona di errore | MSG_ERROR |
| Icona di errore critico | MSG_CRITICALERROR |
| Icona di avvertenza | MSG_WARNING |
| Icona di informazioni | MSG_INFORMATION |
| Icona con punto interrogativo | MSG_QUESTION |

**Nota:** Tutte le costanti delle caselle di dialogo di risposta utente in Tabella 110 vengono definite nell'interfaccia `ODKConstant`.

Per specificare l'argomento *dialogFlags*, creare una maschera delle costanti di dialogo di risposta utente che descrivano l'aspetto della casella di dialogo di risposta utente. Ad esempio, la seguente chiamata a `sendMsg()` crea una casella di dialogo di risposta utente che visualizza un'icona di errore così come i pulsanti Riprova o Annulla:

```
String msg = new String(bdkUtil.getMsg(
   1002, ODKConstant.XRD_ERROR, params));
bdkUtil.sendMsg(msg,
   ODKConstant.MSG_RETRYCANCEL | ODKConstant.MSG_ERROR);
```

### Vedere anche

sendStatusMsg()

## sendStatusMsg()

Visualizza un messaggio all'utente.

### Sintassi

```
public void sendStatusMsg(String msg);
```

### Parametri

*msg* È il messaggio da inviare all'utente.

### Valori di ritorno

Nessuno.

### Vedere anche

sendMsg()

## trace()

Scrive un messaggio nel file di traccia.

## Sintassi

```
public void trace(int level, int msgType, String message);

public void trace(int level, int msgNum, int msgType);
public void trace(int level, int msgNum, int msgType, msgParameters);
public void trace(int level, int msgNum, int msgType, Vector paramArray);

public void trace(int level, int msgType, BusObjDef boDef);

public void trace(int level, int msgType, AgentProperty[] properties,
   String foreword);
```

## Parametri

*boDef* È la definizione di oggetti business da scrivere nel file di traccia.

*foreword* È String che chiarifica il messaggio prima dell'array di proprietà *properties*— ad esempio, "Queste sono le proprietà per Object Discovery Agent".

*level* È il livello di traccia specificato come una delle seguenti costanti dei livelli di traccia:

```
ODKConstant.TRACELEVEL0
ODKConstant.TRACELEVEL1
ODKConstant.TRACELEVEL2
ODKConstant.TRACELEVEL3
ODKConstant.TRACELEVEL4
ODKConstant.TRACELEVEL5
```

*message* È il messaggio String da scrivere nel file di traccia.

*msgNum* Specifica il numero di messaggio nel file di messaggio.

*msgParameters* È un elenco opzionale di massimo tre valori di parametro String , ciascuno corrispondente ad un parametro nell'elenco di messaggi.

*msgType* È il tipo di messaggio specificato come una delle seguenti costanti del tipo di messaggio :

```
ODKConstant.XRD_FATAL
ODKConstant.XRD_ERROR
ODKConstant.XRD_URGENTWARNING
ODKConstant.XRD_WARNING
ODKConstant.XRD_INFO
ODKConstant.XRD_TRACE
```

*paramArray* Un vettore dei parametri da inserire nel messaggio.

*proprietà* È un array degli oggetti di proprietà agente (AgentProperty) da scrivere nel file di traccia.

## Valori di ritorno

Nessuno.

## Eccezioni

**IllegalArgumentException**
Viene riportato se l'argomento *properties* è null oppure l'argomento *msgType* non è valido.

## Note

Il metodo trace() invia le informazioni specificate nel file di traccia quando il *livello* di traccia è minore o uguale al livello di traccia del sistema di livello di traccia. Il livello di traccia del sistema viene impostato mediante la proprietà di

configurazione TraceLevel, che la Procedura guidata dell'oggetto Business automaticamente include nelle proprietà di configurazione ODA. Un *livello* di traccia pari a zero (0) attiva il log degli errori, cioè trace() invia un messaggio di errore al file di traccia. Un livello di traccia diverso da zero, illustrato in Tabella 111, attiva la traccia, cioè trace() invia un messaggio di traccia al file di traccia.

*Tabella 111. Livelli di traccia per un ODA*

| Livello di traccia | Descrizione | Costante dei livelli di traccia |
|---|---|---|
| 0 | Registrazione di un messaggio di errore. | TRACELEVEL0 |
| 1 | Effettua la traccia ogni qualvolta viene immesso un metodo. Di solito fornisce messaggi di metodo e informazioni chiave per ciascuna definizione di oggetti business. | TRACELEVEL1 |
| 2 | Effettua la traccia delle proprietà dell'agente e i valori ricevuti. | TRACELEVEL2 |
| 3 | Effettua la traccia del nome della definizione di oggetti business. Di solito fornisce le proprietà di oggetti business e i valori ricevuti. | TRACELEVEL3 |
| 4 | Effettua la traccia di un messaggio ogni qualvolta viene immesso un metodo e viene terminato. Registra la creazione di tutti i thread. | TRACELEVEL4 |
| 5 | Indica l'inizializzazione ODA. Fornisce i valori per tutte le proprietà agente richiamate, uno stato dettagliato di ciascun thread creato da ODA e un dump della definizione di oggetti business. | TRACELEVEL5 |

L'utente stabilisce il nome della destinazione di traccia ODA mediante la proprietà di configurazione TraceFileName che ODK include automaticamente con le proprietà di avvio ODA. Pertanto, la traccia non può iniziare fino a dopo che viene avviato il metodo init() (che riceve le proprietà di avvio inizializzate).

Il metodo trace() fornisce i seguenti moduli:

- I primi quattro moduli inviano un messaggio di testo al file di traccia:
  - I primi quattro moduli inviano il *messaggio* di testo specificato per il file di traccia.
  - Il secondo modulo richiama il messaggio con il numero di messaggio specificato (*msgNum*) dal file di messaggi ODA.
  - Il terzo modulo invia anche il messaggio con il numero di messaggio specificato (*msgNum*) dal file di messaggi ODA. Fornisce anche la capacità di inviare fino a tre parametri di messaggi String (*msgParameters*) da inserire nel messaggio prima di inviarlo alla destinazione di traccia.
  - Il quarto modulo invia anche un messaggio dal file di messaggi ODA e fornisce parametri di messaggi. Tuttavia, con questo modulo è possibile inviare i parametri di messaggi come elementi in un vettore Java , *paramArray*.

  Per informazioni sui file di messaggi ODA; consultare "File di messaggi" a pagina 166. Per informazioni sui parametri dei messaggi, consultare "Utilizzo dei valori dei parametri" a pagina 169.
- Il quarto modulo invia un dump della definizione di oggetti business al file di traccia. Questo dump viene formattato nel formato della utility repos_copy e ha il seguente formato di base:

```
[BusinessObjectDefinition]
Name=busObjName
AppSpecificInfo=business-object-level application-specific information
[Attribute]
Name=attribute1
Type=attribute type
Cardinality=n or 1
AppSpecificInfo=attribute-level application-specific information
other attribute properties
[End]
...
```

- Il sesto modulo invia un dump delle *proprietà* dell'agente specificato al file di traccia. Questo dump. L'argomento *forward* fornisce il testo introduttivo che chiarifica il messaggio.

### Vedere anche

getMsg()

## Metodi non approvati

Alcuni metodi nella classe `ODKUtility` erano supportate nelle versioni precedenti ma non sono più supportate. Tali *metodi non approvati* non genereranno errori, ma IBM ne sconsiglia l'utilizzo e consiglia di migrare il codice esistente ai nuovi metodi. I metodi non approvati potrebbero essere eliminati in un futuro release.

Tabella 112 elenca i metodi non approvati per la classe `ODKUtility`. Se si sta scrivendo un nuovo ODA (non modificando l'ODA esistente), è possibile ignorare questa sezione.

*Tabella 112. Metodi non approvati della classe ODKUtility*

| Metodo non approvato | Metodo sostitutivo |
|---|---|
| Tutti i metodi che supportano operazioni di filtro:<br>• `filterData()`<br>• `getFilter()`<br>• `setFilter()` | Un ODA supporta ancora operazioni di filtro al livello utente, ma non a livello di programma. Per ulteriori informazioni, consultare "Uso de filtro" a pagina 85. A livello di programma, la funzione del modello di ricerca fornisce la capacità di ridurre il numero di nodi secondari per un nodo principale particolare. Per ulteriori informazioni, consultare "Implementazione della funzione del modello di ricerca" a pagina 124. |

# Capitolo 26. Classe TreeNode

Il API ODK (Object Discovery Agent Development Kit) fornisce la classe `TreeNode` per rappresentare i i nodi tree. ODA (Object Discovery Agent) genera un array di nodi tree affinché la Procedura guidata dell'oggetto Business possa visualizzare una gerarchia di nodi di origine all'utente. L'utente si sposta tra i nodi di questa gerarchia di nodi d'origine per selezionare gli oggetti le cui definizioni di oggetti business stanno per essere generate da ODA.

La classe `TreeNode` definisce i seguenti valori:

- "variabili membro"
- "Metodo" a pagina 294

La classe `TreeNode` implementa l'interfaccia `ODKConstant`. Pertanto, tutte le costanti definite in `ODKConstant` sono disponibili per un oggetto `TreeNode`. Per un elenco di costanti definite dall'interfaccia `ODKConstant` consultare Capitolo 23, "Interfaccia ODKConstant", a pagina 269.

## variabili membro

Tabella 113 riepiloga le variabili membro della classe `TreeNode`.

*Tabella 113. variabili membro della classe TreeNode.*

| variabile membro | Descrizione | Pagina |
|---|---|---|
| `descrizione` | Contiene una descrizione del nodo tree. | 291 |
| `isExpandable` | Specifica dse il nodo tree è espandibile, cioè se vi sono elementi al di sotto del livello corrente. | 291 |
| `isGeneratable` | Specifica se il nodo tree è generabile, cioè se il nodo può essere convertito in una definizione di oggetto business. | 292 |
| `nome` | Contiene il nome del nodo tree. | 292 |
| `nodi` | Contiene la gerarchia espansa dei nodi tree. | 292 |
| `polymorphicNature` | Definisce la natura del nodo, cioè se è "normale" (espandibile o leaf) oppure "file". | 293 |

### descrizione

Contiene una descrizione del nodo tree.

#### Tipo

```
public String description
```

#### Note

La variabile membro della `descrizione` visualizza la colonna Descrizione della casella di dialogo per la selezione dell'origine.

### isExpandable

Specifica se il nodo tree è espandibile, cioè se vi sono nodi al di sotto del livello corrente.

### Tipo
```
public boolean isExpandable
```

### Note
La variabile membro `isExpandable` indica se un nodo espanibile, come viene indicato da Tabella 114.

*Tabella 114. Tipi di nodi*

| Tipo di nodi | Descrizione | Valore di isExpandable |
|---|---|---|
| Espandibile | Il nodo ha nodi secondari | `true` |
| Leaf (terminale) | Il nodo non ha nodi secondari, ma è il punto terminale di un ramo della gerarchia di nodi d'origine. | `false` |

Solo i nodi di natura normale (nodi che hanno la variabile membro `polymorphicNature` impostata su `NODE_NATURE_NORMAL` ) possono avere `isExpandable` impostato su `true`.

## isGeneratable

Specifica se il nodo tree è generabile, cioè se l'utente può selezionare questo nodo come uno per il quale viene generato contenuto da ODA.

### Tipo
```
public boolean isGeneratable
```

## nome

Contiene il nome del nodo tree.

### Tipo
```
public String name
```

### Note
La variabile membro del `nome` visualizza la colonna Nome della casella di dialogo per la selezione dell'origine.

## nodi

Contiene la gerarchia espansa del nodi tree secondari.

### Tipo
```
public TreeNode[] nodes
```

### Note
La variabile membro `nodi` contiene un array degli oggetti `TreeNode`, un oggetto per ciascuno di questi valori secondari del nodo principale. Un nodo secondario può, a turno, contenere nodi secondari (grandchildren di questo nodo principale). Questo mmebro viene utilizzato solo se il nodo è espandibile (non uno leaf), cioè se la variabile membro `isExpandable` è `true`.

## polymorphicNature

Indica le azioni valideche l'utente può intraprendere sulle nodo tree.

### Tipo

```
public int polymorphicNature
```

### Note

La variabile membro `polymorphicNature` determina quali azioni l'utente può intraprendere sul nodo quando viene visualizzato nella casella di dialogo per la selezione dell'origine della Procedura guidata dell'oggetto Business. Questa variabile contiene una costante della natura del nodo intera per indicare la natura del nodo tree. Queste costanti della natura del nodo vengono definite nell'interfaccia `ODKConstant` come indicato da Tabella 115.

*Tabella 115. Natura del nodo tree*

| Natura del nodo tree | Descrizione | Costante della natura del nodo |
|---|---|---|
| Normale | L'utente può intraprendere una delle seguenti azioni:<br>• L'utente può selezionare il nodo, se il nodo è un nodo leaf (terminale). Solo i nodi leaf possono essere selezionati per essere generati nel contenuto.<br>• L'utente può espandere il nodo per visualizzare altri nodi. La Procedura guidata dell'oggetto Business visualizza un segno più (+) a sinistra del nome di un nodo espandibile. | `NODE_NATURE_NORMAL` |
| File | L'utente può associare un file dal file system locale con il nodo. La Procedura guidata dell'oggetto Business attiva la voce di menu **Associa file** nel menu a comparsa che viene visualizzato quando l'utente fa clic con il pulsante destro del mouse sul nome del nodo. Questa voce di menu apre una finestra per la visualizzazione dei file di sistema. Da questa finestra, l'utente può selezionare quale file associare con il nodo.<br><br>Per un nodo tree che ha una natura di nodo file, ODA può utilizzare il `metodo getClientFile()` (definito nella classe `ODKUtility` class) per ottenere i contenuti del file selezionato dall'utente. | `NODE_NATURE_FILE` |

**Nota:** Poiché la classe `TreeNode` implementa l'interfaccia `ODKConstant`, le costanti della natura del nodo sono disponibili per la variabile membro `polymorphicNature` senza essere qualificate con il nome `ODKConstant`.

Per ulteriori informazioni sulle nature dei nodi, consultare "Costruzione dei nodi tree" a pagina 126.

## Metodo

Tabella 116 riepiloga il metodo della classe TreeNode.

*Tabella 116. Il metodo del membro della claase TreeNode.*

| Metodo del membro | Descrizione | Pagina |
|---|---|---|
| TreeNode() | Crea l'oggetto del nodo tree. | pagina 291 |

# TreeNode()

Crea un oggetto del nodo tree.

### Sintassi

```
public TreeNode(String name, String desc, boolean isGen, boolean isExp);
public TreeNode(String name, String desc, boolean isGen, boolean isExp,
   TreeNode[] treeNodes);
public TreeNode(String name, String desc, boolean isGen, boolean isExp,
   TreeNode[] treeNodes, int nodeNature);
```

### Parametri

*desc* Specifica la descrizione del nodo: il valore di questo parametro inizializza la variabile membro della descrizione ("descrizione" a pagina 291).

*isGen* Specifica se il nodo è "generabile", cioè se il nodo può essere convertito in una definizione di oggetto business; il valore di questo parametro inizializza la variabile membro isGeneratable ("isGeneratable" a pagina 292).

*isExp* Specifica se il nodo è espandibile, cioè se il nodo è leaf o meno; il valore di questo parametro inizializza la variabile membro isExpandable ("isExpandable" a pagina 291).

*name* Specifica il nome del nodo; il valore di questo parametro inizializza la variabile membro name ("nome" a pagina 292).

*nodeNature* Indica la natura del nodo, come una delle seguenti costanti della natura del nodo:

```
ODKConstant.NODE_NATURE_FILE
ODKConstant.NODE_NATURE_NORMAL
```

*treeNodes* Specifica la gerarchia completa espansa dei nodi; il valore di questo parametro inizializza la variabile membro nodes ("nodi" a pagina 292).

### Valori di ritorno

Un oggetto creato come nuova istanza TreeNode.

### Note

Il metodo TreeNode() fornisce i seguenti moduli per creare l'istanza di un nodo tree:

- Il primo modulo del costruttore consente di specifica il nome e la descrizione del nodo tree ed anche se è generabile o espandibile. In tale modulo, l'array dei nodi secondari (la variabile nodes ) viene inizializzata su null e la natura del nodo (la variabile membro polymorphicNature) viene inizializzata su "normal". Utilizzare questo modulo per inizializzare un nodo leaf.

- Il secondo modulo del costruttore consente di specificare l'array dei nodi secondari (oltre ai valori specificati nel primo modulo). In questo modulo, la natura del nodo viene inizializzata su "normal". Utilizzare questo modulo per inizializzare un nodo espandibile.
- Il terzo modulo del costruttore consente di specificare la natura del nodo (oltre ai valori specificati nel primo e secondo modulo). Utilizzare questo modulo per inizializzare un nodo della natura file.

Per ulteriori informazioni, consultare "Costruzione dei nodi tree" a pagina 126.

# Informazioni particolari

Queste informazioni sono state sviluppate per prodotti e servizi offerti negli Stati Uniti. È possibile che negli altri paesi l'IBM non offra i prodotti, le funzioni o i servizi illustrati in questo documento. Consultare il rappresentante locale IBM per informazioni sui prodotti e sui servizi disponibili attualmente nel proprio paese. Qualunque riferimento relativo a prodotti, programmi o servizi IBM non implica che solo quei prodotti, programmi o servizi IBM possano essere utilizzati. In sostituzione a quelli forniti dall'IBM possono essere utilizzati prodotti, programmi o servizi funzionalmente equivalenti che non comportino violazione dei diritti di proprietà intellettuale e di altri diritti dell'IBM. È comunque responsabilità dell'utente valutare e verificare la possibilità di utilizzare altri programmi e/o prodotti, fatta eccezione per quelli espressamente indicati dall'IBM. L'IBM può avere brevetti o domande di brevetto in corso relativi a quanto trattato nel presente documento. La fornitura di questa pubblicazione non implica la concessione di alcuna licenza su di essi. E' possibile inviare domande di licenza, per iscritto a:

*IBM Director of Commercial Relations*
*IBM Corporation*
*North Castle Drive*
*Armonk, NY 10504-1785*
*Deutschland*

Per domande di autorizzazioni relative a informazioni DBCS, contattare IBM Intellectual Property Department nel proprio paese oppure inviare le domande a:

*IBM World Trade Asia Corporation Licensing*
*2-31 Roppongi 3-chome, Minato-ku*
*Tokyo 106-0032, Japan*

Il seguente paragrafo non è valido per il Regno Unito o per tutti i paesi le cui leggi nazionali siano in contrasto con le disposizioni in esso contenute: L'INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE QUESTA PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA", SENZA ALCUNA GARANZIA, ESPLICITA O IMPLICITA, IVI INCLUSE EVENTUALI GARANZIE DI COMMERCIABILITA' ED IDONEITA' AD UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni; quindi la presente dichiarazione potrebbe essere non essere a voi applicabile. Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le informazioni incluse in questo documento vengono modificate su base periodica; tali modifiche verranno incorporate nelle nuove edizioni della pubblicazione. L'IBM si riserva il diritto di apportare miglioramenti e/o modifiche al prodotto o al programma descritto nel manuale in qualsiasi momento e senza preavviso. Tutti i riferimenti a siti Web non dell'IBM contenuti in questo documento sono forniti solo per consultazione e non rappresentano in alcun modo un'approvazione di tali siti Web. I materiali disponibile presso i siti Web non fanno parte di questo prodotto e l'utilizzo di questi è a discrezione dell'utente. Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente dall'IBM e diventeranno esclusiva della stessa. Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire: (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

*IBM Corporation*
*577 Airport Blvd., Suite 800*
*Burlingame, CA 94010*
*U.S.A*

Tali informazioni possono essere disponibili, in base ad appropriate clausole e condizioni, includendo in alcuni casi, il pagamento di una tassa. Il programma su licenza descritto in questo manuale e tutto il materiale su licenza ad esso relativo sono forniti dall'IBM nel rispetto delle condizioni previste dalla licenza d'uso. Tutti i dati contenuti in questa pubblicazione sono stati determinati in ambiente controllato. Pertanto, i risultati ottenuti in ambienti operativi diversi possono variare in modo considerevole. Alcune misurazioni potrebbero essere state fatte su sistemi di livello di sviluppo per cui non si garantisce che queste saranno uguali su tutti i sistemi disponibili. Inoltre, alcune misurazioni possono essere state stimate tramite estrapolazione. I risultati attuali possono quindi variare. Gli utenti del presente documento dovranno verificare i dati applicabili per i propri ambienti specifici. Le informazioni relative a prodotti non-IBM sono state ottenute dai fornitori di tali prodotti. L'IBM non ha verificato tali prodotti e, pertanto, non può garantirne l'accuratezza delle prestazioni. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti. Tutti le dichiarazioni riguardanti la direzione o le decisioni future di IBM sono soggette a variazione o ritiro senza preavviso e costituiscono solo degli obiettivi. Questa pubblicazione contiene esempi di dati e prospetti utilizzati quotidianamente nelle operazioni aziendali. Pertanto, può contenere nomi di persone, società, marchi e prodotti. Tutti i nomi contenuti nella pubblicazione sono fittizi e ogni riferimento a nomi e indirizzi reali è puramente casuale. LICENZA DI COPYRIGHT: Queste informazioni contengono programmi applicativi di esempio in lingua originale, che illustrano le tecniche di programmazione su diverse piatteforme operative. E' possibile copiare, modificare e distribuire queste esempi di programmi sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in modo conforme alle API (Application Programming Interface) a seconda della piattaforma operativa per cui gli esempi dei programmi sono stati scritti. Questi esempi non sono stati testati approfonditamente tenendo conto di tutte le condizioni possibili. La IBM, quindi, non può garantire o assicurare l'affidabilità, la praticità o il funzionamento di questi programmi. Se si stanno visualizzando queste informazioni come softcopy, 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 all'utente di scrivere il software di applicazione che ottiene i servizi di questi strumenti del programma'. Tuttavia, queste informazioni possono contenere anche diagnosi, modifiche ed informazioni sull'ottimizzazione. Le informazioni sulle diagnosi, le modifiche e sull'ottimizzazione vengono fornite come supporto per l'esecuzione del debug del software applicativo.

**Avvertenza:** Non utilizzare queste informazioni sulle diagnosi, le modifiche e sull'ottimizzazione come un'interfaccia di programmazione poiché sono soggette a cambiamenti.

## Marchi e marchi di servizio

I seguenti termini sono marchi della International Business Machines Corporation negli Stati Uniti e/o in altre nazioni: 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 ed il logo Windows sono marchi della Microsoft Corporation negli Stati Uniti e/o in altri paesi. MMX, Pentium e ProShare sono marchi o marchi registrati della Intel Corporation negli Stati Uniti e/o negli altri paesi. Java e tutti i marchi basati su Java sono marchi della Sun Microsystems, Inc. negli Stati Uniti e/o negli altri paesi. Linux è un marchio della Linus Torvalds negli Stati Uniti e/o negli altri paesi. Altri nomi di società, di servizi e prodotti possono essere marchi o marchi di servizi di altre società.

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


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

# Indice analitico

## Caratteri speciali



## A



## B



## C

# D



# E

# F



# G



# I

## J



## L



## M

# N



# O

# P



# R

# S



# T



# V

Printed in Denmark by IBM Danmark A/S