Generare un rich client Java che utilizza un servizio Web

Indice

Generazione di un rich client Java che utilizza un servizio Web

Questa esercitazione mostra come utilizzare l'editor visivo Java per generare un rich client che si connette al servizio Web. Il client generato nell'esercitazione è chiamato My Company Directory.

My Company Directory è un'applicazione Java utilizzata per gestire le informazioni relative ai dipendenti di un'azienda. L'applicazione si connette a un servizio Web di esempio che fornisce i metodi per la creazione, il richiamo, l'aggiornamento e l'eliminazione dei record dei dipendenti.

Il client viene generato visivamente nell'editor visivo Java utilizzando i componenti Swing. L'editor visivo Java fornisce un insieme di classi helper (origini dati, oggetti dati e binder) per connettersi e poter utilizzare un servizio Web. Il servizio Web viene distribuito localmente sull'installazione di IBM WebSphere Application Server v6.0 e gli strumenti forniti consentono di generare un proxy Java per il client basato su un file WSDL (Web Services Description Language).

Visualizzail prodotto finito

Obiettivi

In questa esercitazione verranno completate le seguenti attività:

- Utilizzo dell'editor visivo Java per progettare e organizzare il layout di un'interfaccia utente
- Associazione di elementi di interfaccia agli oggetti dati e a un servizio Web

Tempo richiesto

2 ore e 15 minuti

Informazioni correlate

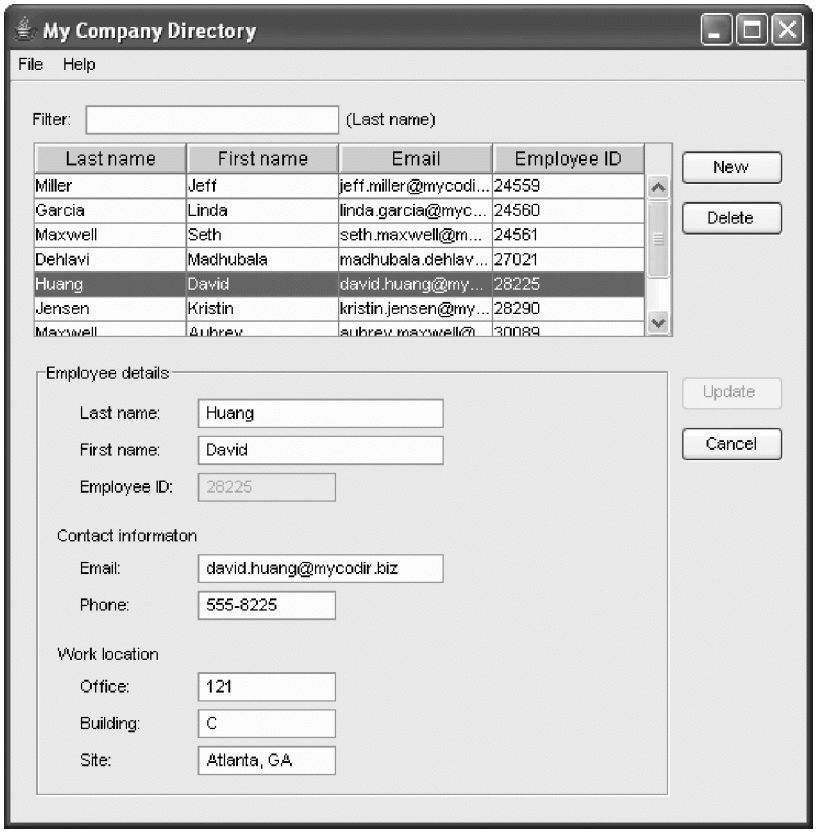
Versione PDF

Esercitazione: Hello World Java

Introduzione: Generazione di un rich client Java che utilizza un servizio Web

La GUI (Graphical User Interface) per il client è stata in parte già generata visivamente, utilizzando i componenti Swing. Nel primo modulo, verrà terminata la disposizione dei componenti principali della GUI utilizzando l'editor visivo Java. Nel secondo modulo, verrà eseguito il collegamento dei componenti GUI all'origine dati del servizio Web, ai servizi e agli oggetti restituiti dall'origine dati. Durante la generazione dell'applicazione nell'editor visivo Java, verranno utilizzati origini dati, oggetti dati e binder che rappresentano le classi helper generate dall'editor visivo Java e utilizzate dall'applicazione.

Visualizza un'immagine del prodotto finito:



Obiettivi

In questa esercitazione verranno completate le seguenti attività:

- Utilizzo dell'editor visivo Java per progettare e organizzare il layout di un'interfaccia utente
- Associazione di elementi di interfaccia agli oggetti dati e a un servizio Web

Tempo richiesto

Per completare l'intera esercitazione, saranno necessarie circa 2 ore e 30 minuti.

Requisiti del sistema

- WebSphere Application Server v6.1. Questo server può anche essere installato con il prodotto, o è possibile utilizzare l'installazione autonoma. Lo scenario in questa esercitazione chiede di distribuire un servizio Web di esempio su un WebSphere Application Server in esecuzione localmente.

Il servizio Web di esempio potrebbe essere in esecuzione su altri server, ma questa esercitazione è stata solo verificata con WebSphere Application Server v6.0 e v6.1.

Prerequisiti

È necessario avere conoscenza di:

- Sviluppo Java di base
- Nozioni fondamentali sui servizi Web
- Nozioni fondamentali sul workbench, ad esempio, come utilizzare i progetti e navigare nelle prospettive e nelle viste

Modulo 1: Progettazione della GUI del client in un editor visivo

Questo modulo insegna ad utilizzare l'editor visivo Java per aggiungere un componente visivo ad un'applicazione, disporlo visivamente ed impostarne i vincoli. La lezione finale di questo modulo mostra come eseguire il file Java per visualizzarne l'aspetto in una applicazione reale.

Attenzione: Prima di iniziare questo modulo, è necessario conoscere i prerequisiti indicati nell'introduzione all'esercitazione.

Obiettivi

Dopo aver completato le lezioni in questo modulo, si disporrà delle conoscenze necessarie per svolgere le seguenti attività:

- Aggiungere e disporre una JTable in un'interfaccia Java
- Eseguire una classe visiva per verificare il funzionamento del proprio lavoro

Tempo richiesto

Il completamento di questo modulo richiede all'incirca 15 minuti.

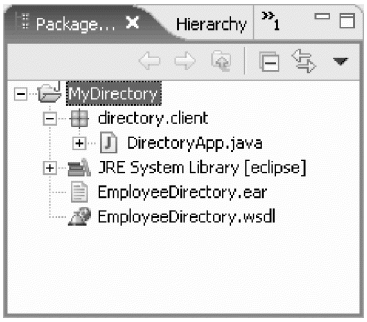
Lezione 1.1: Importazione del progetto Java MyDirectory

In questa lezione, verrà impostato il progetto MyDirectory importando un progetto nell'area di lavoro. Il progetto include una singola classe Java e altri file che verranno utilizzati successivamente.

Poiché l'attenzione principale di questa esercitazione è il binding dei componenti visivi ad un servizio Web, la maggior parte della GUI Java per l'applicazione ″My Company Directory″ è già stata progettato per l'utente.

Il progetto MyDirectory è il progetto Java principale che verrà utilizzato in questa esercitazione. Contiene il file DirectoryApp.java, il file Java che a sua volta contiene l'applicazione Java principale in corso di generazione. Queste esercitazione includono diverse versioni del progetto MyDirectory per il supporto: uno per l'avvio di ogni modulo ed una versione finita del progetto completo.

1. Importa il progetto MyDirectory.
2. In Esplora pacchetti della prospettiva Java, assicurarsi che il progetto MyDirectory appaia come nella seguente immagine:



Riepilogo della lezione

In questa lezione si è importato il progetto MyDirectory di esempio che agisce come punto iniziale per questa esercitazione.

Il progetto MyDirectory include le seguenti risorse:

- DirectoryApp.java: un file che contiene l'applicazione da sviluppare nel corso di questa esercitazione. Il file DirectoryApp.java si trova nel pacchetto Java directory.client.

- EmployeeDirectory.ear: un'applicazione enterprise che contiene il servizio Web di esempio. Nel Modulo 2, questo servizio Web verrà distribuito sull'installazione locale di WebSphere Application Server v6.0.
- EmployeeDirectory.wsdl: un file XML che utilizza il linguaggio WSDL (Web Services Description Language) per descrivere il servizio Web di esempio che verrà distribuito. Nel Modulo 2, verrà utilizzato il file WSDL per generare un proxy Java che verrà utilizzato dall'applicazione.

Lezione 1.2: Aggiunta e disposizione della tabella dipendenti

In questa lezione verrà utilizzato l'editor visivo Java per aggiungere un JScrollPane ed una JTable all'applicazione. Negli esercizi successivi, verrà programmata la JTable per ottenere i dati da un servizio Web che restituisce un elenco di tutti i dipendenti nella struttura dell'azienda.

Dopo aver aggiunto la JTable, verrà utilizzata la vista di progettazione dell'editor visivo Java per personalizzare il layout della JTable nel modo seguente:

- Espandere la JTable su tre celle in orizzontale e due celle in verticale
- Aggiungere un rientro a sinistra di 15 pixel
- Ridenominare JTable in employeesTable.

Mostra

Apertura del file DirectoryApp.java nell'editor visivo Java

Per aprire il file DirectoryApp.java nell'editor visivo Java, procedere come segue:

1. Nella vista Esplora pacchetti della prospettiva Java, espandere il progetto MyDirectory ed il pacchetto directory.client.
2. Selezionare il file DirectoryApp.java con il tasto destro del mouse e scegliere **Apri con → Editor visivo**. L'editor visivo Java carica la classe Java e visualizza la progettazione sull'area di disegno grafica.

Suggerimento:

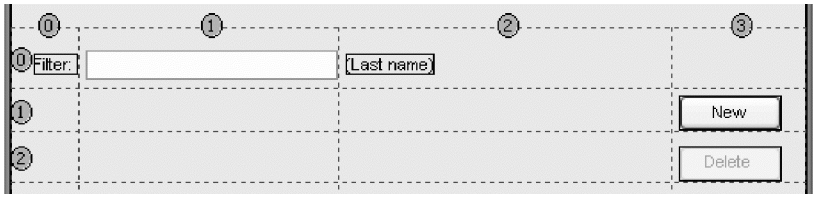
- Per modificare l'aspetto utilizzato dall'editor visivo Java, scegliere **Finestra → Preferenze → Java → Editor visivo** e specificare l'aspetto Swing. Il nuovo aspetto verrà applicato all'apertura successiva della classe. Questa esercitazione utilizza l'aspetto Windows.
- Affinché l'Editor visivo venga utilizzato come editor predefinito per tutti i file Java, scegliere **Finestra → Preferenze**, passare alla pagina **Workbench → Associazioni file** e definire le preferenze.

Aggiunta di una JTable in un JScrollPane

La finestra principale di DirectoryApp.java utilizza un JFrame con JPanel per il riquadro del contenuto principale. JPanel nell'applicazione è denominato jContentPane. jContentPane è stato impostato per utilizzare un tipo di gestore di layout denominato GridBagLayout. GridBagLayout è uno schema di layout basato su una griglia di celle che può essere occupata dai componenti visivi. L'editor visivo Java semplifica l'utilizzo di GridBagLayout consentendo di visualizzare i bordi della griglia. Inoltre mostra gli indicatori di posizione quando nella griglia vengono trascinati nuovi componenti, e mostra gli handler sui componenti che si stanno ridimensionando o spostando sul GridBagLayout.

Per aggiungere la tabella dei dipendenti (javax.swing.JTable) all'interfaccia utente DirectoryApp.java, procedere come segue:

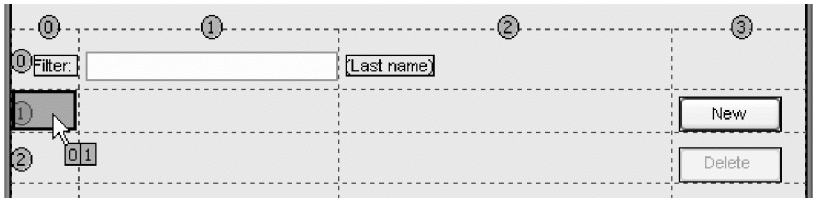
1. Selezionare jContentPane con il tasto destro del mouse nella vista Progettazione o nella vista Bean Java e scegliere **Mostra griglia**. Una linea rossa punteggiata mostra il bordo della griglia e cerchi blu, numerati, indicano i numeri delle righe e delle colonne. Ad esempio, il pulsante **Nuovo** occupa la cella alla riga 1 (griglia y) e la colonna 3 (griglia x).



2. Nella tavolozza dell'editor visivo Java, selezionare il componente **JTable su JScrollPane** Swing, appartenente alla categoria nel cassetto **Componenti Swing** della tavolozza.

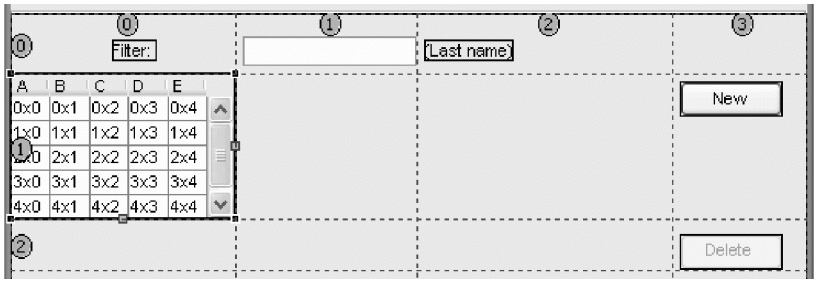
   Suggerimento: Per impostazione predefinita, la tavolozza è ridotta a icona sul lato destro dell'area di progettazione. È possibile ridimensionare e spostare la tavolozza.

3. Spostare il puntatore del mouse sulla cella nella griglia alla colonna 0, riga 1:



   - Quando si sposta il puntatore del mouse nella griglia, vengono visualizzati due quadrati numerati che indicano le coordinate x e y nella griglia in base alla posizione del puntatore.
   - Se si sposta il puntatore direttamente sul bordo della griglia, sarà possibile creare nuove righe e colonne e, come conseguenza, le righe e le colonne esistenti verranno rinumerate. In questo caso, i quadrati gialli sul puntatore del mouse, le barre gialle tra le griglie e le etichette gialle delle colonne e delle righe indicano questo comportamento ed evidenziano l'impatto del posizionamento.

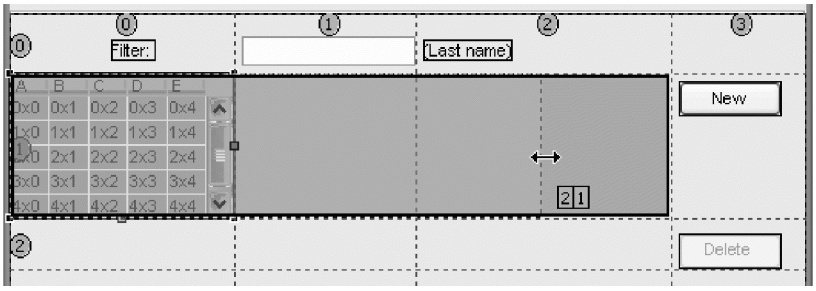
4. Fare clic con il tasto sinistro del mouse per trascinare il JScrollPane e la JTable nella cella nella colonna 0 e riga 1:



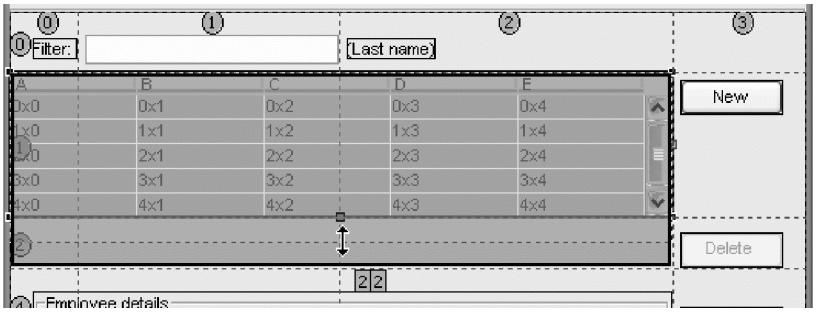
Unione di JScrollPane e JTable in più colonne e righe della griglia

A questo punto, impostare JScrollPane (e le rispettive JTable secondarie) in modo che occupi tre colonne e due righe per ottenere un miglior comportamento di spaziatura e ridimensionamento. Affinché la tabella occupi le colonne e le righe, procedere come segue:

1. Selezionare il JScrollPane nell'area di progettazione o nella vista Bean Java (dovrebbe ancora essere selezionato perché è stato appena aggiunto). Osservare i piccoli quadrati verdi in basso e a destra del JScrollPane. Per trascinare JScrollPane affinché occupi più righe e colonne, verranno utilizzati questi handle di ridimensionamento.
2. Fare clic e tenere premuto il tasto sinistro del mouse sull'handle verde a destra del JScrollPane.
3. Trascinare il puntatore del mouse verso destra fino a quando la posizione indica la colonna 2, riga 1. Un'ombreggiatura grigia indica le celle che verranno occupate dal componente quando verrà rilasciato il tasto del mouse.



4. Rilasciare il tasto del mouse. Il JScrollPane occuperà tre colonne.
5. Ripetere lo stesso procedimento per spostare l'handle inferiore del JScrollPane affinché arrivi a occupare la riga 2:



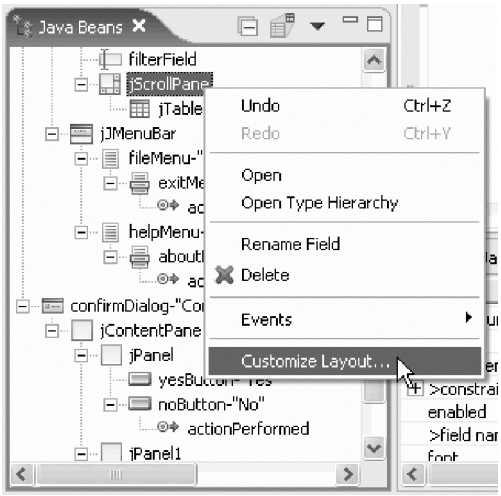
Personalizzazione della spaziatura del JScrollPane nella GridBag

Un'altra funzione del gestore di GridBagLayout è la possibilità di specificare diversi vincoli per personalizzare ulteriormente il layout. Ad esempio, è possibile specificare i seguenti vincoli:

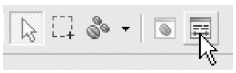
- **ancoraggio:** è possibile assegnare ad un componente un orientamento ancorato nella cella, che influenzerà gli spostamenti del componente quando l'applicazione viene ridimensionata da un utente. Ad esempio un componente potrebbe essere ancorato in alto a sinistra, al centro a sinistra, al centro, o in basso a destra.
- **riempimento:** è possibile specificare che un componente occupi tutto lo spazio disponibile nella propria cella o nelle celle orizzontali, verticali o entrambe.
- **rientri:** è possibile assegnare ad un componente un particolare rientro sul margine in alto, in basso, a sinistra e a destra per indicare una spaziatura tra il componente e il margine sinistro della griglia.

Per personalizzare l'ancoraggio, il riempimento e i rientri per il JScrollPane, procedere come segue:

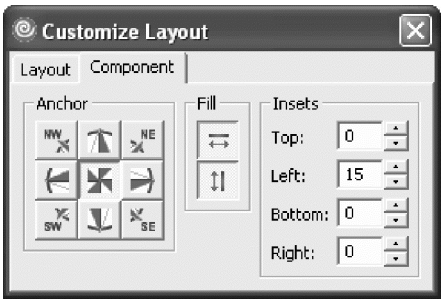
1. Selezionare JScrollPane con il tasto destro del mouse nella vista Progettazione o nella vista Bean Java e scegliere **Mostra griglia**.



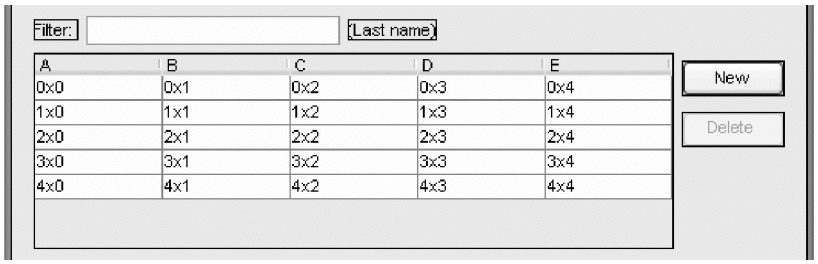
Suggerimento: La finestra Personalizzazione layout può rimanere aperta durante la selezione e la modifica del layout dei diversi componenti. È possibile aprire la finestra Personalizzazione layout in qualsiasi momento scegliendo il pulsante Personalizza layout nella barra dei menu:



2. Nella scheda Componente della finestra Personalizzazione layout, assicurarsi che il pulsante Centro di ancoraggio sia premuto.
3. Assicurarsi che i pulsanti **Riempimento orizzontale** e **Riempimento verticale** siano premuti.
4. Aggiungere un rientro a sinistra di 15 (pixel) per far sì che la spaziatura a sinistra di JScrollPane risulti simile a quella di altri componenti visivi nell'applicazione.



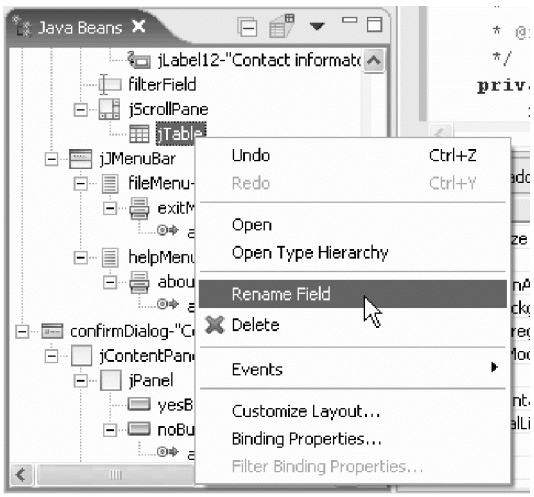
Ad esempio, la tabella adesso è allineata all'etichetta **Filtro**.



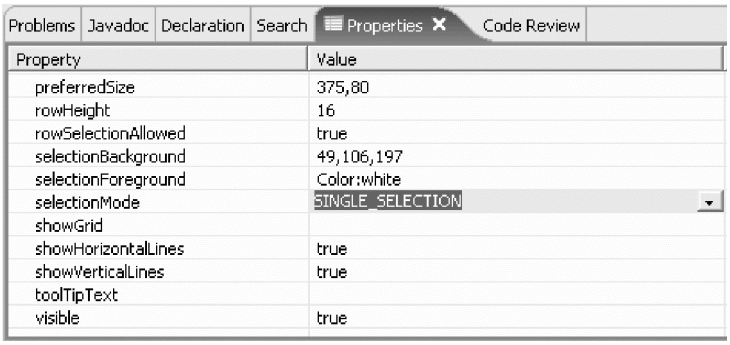
Ridenominazione della nuova JTable con un valore significativo ed impostazione della tabella affinché selezioni una riga singola

Poiché successivamente sarà necessario utilizzare la tabella, ridenominare l'istanza JTable ed il metodo getter. Per ridenominare la tabella, procedere come segue:

1. Nella vista Bean Java, selezionare il componente jTable con il tasto destro del mouse e scegliere **Rinomina campo** dal menu a comparsa.



2. Immettere `employeesTable` e scegliere **OK**. JTable adesso si chiama employeesTable, ed il metodo per l'istanza è getEmployeesTable.
3. Impostare la tabella per consentire di selezionare una singola riga:
   a. Selezionare employeesTable nella vista Progettazione.
   b. Nella vista Proprietà, selezionare la proprietà **selectionMode** ed impostarla su SINGLE_SELECTION.



   c. Salvare il file DirectoryApp.java.

Riepilogo della lezione

In questa lezione è stato illustrato come utilizzare l'editor visivo per aggiungere una tabella ad un'interfaccia utente esistente e come personalizzarne il layout, il posizionamento e la spaziatura.

Lezione 1.3: Esecuzione della classe visiva

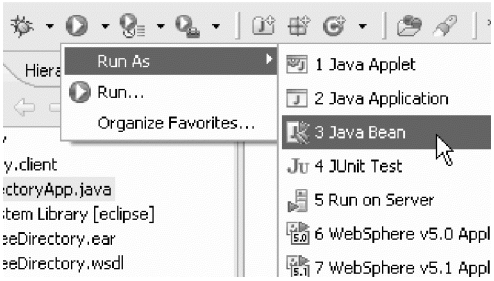
Ora è possibile eseguire l'applicazione Java per visualizzarne l'anteprima. Il workbench e l'editor visivo consentono di eseguire rapidamente l'applicazione ed inoltre è possibile ripetere questa procedura in qualsiasi momento per verificare l'aspetto di runtime reale ed il comportamento della classe.

Mostra

L'editor visivo Java fornisce un'utilità di avvio per bean Java in grado di eseguire le classi senza metodo main(). Quando esegue la classe visiva, avvia l'applicazione in una VM (Virtual Machine) separata. Se si esegue la classe visiva come applicazione Java, l'utilità di avvio tenterà di eseguire il metodo main() nella classe. Per questa esercitazione, l'applicazione include un metodo main() che richiama e mostra la DirectoryApp JFrame, in modo che sia possibile eseguirlo come applicazione o come bean Java.

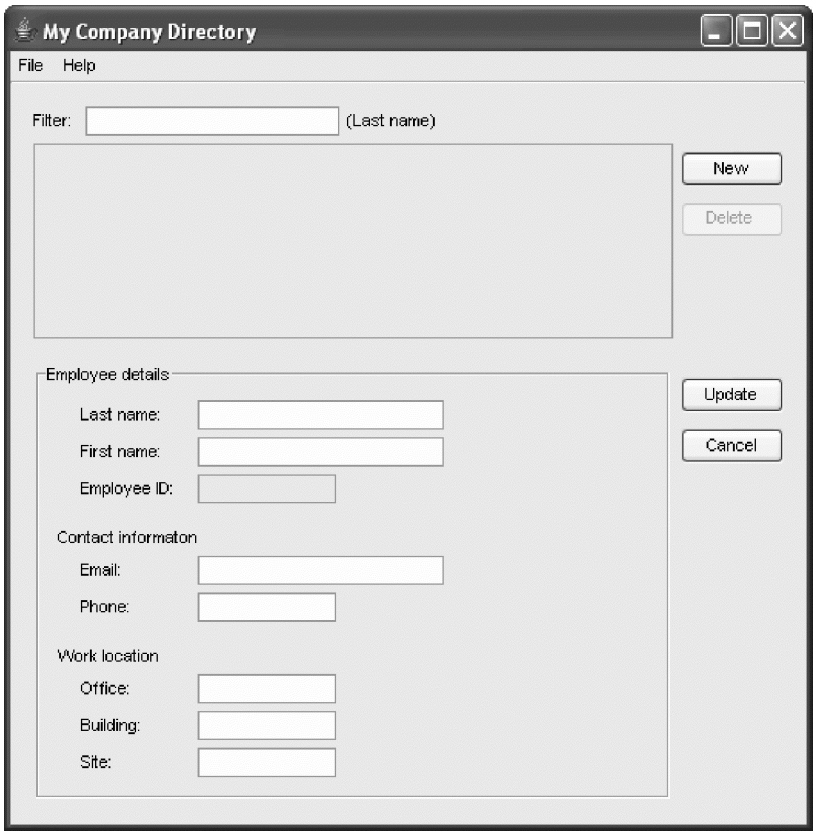
Per eseguire il file DirectoryApp.java come un bean Java, procedere come segue:

1. Assicurarsi che il file DirectoryApp.java sia aperto nell'editor visivo Java.
2. Nella barra dei menu, scegliere **Esegui** → **Esegui come** → **Bean Java**.



Suggerimento: L'applicazione viene aperta sul desktop utilizzando l'aspetto Swing definito nelle preferenze dell'editor visivo (**Finestra** → **Preferenze** → **Java** → **Editor visivo**). In alternativa, scegliere **Esegui** → **Esegui**, e definire l'aspetto della configurazione di avvio per avviare questo bean Java. Se

l'applicazione viene eseguita come applicazione e non come bean, utilizzerà anche l'aspetto Windows, perché è stato definito nel metodo main(). Le schermate utilizzate in questa esercitazione mostrano l'aspetto Windows.



Riepilogo della lezione

Poiché fino ad ora è stata solo progettata l'interfaccia e non sono state programmate connessioni di dati o funzionalità di eventi, non è possibile eseguire alcuna azione con l'applicazione. Tuttavia, è possibile vedere il layout di base e l'aspetto che verrà visualizzato dall'utente. È possibile fare clic su alcuni pulsanti, ma non funzioneranno. I menu File e Guida sono già implementati. È possibile utilizzarli per esaminarne il comportamento ed è possibile rivedere il codice Java per capire come sono implementati con gli eventi actionPerformed.

Lezioni svolte

Questo modulo è una introduzione alla progettazione dell'interfaccia per un rich client mediante l'editor visivo Java. Oltre alla progettazione dell'aspetto visivo del client, affinché il client possa essere utilizzato, sarà necessario svolgere molte altre attività. Sarà necessario includere il comportamento degli eventi o un'altra logica e, in questo caso, il binding degli elementi visivi ad un'origine dati di un tipo determinato.

In questo modulo, sono state descritte le seguenti attività:

- Importazione di un progetto Java mediante l'importazione di scambio progetti
- Aggiunta di una JTable su un JScrollPane alla classe visiva
- Utilizzo del gestore GridBagLayout per disporre visivamente la tabella sul rich client
- Esecuzione dell'applicazione per visualizzare l'aspetto reale del rich client Java

Nel modulo successivo, Modulo 2: verranno associati componenti visivi al servizio Web e l'interfaccia semplice dell'applicazione My Company Directory verrà modificata in un rich client che accede ai metodi

dei servizi Web per la creazione, il richiamo, l'aggiornamento e l'eliminazione dei record dei dipendenti da una struttura aziendale.

Modulo 2: Generazione dei componenti visivi nel servizio Web

Questo modulo illustra come associare gli elementi visivi dell'applicazione My Company Directory (i pulsanti, la tabella dipendenti, i campi e altre azioni) ad un servizio Web. Il servizio Web fornisce la le funzionalità per creare, richiamare, aggiornare ed eliminare dipendenti da una struttura di esempio.

Obiettivi

Dopo aver completato le lezioni in questo modulo, si disporrà delle conoscenze necessarie per svolgere le seguenti attività:

- Associare una tabella all'origine dati di un servizio Web
- Associare campi a oggetti
- Programmare pulsanti con azioni

Questo modulo richiede circa **2 ore**.

Lezione 2.1: Installazione e distribuzione del servizio Web

In questo esercizio, verrà installato un file EAR (enterprise application) di esempio su WebSphere Application Server v6.1 e verrà distribuito il servizio Web EmployeeDirectory. L'applicazione utilizza il servizio Web per creare, richiamare, aggiornare ed eliminare i record dei dipendenti.

Prima di iniziare, completare *una delle seguenti opzioni* per assicurarsi che il progetto MyDirectory si trovi nel punto di partenza appropriato:

- Completare la "Modulo 1: Progettazione della GUI del client in un editor visivo" a pagina 3.

  o

- Importa il progetto MyDirectory al punto di inizio del Modulo 2

  Suggerimento: A meno che durante l'importazione non venga specificato un nome progetto diverso, questo sovrascrive i contenuti del progetto MyDirectory.

Il progetto MyDirectory Java contiene un file EmployeeDirectory.ear. Verrà utilizzata la console di gestione WebSphere per installare l'applicazione enterprise di EmployeeDirectory contenuta nel file EAR. Una volta installata l'applicazione, verrà anche distribuito il servizio Web incluso nell'applicazione. L'applicazione My Company Directory completata utilizzerà questo servizio Web distribuito.

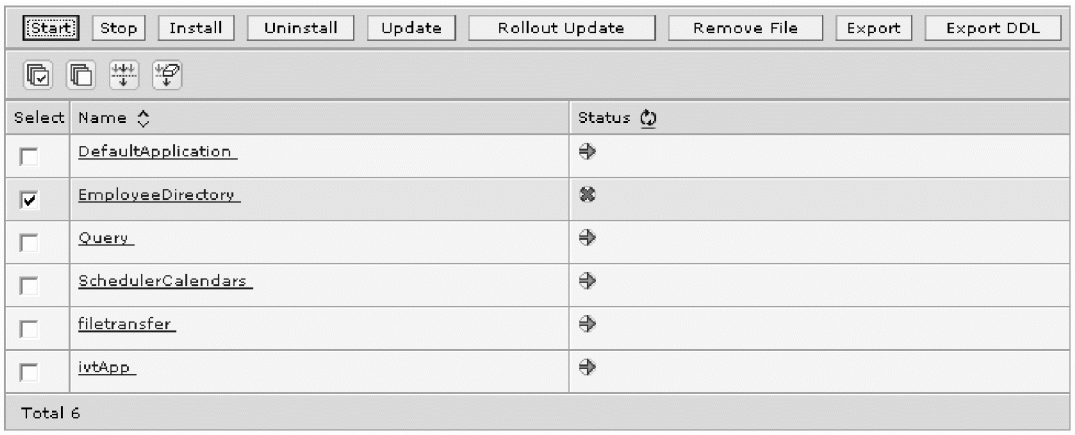
Per installare l'esempio di applicazione EmployeeDirectory e distribuire il servizio Web sull'ambiente di test di WebSphere Application Server v6.1, procedere come segue:

1. Avviare un'istanza del server di applicazioni dal workbench. Il server può essere avviato in diversi modi; in questa procedura verrà avviato dal workbench:
   a. Aprire la vista Server. Per aggiungere la vista Server alla prospettiva Java, scegliere **Finestra → Mostra vista → Altro e selezionare Server → Server**.
   b. La vista Server elenca i server installati ed impostati.
   c. Fare clic con il pulsante destro del mouse sul server e selezionare **Avvia**. Se nella vista Server lo stato del server risulta **Avviato** o se nella console appare il messaggio `Server server1 aperto per e-business`, il server sarà stato avviato correttamente. È adesso possibile eseguire la console di gestione.

   Nota: Se la vista Server non contiene alcuna istanza, creare un nuovo server:
   a. Fare clic con il tasto destro del mouse nella vista Server e selezionare **Nuovo → Server**.

b. Utilizzare la procedure guidata Nuovo server per aggiungere WebSphere Application Server v6.1.

2. Eseguire la Console di gestione WebSphere. Anche in questo caso, la console può essere eseguita in diversi modi, e le seguenti istruzioni descrivono come eseguirla dal workbench:
   a. Nella vista Server, selezionare il server appena avviato con il tasto destro del mouse e scegliere **Esegui console di gestione**. La console di gestione WebSphere viene aperta in una finestra del browser.
   b. Immettere un ID utente e scegliere **Accedi**. Viene aperta la pagina di benvenuto della console di gestione. L'ID utente immesso viene utilizzato solo per tenere traccia delle modifiche dell'utente ai dati di configurazione del server.
3. Utilizzare la console di gestione per installare l'applicazione enterprise EmployeeDirectory.ear presente nel progetto MyDirectory. La console di gestione dispone di una procedura guidata che consente di installare le applicazioni; per impostare tutte le opzioni, scegliere **Avanti** nelle diverse pagine. Per installare l'applicazione enterprise di esempio che contiene il servizio Web per questa esercitazione, procedere come segue:
   a. Sul lato sinistro della Console di gestione, espandere l'opzione **Menu Applicazioni** e scegliere **Installa nuova applicazione**.
   b. Selezionare **File system locale** e nel campo **Specifica percorso** immettere il percorso completo del file EmployeeDirectory.ear contenuto nel progetto MyDirectory. Suggerimento: per ottenere il percorso completo, selezionare il file EmployeeDirectory.ear con il tasto destro del mouse in Esplora pacchetti e selezionare **Proprietà**. Nella pagina Proprietà è riportato il percorso del file che è possibile copiare e incollare nel campo **Specifica percorso**.
   c. Fare clic su **Avanti** fino a raggiungere la pagina **Seleziona opzioni di installazione**.
   d. Selezionare **Distribuisci servizi Web**.
   e. Fare clic su **Avanti** fino a raggiungere la pagina **Riepilogo**, quindi fare clic su **Fine**.
   f. Quando richiesto, scegliere **Salva nella configurazione principale** per applicare le modifiche effettuate alla configurazione locale. Riesaminare le modifiche e scegliere **Salva**.
4. Utilizzare la console di gestione per avviare l'applicazione EmployeeDirectory:
   a. Scegliere **Applicazioni** → **Applicazioni enterprise**. L'applicazione EmployeeDirectory viene elencata come un'applicazione installata sul server, ma lo stato è Arrestato.



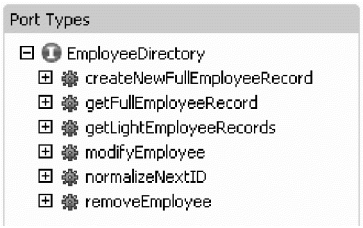
   b. Selezionare la casella di controllo accanto a EmployeeDirectory e scegliere **Avvia**. Viene visualizzato un messaggio che indica che l'applicazione EmployeeDirectory è stata avviata correttamente e l'icona Stato diventa una freccia verde.

L'applicazione EmployeeDirectory è ora in esecuzione sull'host locale sulla porta 9080, ed è possibile accedere al servizio Web. Dopo aver completato questa esercitazione, tornare alla console di gestione, arrestare EmployeeDirectory, quindi disinstallarlo.

Se si apre il file EmployeeDirectory.wsdl contenuto nel progetto MyDirectory (per impostazione predefinita viene aperto nell'editor WSDL grafico), è possibile esaminare il servizio Web distribuito. Se il

file WSDL non viene aperto nell'editor WSDL, è possibile che la funzione Sviluppatore servizi Web non sia attivata nel workbench. È possibile specificare le funzioni del workbench nelle Preferenze (**Finestra → Preferenze → Workbench → Funzioni**).

La seguente immagine dell'editor WSDL mostra le operazioni disponibili nel servizio EmployeeDirectory:



È possibile utilizzare l'editor WSDL per esaminare ciascuna operazione e i messaggi di richiesta e di ritorno corrispondenti. In tal modo è possibile comprendere la struttura del servizio Web e l'uso che ne viene fatto negli esercizi rimanenti.

Lezione 2.2: Bind della tabella dipendenti all'origine dati del servizio Web

L'applicazione My Company Directory visualizza un elenco di tutti i record correnti dei dipendenti nella struttura. I record vengono visualizzati in una JTable (employeesTable) con colonne che è possibile ordinare, che comprendono il nome, il cognome, l'e-mail e l'ID dipendente. Per ottenere i record per la tabella, è necessario associare employeesTable all'oggetto dati restituito dall'origine dati del servizio Web.

Mostra

Panoramica degli oggetti dati, origini dati e binder

Per ottenere un'oggetto dati locale che la employeesTable possa utilizzare, verrà utilizzato l'editor visivo per aggiungere un'origine dati all'applicazione. L'origine dati si connette al proxy dei servizi Web di esempio e rileva i metodi di servizio disponibili per l'applicazione. Verrà quindi scelto il metodo di servizio getLightEmployeeRecord reso disponibile dall'origine dati. Infine, employeesTable verrà associata nell'applicazione ai campi restituiti nell'oggetto dati (lightEmployeeRecordRows).

È possibile creare tutte queste origini dati e gli oggetti dati in modo rapido e semplice mediante le classi binder integrate dell'editor visivo Java. L'editor visivo fornisce un insieme di classi ed interfacce generiche generate nel progetto durante il collegamento dei componenti visivi ai factory di dati. Le classi binder vengono generate per impostazione predefinita nel pacchetto jve.generated. L'editor visivo fornisce le classi binder come un'implementazione generica che è possibile personalizzare e migliorare per soddisfare le necessità dell'applicazione. Questa esercitazione dimostra la potenza e la flessibilità delle classi binder predefinite, anche per operazioni molto semplici.

Importante: Prima di cominciare questo esercizio, si consiglia di leggere i seguenti argomenti della guida. Tali argomenti contengono informazioni utili sulla funzionalità e la logica degli oggetti dati, origini dati e binder forniti dall'editor:

- Panoramica sui binder di dati
- Riferimento all'API binder

Per questa esercitazione, verrà utilizzata un'origine dati del servizio Web, diversi tipi di oggetti dati e diversi tipi di binder nell'applicazione. Quando si aggiungono istanze di questi oggetti all'applicazione, l'editor visivo aggiunge le classi necessarie al pacchetto jve.generated nel progetto dove è possibile estendere, sostituire o riscrivere la logica del binding dei dati. L'editor visivo Java fornisce il supporto visivo per il binding di oggetti mostrando sull'area a formato libero della vista Progettazione, gli oggetti

dati, le origini dati e i binder che l'applicazione sta utilizzando. L'editor visivo traccia linee tra i componenti visivi, gli oggetti dati e le origini dati per mostrare i collegamenti correnti per ciascun oggetto selezionato.

Il seguente diagramma costituisce una semplice panoramica dell'interazione reciproca dei diversi componenti visivi, binder, oggetti dati e origini dati. L'applicazione creata in questa esercitazione illustra un uso leggermente più complesso e creativo dei binder. Questo diagramma non rappresenta esattamente i binder, gli oggetti dati e le origini dati nell'applicazione di esempio che si sta creando.

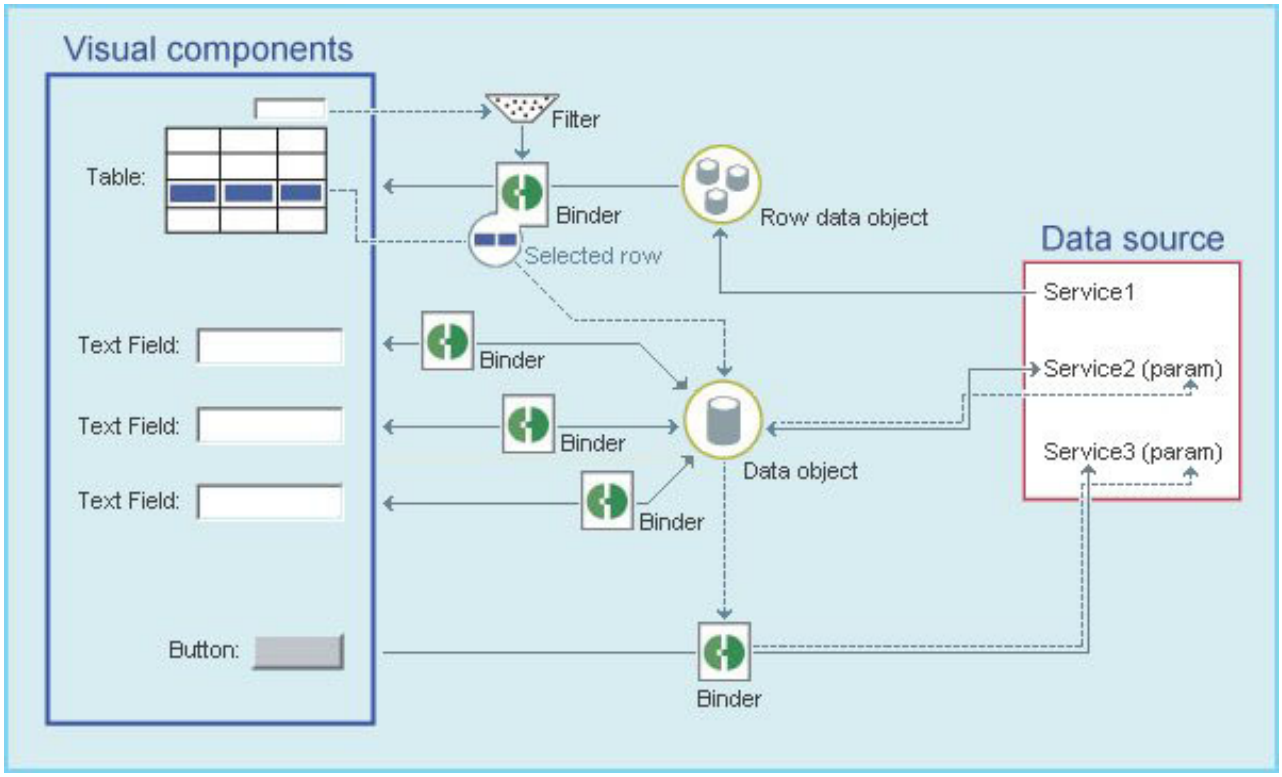


Figura 1. Questo diagramma illustra un semplice rapporto tra i componenti visivi,i binder, gli oggetti di dati e le origini dati

Nella figura 1, ciascun componente visivo ha il proprio binder che lo associa a un oggetto dati oppure, nel caso di un pulsante, a un'origine dati. I binder per i campi di testo collegano il campo a una particolare proprietà dell'oggetto dati. Sia l'oggetto dati riga sia l'oggetto dati di questo diagramma ricavano i propri dati da chiamate dirette a un servizio che risiede sull'origine dati. L'oggetto dati per i campi di testo utilizza un valore chiave ricavato dalla riga selezionata nella tabella come argomento per la chiamata a Service2, che restituisce un record completo che include presumibilmente più informazioni sulla riga selezionata nella tabella. Tale record completo, a volta, viene utilizzato come argomento per il binder di azione del pulsante quando richiama Service3, che può essere, ad esempio, un metodo che aggiorna i valori immessi nei campi. Per spiegazioni più dettagliate sugli oggetti dati, i binder di dati e le origini dati, seguire i collegamenti forniti precedentemente.

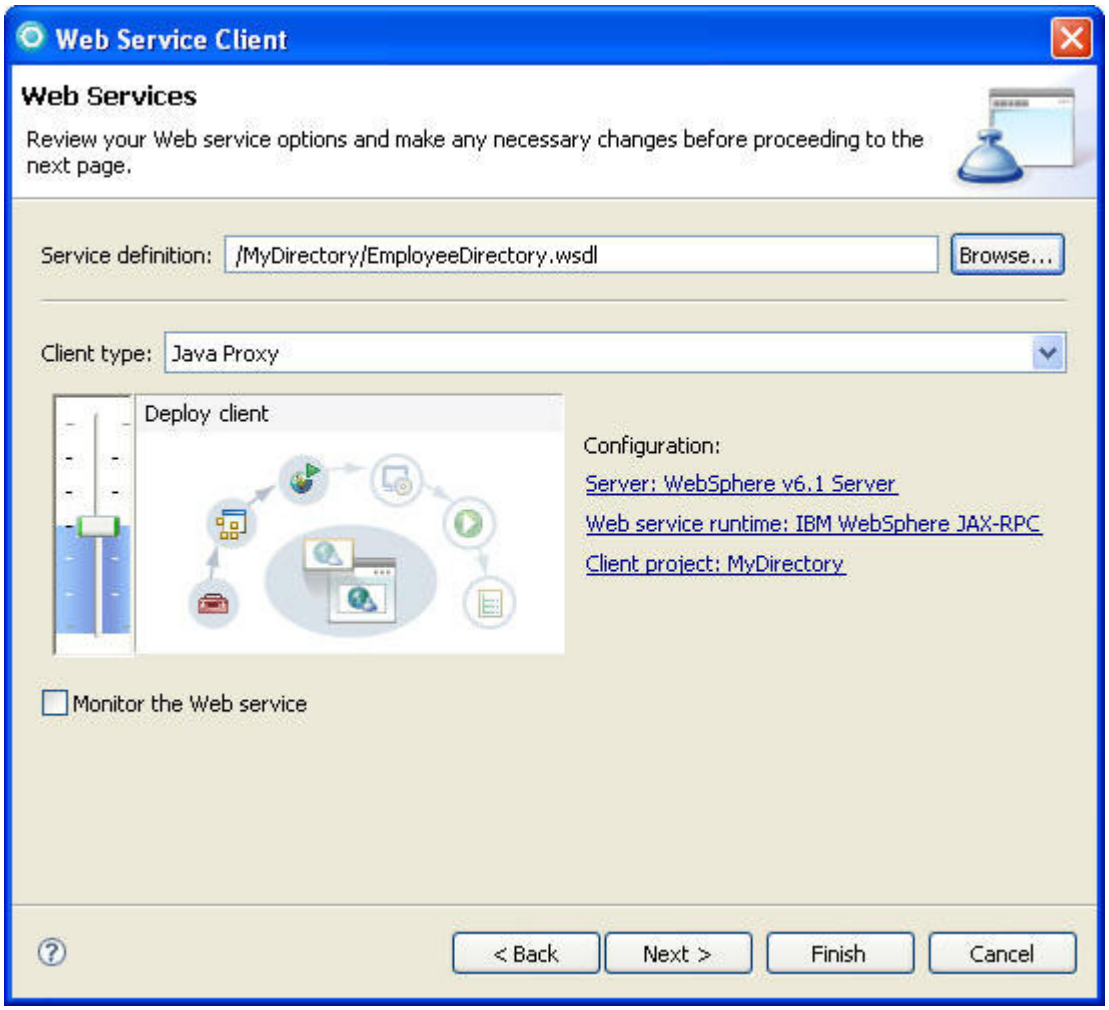
Generazione di un proxy Java del servizio Web nel progetto utilizzando il file WSDL fornito

Per utilizzare il servizio Web in esecuzione su un server, l'applicazione Java richiede un proxy Java, o client, con cui interagire. Utilizzando un file WSDL, è possibile generare un proxy Java nel progetto Java utilizzando la procedura guidata Client di servizi Web. Il progetto MyDirectory contiene il file EmployeeDirectory.wsdl che verrà utilizzato per generare questo proxy. Una volta generato il proxy Java, sarà possibile creare un'origine dati che rappresenta il servizio Web e il binding visivo dei componenti.

Importante: Il file WSDL utilizzato in questo esercizio prevede che il servizio Web sia stato distribuito su WebSphere Application Server installato localmente e che sia stata utilizzata la porta predefinita per localhost (http://localhost:9080). Se il file EAR è stato distribuito in modo diverso, sarà necessario modificare il file WSDL per poter procedere.

Per generare un proxy Java di servizi Web nel progetto, procedere come segue:

1. Nel menu principale, scegliere **File** → **Nuovo** → **Altro** e selezionare la procedura guidata **Servizi Web** → **Client di servizi Web**. Se la categoria servizi Web non viene visualizzata, selezionare **Mostra tutte le procedure guidate**.
2. Utilizzare la procedura guidata per definire il client del servizio Web:
   a. Per la **Definizione servizio**, immettere il file WSDL fornito nel progetto MyDirectory: `/MyDirectory/EmployeeDirectory.wsdl`
   b. Nel campo **Tipo client**, selezionare **Proxy Java**.
   c. Impostare la barra scorrevole su **Distribuisci client**.
   d. Verificare che il server ed il runtime del servizio Web sia correttamente impostato per il server che si sta eseguendo. Questa esercitazione è stata verificata su WebSphere v6.0 e WebSphere v6.1 con il runtime IBM WebSphere JAX-RPC.
   e. Verificare che il client proxy Java sia emesso nel progetto MyDirectory.



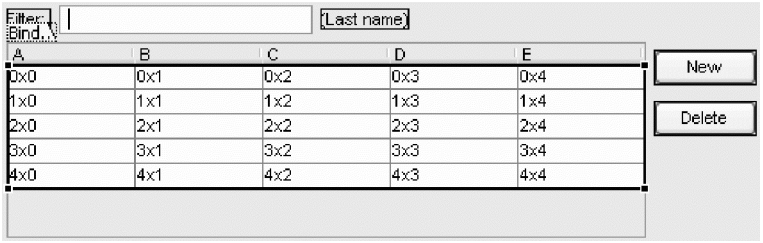
3. Scegliere **Fine**. La proceduta guidata Client di servizi Web genera il proxy Java in un nuovo pacchetto (directory.service) nel progetto.

Bind di employeesTable a un oggetto dati restituito dal servizio Web

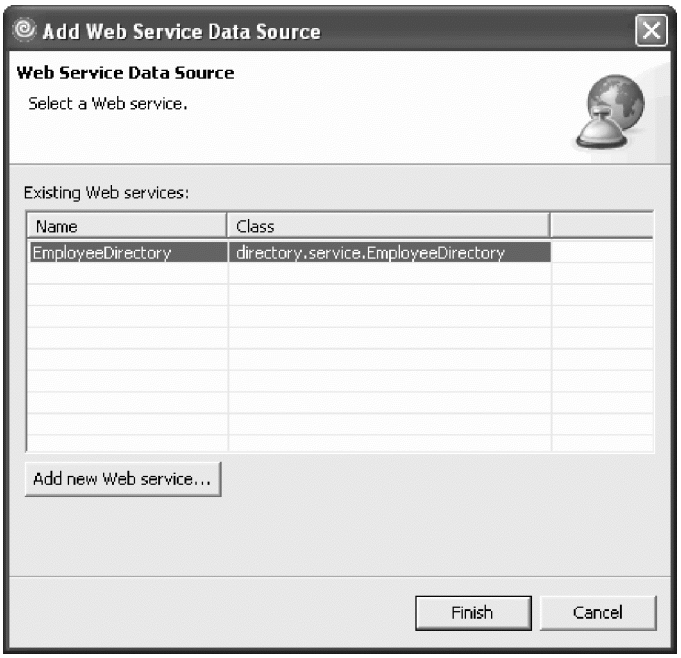
Poiché employeesTable è il primo componente visivo di cui viene eseguito il binding in questa applicazione, sarà necessario creare un'origine dati che faccia riferimento al proxy di servizi Web di esempio appena aggiunto al progetto. Quando si esegue il binding degli altri componenti visivi negli esercizi successivi, verrà riutilizzata questa origine dati. In questa fase, vengono aggiunti l'origine dati dei servizi Web e l'oggetto dati lightEmployeeRecordRows.

Per eseguire il collegamento della tabella dipendenti:

1. Nella vista Bean Java o Progettazione, selezionare employeesTable. Assicurarsi che non venga selezionato il JScrollPane principale. Al di sopra di employeesTable nell'area di progettazione, viene visualizzata una piccola scheda **Collegamento**.

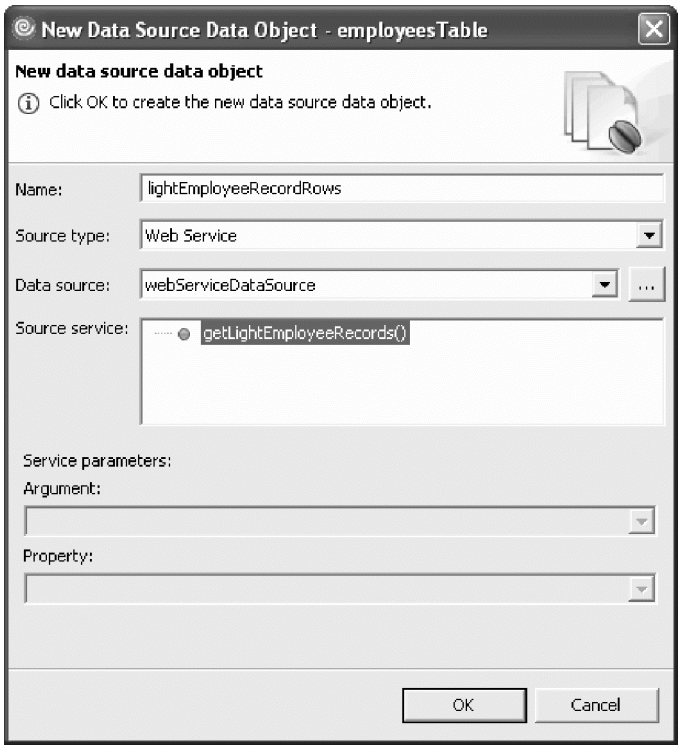


2. Fare clic sulla scheda **Collegamento** nella employeesTable. Oppure, è possibile fare clic con il tasto destro del mouse su employeesTable e selezionare **Proprietà di binding**.
3. Poiché l'applicazione non contiene oggetti dati, è necessario aggiungerne uno nuovo. Scegliere **Nuovo oggetto dati dell'origine dati**.
4. Nel campo **Tipo di origine**, selezionare **Servizio Web**.
5. Poiché all'applicazione non è stata ancora aggiunta l'origine dati del servizio Web, sarà necessario aggiungerla adesso. Accanto al campo **Origine dati**, fare clic sul pulsante **...** per aprire la finestra Aggiunta origine dati del servizio Web in cui è possibile ricercare client di servizi Web o proxy disponibili nel progetto.
6. Selezionare il servizio Web EmployeeDirectory e scegliere Fine. Una nuova origine dati viene aggiunta al file DirectoryApp.java.



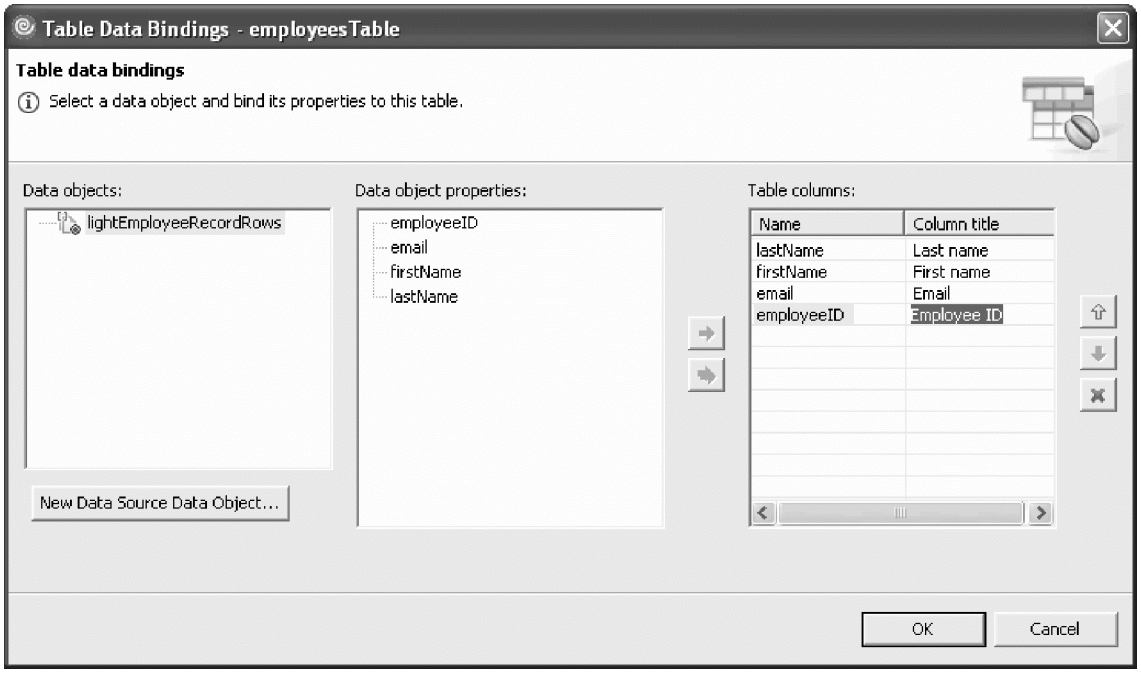
7. Nella finestra Nuovo oggetto dati dell'origine dati, selezionare getLightEmployeeRecords() nel campo Servizio di origine ed accettare il nome predefinito per il nuovo oggetto dati:

lightEmployeeRecordRows. Non è necessario alcun parametro per questo metodo di servizio. Scegliere OK. Viene creato il nuovo oggetto dati e viene visualizzato nell'area a formato libero della vista Progettazione.



Suggerimento: Poiché viene eseguito il binding di una tabella, la finestra Nuovo oggetto dati dell'origine dati conterrà solo i servizi che restituiscono oggetti dati. In questo caso, il metodo getLightEmployeeRecords() sarà l'unico servizio disponibile che restituisce una matrice di oggetti.

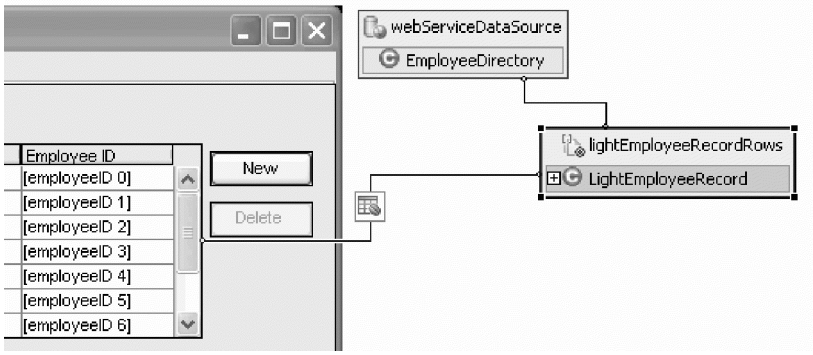
8. Nella finestra Binding di dati tabella, selezionare l'oggetto dati lightEmployeeRecordRows.
9. Selezionare le proprietà dell'oggetto dati lightEmployeeRecordRows che si desidera visualizzare nella employeeRecords:



a. Scegliere la doppia freccia ⇨ per aggiungere tutte le proprietà dell'oggetto all'elenco **Colonne della tabella**.
b. Usare le frecce su e giù per disporre le colonne nell'ordine seguente: lastName, firstName, email, employeeID
c. Ridenominare i titoli delle colonne in `Cognome`, `Nome`, `Email`, `ID Dipendente`

   Suggerimento: Una volta completato il binding della tabella, è sempre possibile tornare alle proprietà di binding e ridenominare e riordinare le colonne.
d. Scegliere **OK**.

La employeesTable è ora associata all'oggetto dati lightEmployeeRecordRows utilizzando JRowTableBinder. Scegliendo l'oggetto dati lightEmployeeRecordRows nell'area a formato libero, l'editor visivo traccia una linea dall'oggetto dati alla tabella. Sulla linea, JRowTableBinder è rappresentato dall'icona del binder di tabella . Un'altra linea indica che l'oggetto dati utilizza webServiceDataSource come origine dati.



Riepilogo della lezione

Rivedere le modifiche al progetto e all'applicazione. Durante questa lezione, sono stati aggiunti l'origine dati del servizio Web, l'oggetto dati e il binder che associa employeesTable all'oggetto dati.

Osservare il nuovo pacchetto (jve.generated) creato nel progetto in cui sono contenute tutte le classi binder generate dall'editor visivo Java. Osservare inoltre il nuovo pacchetto (directory.service) che contiene il proxy Java per il servizio Web. Descrivere o riepilogare la lezione.

Adesso, se si esegue l'applicazione My Company Directory, il sevizio Web inserisce nella tabella Dipendenti i record dei dipendenti esistenti.

Lezione 2.3: Bind dei campi dei dettagli alla selezione della tabella

Nell'esercizio precedente, è stato eseguito il binding della employeesTable all'oggetto dati lightEmployeeRecordRows restituito dal servizio getLightEmployeeRecords() nel servizio Web. A questo punto, sarà necessario inserire i dettagli nei campi in base al dipendente selezionato nella tabella.

Per ottenere ulteriori dettagli per ogni dipendente selezionato, viene utilizzato un altro oggetto dati. L'oggetto dati selectedEmployeeRecord che verrà aggiunto, viene restituito dal servizio getFullEmployeeRecord(). Questo servizio utilizza l'ID del dipendente selezionato nella tabella come parametro e raccoglie altri dettagli sul dipendente, tra cui numero di telefono e reparto.

Il JRowTableBinder utilizzato quando si è eseguita l'associazione della tabella all'oggetto dati semplifica questa fase. Il JRowTableBinder inserisce l'elemento selezionato in una tabella come oggetto dati separato che può essere utilizzato come parametro per il metodo getFullEmployeeRecord(java.lang.Integer). Sarà quindi possibile associare facilmente ciascun campo di testo alla proprietà corrispondente nell'oggetto dati selectedEmployeeRecord.

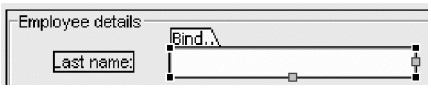
Altre informazioni su questo servizio Web: Il servizio Web include due servizi per ottenere tutti i dettagli di ciascun dipendente. La tabella riporta tutti i dipendenti e solo alcuni dati. Quando viene selezionato un singolo dipendente, è possibile richiamare le informazioni rimanenti relative al solo dipendente selezionato. Se il servizio Web invia tutti i dati per ciascun dipendente richiesti dalla tabella, il traffico Web potrebbe risultare appesantito e rallentare le prestazioni dell'applicazione.

Ad esempio, se il record del dipendente include una foto o un allegato, tali elementi non dovranno essere richiamati quando si richiede l'intero elenco di dipendenti. Quindi, il servizio getLightEmployeeRecord verrà utilizzato per inserire dati nella tabella e getFullEmployeeRecord otterrà il record completo per il dipendente selezionato nella tabella.

Bind del campo Cognome

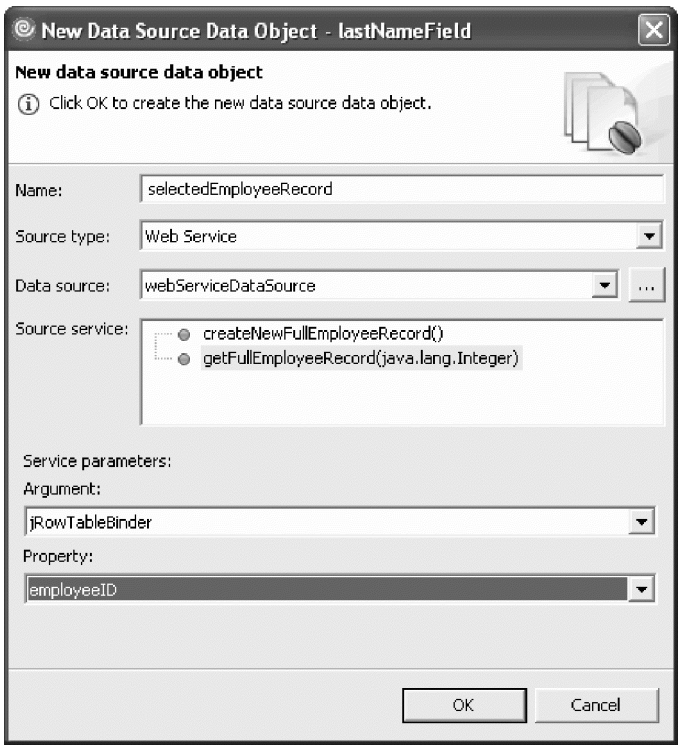
In questa fase verrà eseguito il binding del campo **Cognome** alla proprietà lastName nell'oggetto dati selectedEmployeeRecord:

1. Nella vista Bean Java o Progettazione, selezionare il JTextField per il cognome (lastNameField). L'area di progettazione conterrà la scheda **Binding** sul campo di testo.

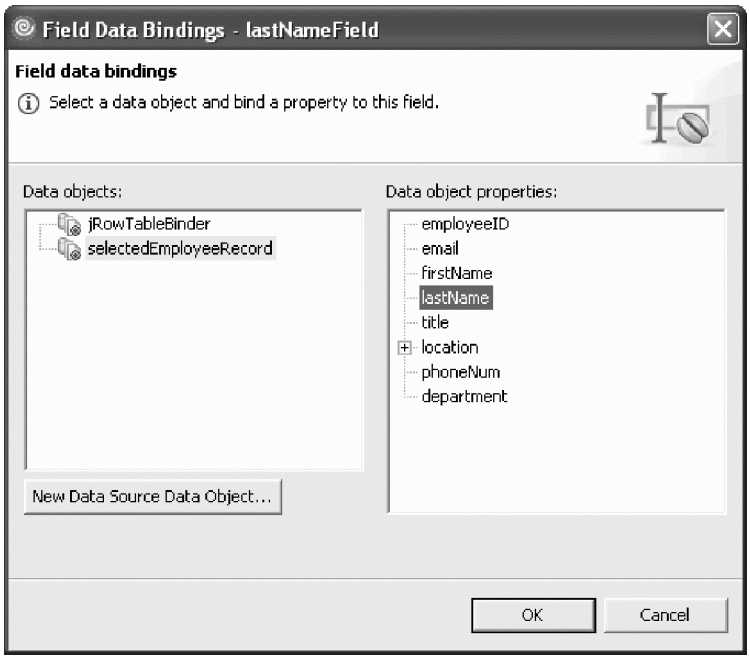


2. Scegliere la scheda **Binding** per aprire la finestra Binding di dati campo.
3. Scegliere **Nuovo oggetto dati dell'origine dati**. Anche se l'oggetto dati jRowTableBinder esistente restituisce il cognome corretto, non restituirà il record di dipendente completo. Per rappresentare il record dipendente completo, sarà necessario creare un nuovo oggetto dati.
4. Nel campo **Tipo di origine**, assicurarsi che **Servizio Web** sia selezionato e per **Origine dati**, assicurarsi che **webServiceDataSource** sia selezionato.
5. Nell'elenco **Servizio di origine**, selezionare getFullEmployeeRecord(java.lang.Integer). La finestra Nuovo oggetto dati dell'origine dati, contiene i servizi che restituiscono gli oggetti dati compatibili con un campo di testo.
6. Nel campo **Nome**, immettere `selectedEmployeeRecord`.
7. Nel campo **Argomento**, selezionare jRowTableBinder, e nel campo **Proprietà**, selezionare employeeID. L'ID dipendente della riga selezionata è adesso impostato come argomento del metodo di servizio getFullEmployeeRecord().

   Nota: getFullEmployeeRecord(java.lang.Integer) richiede un numero intero come argomento. Per richiamare un record completo, utilizzare l'ID dipendente della selezione corrente nella tabella dipendenti. Quando si esegue il binding alla tabella, l'editor visivo genera automaticamente un jRowTableBinder, che attende la selezione corrente nella tabella dei dipendenti. Per il parametro integer, verrà utilizzato l'employeeID della riga selezionata nel jRowTableBinder.

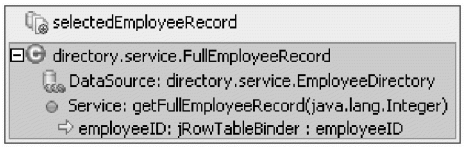


8. Scegliere **OK**.
9. Nella finestra Binding di dati campo, assicurarsi che selectedEmployeeRecord sia selezionato nell'elenco **Oggetti dati**. L'oggetto dati selectedEmployeeRecord ha più proprietà disponibili dell'oggetto dati jRowTableBinder.
10. Nell'elenco **Proprietà oggetto dati**, selezionare la proprietà lastName.



11. Scegliere **OK**. Il campo Cognome dell'applicazione viene associato alla proprietà lastName dell'oggetto dati selectedEmployeeRecord, restituito da getFullEmployeeRecord().

    Viene creato il nuovo oggetto dati selectedEmployeeRecord e viene aggiunto all'applicazione. All'area a formato libero della vista Progettazione, viene aggiunta una rappresentazione visiva dell'oggetto dati, come mostrato nella seguente immagine:



Adesso, quando viene selezionato il campo lastName nell'area di progettazione, una linea indica che è associato a selectedEmployeeRecord. Al centro della linea, l'icona del binder di testo rappresenta il SwingTextComponentBinder utilizzato per questo binding. Se si seleziona la linea o l'icona che rappresenta il binder nell'area di progettazione, è possibile visualizzare le proprietà del binder nella vista Proprietà.

Bind dei campi dei dettagli rimanenti

Per eseguire il binding dei campi dei dettagli rimanenti di un dipendente, viene seguita una procedura simile a quella relativa al campo Cognome, ma non sarà necessario aggiungere l'oggetto dati. Poiché l'oggetto dati selectedEmployeeRecord è stato già aggiunto, associare semplicemente ciascun campo alla proprietà corrispondente nell'oggetto dati selectedEmployeeRecord.

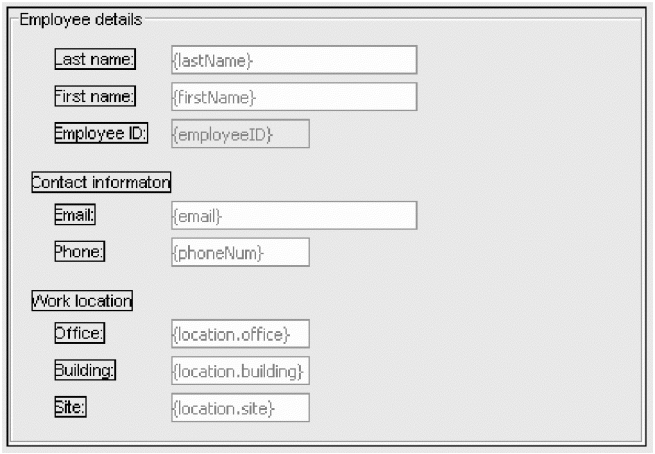
Per eseguire il binding dei campi, attenersi alle seguenti istruzioni per ciascun campo nella sezione Dipendenti dell'applicazione:

1. Selezionare il campo nella vista Progettazione e fare clic sulla scheda **Binding**.
2. Nella finestra Binding di dati campo, selezionare selectedEmployeeRecord dall'elenco **Oggetti dati**.
3. Nell'elenco **Proprietà oggetto dati**, selezionare la proprietà del campo per il quale si sta eseguendo il binding. Il seguente grafico mostra la proprietà a cui associare ciascun campo di testo:

Campo	**Proprietà nell'oggetto dati selectedEmployeeRecord**
lastNameField	lastName
firstNameField	firstName
idField	employeeID
emailField	email
phoneField	phoneNum
officeField	location.office
buildingField	location.building
siteField	location.site

4. Scegliere **OK**.

Quando il binding dei campi di testo viene completato, l'area di progettazione risulterà simile alla seguente immagine:



Impostazione del campo ID dipendente in sola lettura

Il campo ID dipendente è disabilitato perché la proprietà modificabile nel campo è impostata su false. Tuttavia, il comportamento predefinito del binder del campo cambia lo stato del campo quando l'oggetto dati contiene un valore. È possibile disattivare questo comportamento del binder in modo che il campo rimanga nello stato iniziale di sola lettura.

Per impedire che il binder modifichi automaticamente la proprietà editabile, procedere come segue

1. Selezionare il campo ID dipendente. Nell'area di progettazione viene visualizzata una linea e un'icona che rappresenta il binder del campo.
2. Fare clic sull'icona del binder per il campo ID dipendente.
3. Nella vista Proprietà, modificare la proprietà autoEditable in **false**. Premere **Invio**.

Riepilogo della lezione

Adesso, quando si esegue l'applicazione e si seleziona il dipendente dalla tabella, i dettagli di quel record dipendente verranno visualizzati nei campi dei dettagli.

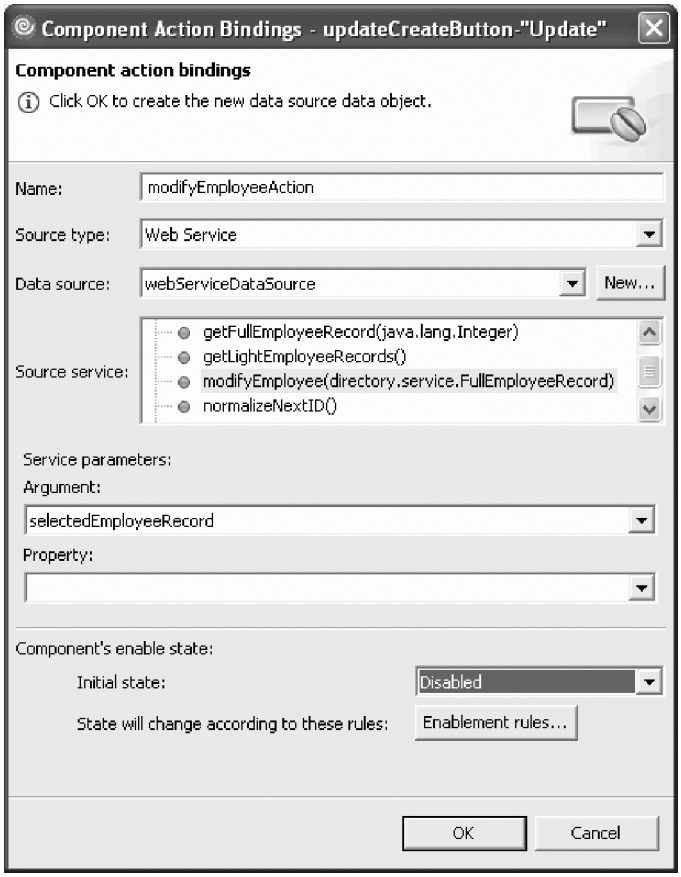
Lezione 2.4: Bind del pulsante Aggiorna ad un binder di azione

L'editor visivo Java fornisce binder di azioni per il richiamo di un servizio su un'origine dati facendo clic su un pulsante. Ad esempio, facendo clic su Aggiorna, l'applicazione dovrebbe eseguire un metodo modifyEmployee() sul servizio Web in modo che le modifiche vengano inserite nei campi dei dettagli. In questa lezione il pulsante Aggiorna verrà associato ad un binder di azione.

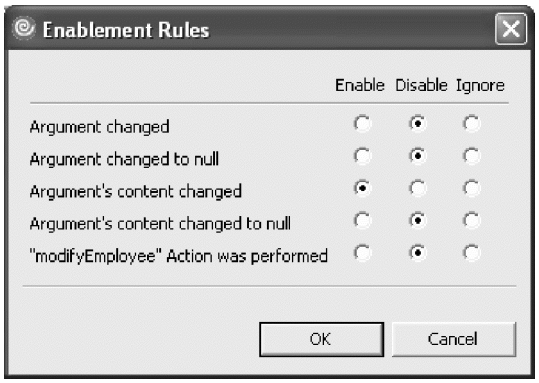
Per associare il pulsante Aggiorna:

1. Selezionare il pulsante **Aggiorna** nell'area di progettazione e scegliere la scheda **Binding** per aprire la finestra Binding azioni componente.

2. Nel campo **Tipo di origine**, selezionare **Servizio Web**.
3. Nel campo **Origine dati**, selezionare **webServiceDataSource**.
4. Nell'elenco **Servizio di origine**, selezionare **modifyEmployee(directory.service.FullEmployeeRecord)**.
5. Il campo **Nome** viene automaticamente modificato in **modifyEmployeeAction**. Accettare questo valore predefinito.
6. Nel campo **Argomento**, selezionare **selectedEmployeeRecord**.
7. Poiché il metodo modifyEmployee() utilizza un record di dipendente completo come argomento, lasciare vuoto il campo **Proprietà**.
8. Impostare lo **Stato iniziale** del pulsante su **Disabilitato**.



9. Per definire le modifiche allo stato del pulsante, scegliere **Abilitazione regole**. Specificare che il pulsante deve essere abilitato solo quando viene modificato il contenuto dell'argomento e disabilitato in tutte le altre istanze. Scegliere **OK**.



   Indica che il pulsante **Aggiorna** rimane disabilitato fino a quando il contenuto di selectedEmployeeRecord non viene modificato. In altre parole, quando viene immesso un nuovo valore in uno dei campi dei dettagli, associati a selectedEmployeeRecord, il binder abiliterà il pulsante. Se si seleziona un nuovo record o si sceglie **Aggiorna**, il pulsante verrà nuovamente disabilitato.

10. Scegliere **OK**.

Per il pulsante **Aggiorna** viene aggiunto un nuovo binder SwingDataServiceAction. Selezionando il pulsante nell'area di progettazione, l'editor visivo traccia una linea che indica che il pulsante è associato all'origine dati del servizio Web. Una freccia punteggiata, rosa, va dall'oggetto selectedEmployeeRecord alla linea. Questa freccia indica che il selectedEmployeeRecord è l'argomento per la chiamata al servizio.

Riepilogo della lezione

Adesso, quando si esegue l'applicazione, è possibile aggiornare i record dei dipendenti.

Selezionare un dipendente nella tabella e modificarne il cognome. Quando il cognome viene modificato, il pulsante **Aggiorna** viene abilitato. Scegliendo **Aggiorna**, il servizio modifyEmployee viene richiamato e il dipendente viene aggiornato. Il nuovo cognome viene visualizzato nella tabella dei dipendenti.

Lezione 2.5: Abilitazione del pulsante Elimina e della finestra di conferma

In questa lezione, l'applicazione My Company Directory viene programmata per eliminare un record di dipendente.

L'elenco seguente descrive il comportamento dell'applicazione:

- Quando si seleziona un dipendente nella tabella, il pulsante **Elimina** viene abilitato.
- Quando si fa clic sul pulsante **Elimina**, viene visualizzata la finestra di dialogo Conferma eliminazione che richiede di confermare dell'eliminazione.
- Scegliendo **Sì** nella finestra di dialogo Conferma eliminazione, il record del dipendente viene eliminato, la finestra Conferma eliminazione viene chiusa e l'elenco di dipendenti viene aggiornato.
- Scegliendo **No**, l'eliminazione viene annullata e la finestra di dialogo Conferma eliminazione viene chiusa.

Programmazione del pulsante Elimina in modo che venga abilitato o disabilitato in base alla selezione di una riga nella tabella

Per programmare l'abilitazione e la disabilitazione del pulsante Elimina, aggiungere un listener alla tabella che abiliti il pulsante in base alla selezione di una riga.

1. Selezionare employeesTable nella vista Bean Java. La vista di origine evidenzia la riga seguente:

   ```
   employeesTable = new JTable();
   ```

2. Dopo questa riga, aggiungere un nuovo evento ListSelectionListener e valueChanged ad employeesTable:

   ```
   employeesTable.getSelectionModel().addListSelectionListener(new ListSelectionListener() {
      public void valueChanged(ListSelectionEvent e) {
        getDeleteButton().setEnabled(getEmployeesTable().getSelectedRowCount() != 0);
      }
   });
   ```

3. Dopo aver aggiunto le righe di codice, l'editor di origine le contrassegna come errori fino a quando non si importa ListSelectListener e ListSelectionEvent. Per aggiungere le importazioni richieste, scegliere **Origine** → **Organizza importazioni** nel menu principale. Le seguenti righe vengono aggiunte alla sezione delle importazioni della classe:

   ```
   import javax.swing.event.ListSelectionEvent;
   import javax.swing.event.ListSelectionListener;
   ```

Quando si seleziona una riga nella tabella, il pulsante **Elimina** viene abilitato.

Programmazione della finestra di Conferma eliminazione affinché venga aperta quando si fa clic su Elimina

Aggiungere un evento actionPerformed al pulsante Elimina e programmare l'evento affinché apra la finestra di dialogo Conferma eliminazione.

1. Selezionare il pulsante **Elimina** con il pulsante destro del mouse e scegliere **Eventi** → **actionPerformed**. Il seguente stub di evento viene aggiunto al metodo getDeleteButton():

```
deleteButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent e) {
     System.out.println("actionPerformed()");
     // TODO Auto-generated Event stub actionPerformed()
   }
});
```

2. Sostituire questo stub generato con il seguente codice, che imposta la finestra Conferma eliminazione in modo che sia visibile quando si fa clic sul pulsante:

```
deleteButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent e) {
     getConfirmDialog().setVisible(true);
   }
});
```

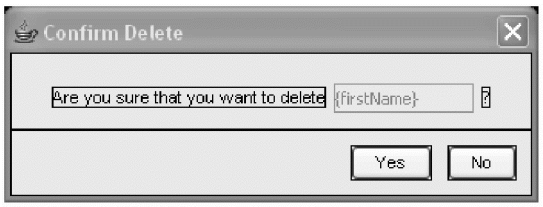
Binding del campo di testo nella finestra Conferma eliminazione

Eseguire il binding del campo di testo nella finestra Conferma eliminazione per visualizzare il nome del dipendente da eliminare.

1. Nella vista Bean Java o nell'area di progettazione, selezionare il campo di testo employeeToDeleteField e fare clic sulla scheda **Binding**.

2. Nella finestra Binding di dati campo, selezionare l'oggetto dati selectedEmployeeRecord e il campo firstName, quindi scegliere **OK**.

   Il campo di testo è ora associato alla colonna firstName della riga selezionata in employeesTable.



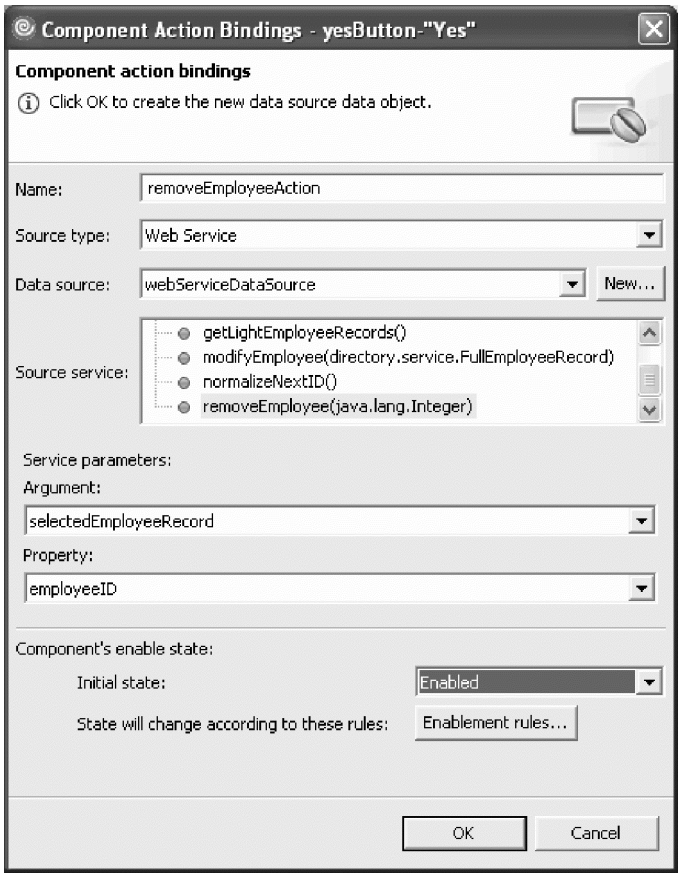
3. Per assicurarsi che questo campo sia di sola lettura, impostare la proprietà **autoEditable** per il binder del campo su **false**.

Binding del pulsante Sì per eseguire l'eliminazione

Eseguire il binding del pulsante **Sì** per richiamare il metodo removeEmployee(java.lang.Integer) sul servizio Web.

1. Selezionare il pulsante **Sì** nell'area di progettazione e scegliere la scheda **Binding** per aprire la finestra Binding azioni componente.
2. Nel campo **Tipo di origine**, selezionare **Servizio Web**.
3. Nel campo **Origine dati**, selezionare **webServiceDataSource**.
4. Nell'elenco **Servizio di origine**, selezionare **removeEmployee(java.lang.Integer)**.
5. Il campo **Nome** viene automaticamente modificato in **removeEmployeeAction**. Accettare questo valore predefinito.
6. Nel campo **Argomento**, selezionare **selectedEmployeeRecord**.
7. Nel campo **Proprietà**, selezionare **employeeID**. Poiché il metodo removeEmployee() utilizza un numero intero come argomento, utilizzare l'ID dipendente del selectedEmployeeRecord.
8. Impostare lo **Stato iniziale** del pulsante su **Abilitato**.
9. Per **Regole di abilitazione**, selezionare **Ignora** per ciascuna condizione.

Lo stato di questo componente indica che il pulsante Sì sarà sempre abilitato, perché non c'è bisogno di modificarlo.



10. Scegliere **OK**.

Aggiunta di un evento per nascondere la finestra Conferma eliminazione dopo l'eliminazione del dipendente

A questo punto, è necessario aggiungere un evento al *binder* del pulsante **Sì** (non allo stesso pulsante **Sì**). Inoltre, bisognerà impostare la finestra Conferma eliminazione affinché venga chiusa dopo la rimozione del dipendente, ovvero dopo che il binder ha richiamato correttamente il servizio sull'origine dati.

Aggiungere il seguente codice al metodo getRemoveEmployeeAction():

```
removeEmployeeAction.addActionBinderListener(new jve.generated.IActionBinder.ActionBinderListener() {
   public void afterActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {
     getConfirmDialog().setVisible(false);
   }
   public void beforeActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {}
});
```

Il codice di evento nasconde la finestra Conferma eliminazione dopo aver eseguito l'azione del binder.

Riepilogo della lezione

Adesso, quando viene eseguita l'applicazione My Company Directory, sarà possibile selezionare un dipendente nella tabella, fare clic su **Elimina** e scegliere **Sì** per confermare l'eliminazione. Il record del dipendente verrà eliminato dalla struttura e l'elenco dei dipendenti verrà aggiornato.

Lezione 2.6: Impostare azioni e binding per l'aggiunta di un nuovo dipendente

In questa lezione verrà abilitata l'applicazione My Company Directory per l'aggiunta del record di un nuovo dipendente.

Poiché il comportamento dell'applicazione è più complicato e dinamico per l'aggiunta di un nuovo dipendente, questo esercizio è più complesso e verrà richiesto di effettuare alcune modifiche manuali al codice di origine. Inoltre, questo esercizio mostra alcune funzioni avanzate degli oggetti dati, e fornisce un esempio dei diversi modi in cui è possibile utilizzare i binder e gli oggetti dati.

Il seguente elenco descrive il comportamento previsto dell'applicazione:

- Scegliendo **Nuovo**, si verifica il seguente comportamento:
  - La selezione viene cancellata nella tabella dei dipendenti e la tabella viene disabilitata.
  - La cancellazione della selezione nella tabella fa sì che il pulsante **Elimina** venga disabilitato.
  - Viene disabilitato il campo **Filtro**.
  - I valori nel campo dei dettagli vengono cancellati, fatta eccezione per il nuovo ID dipendente.
  - Il testo del pulsante **Aggiorna** diventa **Aggiungi**.
- Scegliendo **Aggiungi**, si verifica il seguente comportamento:
  - I valori immessi nei campi dei dettagli vengono aggiunti alla struttura come nuovo record di dipendente.
  - La tabella viene abilitata e i valori vengono aggiornati.
  - Il campo **Filtro** viene abilitato.
  - Il testo del pulsante **Aggiungi** diventa **Aggiorna**.

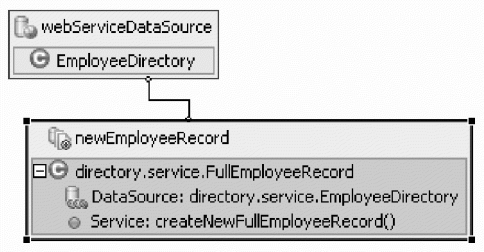
Aggiunta di un nuovo Oggetto dati dell'origine dati che richiama createNewFullEmployeeRecord()

Il servizio Web di esempio fornisce un servizio createNewFullEmployeeRecord che a sua volta fornisce un record di dipendente nuovo, vuoto in cui verrà inserito il primo numero di ID dipendente disponibile. In questo record vuoto possono successivamente essere inserite le informazioni relative al nuovo dipendente e inoltrate al servizio Web.

1. Nella tavolozza dell'editor visivo Java, espandere il cassetto Oggetti di dati e selezionare **Oggetto dati dell'origine Java**.
2. Spostare il mouse sull'area vuota della vista di progettazione o sull'area a formato libero e fare clic con il pulsante sinistro del mouse per rilasciare l'Oggetto dati dell'origine Java. Un nuovo Oggetto dati dell'origine dati viene aggiunto e mostrato nell'area a formato libero:

3. Selezionare l'oggetto dati dell'origine dati con il pulsante destro del mouse e scegliere **Rinomina campo**. Rinominare l'oggetto dati in `newEmployeeRecord`.
4. Selezionare l'oggetto dati newEmployeeRecord con il pulsante destro del mouse e selezionare **Proprietà di binding**. Viene visualizzata la finestra Binding di dati.
5. Nel campo **Origine dati**, selezionare webServiceDataSource
6. Nel campo **Servizio**, selezionare createNewFullEmployeeRecord()
7. Scegliere **OK**.

Nell'area a formato libero, viene visualizzato l'oggetto dati dell'origine dati newEmployeeRecord associato al servizio Web.



Aggiunta di un Oggetto dati di base per facilitare l'alternanza degli oggetti dati

Poiché i campi dei dettagli e il pulsante Aggiorna devono alternare le modalità (sia per eseguire un aggiornamento che per la creazione di un nuovo dipendente), dovranno essere associati a due diversi oggetti dati in due momenti diversi. Per semplificare queste operazioni, aggiungere un oggetto dati di base chiamato switchingDataObject. Tale oggetto verrà utilizzato per alternare il binding per i campi di testo tra selectedEmployeeRecord e newEmployeeRecord.

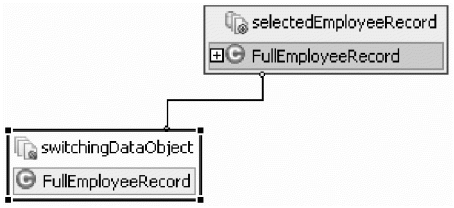
Il nuovo oggetto dati di base, indica semplicemente un altro oggetto dati (selectedEmployeeRecord) definito in uno degli esercizi precedenti. Questo nuovo oggetto dati risulterà utile durante la creazione di un metodo che indica all'oggetto di utilizzare il newEmployeeRecord creato precedentemente. In altre parole, l'oggetto dati di base funzionerà come oggetto dati intermedio che passa dall'oggetto dati selectedEmployeeRecord all'oggetto dati newEmployeeRecord, consentendo ai componenti visivi dell'applicazione di utilizzare due diversi oggetti dati.

1. Nella tavolozza dell'editor visivo, selezionare **Oggetto dati di base**, e rilasciarlo nell'area a formato libero. Viene aggiunto un basicDataObject.

2. Ridenominare l'oggetto dati in `switchingDataObject`
3. Nella vista Proprietà di switchingDataObject, impostare la proprietà **sourceObject** su **selectedEmployeeRecord**. È possibile selezionare selectedEmployeeRecord dal menu a discesa nella colonna Valore della proprietà.

   Adesso, switchingDataObject fa riferimento a selectedEmployeeRecord e ne rispecchia gli stessi valori:

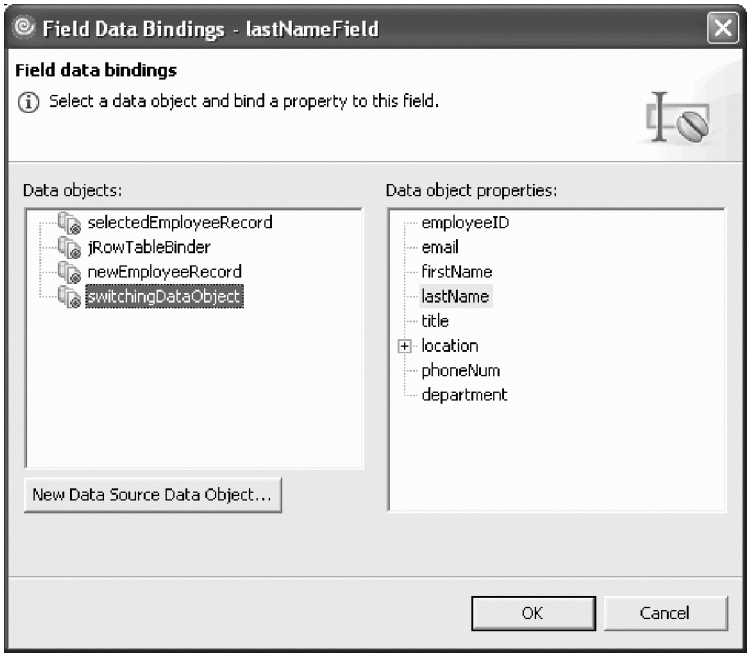
   

Rebinding di ciascun campo di dipendente a switchingDataObject

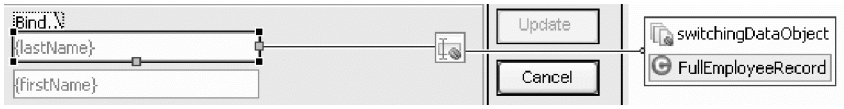
Anche se i campi dei dettagli del dipendente sono già associati a selectedEmployeeRecord, adesso verranno associati a switchingDataObject. Dopo aver associato i campi, sarà possibile passare dinamicamente da un oggetto dati all'altro per i campi, in base alle modifiche apportate al record di un dipendente esistente o all'aggiunta di un nuovo record.

Per ciascuno dei campi nella sezione Dettagli dipendente, procedere come segue:

1. Selezionare il campo e fare clic sulla scheda **Binding**.
2. Nella finestra Binding di dati campo, selezionare switchingDataObject. Precedentemente sono stati associati i campi a selectedEmployeeRecord.



3. Assicurarsi che il campo sia ancora associato alla proprietà dell'oggetto dati e scegliere **OK**. Se si seleziona il campo nella vista di progettazione, è possibile vedere che le linee del binder adesso indicano l'oggetto switchingDataObject.



Definizione di un indicatore e di un metodo per l'aggiornamento e l'alternanza delle modalità

Il seguente metodo updateMode() controlla se il modo è impostato su Nuovo, quindi modifica alcuni comportamenti dell'applicazione di conseguenza. Per impostazione predefinita, l'indicatore booleano isNewMode è impostato su false, il metodo updateMode() abilita la tabella dipendenti e il campo dei filtri e imposta il testo sul pulsante Aggiorna su "Aggiorna". Se isNewMode è impostato su true, la tabella dei dipendenti viene disabilitata e qualsiasi selezione sarà annullata, il campo filtro viene disabilitato e il testo sul pulsante Aggiorna viene impostato su "Aggiungi".

Aggiungere il seguente codice alla classe DirectoryApp.java prima dell'ultima parentesi graffa chiusa:

```
private boolean isNewMode = false;
private void updateMode() {
   if (isNewMode) {
      getEmployeesTable().clearSelection();
      getEmployeesTable().setEnabled(false);
      getFilterField().setEditable(false);
      getUpdateCreateButton().setText("Add");
   } else {
      getEmployeesTable().setEnabled(true);
      getFilterField().setEditable(true);
      getUpdateCreateButton().setText("Update");
   }
}
```

Aggiunta di un evento actionPerformed al pulsante Nuovo

In questa fase, verrà aggiunto il codice di evento per il pulsante **Nuovo**. L'evento indica a switchingDataObject di utilizzare l'oggetto dati newEmployeeRecord, imposta l'indicatore di modalità su "new" ed esegue il metodo updateMode() aggiunto precedentemente.

1. Nella vista di progettazione, selezionare il pulsante **Nuovo** con il tasto destro del mouse e scegliere **Eventi** → **actionPerformed**. Nel metodo getNewButton() viene generato il seguente codice:

```
newButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent e) {
      System.out.println("actionPerformed()"); // TODO Auto-generated Event stub actionPerformed()
   }
});
```

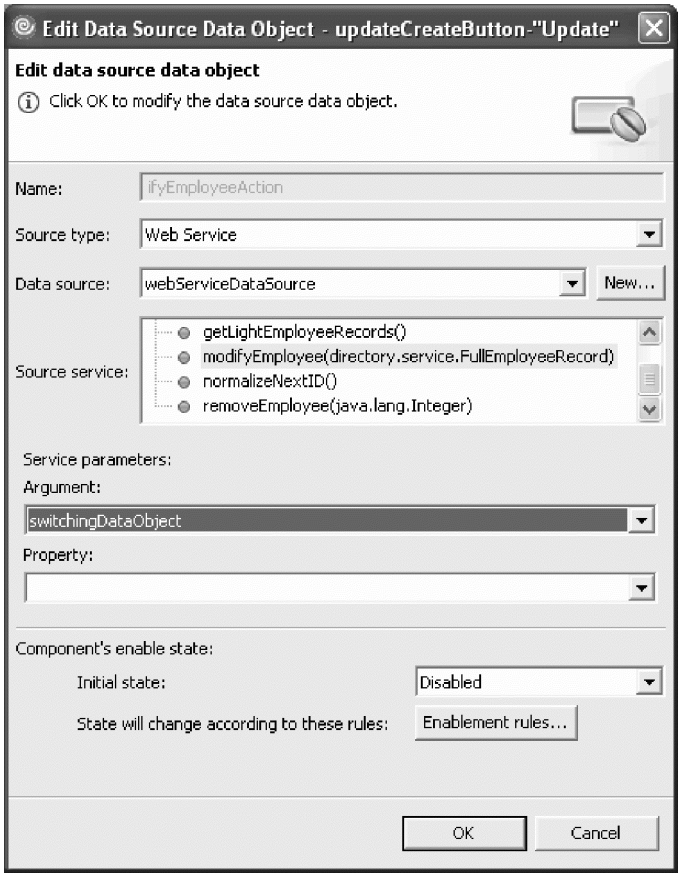
2. Sostituire lo stub generato con il seguente codice:

```
newButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent e) {
      getSwitchingDataObject().setSourceObject(getNewEmployeeRecord());
      getNewEmployeeRecord().refresh();
      isNewMode = true; //sets application to new mode
      updateMode(); //changes UI according to new mode
      getLastNameField().grabFocus();
   }
});
```

Rebinding del pulsante Aggiorna

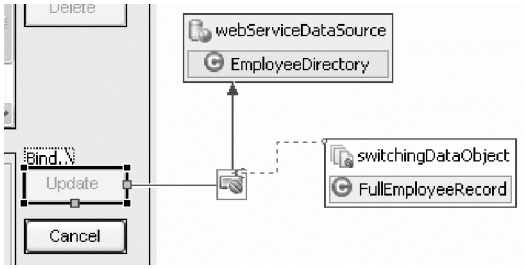
In una delle lezioni precedenti, è stato programmato il pulsante **Aggiorna** affinché utilizzi il metodo modifyEmployee sul servizio Web. Questa azione viene implementata come una SwingDataServiceAction. Una delle proprietà di SwingDataServiceAction è l'oggetto di origine, che agisce come argomento per il servizio. L'oggetto di origine per l'azione di modifica è al momento impostato su selectedEmployeeRecord. Per programmare il pulsante affinché controlli sia l'aggiornamento che l'aggiunta, è necessario riconfigurare l'azione del pulsante per utilizzare switchingDataObject come argomento nel servizio modifyEmployee.

1. Nella vista di progettazione, selezionare il pulsante **Aggiorna**. Osservare la freccia rosa, punteggiata che mostra che selectedEmployeeRecord è l'argomento per la chiamata al servizio.
2. Fare clic sulla scheda **Binding** nel pulsante **Aggiorna**.
3. Nel campo **Argomento**, selezionare switchingDataObject.



4. Scegliere **OK**.

   A questo punto, osservare che l'azione del pulsante è configurata per utilizzare switchingDataObject come argomento per il metodo modifyEmployee:



Aggiunta di un evento al binder del pulsante Aggiorna per reimpostare la modalità

Dopo aver fatto clic su **Aggiorna** e una volta che l'azione è stata completata sul servizio Web, ripristinare la modalità e il comportamento predefinito dell'applicazione. Per ottenere questo risultato, aggiungere un listener di eventi al binder di azioni del pulsante che aggiornerà la modalità e aggiornerà la tabella dopo aver eseguito l'aggiornamento o l'aggiunta.

Aggiungere il seguente codice al metodo getModifyEmployeeAction() per il pulsante Aggiorna:

```
modifyEmployeeAction.addActionBinderListener(new jve.generated.IActionBinder.ActionBinderListener() {
    public void afterActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {
      if (isNewMode) {
        //Go back to using the selectedEmployeeRecord
        getSwitchingDataObject().setSourceObject(getSelectedEmployeeRecord());
        //Revert out of new mode
        isNewMode = false;
        updateMode();
      }
      // Refresh the table's data object
```

```
      getLightEmployeeRecordRows().refresh();
    }
    public void beforeActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {}
});
```

Riepilogo della lezione

Adesso, quando si esegue l'applicazione My Company Directory, sarà possibile aggiungere un nuovo record di dipendente scegliendo **Nuovo**.

Lezione 2.7: Programmazione del comportamento del pulsante Annulla

Quando su utilizza l'applicazione, è necessario poter annullare facilmente le modifiche apportate ai record dei dipendenti se si decide di non inoltrare le modifiche. In altre parole, deve essere possibile annullare e cancellare i campi modificati per poter ricominciare dalla situazione originale. Per aggiungere questa funzionalità, impostare alcuni eventi actionPerformed sul pulsante **Annulla**.

Il seguente elenco descrive il comportamento previsto dal pulsante **Annulla**:

- Se si fa clic sul pulsante **Annulla** nella nuova modalità, l'applicazione esce dalla nuova modalità.
- Se si fa clic sul pulsante **Annulla** durante la modifica di un record di dipendente, tutti i valori modificati verranno ripristinati sui valori originali.

Per aggiungere un evento actionPerformed al pulsante **Annulla** per eseguire il comportamento richiesto, procedere come segue:

1. Nella vista di progettazione, selezionare il pulsante **Annulla** con il pulsante destro del mouse e scegliere **Eventi** → **actionPerformed**. Nel metodo getCancelButton() viene generato il seguente codice:

   ```
   cancelButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent e) {
         System.out.println("actionPerformed()"); // TODO Auto-generated Event stub actionPerformed()
      }
   });
   ```

2. Sostituire lo stub di evento generato con il seguente codice:

   ```
   cancelButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent e) {
         if (isNewMode) {
            getSwitchingDataObject().setSourceObject(getSelectedEmployeeRecord());
            isNewMode = false;
            updateMode();
         } else {
            getSelectedEmployeeRecord().refresh();
         }
      }
   });
   ```

Riepilogo della lezione

In questa lezione è stato illustrato come programmare il pulsante **Annulla** con gli eventi actionPerformed.

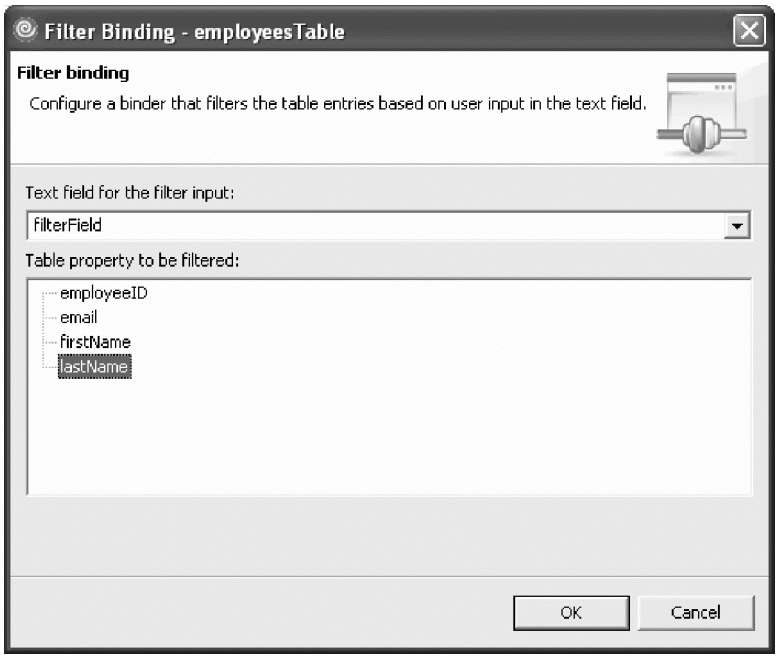
Lezione 2.8: Impostazione di un filtro nella tabella dipendenti

È possibile utilizzare un Binder di filtro testo per filtrare i contenuti della tabella dipendenti. Il filtro utilizza l'input da un campo di testo e filtra la tabella in base a una particolare proprietà, o colonna nella tabella.

Nell'applicazione verranno utilizzati i caratteri immessi nel campo **Filtro** per filtrare il contenuto in base al cognome del dipendente. Se i valori esatti immessi nel campo **Filtro** sono presenti nel cognome di un record di dipendente, il record verrà visualizzato nella tabella.

Per creare un filtro per la tabella, procedere come segue:

1. Selezionare l'icona del binder per employeesTable e selezionare **Proprietà di binding del filtro**. La casella di dialogo del binding del filtro si apre.
2. Nell'elenco **Campo di testo per l'input del filtro**, selezionare **filterField**.
3. Nell'elenco **Proprietà tabella da filtrare**, selezionare **lastName**.



4. Scegliere **OK**.

Viene generato un nuovo SwingPropertyFilter. La proprietà filtro sul binder della tabella è impostata affinché venga utilizzato il nuovo filtro. Il nuovo filtro è configurato per utilizzare il campo **Filtro** come input, e per filtrare la proprietà lastName della tabella.

Riepilogo della lezione

In questa lezione è stato illustrato come impostare un filtro per una tabella.

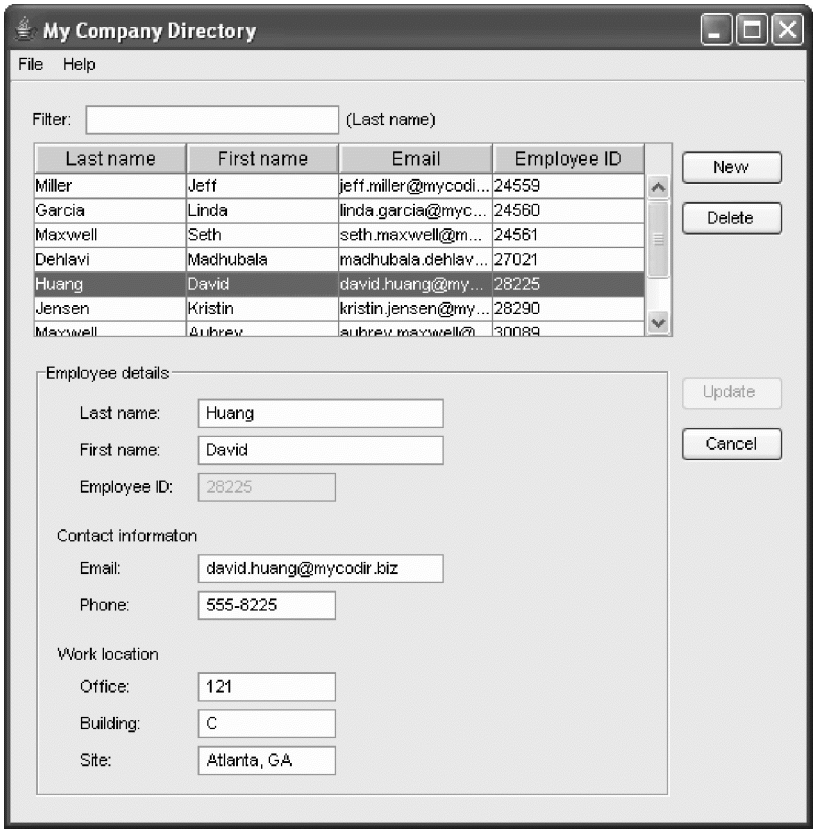
Adesso, quando si esegue l'applicazione My Company Directory, sarà possibile immettere una stringa nel campo **Filtro**, in modo che nella tabella vengano visualizzate solo le righe in cui il cognome contiene la stringa specificata.

Congratulazioni. L'applicazione Directory dell'azienda è stata completata.

Riepilogo: Generazione di un rich client Java che utilizza un servizio Web

Congratulazioni. È stato illustrato come utilizzare l'editor visivo Java per generare l'applicazione My Company Directory, un rich client Java che si connette ad un servizio Web di esempio per gestire le informazioni relative ai dipendenti di un'azienda.

Visualizza un'immagine del prodotto finito:



Lezioni svolte

Per completare l'interfaccia utente grafica, è stato utilizzato l'editor visivo e GridBagLayout per organizzare la tabella dipendenti. Quindi è stato eseguito il binding della tabella, dei campi, dei pulsanti agli oggetti e alle origini dati appropriati per far sì che l'applicazione possa funzionare con un proxy Java di servizi Java generato dall'utente. Inoltre è stato creato del codice complesso per far sì che l'applicazione funzioni correttamente e che risulti semplice e di facile comprensione. Inoltre, si è appreso come installare un'applicazione enterprise su Application Server v6.0 e a distribuire un servizio Web.

Infine, si è avuta una visione generale della classe di binder fornita con l'editor visivo Java per la gestione dei dati. A questo punto si dispone delle conoscenze necessarie per cominciare a sperimentare i diversi usi dei binder.

È adesso possibile completare le seguenti attività:

- Uso dell'editor visivo Java per la disposizione dei componenti in un GridBagLayout.
- Esecuzione di una classe visiva come bean Java.
- Binding dei componenti visivi dell'applicazione Java ai metodi e agli oggetti dati restituiti da un servizio Web.
- Aggiunta di eventi ai componenti visivi.

Ulteriori risorse

Importazione di una versione finita dell'applicazione My Directory

Questo progetto include l'applicazione finita, il pacchetto jve.generated con le classi binder ed il client Java del servizio Web configurate per WebSphere Application Server v6.1. Se si importa questo progetto

finito senza lavorare nell'esercitazione, potrebbe essere necessario configurare la variabile del percorso del build Java. È necessario puntare al file JAR del client thin dei servizi Web di WebSphere v6.1.

Suggerimento: A meno che durante l'importazione non venga specificato un nome progetto diverso, questo sovrascrive i contenuti del progetto MyDirectory.