Einen Rich-Java-Client erstellen, der einen Web-Service verwendet

Inhaltsverzeichnis

Einen Rich-Java-Client erstellen, der einen Web-Service verwendet

Mit Hilfe dieses Lernprogramms lernen Sie, den Java Visual Editor zum Erstellen eines Rich-Java-Clients zu verwenden, der eine Verbindung zu einem Web-Service herstellt. Der von Ihnen im Lernprogramm erstellte Client heißt 'My Company Directory'.

Bei 'My Company Directory' handelt es sich um eine Java-Anwendung, die zum Verwalten des Mitarbeiterverzeichnisses eines Unternehmens verwendet wird. Die Anwendung stellt eine Verbindung zu einem Beispiel-Web-Service her, der Methoden zum Erstellen, Abrufen, Aktualisieren und Löschen von Mitarbeiterdatensätzen bereitstellt.

Der Client wird im Java Visual Editor mit Hilfe von Swing-Komponenten visuell erstellt. Der Java Visual Editor bietet eine Reihe von Helper-Klassen (Datenquellen, Datenobjekte und Binder) für die Verbindung und Arbeit mit dem Web-Service. Der Web-Service wird lokal in Ihrer eigenen Installation von IBM WebSphere Application Server v6.0 implementiert. Die Tools helfen Ihnen beim Generieren eines Java-Proxy für Ihren Client auf der Basis einer WSDL-Datei (Web Services Description Language).

Fertiges Produkt anzeigen

Lernziele

Mit diesem Lernprogramm lernen Sie Folgendes:

- Verwenden des Java Visual Editor zum Entwerfen und Erstellen eines Layouts einer Benutzerschnittstelle
- Binden von Schnittstellenelementen an Datenobjekte und einen Web-Service

Erforderliche Zeit

2 Sunden 15 Minuten

Zugehörige Informationen

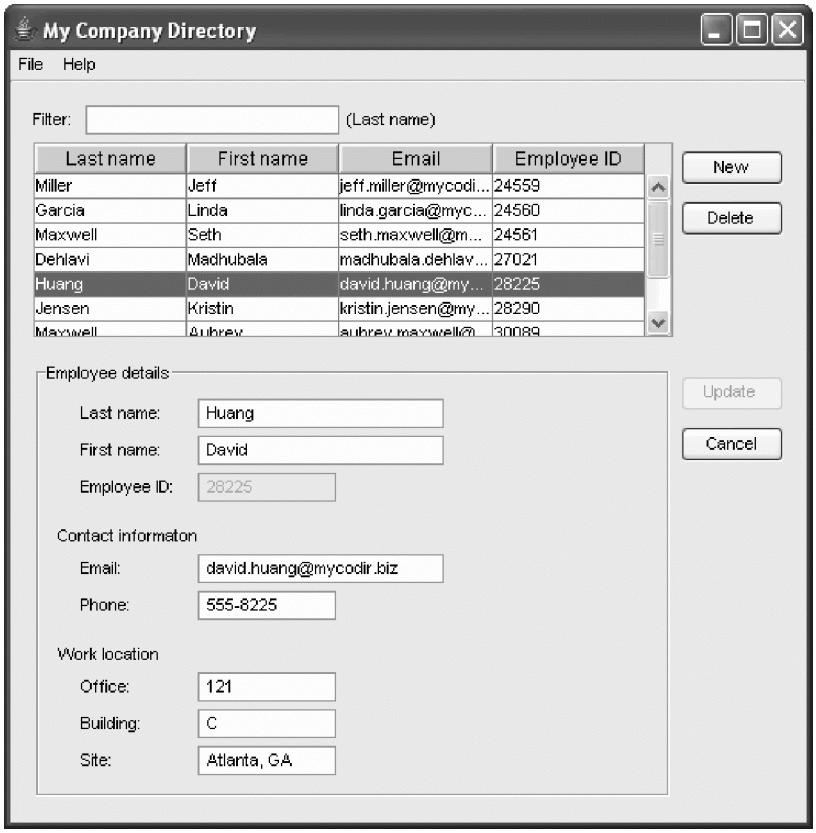
PDF-Version anzeigen

Lernprogramm: Hello World Java

Einführung: Einen Rich-Java-Client erstellen, der einen Web-Service verwendet

Die grafische Benutzerschnittstelle (GUI, Graphical User Interface) für den Client wurde unter Verwendung von Swing-Komponenten zum größten Teil visuell vorgefertigt. Im ersten Modul entwerfen Sie die wichtigsten GUI-Komponenten unter Verwendung des Java Visual Editor. Im zweiten Modul binden Sie die GUI-Komponenten an die Web-Service-Datenquelle, Services und die von der Datenquelle zurückgegebenen Objekte. Beim Erstellen der Anwendung im Java Visual Editor verwenden Sie Datenquellen, Datenobjekte und Binder, die Instanzen von Helper-Klassen sind. Die Helper-Klassen werden vom Java Visual Editor generiert und von Ihrer Anwendung verwendet.

Ein Bild des fertigen Produkts anzeigen:



Lernziele

Mit diesem Lernprogramm lernen Sie Folgendes:

- Verwenden des Java Visual Editor zum Entwerfen und Erstellen eines Layouts einer Benutzerschnittstelle
- Binden von Schnittstellenelementen an Datenobjekte und einen Web-Service

Erforderliche Zeit

Für die Bearbeitung des gesamten Lernprogramms sind ca. 2 Stunden 30 Minuten zu veranschlagen.

Systemvoraussetzungen

- WebSphere Application Server v6.1. Dieser Server wurde unter Umständen bereits mit Ihrem Produkt installiert. Sie können allerdings auch Ihre eigenständige Installation verwenden. Das Szenario in diesem Lernprogramm erfordert, dass Sie einen Beispiel-Web-Service auf dem WebSphere Application Server implementieren, der lokal aktiv ist.

Es kann sein, dass der Beispiel-Web-Service auch auf anderen Servern funktioniert, dieses Lernprogramm wurde jedoch nur mit WebSphere Application Server v6.0 und v6.1 getestet.

Voraussetzungen

Sie müssen mit den folgenden Konzepten vertraut sein:

- Grundlagen der Java-Entwicklung
- Grundlegende Web-Service-Prinzipien

- Grundlegende Workbench-Fertigkeiten, z. B. Arbeiten mit Projekten und Navigieren durch Perspektiven und Sichten

Modul 1: Client-GUI im Visual Editor entwerfen

In diesem Modul lernen Sie, wie Sie den Java Visual Editor verwenden können, um eine visuelle Komponente zu einer Anwendung hinzuzufügen, diese grafisch anzuordnen und Zusammenstellungseinschränkungen zu konfigurieren. Die abschließende Lerneinheit dieses Moduls zeigt, wie Sie die Java-Datei ausführen, um zu sehen, wie sie als eigentliche Anwendung aussehen wird.

Hinweis: Bevor Sie mit diesem Modul beginnen, sollten Sie die erforderlichen Vorkenntnisse besitzen, die in der Einführung zum Lernprogramm beschrieben wurden.

Lernziele

Nach den Lerneinheiten in diesem Modul verstehen Sie folgende Konzepte und Vorgehensweisen:

- Eine JTable in einer Java-Schnittstelle hinzufügen und ein Layout dafür erstellen
- Eine visuelle Klasse ausführen, um die eigene Arbeit zu testen

Erforderliche Zeit

Für die Bearbeitung dieses Moduls benötigen Sie ca. 15 Minuten.

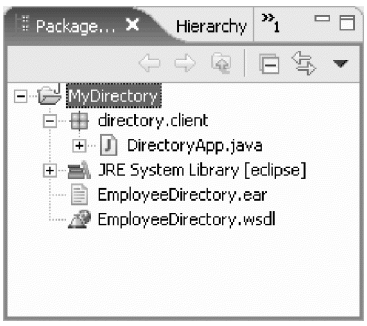
Lerneinheit 1.1: Das Java-Projekt konfigurieren

In dieser Lerneinheit konfigurieren Sie das Projekt 'MyDirectory', indem Sie ein Projekt in Ihren Arbeitsbereich importieren. Das Projekt umfasst eine einzige Java-Klasse, sowie weitere Dateien, die zu einem späteren Zeitpunkt verwendet werden.

Da der Hauptfokus dieses Lernprogramms darauf liegt, visuelle Komponenten an einen Web-Service zu binden, wurde der Großteil der Java-GUI für die Anwendung "My Company Directory" bereits für Sie entworfen.

Das Projekt 'MyDirectory' ist das Java-Projekt, mit dem Sie in diesem Lernprogramm hauptsächlich arbeiten werden. Es enthält die Datei DirectoryApp.java. Diese Java-Datei enthält ihrerseits die Java-Hauptanwendung, die Sie erstellen. Dieses Lernprogramm enthält mehrere Versionen von 'MyDirectory', um Ihnen zu helfen: eine Version für den Beginn der einzelnen Module, sowie eine fertige Version des abgeschlossenen Projekts.

1. Projekt 'MyDirectory' importieren.
2. Stellen Sie im Paketexplorer der Java-Perspektive sicher, dass die Darstellung des Projekts 'MyDirectory' der folgenden Abbildung entspricht:



Prüfpunkt für die Lerneinheit

In dieser Lerneinheit importieren Sie das Beispielprojekt 'MyDirectory', das als Ausgangspunkt für dieses Lernprogramm fungiert.

Das Projekt 'MyDirectory' umfasst die folgenden Ressourcen:

- DirectoryApp.java: Eine Java-Datei, die die Anwendung enthält, die Sie in diesem Lernprogramm entwickeln. Die Datei DirectoryApp.java befindet sich in einem Java-Paket mit dem Namen 'directory.client'.
- EmployeeDirectory.ear: Eine Unternehmensanwendung, die den Beispiel-Web-Service enthält. In Modul 2 implementieren Sie diesen Web-Service in einer lokalen Installation von WebSphere Application Server v6.0.
- EmployeeDirectory.wsdl: Eine XML-Datei, die WSDL (Web Services Description Language) zum Beschreiben des Beispiel-Web-Service verwendet, den Sie implementieren. In Modul 2 verwenden Sie diese WSDL-Datei zum Generieren eines Java-Proxy, der von Ihrer Anwendung verwendet wird.

Lerneinheit 1.2: Die Mitarbeitertabelle hinzufügen und anordnen

In dieser Lerneinheit verwenden Sie den Java Visual Editor, um ein 'JScrollPane' und eine 'JTable' zur Anwendung hinzuzufügen. In späteren Übungen programmieren Sie die JTable für den Abruf der Daten von einem Web-Service, der eine Liste aller Mitarbeiter im Unternehmensverzeichnis zurückgibt.

Nach dem Hinzufügen der JTable passen Sie das Layout der JTable in der Entwurfssicht des Java Visual Editor folgendermaßen an:

- Ziehen Sie die JTable horizontal über drei Zellen und vertikal über zwei Zellen.
- Fügen Sie links eine Einrückung von 15 Pixeln ein.
- Benennen Sie die JTable in 'employeesTable' um.

Zeigen

Öffnen Sie die Datei 'DirectoryApp.java' im Java Visual Editor

Gehen Sie wie folgt vor, um die Datei DirectoryApp.java im Java Visual Editor zu öffnen:

1. Erweitern Sie in der Sicht 'Paketexplorer' der Java-Perspektive das Projekt 'MyDirectory' und das Paket 'directory.client'.
2. Klicken Sie mit der rechten Maustaste auf die Datei DirectoryApp.java und wählen Sie **Öffnen mit → Visual Editor** aus. Der Java Visual Editor lädt die Java-Klasse und zeigt den Entwurf im grafischen Erstellungsbereich an.

Tipp:

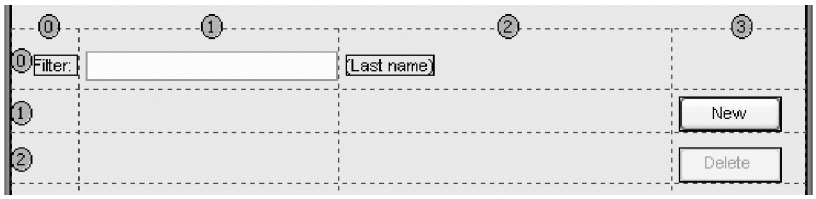
- Wählen Sie zum Ändern der von Java Visual Editor verwendeten Darstellung und Funktionsweise **Fenster → Benutzervorgaben → Java → Visual Editor** aus, und geben Sie eine Swing-Darstellung und -Funktionsweise an. Die angegebene Benutzervorgabe wird beim nächsten Öffnen der Klasse wirksam. In diesem Lernprogramm wird die Windows-Darstellung und -Funktionsweise verwendet.
- Wenn Sie den Visual Editor als Standardeditor für alle Java-Dateien verwenden möchten, können Sie auf **Fenster → Benutzervorgaben** klicken und zur Seite **Workbench → Dateizuordnungen** wechseln, um die gewünschte Benutzervorgabe zu definieren.

Eine JTable auf einem JScrollPane hinzufügen

Für das Hauptfenster von DirectoryApp.java wird ein JFrame mit einem JPanel als Hauptinhaltsteilfenster verwendet. Das JPanel in Ihrer Anwendung heißt jContentPane. Das jContentPane wurde für die Verwendung eines Typs von Layout-Manager mit dem Namen GridBagLayout definiert. Das GridBagLayout ist ein leistungsfähiges Layoutschema auf Basis eines Zellengitters, das von visuellen Komponenten belegt werden kann. Der Java Visual Editor erleichtert die Arbeit mit GridBagLayout durch die Anzeige der Gitterbegrenzungen. Darüber hinaus werden beim Ablegen neuer Komponenten im Gitter Positionsmarker angezeigt, und an Komponenten, deren Größe geändert wird oder die im GridBagLayout bewegt werden, werden Steuerungselemente angezeigt.

Gehen Sie wie folgt vor, um die Mitarbeitertabelle (eine javax.swing.JTable) der Benutzerschnittstelle DirectoryApp.java hinzuzufügen:

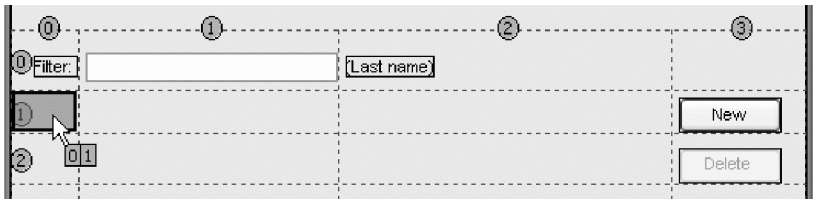
1. Klicken Sie mit der rechten Maustaste auf das jContentPane in der Entwurfssicht oder in der Sicht 'Java-Beans', und wählen Sie **Gitter anzeigen** aus. Eine rot gepunktete Linie zeigt die Gitterbegrenzung an, und blaue Kreise mit Zahlen geben die Zeilen- und Spaltennummern an. Im Beispiel belegt die Schaltfläche **Neu** die Zelle in Zeile 1 (Gitter y) und Spalte 3 (Gitter x).



2. Wählen Sie in der Palette des Java Visual Editor die Swing-Komponente **JTable on JScrollPane** aus, die dem Ablagefach **Swing-Komponenten** der Palette zugeordnet ist.

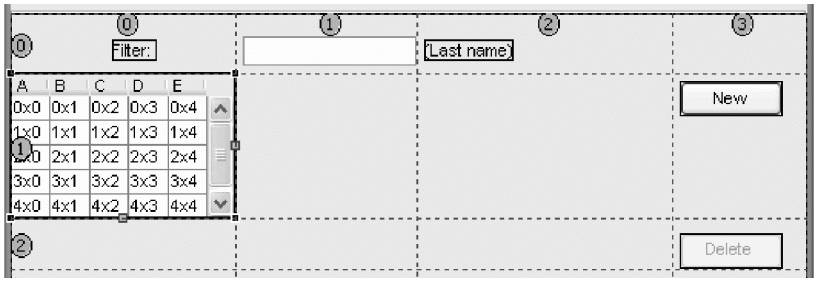
   Tipp: Standardmäßig ist die Palette auf der rechten Seite des Entwurfsbereichs ausgeblendet. Sie können die Größe der Palette ändern und sie versetzen.

3. Bewegen Sie den Mauszeiger auf die Zelle im Gitter in Spalte 0, Zeile 1:



   - Beim Bewegen des Mauszeigers über dem Gitter werden zwei nummerierte Quadrate eingeblendet, die die x- und die y-Koordinate im Gitter basierend auf der Position des Mauszeigers angeben.
   - Wenn Sie den Mauszeiger direkt auf eine Gitterbegrenzung schieben, können neue Zeilen und Spalten erstellt werden. Vorhandene Zeilen und Spalten werden dann umnummeriert. In diesem Fall deuten gelbe Quadrate auf dem Mauszeiger, gelbe Balken zwischen den Gittern und gelbe Spalten- und Zeilenbeschriftungen auf dieses Verhalten hin und verdeutlichen die Auswirkung, die diese Platzierung haben wird.

4. Klicken Sie mit der linken Maustaste, um das JScrollPane und die JTable an die Zelle bei Spalte 0 und Zeile 1 zu übergeben:

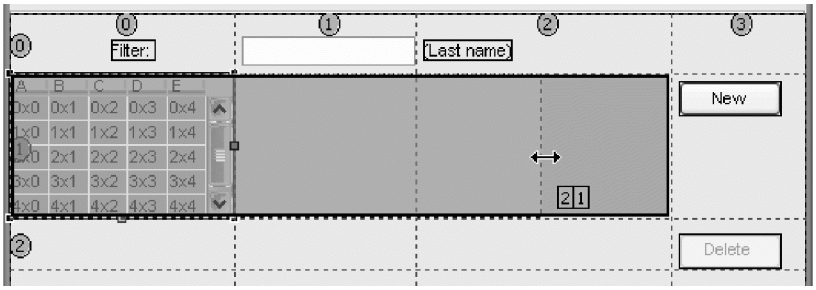


JScrollPane und JTable über mehrere Spalten und Zeilen des Gitters ausdehnen

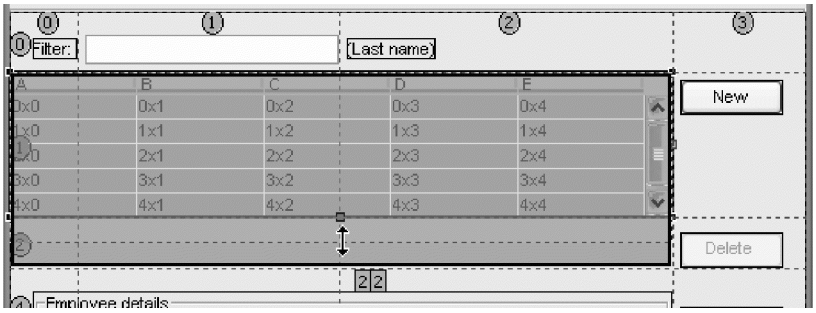
Nun müssen Sie das JScrollPane (und dessen untergeordnetes Element JTable) für ein besseres Abstands- und Größenänderungsverhalten über drei Spalten und zwei Zeilen ausdehnen. Gehen Sie wie folgt vor, um die Tabelle über die Spalten und Zeilen auszudehnen:

1. Wählen Sie das JScrollPane im Entwurfsbereich oder in der Sicht 'Java-Beans' aus (es müsste noch ausgewählt sein, weil Sie es soeben hinzugefügt haben). Beachten Sie die kleinen grünen Quadrate auf der rechten und der unteren Seite des JScrollPane. Sie verwenden diese Größensteuerungselemente zum Ziehen des JScrollPane, damit es sich über mehrere Spalten und Zeilen ausdehnt.
2. Drücken Sie die linke Maustaste auf dem grünen Steuerungselement auf der rechten Seite des JScrollPane und halten Sie sie gedrückt.

3. Ziehen Sie den Mauszeiger nach rechts, bis die Position auf Spalte 2, Zeile 1 deutet. Eine dunkelgraue Schattierung markiert ebenfalls die Zellen, die beim Loslassen der Maustaste von der Komponente belegt werden.



4. Lassen Sie die Maustaste los. Das JScrollPane erstreckt sich nun über drei Spalten.
5. Wiederholen Sie den Prozess entsprechend, um das Steuerungselement auf der unteren Seite des JScrollPane zu ziehen, bis sich das JScrollPane in Zeile 2 ausdehnt:



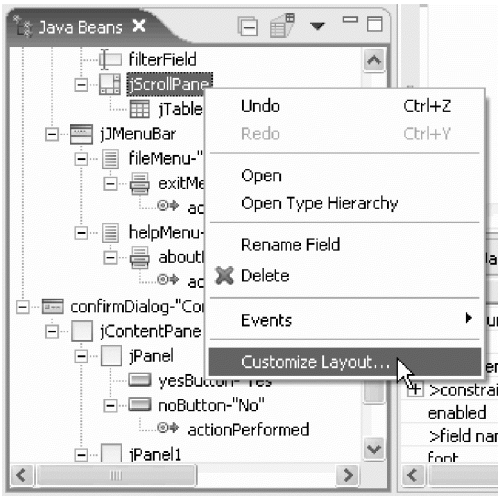
Abstände des JScrollPane innerhalb des GridBag anpassen

Ein weitere Funktion des GridBagLayout-Managers ermöglicht das Angeben verschiedener Integritätsbedingungen, um das Layout weiter anzupassen. Sie können beispielsweise die folgenden Integritätsbedingungen angeben:

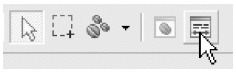
- **Anker:** Einer Komponente kann eine Ankerausrichtung innerhalb ihrer Zelle zugeordnet werden. Diese Ausrichtung hat einen Einfluss darauf, wie die Komponente bewegt wird, wenn die Größe der Anwendung von einem Benutzer geändert wird. Der Anker einer Komponente kann beispielsweise oben links, in der Mitte links, im Mittelpunkt oder unten rechts gesetzt werden.
- **Füllung:** Es kann festgelegt werden, dass eine Komponente den gesamten verfügbaren Platz innerhalb ihrer Zelle(n) entweder horizontal, vertikal oder in beiden Richtungen belegt.
- **Insets:** Einer Komponente kann oben, unten, links und rechts eine eigene Auffüllung zugeordnet werden, um einen Abstand zwischen der Komponente und der Gitterkante einzuräumen.

Gehen Sie wie folgt vor, um Anker, Füllung und Insets für das JScrollPane anzupassen:

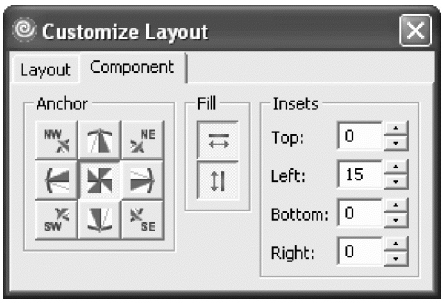
1. Klicken Sie mit der rechten Maustaste auf das JScrollPane in der Entwurfssicht oder der Sicht 'JavaBeans', und wählen Sie **Layout anpassen** aus.



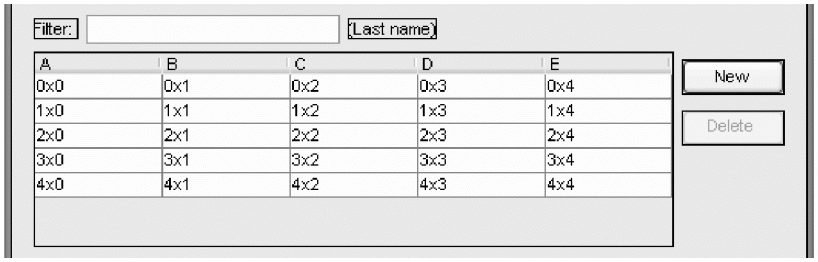
Tipp: Das Dialogfenster 'Layout anpassen' kann beim Auswählen und Ändern des Layouts für verschiedene Komponenten geöffnet bleiben. Sie können das Dialogfenster 'Layout anpassen' jederzeit öffnen, indem Sie auf die Schaltfläche 'Layout anpassen' in der Menüleiste klicken:



2. Stellen Sie auf der Registerkarte 'Komponente' des Dialogfensters 'Layout anpassen' sicher, dass die Schaltfläche 'Anker Mitte' ausgewählt ist.
3. Stellen Sie sicher, dass die Schaltflächen **Horizontal füllen** und **Vertikal füllen** ausgewählt sind.
4. Geben Sie ein linkes Inset von 15 (Pixel) ein, um den Abstand auf der linken Seite des JScrollPane den anderen visuellen Komponenten in der Anwendung anzugleichen.



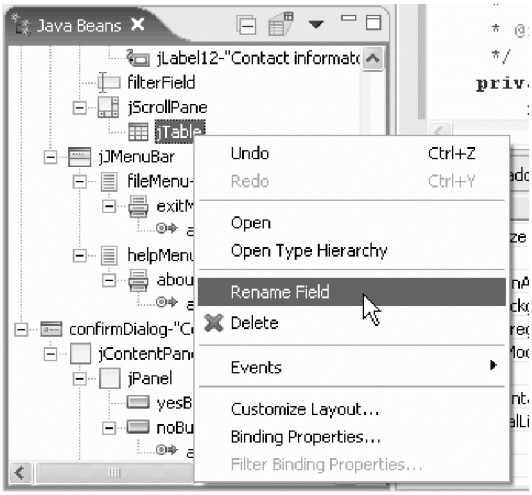
Die Tabelle ist nun beispielsweise auf das Feld **Filter** ausgerichtet.



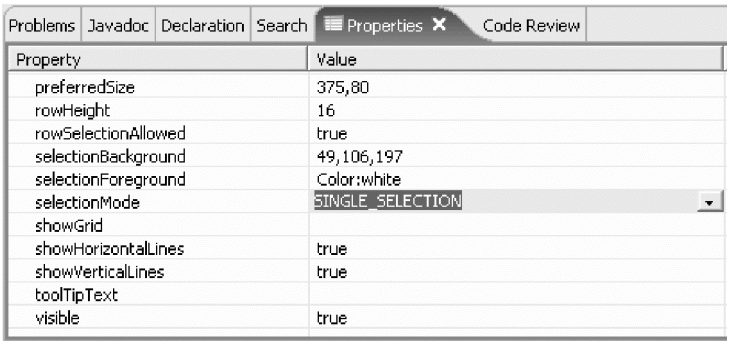
Die neue JTable in einen sinnvollen Wert umbenennen und für das Auswählen einer einzelnen Zeile definieren

Da Sie später mit der Tabelle arbeiten werden, ist es sinnvoll, die JTable-Instanz und ihre Getter-Methode umzubenennen. Gehen Sie wie folgt vor, um die Tabelle umzubenennen:

1. Klicken Sie in der Sicht 'Java-Beans' mit der rechten Maustaste auf die Komponente 'jTable', und wählen Sie im Popup-Menü den Menüpunkt **Feld umbenennen** aus.



2. Geben Sie `employeesTable` ein, und klicken Sie auf **OK**. Die JTable hat nun den Namen 'employeesTable', und die Methode für ihre Instanziierung heißt 'getEmployeesTable'.
3. Richten Sie die Tabelle so ein, dass nur eine Zeile ausgewählt werden kann:
   a. Wählen Sie die employeesTable in der Entwurfssicht aus.
   b. Wählen Sie in der Eigenschaftensicht die Eigenschaft **selectionMode** aus, und setzen Sie sie auf SINGLE_SELECTION.



   c. Speichern Sie die Datei DirectoryApp.java.

Prüfpunkt für die Lerneinheit

In dieser Lerneinheit haben Sie gelernt, wie der Visual Editor zum Hinzufügen einer Tabelle zu einer vorhandenen Benutzerschnittstelle verwendet wird. Anschließend haben Sie gelernt, das Layout, die Position und den Abstand anzupassen.

Lerneinheit 1.3: Die visuelle Klasse ausführen

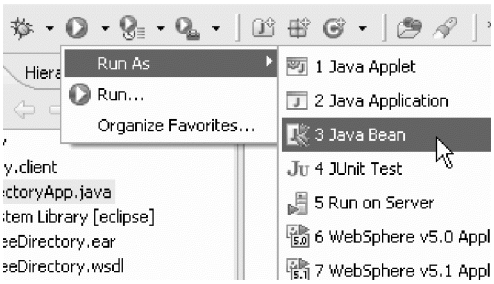
Nun können Sie die Java-Anwendung ausführen, um eine Vorschau der Darstellung aufzurufen. Die Workbench und der Visual Editor machen das rasche Ausführen Ihrer Anwendung sehr leicht, und Sie können diese Schritte in der Entwicklung jederzeit wiederholen, um die tatsächliche Laufzeitdarstellung und das Laufzeitverhalten der Klasse zu testen.

Zeigen

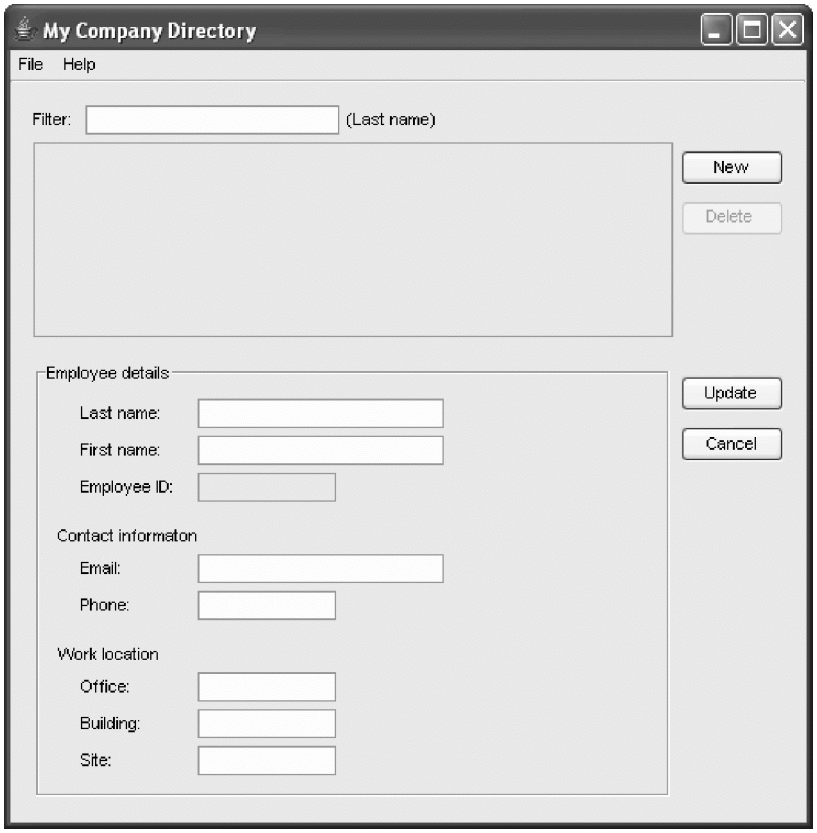
Der Java Visual Editor stellt ein Java-Bean-Startprogramm bereit, das es ermöglicht, Klassen ohne main()-Methode auszuführen. Wenn es die visuelle Klasse ausführt, wird die Anwendung in einer gesonderten virtuellen Maschine (VM, Virtual Machine) aufgerufen. Wenn Sie eine visuelle Klasse als Java-Anwendung ausführen, versucht das Startprogramm, die main()-Methode in der Klasse auszuführen. Für dieses Lernprogramm enthält Ihre Anwendung eine main()-Methode, die den DirectoryApp-JFrame aufruft und anzeigt, sodass Sie ihn als Anwendung oder als Java-Bean ausführen können.

Gehen Sie wie folgt vor, um die Datei DirectoryApp.java als Java-Bean auszuführen:

1. Stellen Sie sicher, dass die Datei DirectoryApp.java im Java Visual Editor geöffnet ist.
2. Klicken Sie in der Menüleiste auf **Ausführen** → **Ausführen als** → **Java-Bean**.



Tipp: Die Anwendung wird auf der Arbeitsoberfläche mit der Swing-Darstellung und -Funktionsweise geöffnet, die Sie in den Benutzervorgaben des Visual Editor (**Fenster** → **Benutzervorgaben** → **Java** → **Visual Editor**) definiert haben. Alternativ können Sie auf **Ausführen** → **Ausführen** klicken und die Darstellung und Funktionsweise für die betreffende Startkonfiguration zum Starten dieser Java-Bean definieren. Wenn Sie diese Anwendung nicht als Bean sondern als Anwendung ausführen, wird auch die Windows-Darstellung und -Funktionsweise verwendet, da diese in der main()-Methode definiert ist. In den in diesem Lernprogramm verwendeten Screenshots wird die Windows-Darstellung und -Funktionsweise verwendet.



Prüfpunkt für die Lerneinheit

Da Sie bisher nur die Schnittstelle entworfen, aber noch keine Datenverbindung oder Ereignisfunktionalität programmiert haben, können Sie mit der Anwendung noch nicht arbeiten. Sie können allerdings das grundlegende Layout und die Darstellung anschauen, die dem Benutzer angezeigt werden. Sie können versuchen, auf verschiedene Schaltflächen zu klicken und werden bemerken, dass hierdurch keine Aktion

ausgeführt wird. Das Menü 'Datei' und das Menü 'Hilfe' sind jedoch bereits implementiert. Sie können diese Menüs ausprobieren, um ihre Möglichkeiten zu erforschen, und Sie können den Java-Code überprüfen, um zu sehen, wie sie in actionPerformed-Ereignisse implementiert sind.

Durchgeführte Lerneinheiten

In diesem Modul erhielten Sie eine Einführung in das Entwerfen der Schnittstelle für einen Rich Client unter Verwendung des Java Visual Editor. Neben dem Entwerfen der grafischen Darstellung eines Clients sind jedoch noch viele weitere Arbeiten erforderlich, bevor der Client tatsächlich verwendet werden kann. Sie müssen normalerweise Ereignisverhalten oder andere Logik einschließen sowie, im vorliegenden Fall, das Binden der visuellen Elemente an eine Datenquelle.

In diesem Modul haben Sie gelernt, die folgenden Tasks auszuführen:

- Ein Java-Projekt unter Verwendung des Projektaustauschimports zu importieren
- Der visuellen Klasse eine JTable in einem JScrollPane hinzuzufügen
- Den GridBagLayout-Manager zum visuellen Gestalten der Tabelle auf dem Rich Client zu verwenden
- Die Anwendung auszuführen, um die tatsächliche Darstellung des Rich-Java-Clients zu sehen

Im nächsten Modul, 'Modul 2: Visuelle Komponenten an den Web-Service binden', verwandeln Sie die einfache Schnittstelle 'My Company Directory' in einen leistungsfähigen Rich Client, der auf Web-Service-Methoden zugreift, um Mitarbeiterdatensätze eines Unternehmensverzeichnisses zu erstellen, abzurufen, zu aktualisieren und zu löschen.

Modul 2: Visuelle Komponenten an den Web-Service binden

In diesem Modul lernen Sie, die visuellen Elemente von 'My Company Directory' (Schaltflächen, Mitarbeitertabelle, Felder und andere Aktionen) an einen Web-Service zu binden. Der Web-Service stellt die eigentliche Funktionalität zum Erstellen, Abrufen, Aktualisieren und Löschen von Mitarbeitern in bzw. aus dem Beispielverzeichnis bereit.

Lernziele

Nach den Lerneinheiten in diesem Modul verstehen Sie folgende Konzepte und Vorgehensweisen:

- Binden einer Tabelle an eine Web-Service-Datenquelle
- Binden von Feldern an Objekte
- Programmschaltflächen mit Aktionen

Es dauert ungefähr **2 Stunden**, um dieses Modul durchzuarbeiten.

Lerneinheit 2.1: Den Web-Service installieren und implementieren

In dieser Übung installieren Sie eine Beispiel-Unternehmensanwendungsdatei (EAR-Datei) auf WebSphere Application Server v6.1 und implementieren den Web-Service 'EmployeeDirectory'. Die Anwendung verwendet diesen Web-Service zum Erstellen, Lesen, Aktualisieren und Löschen von Mitarbeiterdatensätzen.

Bevor Sie beginnen, müssen Sie *eine der folgenden Aufgaben* ausführen, um sicherzustellen, dass sich das Projekt 'MyDirectory' am richtigen Ausgangspunkt befindet:

- Führen Sie „Modul 1: Client-GUI im Visual Editor entwerfen" auf Seite 3 durch.

  oder
- Das Projekt 'MyDirectory' am Ausgangspunkt von Modul 2 importieren

  Tipp: Sofern Sie beim Import keinen anderen Projektnamen angeben, werden hierdurch die Inhalte Ihres Projekts 'MyDirectory' überschrieben.

Das Java-Projekt 'MyDirectory' enthält eine Datei namens 'EmployeeDirectory.ear'. Sie verwenden die WebSphere-Administrationskonsole zum Installieren der Unternehmensanwendung 'EmployeeDirectory', die in der EAR-Datei enthalten ist. Wenn Sie die Anwendung installieren, implementieren Sie auch den in der Anwendung enthaltenen Web-Service. Die fertige Anwendung 'My Company Directory' verwendet diesen implementierten Web-Service.

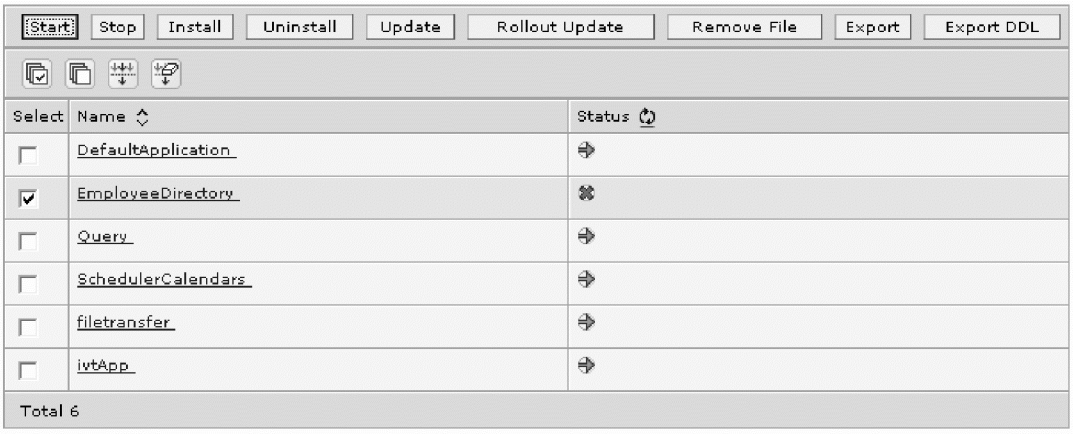
Gehen Sie folgendermaßen vor, um die Beispielanwendung 'EmployeeDirectory' zu installieren und den Web-Service in ihrer WebSphere Application Server v6.1-Umgebung zu implementieren:

1. Starten Sie ein Exemplar Ihres Anwendungsservers von der Workbench aus. Der Server kann auf verschiedene Arten gestartet werden. Die folgenden Schritte beschreiben das Starten von der Workbench:
   a. Öffnen Sie die Sicht 'Server'. Klicken Sie zum Hinzufügen der Sicht 'Server' zur Java-Perspektive auf **Fenster → Sicht anzeigen → Andere und wählen Sie Server → Server** aus.
   b. Die Sicht 'Server' führt die Server auf, die installiert und konfiguriert sind.
   c. Klicken Sie mit der rechten Maustaste auf Ihren Server, und wählen Sie **Starten** aus. Wenn in der Sicht 'Server' als Serverstatus **Gestartet** angezeigt wird, oder wenn in der Konsole der Hinweis `Server 'server1' ist für e-business bereit` angezeigt wird, wurde der Server erfolgreich gestartet. Sie können die Administrationskonsole nun ausführen.

   Anmerkung: Wenn in der Sicht 'Server' keine Serverinstanz vorhanden ist, müssen Sie einen neuen Server erstellen:
   a. Klicken Sie mit der rechten Maustaste auf die Sicht 'Server', und wählen Sie **Neu → Server** aus.
   b. Verwenden Sie den Assistenten 'Neuer Server', um WebSphere Application Server v6.1 hinzuzufügen.
2. Führen Sie die WebSphere-Verwaltungskonsole aus. Auch hier stehen andere Möglichkeiten zum Ausführen der Verwaltungskonsole zur Verfügung. Die vorliegenden Anweisungen beschreiben jedoch das Ausführen von der Workbench:
   a. Klicken Sie in der Sicht 'Server' mit der rechten Maustaste auf den soeben gestarteten Server, und wählen Sie **Administrationskonsole ausführen** aus. Die WebSphere-Administrationskonsole wird daraufhin in einem Browserfenster geöffnet.
   b. Geben Sie eine Benutzer-ID ein, und klicken Sie auf **Anmelden**. Die Seite 'Willkommen' der Administrationskonsole wird geöffnet. Die von Ihnen eingegebene Benutzer-ID wird nur verwendet, um benutzerspezifische Änderungen an den Konfigurationsdaten des Servers zu protokollieren.
3. Verwenden Sie die Administrationskonsole zum Installieren der Unternehmensanwendung 'EmployeeDirectory.ear', die sich im Projekt 'MyDirectory' befindet. In der Administrationskonsole wird eine Assistentenlösung verwendet, um Ihnen das Installieren von Anwendungen zu erleichtern. Bei diesem Assistenten klicken Sie auf **Weiter**, um von einer Seite zur nächsten zu gelangen, bis alle Optionen festgelegt sind. Gehen Sie wie folgt vor, um die Beispielunternehmensanwendung zu installieren, die den Web-Service für dieses Lernprogramm enthält:
   a. Erweitern Sie auf der linken Seite der Administrationskonsole die Option **Anwendungsmenü**, und klicken Sie anschließend auf **Neue Anwendung installieren**.
   b. Wählen Sie **Lokales Dateisystem** aus, und geben Sie im Feld **Pfad angeben** den vollständigen Pfad zur Datei EmployeeDirectory.ear ein, die sich in Ihrem Projekt 'MyDirectory' befindet. Tipp: Sie können den vollständigen Pfad abrufen, indem Sie mit der rechten Maustaste auf die Datei EmployeeDirectory.ear im Paketexplorer klicken, und die Option **Eigenschaften** auswählen. Auf der Seite 'Eigenschaften' wird die Position der Datei aufgelistet. Sie können diese Position kopieren und in das Feld **Pfad angeben** einfügen.
   c. Klicken Sie auf **Weiter**, bis Sie auf die Seite **Installationsoptionen auswählen** gelangen.
   d. Wählen Sie **Web-Services implementieren** aus.
   e. Klicken Sie auf **Weiter**, bis Sie auf die Seite **Zusammenfassung** gelangen, und klicken Sie anschließend auf **Fertig stellen**.

f. Klicken Sie auf den Link **In Masterkonfiguration speichern**, wenn Sie aufgefordert werden, die von Ihnen vorgenommenen Änderungen auf die lokale Konfiguration anzuwenden. Überprüfen Sie die Änderungen, und klicken Sie auf die Schaltfläche **Speichern**.

4. Verwenden Sie die Administrationskonsole, um die Anwendung 'EmployeeDirectory' zu starten:
   a. Klicken Sie auf **Anwendungen → Unternehmensanwendungen**. Die Anwendung 'EmployeeDirectory' wird als installierte Anwendung auf dem Server aufgelistet, ihr Status ist jedoch 'Gestoppt'.



   b. Wählen Sie das Markierungsfeld neben 'EmployeeDirectory' aus, und klicken Sie auf **Starten**. Eine Nachricht weist darauf hin, dass die Anwendung 'EmployeeDirectory' erfolgreich gestartet wurde, und das Symbol für den Status ändert sich in einen grünen Pfeil.

Die Anwendung 'EmployeeDirectory' wird auf dem lokalen Host bei Port 9080 ausgeführt, und der Zugriff auf den Web-Service ist nun möglich. Nach dem Beenden dieses Lernprogramms können Sie zur Administrationskonsole zurückkehren, die Anwendung 'EmployeeDirectory' stoppen und sie deinstallieren.

Wenn Sie die Datei 'EmployeeDirectory.wsdl' öffnen, die sich im Projekt 'MyDirectory' befindet (sie wird standardmäßig im grafischen WSDL-Editor geöffnet), können Sie den soeben implementierten Web-Service überprüfen. Wenn die WSDL-Datei nicht im WSDL-Editor geöffnet wird, ist die Web Service Developer-Funktionalität in der Workbench möglicherweise nicht aktiviert. Workbench-Funktionalitäten können Sie in den Benutzervorgaben angeben (**Fenster → Benutzervorgaben → Workbench → Funktionalitäten**).

In der folgenden Grafik aus dem WSDL-Editor werden die Operationen angezeigt, die im EmployeeDirectory-Service verfügbar sind:

Sie können den WSDL-Editor verwenden, um die einzelnen Operationen und deren zugehörige Anforderungs- und Antwortnachrichten zu überprüfen. Dies kann Ihnen dabei helfen, den Web-Service und seine Verwendung in den verbleibenden Übungen besser zu verstehen.

Lerneinheit 2.2: Die Mitarbeitertabelle an die Datenquelle des Web-Service binden

In der Anwendung 'My Company Directory' wird eine Liste aller momentan im Verzeichnis vorhandenen Mitarbeiterdatensätze angezeigt. Die Datensätze werden in einer JTable (employeesTable) mit sortierbaren

Spalten angezeigt und enthalten Nachname, Vorname, E-Mail und Mitarbeiter-ID. Zum Abrufen der Datensätze für die Tabelle müssen Sie die employeesTable an ein Datenobjekt binden, das von der Datenquelle des Beispiel-Web-Service zurückgegeben wird.

Zeigen

Übersicht über Datenobjekte, Datenquellen und Binder

Sie verwenden zum Abrufen eines lokalen Datenobjekts, mit dem die employeesTable arbeiten kann, den Visual Editor, um Ihrer Anwendung eine Datenquelle hinzuzufügen. Die Datenquelle stellt eine Verbindung zum Beispiel-Web-Service-Proxy her und erkennt die für die Anwendung verfügbaren Servicemethoden. Sie wählen daraufhin die Servicemethode 'getLightEmployeeRecord' aus, die von der Datenquelle verfügbar gemacht wird. Schließlich binden Sie die employeesTable in Ihrer Anwendung an die Felder, die im Zeilendatenobjekt (lightEmployeeRecordRows) zurückgegeben werden.

Sie können alle diese Datenquellen und Datenobjekte rasch und einfach unter Verwendung der integrierten Binder-Klassen des Java Visual Editor erstellen. Der Visual Editor stellt eine Reihe generischer Schnittstellen und Klassen bereit, die in Ihr Projekt generiert werden, wenn Sie visuelle Komponenten an Datenfactorys binden. Die Binder-Klassen werden standardmäßig in ein Paket namens jve.generated generiert. Der Visual Editor stellt die Binder-Klassen als generische Implementierung bereit, die Sie den Anforderungen Ihrer Anwendung entsprechend weiter anpassen und erweitern können. In diesem Lernprogramm werden Potenzial und Flexibilität der standardmäßigen Binder-Klassen veranschaulicht, die bereits bei einer grundlegenden und einfachen Verwendung dieser Klassen gegeben sind.

Wichtig: Es wird dringend empfohlen, vor dem Beginn dieser Übung die folgenden Hilfethemen zu lesen. Diese Themen helfen Ihnen, mehr über die Funktionalität und die Logik zu erfahren, die hinter den vom Java Visual Editor bereitgestellten Datenobjekten, Datenquellen und Bindern stehen.

- Datenbinder - Übersicht
- Binder-API - Referenz

In diesem Lernprogramm verwenden Sie eine Web-Service-Datenquelle, mehrere Typen von Datenobjekten und verschiedene Typen von Bindern in der Anwendung. Wenn Sie Instanzen von diesen Objekten der Anwendung hinzufügen, fügt der Visual Editor die erforderlichen Klassen dem Paket 'jve.generated' in Ihrem Projekt hinzu. Dort können Sie die Datenbindungslogik erweitern, ersetzen oder neu schreiben. Der Java Visual Editor bietet visuelle Unterstützung für die Bindungsobjekte, indem er im unformatierten Bereich der Entwurfssicht die Datenobjekte, Datenquellen und Binder anzeigt, die von Ihrer Anwendung verwendet werden. Der Visual Editor zeichnet Linien zwischen visuellen Komponenten und den Datenobjekten und Datenquellen, um die aktuellen Bindungen für ein ausgewähltes Objekt zu veranschaulichen.

Das folgende Diagramm ist eine einfache Übersicht, in der die Interaktion zwischen visuellen Komponenten, Bindern, Datenobjekten und Datenquellen dargestellt ist. Die Anwendung, die Sie im Rahmen dieses Lernprogramms erstellen, beinhaltet eine etwas komplexere und kreativere Verwendung der Binder. Dieses Diagramm stellt nicht genau die Binder, Datenobjekte und Datenquellen in der von Ihnen erstellten Beispielanwendung dar.

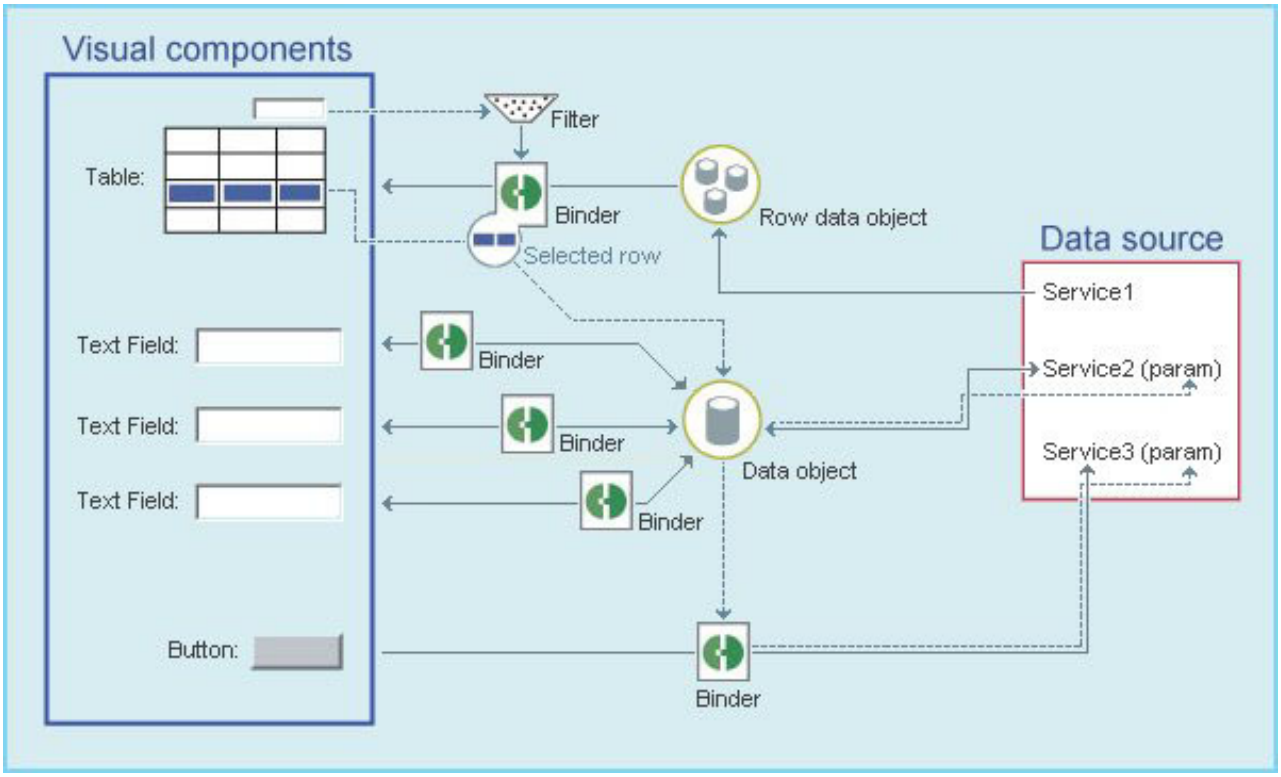


Abbildung 1. Dieses Diagramm veranschaulicht eine beispielhafte Beziehung zwischen visuellen Komponenten, Bindern, Datenobjekten und Datenquellen.

In Abbildung 1 verfügt jede visuelle Komponente über einen eigenen Binder, der sie einem Datenobjekt oder, im Fall der Schaltfläche, einer Datenquelle zuordnet. Die Binder für die Textfelder binden das Feld an eine bestimmte Eigenschaft des Datenobjekts. Sowohl das Zeilendatenobjekt als auch das Datenobjekt in dem vorliegenden Diagramm rufen ihre Daten anhand von direkten Aufrufen eines Service in der Datenquelle ab. Das Datenobjekt für das Textfeld verwendet einen Schlüsselwert aus der ausgewählten Zeile in der Tabelle als Argument für den Aufruf von Service2, der einen vollständigen Datensatz zurückgibt. Dieser Datensatz enthält vermutlich weitere Informationen über die ausgewählte Zeile in der Tabelle. Der vollständige Datensatz wird seinerseits als Argument für den Aktionsbinder der Schaltfläche beim Aufrufen von Service3 verwendet. Bei diesem Service kann es sich um eine Methode handeln, die die in die Felder eingegebenen Werte aktualisiert. Wenn Sie detailliertere Erläuterungen zu den Datenobjekten, Datenbindern und Datenquellen benötigen, folgen Sie den weiter oben angegebenen Links.

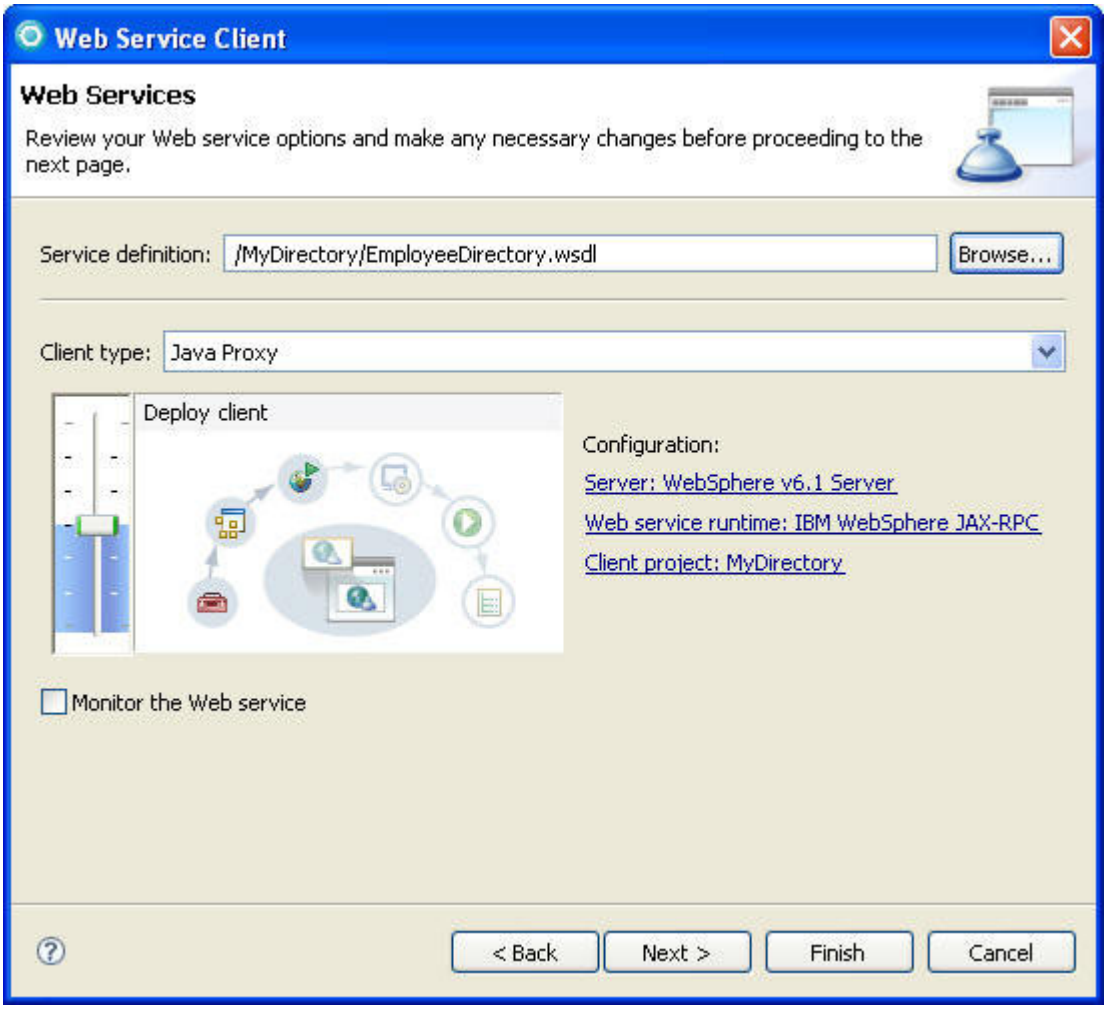
Ein Web-Service-Java-Proxy in Ihrem Projekt anhand der bereitgestellten WSDL-Datei erstellen

Für die Arbeit mit dem auf einem Server ausgeführten Web-Service benötigt die Java-Anwendung einen Java-Proxy oder -Client für die Interaktion. Bei Verwendung einer WSDL-Datei können Sie mit Hilfe des Assistenten für Web-Service-Clients einen Java-Proxy im Java-Projekt generieren. Ihr Projekt 'MyDirectory' enthält die Datei EmployeeDirectory.wsdl, die Sie zum Generieren dieses Proxys verwenden. Nach dem Generieren des Java-Proxys können Sie eine Datenquelle erstellen, die den Web-Service darstellt, und mit dem Binden visueller Komponenten beginnen.

Wichtig: Bei der in dieser Übung verwendeten WSDL-Datei wird davon ausgegangen, dass der Web-Service in einer lokalen Installation von WebSphere Application Server implementiert und der Standardport für 'localhost' (http://localhost:9080) verwendet wurde. Wenn Sie die EAR-Datei auf andere Art und Weise implementiert haben, müssen Sie die WSDL-Datei zuerst entsprechend bearbeiten, bevor Sie fortfahren können.

Gehen Sie wie folgt vor, um den Web-Service-Java-Proxy in Ihrem Projekt zu generieren:

1. Klicken Sie im Hauptmenü auf **Datei** → **Neu** → **Andere**, und wählen Sie den Assistenten **Web-Services** → **Web-Service-Client** aus. Falls die Kategorie 'Web-Services' nicht angezeigt wird, wählen Sie die Option **Alle Assistenten anzeigen** aus.
2. Verwenden Sie den Assistenten, um den Web-Service-Client zu definieren:
   a. Geben Sie als **Servicedefinition**, die WSDL-Datei an, die in Ihrem MyDirectory-Projekt bereitgestellt ist: `/MyDirectory/EmployeeDirectory.wsdl`
   b. Wählen Sie im Feld **Clienttyp** die Option **Java-Proxy** aus.
   c. Stellen Sie die Schiebeleiste auf **Client implementieren** ein.
   d. Stellen Sie sicher, dass der Server und die Web-Service-Laufzeit für den von Ihnen verwendeten Server korrekt eingestellt sind. Dieses Lernprogramm wurde für WebSphere v6.0 und WebSphere v6.1 mit der IBM WebSphere JAX-RPC-Laufzeit getestet.
   e. Stellen Sie sicher, dass der Java-Proxy-Client an das Projekt 'MyDirectory' ausgegeben wird.



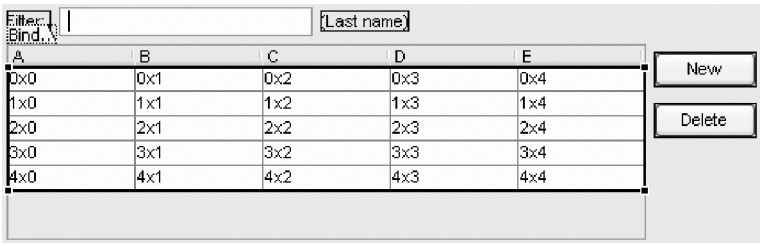
3. Klicken Sie auf **Fertig stellen**. Der Assistent für den Web-Service-Client generiert den Java-Proxy in einem neuen Paket (directory.service) in Ihrem Projekt.

Die employeesTable an ein Zeilendatenobjekt binden, das vom Web-Service zurückgegeben wird

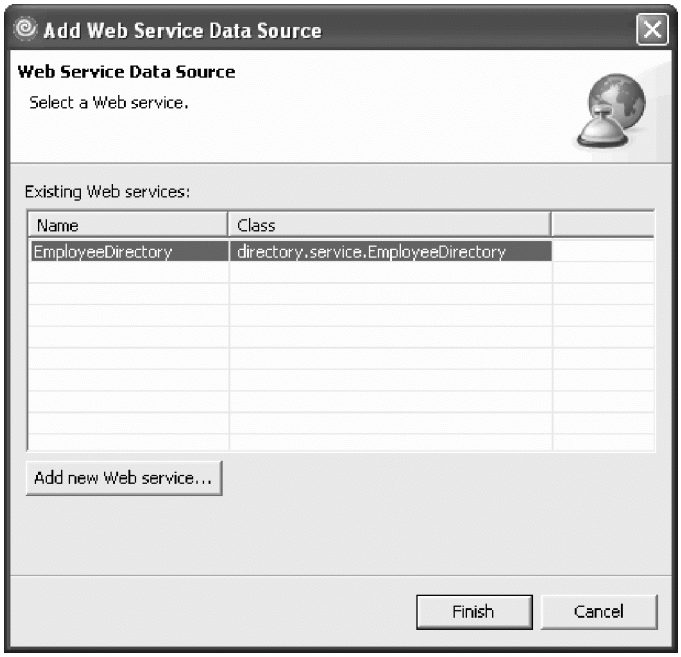
Da die employeesTable die erste visuelle Komponente ist, die Sie in dieser Anwendung binden, müssen Sie eine Datenquelle erstellen, die auf den soeben dem Projekt hinzugefügten Beispiel-Web-Service-Proxy verweist. Wenn Sie in späteren Übungen weitere visuelle Komponenten binden, werden Sie diese Datenquelle wiederverwenden. In diesem Schritt fügen Sie die Web-Service-Datenquelle und das Datenobjekt 'lightEmployeeRecordRows' hinzu.

Gehen Sie wie folgt vor, um die Mitarbeitertabelle zu binden:

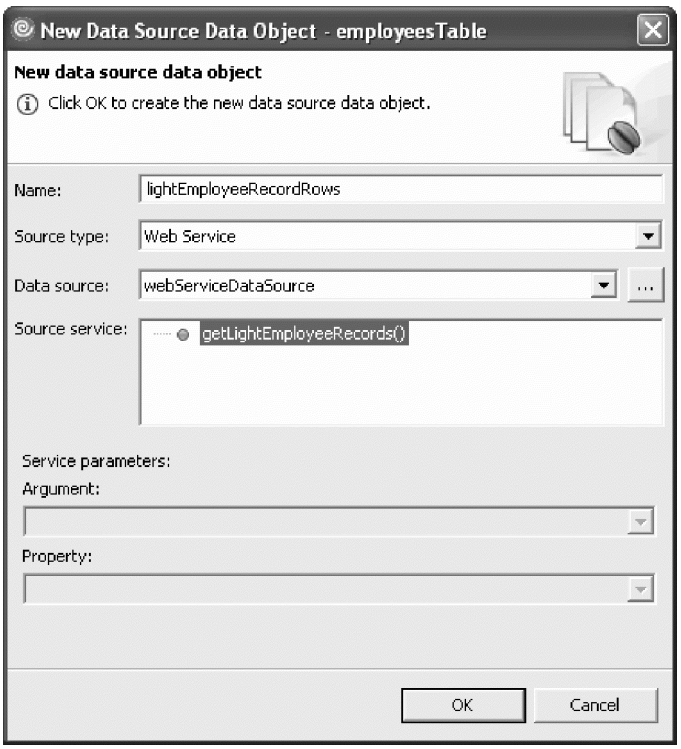
1. Wählen Sie in der Sicht 'Java-Beans' oder in der Entwurfssicht die employeesTable aus. (Stellen Sie sicher, dass Sie nicht das übergeordnete JScrollPane-Element auswählen.) Daraufhin wird oben in der employeesTable im Entwurfsbereich eine kleine Registerkarte mit der Aufschrift **Binden** angezeigt.



2. Klicken Sie auf die Registerkarte **Binden** in der employeesTable. Alternativ können Sie mit der rechten Maustaste auf die employeesTable klicken und die Option **Binding-Eigenschaften** auswählen.
3. Da sich in der Anwendung keine Datenobjekte befinden, müssen Sie ein neues hinzufügen. Klicken Sie auf **Neues Datenquellendatenobjekt**.
4. Wählen Sie im Feld **Quellentyp** den Wert **Web-Service** aus.
5. Da Sie die Web-Service-Datenquelle noch nicht Ihrer Anwendung hinzugefügt haben, müssen Sie sie jetzt hinzufügen. Klicken Sie neben dem Feld **Datenquelle** auf die Schaltfläche **...**, um das Dialogfenster 'Web-Service-Datenquelle hinzufügen' zu öffnen, in dem nach verfügbaren Web-Service-Clients oder Proxys in Ihrem Projekt gesucht wird.
6. Wählen Sie den Web-Service 'EmployeeDirectory' aus, und klicken Sie danach auf 'Fertig stellen'. Der Datei DirectoryApp.java wird daraufhin eine neue Datenquelle hinzugefügt.

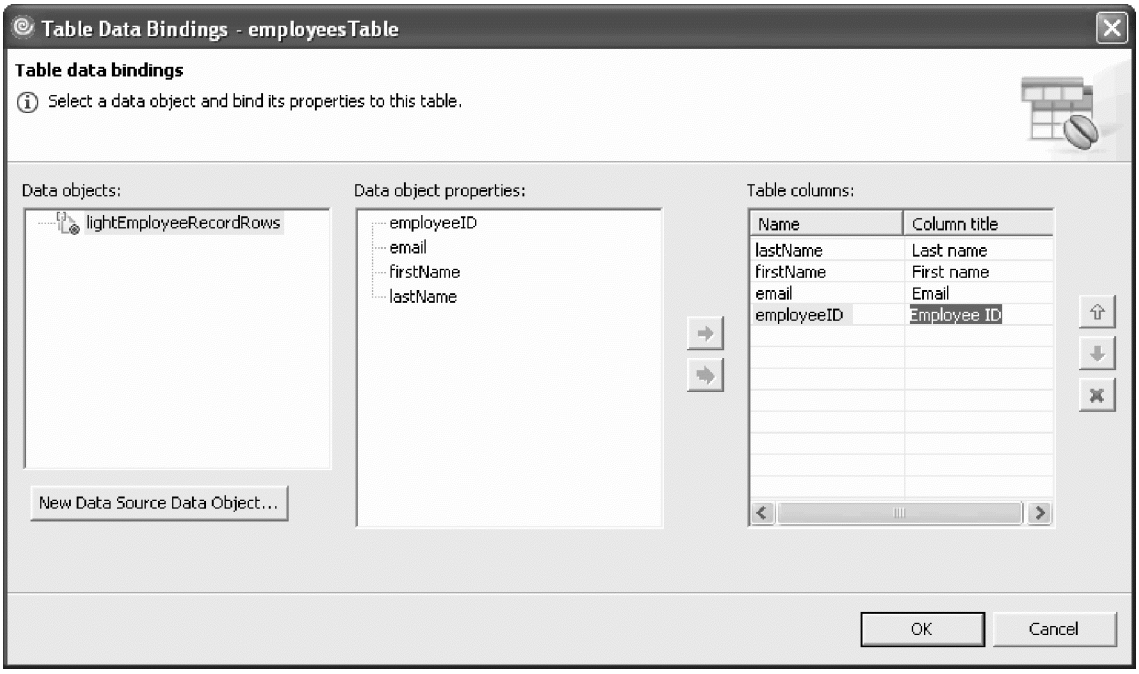


7. Wählen Sie im Dialogfenster 'Neues Datenquellendatenobjekt' den Wert 'getLightEmployeeRecords()' im Feld 'Quellenservice' aus, und akzeptieren Sie den Standardnamen für das neue Datenobjekt: lightEmployeeRecordRows. Für diese Servicemethode sind keine Parameter erforderlich. Klicken Sie auf 'OK'. Das neue Datenobjekt wird erstellt und im unformatierten Bereich der Entwurfssicht angezeigt.



Tipp: Da Sie eine Tabelle binden, werden im Dialogfenster 'Neues Datenquellendatenobjekt' nur Services angezeigt, die Zeilendatenobjekte zurückgeben. Im vorliegenden Fall ist die Methode getLightEmployeeRecords() der einzige verfügbare Service, der einen Array von Objekten zurückgibt.

8. Wählen Sie im Dialogfenster 'Tabellen-Daten-Bindings' das Datenobjekt 'lightEmployeeRecordRows' aus.
9. Nun müssen Sie die Eigenschaften des Datenobjekts 'lightEmployeeRecordRows' auswählen, die in employeesTable angezeigt werden sollen:



a. Klicken Sie auf die Schaltfläche mit dem Doppelpfeil ⇨, um alle Objekteigenschaften der Liste **Tabellenspalten** hinzuzufügen.

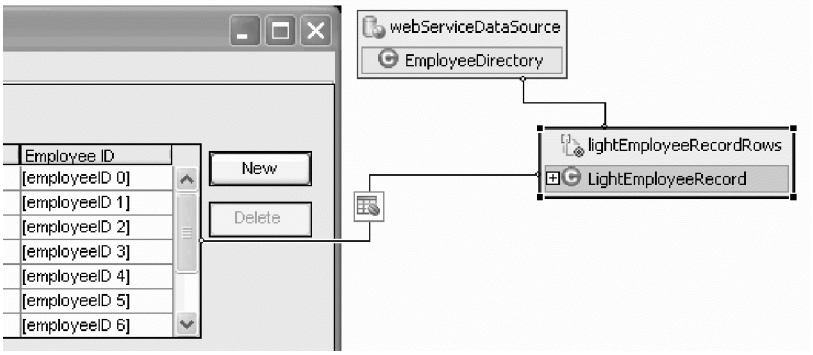
b. Verwenden Sie den Aufwärts- und den Abwärtspfeil, um die Spalten in der folgenden Reihenfolge von oben nach unten anzuordnen: lastName, firstName, email, employeeID

c. Benennen Sie die Spaltentitel folgendermaßen um: `Nachname`, `Vorname`, `E-Mail`, `Mitarbeiter-ID`

   Tipp: Nach dem Binden der Tabelle können Sie jederzeit zu den Binding-Eigenschaften zurückkehren und die Spalten erneut umbenennen oder neu anordnen.

d. Klicken Sie auf **OK**.

Die employeesTable ist nun unter Verwendung eines JRowTableBinder an das Datenobjekt 'lightEmployeeRecordRows' gebunden. Wenn Sie im unformatiert Bereich auf das Datenobjekt 'lightEmployeeRecordRows' klicken, zeichnet der Visual Editor eine Linie vom Datenobjekt zur Tabelle. Der JRowTableBinder wird auf dieser Linie durch das Tabellenbindersymbol dargestellt. Eine weitere Linie gibt an, dass das Datenobjekt die webServiceDataSource als Datenquelle verwendet.



Prüfpunkt für die Lerneinheit

Beachten Sie die Änderungen am Projekt und an der Anwendung. In dieser Lerneinheit haben Sie die Web-Service-Datenquelle, ein Zeilendatenobjekt und einen Binder hinzugefügt, der die employeesTable an das Zeilendatenobjekt bindet.

Überprüfen Sie das neue Paket (jve.generated), das im Projekt erstellt wurde und alle vom Java Visual Editor generierten Binderklassen enthalten soll. Beachten Sie darüber hinaus das neue Paket (directory-.service), das den Java-Proxy für den Web-Service enthält. Beschreiben Sie oder fassen Sie zusammen, was Sie in dieser Lerneinheit gelernt haben.

Wenn Sie die Anwendung 'My Company Directory' jetzt ausführen, wird die Mitarbeitertabelle vom Web-Service mit den vorhandenen Mitarbeiterdatensätzen gefüllt.

Lerneinheit 2.3: Die Detailfelder an die Tabellenauswahl binden

In der vorhergehenden Übung haben Sie die employeesTable an das Datenobjekt 'lightEmployeeRecordRows' gebunden, das seinerseits vom Service 'getLightEmployeeRecords()' im Web-Service zurückgegeben wurde. Nun müssen Sie die Detailfelder anhand des in der Tabelle ausgewählten Mitarbeiters ausfüllen.

Zum Abrufen zusätzlicher Details für jeden ausgewählten Mitarbeiter wird ein anderes Datenobjekt verwendet. Das Datenobjekt 'selectedEmployeeRecord', das Sie hinzufügen, wird vom Service 'getFullEmplo-

yeeRecord()' zurückgegeben. Dieser Service verwendet die ID des in der Tabelle ausgewählten Mitarbeiters als Parameter und ruft zusätzliche Details über den Mitarbeiter, einschließlich Telefonnummer und Arbeitsplatz, ab.

Der JRowTableBinder, der beim Binden der Tabelle an das Zeilendatenobjekt verwendet wurde, vereinfacht diesen Schritt. Der JRowTableBinder macht das in der Tabelle ausgewählte Element als gesondertes Datenobjekt zugänglich, das als Parameter für die Methode 'getFullEmployeeRecord(java.lang.Integer)' verwendet werden kann. Sie können dann jedes Textfeld einfach an die entsprechende Eigenschaft im Datenobjekt 'selectedEmployeeRecord' binden.

Weitere Informationen über diesen Web-Service: Der Web-Service enthält zwei Services zum Abrufen aller Details für die einzelnen Mitarbeiter. In der Tabelle werden alle Mitarbeiter aufgelistet, und es wird nur ein Teil der Daten in der Tabelle angezeigt. Wenn ein einzelner Mitarbeiter ausgewählt wird, können Sie die restlichen Mitarbeiterinformationen nur für diesen ausgewählten Mitarbeiter abrufen. Wenn der Web-Service alle Daten zu den einzelnen Mitarbeitern sendet, wenn die Tabelle Daten anfordert, kann dies zu einem starken Webdatenverkehr und damit zu einer langsamen Verarbeitung der Anwendung führen.

Wenn der Mitarbeiterdatensatz beispielsweise ein Foto oder einen Anhang enthält, möchten Sie beim einfachen Abrufen der vollständigen Mitarbeiterliste nicht sämtliche Fotos abrufen. Daher wird der Service 'getLightEmployeeRecord' verwendet, um die Tabelle zu füllen, und der getFullEmployeeRecord ruft den vollständigen Datensatz für den Mitarbeiter ab, der in der Tabelle ausgewählt wird.

Das Feld 'Nachname' binden

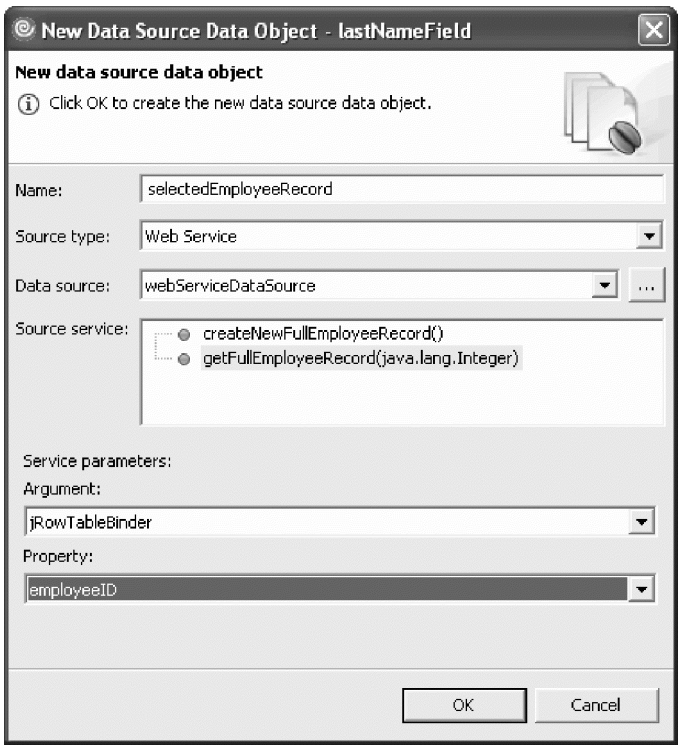
In diesem Schritt binden Sie das Feld **Nachname** an die Eigenschaft 'lastName' im Datenobjekt 'selectedEmployeeRecord'.

1. Wählen Sie in der Sicht 'Java-Beans' oder in der Entwurfssicht das JTextField für den Nachnamen (lastNameField) aus. Im Entwurfsbereich wird über dem Textfeld die Registerkarte **Binden** angezeigt.

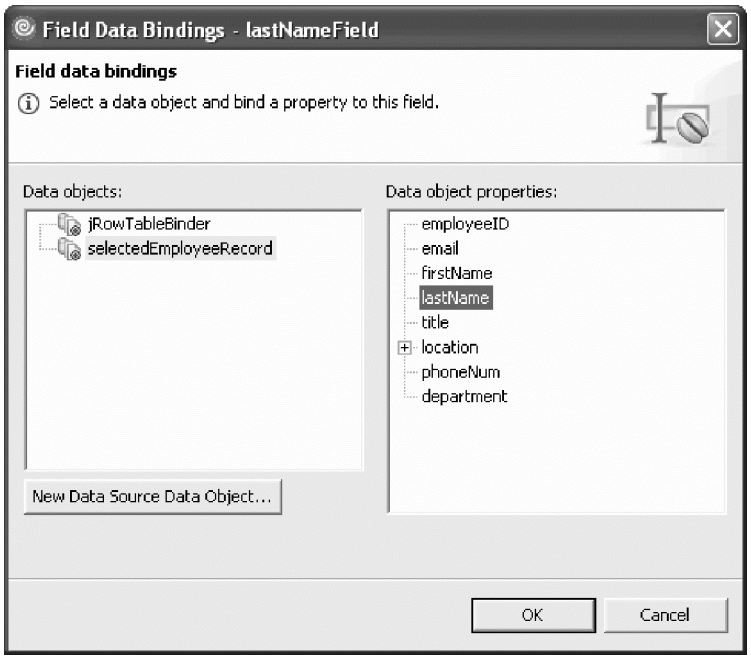
   

2. Klicken Sie auf die Registerkarte **Binden**, um das Dialogfenster 'Felddaten-Bindings' zu öffnen.
3. Klicken Sie auf **Neues Datenquellendatenobjekt**. Das vorhandene Datenobjekt 'jRowTableBinder' gibt zwar den richtigen Nachnamen zurück, es enthält aber nicht den vollständigen Mitarbeiterdatensatz. Sie müssen ein neues Datenobjekt erstellen, das den vollständigen Mitarbeiterdatensatz darstellt.
4. Stellen Sie im Feld **Quellentyp** sicher, dass die Option **Web-Service** ausgewählt ist, und dass für **Datenquelle** der Wert **webServiceDataSource** ausgewählt ist.
5. Wählen Sie in der Liste **Quellenservice** den Eintrag 'getFullEmployeeRecord(java.lang.Integer)' aus. Im Dialogfenster 'Neues Datenquellendatenobjekt' werden die Services aufgelistet, die Datenobjekte zurückgeben, die mit einem Textfeld kompatibel sind.
6. Geben Sie im Feld **Name** den Wert `selectedEmployeeRecord` ein.
7. Wählen Sie im Feld **Argument** den Wert 'jRowTableBinder' und im Feld **Eigenschaft** den Wert 'employeeID' aus. Die Mitarbeiter-ID der ausgewählten Zeile ist nun als Argument für die Servicemethode 'getFullEmployeeRecord()' festgelegt.

   Anmerkung: getFullEmployeeRecord(java.lang.Integer) erfordert eine ganze Zahl als Argument. Sie möchten die Mitarbeiter-ID der aktuellen Auswahl in der Mitarbeitertabelle zum Abrufen eines vollständigen Datensatzes verwenden. Beim Binden der Tabelle generierte der Visual Editor automatisch den jRowTableBinder, der für die aktuelle Auswahl in der Mitarbeitertabelle empfangsbereit ist. Als Ganzzahlparameter verwenden Sie die employeeID der ausgewählten Zeile in jRowTableBinder.



8. Klicken Sie auf **OK**.
9. Stellen Sie im Dialogfenster 'Felddaten-Bindings' sicher, dass 'selectedEmployeeRecord' in der Liste **Datenobjekte** ausgewählt ist. Beachten Sie, dass für das Datenobjekt 'selectedEmployeeRecord' mehr Eigenschaften verfügbar sind als für das Datenobjekt 'jRowTableBinder'.
10. Wählen Sie in der Liste **Datenobjekteigenschaften** die Eigenschaft 'lastName' aus.



11. Klicken Sie auf **OK**. Das Feld 'Nachname' in Ihrer Anwendung ist nun an die Eigenschaft 'lastName' des Datenobjekts 'selectedEmployeeRecord' gebunden, das seinerseits von getFullEmployeeRecord() zurückgegeben wird.

Ein neues Datenobjekt namens 'selectedEmployeeRecord' wird erstellt und der Anwendung hinzugefügt. Eine visuelle Darstellung des Datenobjekts wird dem unformatierten Bereich der Entwurfssicht hinzugefügt. Dies ist in der folgenden Abbildung dargestellt.

Wenn Sie nun das Feld 'lastName' im Entwurfsbereich auswählen, wird durch eine Linie angezeigt, dass es an den selectedEmployeeRecord gebunden ist. In der Mitte der Linie wird durch das Textbindersymbol der SwingTextComponentBinder dargestellt, der für diese Bindung verwendet wird. Wenn Sie die Linie oder das Symbol für den Binder im Entwurfsbereich auswählen, können Sie die Eigenschaften des Binders in der Sicht 'Eigenschaften' überprüfen.

Die verbleibenden Detailfelder binden

Beim Binden der einzelnen verbleibenden Detailfelder für einen Mitarbeiter gehen Sie ähnlich wie beim Feld für den Nachnamen vor, Sie müssen nur das Datenobjekt nicht hinzufügen. Da Sie das Datenobjekt 'selectedEmployeeRecord' bereits hinzugefügt haben, können Sie einfach jedes Feld an seine entsprechende Eigenschaft im Datenobjekt 'selectedEmployeeRecord' binden.

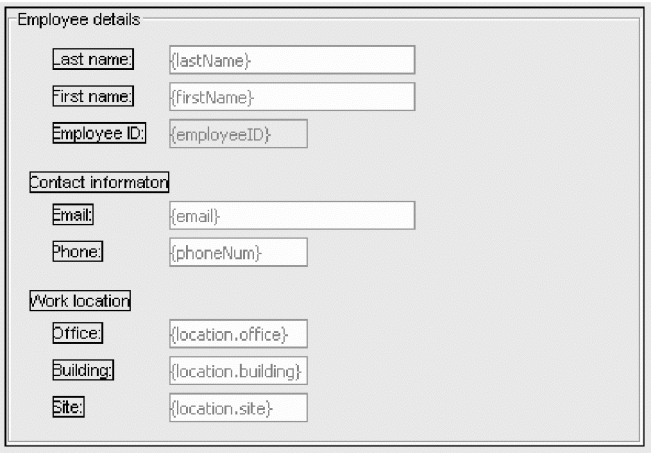
Führen Sie zum Binden der Felder für jedes Feld die folgenden Schritte im Abschnitt für Mitarbeiterdetails der Anwendung aus:

1. Wählen Sie das Feld in der Entwurfssicht aus, und klicken Sie auf die Registerkarte **Binden**.
2. Wählen Sie im Dialogfenster 'Felddaten-Bindings' den Eintrag 'selectedEmployeeRecord' in der Liste **Datenobjekte** aus.
3. Wählen Sie in der Liste **Datenobjekteigenschaften** die entsprechende Eigenschaft für das Feld aus, das Sie binden. In der folgenden Grafik wird die Eigenschaft angezeigt, an die die einzelnen Textfelder gebunden werden müssen:

Feld	**Eigenschaft im Datenobjekt 'selectedEmployeeRecord'**
lastNameField	lastName
firstNameField	firstName
idField	employeeID
emailField	email
phoneField	phoneNum
officeField	location.office
buildingField	location.building
siteField	location.site

4. Klicken Sie auf **OK**.

Wenn Sie mit dem Binden der Textfelder fertig sind, sollte der Entwurfsbereich folgendermaßen aussehen:



Das Feld 'Mitarbeiter-ID' schreibgeschützt machen

Das Feld 'Mitarbeiter-ID' ist inaktiviert, da die Eigenschaft 'editable' (editierbar) des Felds auf den Wert 'false' gesetzt ist. Das Standardverhalten des Textfeldbinders ändert dies jedoch in den aktivierten Status des Felds, wenn das Datenobjekt einen Wert enthält. Sie können dieses Binderverhalten ausschalten, sodass das Feld in seinem ursprünglichen schreibgeschützten Zustand verbleibt.

Gehen Sie wie folgt vor, um zu verhindern, dass der Binder die Eigenschaft 'editable' automatisch ändert:

1. Wählen Sie das Feld 'Mitarbeiter-ID' aus. Daraufhin wird im Entwurfsbereich eine Linie mit einem Symbol angezeigt, das den Binder für das Feld darstellt.
2. Klicken Sie auf das Bindersymbol für das Feld 'Mitarbeiter-ID'.
3. Ändern Sie in der Sicht 'Eigenschaften' die Eigenschaft 'autoEditable' in **false**. Drücken Sie die **Eingabetaste**.

Prüfpunkt für die Lerneinheit

Wenn Sie nun die Anwendung ausführen und einen Mitarbeiter aus der Tabelle auswählen, werden die Details dieses Mitarbeiterdatensatzes in den Detailfeldern angezeigt.

Lerneinheit 2.4: Die Schaltfläche 'Aktualisieren' an einen Aktionsbinder binden

Der Java Visual Editor stellt Aktionsbinder bereit. Diese Aktionsbinder rufen einen Service für eine Datenquelle auf, wenn auf eine Schaltfläche geklickt wird. Wenn beispielsweise auf die Schaltfläche 'Aktualisieren' geklickt wird, muss die Anwendung die Methode 'modifyEmployee()' für den Web-Service ausführen, wobei die Änderungen in die Detailfelder eingegeben werden. In dieser Lerneinheit binden Sie die Schaltfläche 'Aktualisieren' an einen Aktionsbinder.

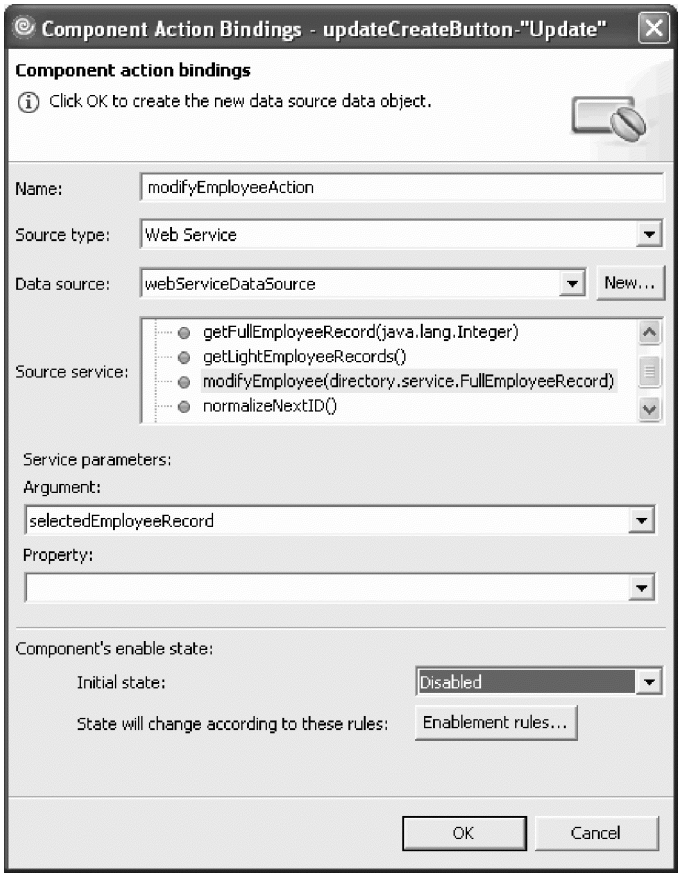
Gehen Sie wie folgt vor, um die Schaltfläche 'Aktualisieren' zu binden:

1. Wählen Sie die Schaltfläche **Aktualisieren** im Entwurfsbereich aus, und klicken Sie auf die Registerkarte **Binden**, um das Dialogfenster 'Komponentenaktion-Bindings' zu öffnen.

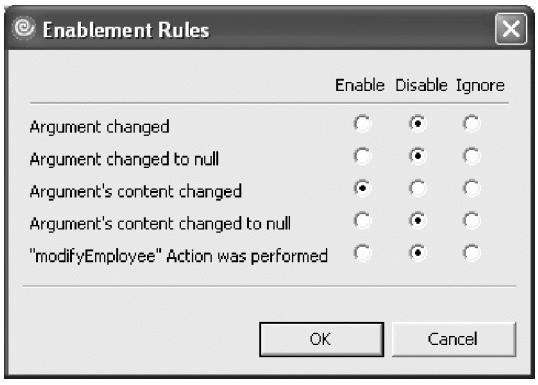
   

2. Wählen Sie im Feld **Quellentyp** die Option **Web-Service** aus.
3. Wählen Sie im Feld **Datenquelle** den Wert **webServiceDataSource** aus.
4. Wählen Sie in der Liste **Quellenservice** den Eintrag **modifyEmployee(directory.service.FullEmployeeRecord)** aus.
5. Das Feld **Name** wird automatisch in **modifyEmployeeAction** geändert. Übernehmen Sie diesen Standardwert.
6. Wählen Sie im Feld **Argument** den Wert **selectedEmployeeRecord** aus.

7. Da die Methode 'modifyEmployee()' einen vollständigen Mitarbeiterdatensatz als Argument verwendet, müssen Sie das Feld **Eigenschaft** leer lassen.
8. Setzen Sie den **Anfangsstatus** der Schaltfläche auf den Wert **Inaktiviert**.



9. Um zu definieren, wie der Status der Schaltfläche geändert wird, klicken Sie auf die Schaltfläche **Aktivierungsregeln**. Geben Sie an, dass die Schaltfläche nur aktiviert ist, wenn sich der Inhalt des Arguments ändert, und in allen anderen Fällen inaktiviert ist. Klicken Sie auf **OK**.



   Dies bedeutet, dass die Schaltfläche **Aktualisieren** inaktiviert bleibt, bis der Inhalt des 'selectedEmployeeRecord' geändert wird. Mit anderen Worten: Sobald Sie einen neuen Wert in eines der Detailfelder eingeben, die an den selectedEmployeeRecord gebunden sind, aktiviert der Binder die Schaltfläche. Wenn Sie einen neuen Datensatz auswählen oder auf **Aktualisieren** klicken, wird die Schaltfläche wieder inaktiviert.
10. Klicken Sie auf **OK**.

Für die Schaltfläche **Aktualisieren** wird ein neuer SwingDataServiceAction-Binder hinzugefügt. Wenn Sie die Schaltfläche im Entwurfsbereich auswählen, zeichnet der Visual Editor eine Linie, die angibt, dass die

Schaltfläche an die Web-Service-Datenquelle gebunden ist. Ein rosa gepunkteter Pfeil zeigt vom Objekt 'selectedEmployeeRecord' zur Linie. Dieser Pfeil gibt an, dass der selectedEmployeeRecord das Argument für den Aufruf des Service ist.

Prüfpunkt für die Lerneinheit

Wenn Sie nun die Anwendung ausführen, können Sie den Datensatz eines Mitarbeiters aktualisieren.

Wählen Sie einen Mitarbeiter in der Tabelle aus, und ändern Sie den Nachnamen. Sobald Sie den Nachnamen ändern, wird die Schaltfläche **Aktualisieren** aktiviert. Wenn Sie auf **Aktualisieren** klicken, wird der Service 'modifyEmployee' aufgerufen, und der Mitarbeiterdatensatz wird aktualisiert. Der neue Nachname wird in der Mitarbeitertabelle angezeigt.

Lerneinheit 2.5: Die Schaltfläche 'Löschen' und das Fenster 'Bestätigungsdialog' aktivieren

In dieser Übung programmieren Sie die Anwendung 'My Company Directory' für das Löschen eines Mitarbeiterdatensatzes.

In der folgenden Liste ist das Verhalten beschrieben, das die Anwendung aufweisen soll:

- Wenn Sie einen Mitarbeiter in der Tabelle auswählen, wird die Schaltfläche **Löschen** aktiviert.
- Wenn Sie auf die Schaltfläche **Löschen** klicken, wird das Dialogfenster 'Löschen bestätigen' geöffnet, in dem Sie aufgefordert werden, den Löschvorgang zu bestätigen.
- Wenn Sie auf die Schaltfläche **Ja** im Dialogfenster 'Löschen bestätigen' klicken, wird der Mitarbeiterdatensatz gelöscht, das Dialogfenster 'Löschen bestätigen' wird geschlossen, und die Liste der Mitarbeiter wird aktualisiert.
- Wenn Sie auf **Nein** klicken, wird der Löschvorgang abgebrochen, und das Dialogfenster 'Löschen bestätigen' wird geschlossen.

Die Schaltfläche 'Löschen' so programmieren, dass sie je nach dem, ob eine Zeile in der Tabelle ausgewählt ist, aktiviert oder inaktiviert wird

Um die Schaltfläche 'Löschen' so zu programmieren, dass sie aktiviert oder inaktiviert sein kann, fügen Sie der Tabelle einen Listener hinzu, der die Schaltfläche aktiviert, wenn eine Zeile ausgewählt wird.

1. Wählen Sie die employeesTable in der Sicht 'Java-Beans' aus. In der Quellensicht ist die folgende Zeile hervorgehoben:

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

2. Fügen Sie unmittelbar nach dieser Zeile der employeesTable einen neuen ListSelectionListener und das Ereignis 'valueChanged' hinzu:

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

3. Nach dem Hinzufügen dieser Codezeilen markiert der Quelleneditor sie als Fehler, bis Sie ListSelectListener und ListSelectionEvent importieren. Klicken Sie zum Hinzufügen der erforderlichen Importe im Hauptmenü auf **Quelle → Importe verwalten**. Daraufhin werden dem Importabschnitt der Klasse die folgenden Zeilen hinzugefügt:

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

Wenn nun eine Zeile in der Tabelle ausgewählt wird, wird die Schaltfläche **Löschen** aktiviert.

Das Dialogfenster 'Löschen bestätigen' so programmieren, dass es beim Klicken auf 'Löschen' geöffnet wird

Fügen Sie der Schaltfläche 'Löschen' das Ereignis 'actionPerformed' hinzu, und programmieren Sie das Ereignis für das Öffnen des Dialogfensters 'Löschen bestätigen'.

1. Klicken Sie mit der rechten Maustaste auf die Schaltfläche **Löschen**, und wählen Sie **Ereignisse → actionPerformed** aus. Der folgende Ereignis-Stub wird der Methode 'getDeleteButton()' hinzugefügt:

```
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()
   }
});
```

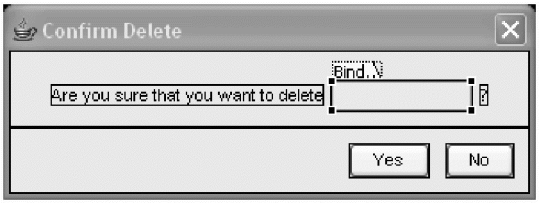
2. Ersetzen Sie diesen generierten Stub durch den folgenden Code, durch den das Dialogfenster 'Löschen bestätigen' angezeigt wird, wenn auf die Schaltfläche geklickt wird:

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

Das Textfeld im Dialogfenster 'Löschen bestätigen' binden

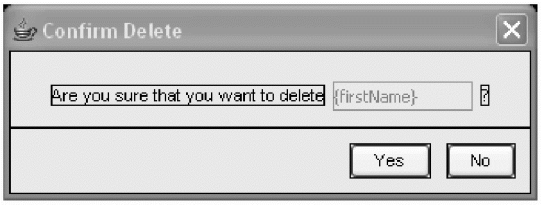
Binden Sie das Textfeld im Dialogfenster 'Löschen bestätigen', um den Vornamen des Mitarbeiters anzuzeigen, der gelöscht werden soll.

1. Wählen Sie in der Sicht 'Java-Beans' oder im Entwurfsbereich das Textfeld 'employeeToDeleteField' aus, und klicken Sie auf die Registerkarte **Binden**.



2. Wählen Sie im Dialogfenster 'Felddaten-Bindings' das Datenobjekt 'selectedEmployeeRecord' und das Feld 'firstName' aus, und klicken Sie anschließend auf **OK**.

   Das Textfeld ist nun an die Spalte 'firstName' der ausgewählten Zeile in der employeesTable gebunden.



3. Um sicherzustellen, dass dieses Feld schreibgeschützt ist, setzen Sie die Eigenschaft **autoEditable** für den Binder des Felds auf **false**.

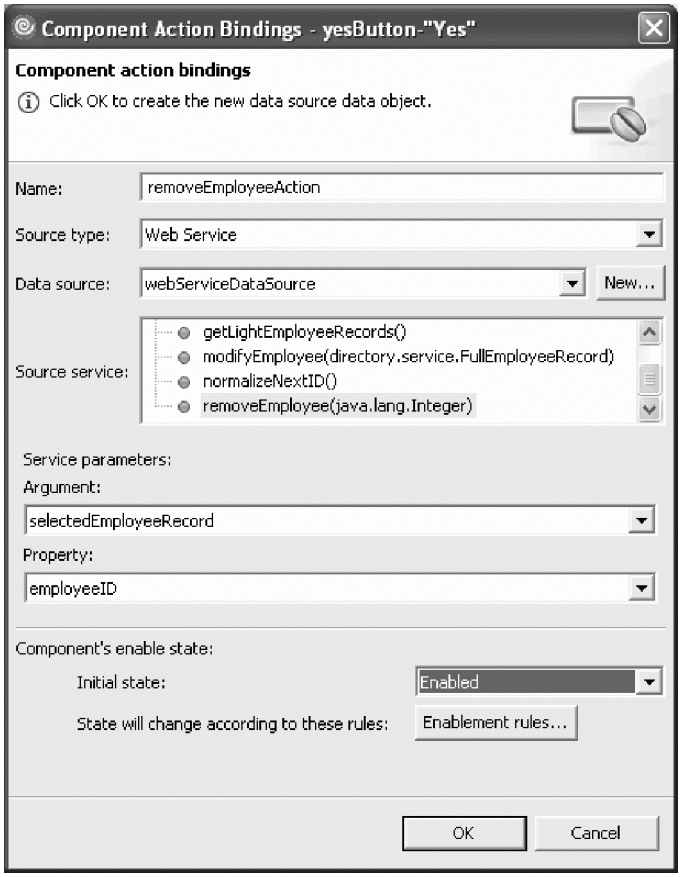
Die Schaltfläche 'Ja' für das Ausführen des Löschvorgangs binden

Binden Sie die Schaltfläche **Ja** für den Aufruf der Methode 'removeEmployee(java.lang.Integer)' für den Web-Service.

1. Wählen Sie die Schaltfläche **Ja** aus, und klicken Sie auf die Registerkarte **Binden**, um das Dialogfenster 'Komponentenaktion-Bindings' zu öffnen.
2. Wählen Sie im Feld **Quellentyp** die Option **Web-Service** aus.
3. Wählen Sie im Feld **Datenquelle** den Wert **webServiceDataSource** aus.
4. Wählen Sie in der Liste **Quellenservice** den Eintrag **removeEmployee(java.lang.Integer)** aus.
5. Das Feld **Name** wird automatisch in **removeEmployeeAction** geändert. Übernehmen Sie diesen Standardwert.
6. Wählen Sie im Feld **Argument** den Wert **selectedEmployeeRecord** aus.

7. Wählen Sie im Feld **Eigenschaft** den Wert **employeeID** aus. Da die Methode 'removeEmployee()' eine ganze Zahl als Argument verlangt, verwenden Sie die Mitarbeiter-ID des selectedEmployeeRecord.
8. Setzen Sie den **Anfangsstatus** der Schaltfläche auf **Aktiviert**.
9. Wählen Sie unter **Aktivierungsregeln** für jede Bedingung **Ignorieren** aus.

   Dieser Komponentenstatus bedeutet, dass die Schaltfläche 'Ja' stets aktiviert ist, da für einen Statuswechsel kein Grund vorliegt.



10. Klicken Sie auf **OK**.

Ein Ereignis hinzufügen, um das Dialogfenster 'Löschen bestätigen' nach dem Löschen des Mitarbeiters auszublenden

In diesem Schritt fügen Sie dem *Binder* der Schaltfläche **Ja** (nicht der Schaltfläche **Ja** selbst) ein Ereignis hinzu. Das Dialogfenster 'Löschen bestätigen' soll geschlossen werden, wenn der Mitarbeiter gelöscht ist, d. h. nachdem der Binder den Service für die Datenquelle erfolgreich aufgerufen hat.

Fügen Sie den folgenden Code zur Methode 'getRemoveEmployeeAction()' hinzu:

```
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) {}
});
```

Durch diesen Ereigniscode wird das Dialogfenster 'Löschen bestätigen' ausgeblendet, wenn die Binderaktion beendet ist.

Prüfpunkt für die Lerneinheit

Wenn Sie nun die Anwendung 'My Company Directory' ausführen, können Sie einen Mitarbeiter in der Tabelle auswählen, auf die Schaltfläche **Löschen** und anschließend auf die Schaltfläche **Ja** klicken, um den Löschvorgang zu bestätigen. Der Mitarbeiterdatensatz wird aus dem Verzeichnis entfernt, und die Liste der Mitarbeiter gibt das Löschen des Datensatzes wieder.

Lerneinheit 2.6: Aktionen und Bindings für das Hinzufügen eines neuen Mitarbeiters konfigurieren

In dieser Lerneinheit ermöglichen Sie es der Anwendung 'My Company Directory', einen neuen Mitarbeiterdatensatz hinzuzufügen.

Das kompliziertere und dynamischere Verhalten der Anwendung für das Hinzufügen eines neuen Mitarbeiters macht auch diese Übung unweigerlich komplexer. Sie werden einige manuelle Änderungen am Quellencode vornehmen müssen. Darüber hinaus werden in dieser Übung einige erweiterte Funktionalitäten der Datenobjekte demonstriert, und es wird Ihnen ein kreatives Beispiel dafür geliefert, wie Sie die Binder und Datenobjekte an Ihre Anforderungen anpassen können.

In der folgenden Liste ist das erforderliche Verhalten der Anwendung beschrieben:

- Wenn Sie auf die Schaltfläche **Neu** klicken, führt dies zu folgendem Verhalten:
  - Die Auswahl in der Mitarbeitertabelle wird aufgehoben, und die Tabelle wird inaktiviert.
  - Das Aufheben der Tabellenauswahl führt zum Inaktivieren der Schaltfläche **Löschen**.
  - Das Feld **Filter** wird inaktiviert.
  - Sämtliche Werte in den Detailfeldern, mit Ausnahme einer neuen Mitarbeiter-ID, werden gelöscht.
  - Der Text auf der Schaltfläche **Aktualisieren** ändert sich in **Hinzufügen**.
- Wenn Sie auf die Schaltfläche **Hinzufügen** klicken, führt dies zu folgendem Verhalten:
  - Die in den Detailfeldern eingegebenen Werte werden dem Verzeichnis als neuer Mitarbeiterdatensatz hinzugefügt.
  - Die Tabelle wird aktiviert, und die Werte werden aktualisiert.
  - Das Feld **Filter** wird aktiviert.
  - Der Text auf der Schaltfläche **Hinzufügen** ändert sich wieder in **Aktualisieren**.

Ein neues Datenquellendatenobjekt hinzufügen, das createNewFullEmployeeRecord() aufruft

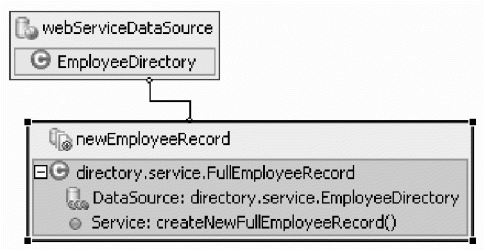
Der Beispiel-Web-Service stellt den Service 'createNewFullEmployeeRecord' mit einem neuen, leeren Mitarbeiterdatensatz zur Verfügung, der mit der nächsten verfügbaren Mitarbeiter-ID gefüllt wird. Dieser leere Datensatz kann anschließend mit den Informationen eines neuen Mitarbeiters gefüllt und an den Web-Service zurück übergeben werden.

1. Erweitern Sie in der Palette des Java Visual Editor das Ablagefach 'Datenobjekte', und wählen Sie **Datenquellendatenobjekt** aus.
2. Schieben Sie den Mauszeiger auf den freien Bereich der Entwurfssicht oder des unformatierten Bereichs, und klicken Sie mit der linken Maustaste, um das Datenquellendatenobjekt zu übergeben. Daraufhin wird ein neues Datenquellendatenobjekt hinzugefügt und im unformatierten Bereich angezeigt:

3. Klicken Sie mit der rechten Maustaste auf das Datenquellendatenobjekt, und wählen Sie **Feld umbenennen** aus. Benennen Sie das Datenobjekt in `newEmployeeRecord` um.
4. Klicken Sie mit der rechten Maustaste auf das Datenobjekt 'newEmployeeRecord', und wählen Sie **Binding-Eigenschaften** aus. Daraufhin wird das Dialogfenster 'Datenbinding' geöffnet.
5. Wählen Sie im Feld **Datenquelle** den Wert 'webServiceDataSource' aus.

6. Wählen Sie im Feld **Service** den Wert 'createNewFullEmployeeRecord()' aus.
7. Klicken Sie auf **OK**.

Im unformatierten Bereich können Sie sehen, dass das Datenquellendatenobjekt 'newEmployeeRecord' an den Web-Service gebunden ist.



Ein Basisdatenobjekt hinzufügen, um das Umschalten von Datenobjekten zu erleichtern

Da bei den Detailfeldern und bei der Schaltfläche 'Aktualisieren' ein Umschalten des Modus (sowohl zum Ausführen einer Aktualisierung als auch zum Erstellen eines neuen Mitarbeiters) erforderlich ist, müssen sie zu unterschiedlichen Zeiten an zwei unterschiedliche Datenobjekte gebunden werden. Um diesen Schritt zu erleichtern, fügen Sie ein Basisdatenobjekt namens 'switchingDataObject' hinzu. Sie verwenden dieses Basisdatenobjekt zum Umschalten des Bindings für die Textfelder zwischen 'selectedEmployeeRecord' und 'newEmployeeRecord'.

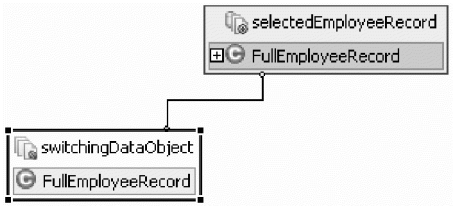
Das neue Basisdatenobjekt zeigt einfach auf ein anderes Datenobjekt (selectedEmployeeRecord), das Sie in einer früheren Übung definiert haben. Dieses neue Datenobjekt wird nützlich, wenn Sie eine Methode erstellen, die diesem Basisdatenobjekt vorgibt, den bereits zu einem früheren Zeitpunkt erstellten newEmployeeRecord zu verwenden. Mit anderen Worten: Dieses Basisdatenobjekt wird als temporäres Datenobjekt fungieren, das zwischen dem Datenobjekt 'selectedEmployeeRecord' und dem Datenobjekt 'newEmployeeRecord' umschaltet und somit den visuellen Komponenten in der Anwendung ermöglicht, mit zwei verschiedenen Datenobjekten zu arbeiten.

1. Wählen Sie in der Palette des Visual Editor das **Basisdatenobjekt** aus und übergeben Sie es an den unformatierten Bereich. Ein basicDataObject wird hinzugefügt.

2. Benennen Sie das Datenobjekt in `switchingDataObject` um.
3. Setzen Sie in der Sicht 'Eigenschaften' für 'switchingDataObject' die Eigenschaft **sourceObject** auf **selectedEmployeeRecord**. Sie können 'selectedEmployeeRecord' im Dropdown-Menü in der Spalte 'Wert' für die Eigenschaft auswählen.

   Nun verweist 'switchingDataObject' auf 'selectedEmployeeRecord' und gibt dieselben Werte wieder:

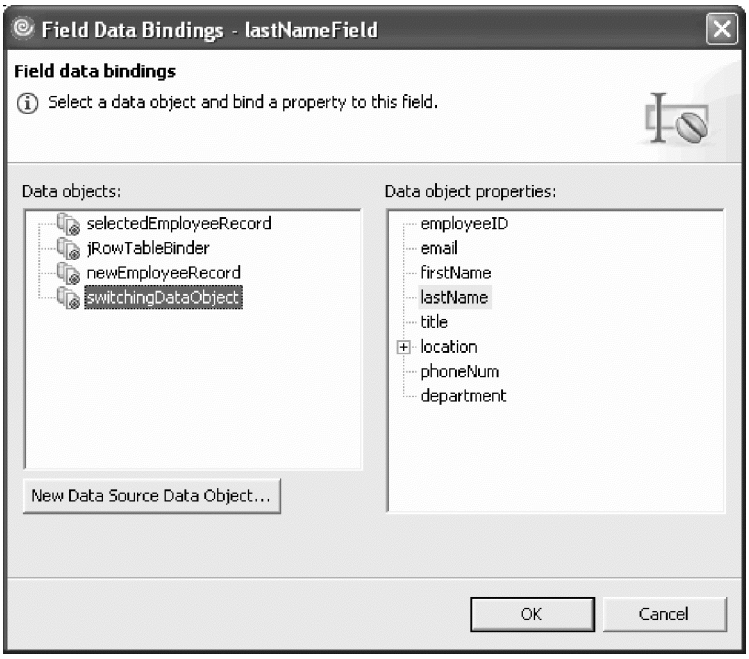
   

Jedes Mitarbeiterfeld erneut an das switchingDataObject binden

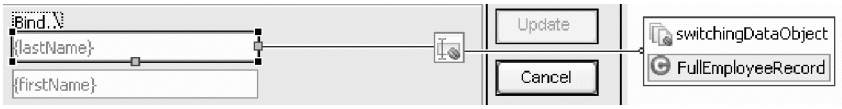
Es sind zwar bereits alle Mitarbeiterdetailfelder an selectedEmployeeRecord gebunden, nun binden Sie sie jedoch an switchingDataObject. Nach dem Binden der Felder können Sie dynamisch zwischen Datenobjekten für die Felder in Abhängigkeit davon umschalten, ob Sie einen vorhandenen Mitarbeiterdatensatz ändern oder einen neuen Mitarbeiterdatensatz hinzufügen.

Führen Sie für jedes Feld im Abschnitt für Mitarbeiterdetails die folgenden Schritte aus:

1. Wählen Sie das Feld aus, und klicken Sie auf die Registerkarte **Binden**.
2. Wählen Sie im Dialogfenster 'Felddaten-Bindings' das switchingDataObject aus. Zuvor haben Sie die Felder an den selectedEmployeeRecord gebunden.



3. Stellen Sie sicher, dass das Feld noch an die richtige Datenobjekteigenschaft gebunden ist, und klicken Sie auf **OK**. Wenn Sie das Feld in der Entwurfssicht auswählen, können Sie sehen, dass die Binderlinien nun auf switchingDataObject zeigen.



Eine Markierung und eine Methode zum Aktualisieren und Umschalten von Modi definieren

Die folgende Methode 'updateMode()' prüft, ob die Modusmarkierung auf 'new' gesetzt ist, und ändert das Verhalten der Anwendung anschließend entsprechend. Standardmäßig ist die Boolesche Markierung 'isNewMode' auf 'false' gesetzt, und die Methode 'updateMode()' aktiviert die Mitarbeitertabelle und das Filterfeld und setzt den Text der Schaltfläche zum Aktualisieren auf 'Aktualisieren'. Wenn 'isNewMode' auf 'true' gesetzt wird, wird die Mitarbeitertabelle inaktiviert, die Auswahl wird ggf. aufgehoben, das Filterfeld wird inaktiviert und der Text der Schaltfläche zum Aktualisieren wird auf 'Hinzufügen' gesetzt.

Fügen Sie der Klasse 'DirectoryApp.java' den folgenden Code unmittelbar vor der letzten schließenden geschweiften Klammer hinzu:

```
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");
   }
}
```

Der Schaltfläche 'Neu' ein actionPerformed-Ereignis hinzufügen

In diesem Schritt fügen Sie Ereigniscode für das Klicken auf die Schaltfläche **Neu** hinzu. Das Ereignis gibt für das switchingDataObject vor, das Datenobjekt 'newEmployeeRecord' zu verwenden, es setzt die Modusmarkierung auf 'new' und führt die Methode 'updateMode()' aus, die Sie im vorherigen Schritt hinzugefügt haben.

1. Klicken Sie in der Entwurfssicht mit der rechten Maustaste auf die Schaltfläche **Neu**, und wählen Sie **Ereignisse → actionPerformed** aus. Der folgende Code wird in der Methode 'getNewButton()' generiert:

```
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()
   }
});
```

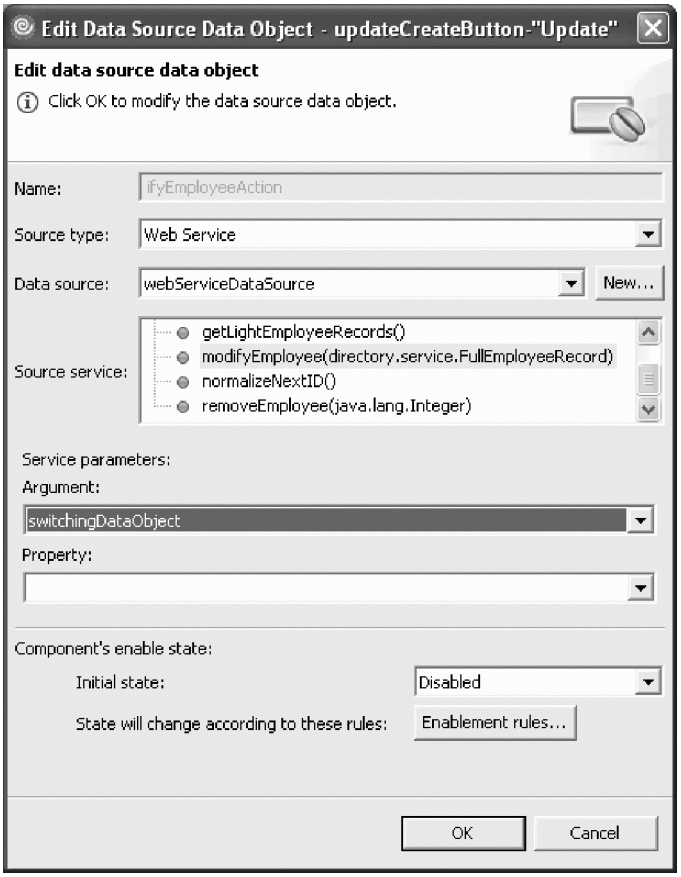
2. Ersetzen Sie diesen generierten Stub durch den folgenden Code:

```
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();
   }
});
```

Die Schaltfläche 'Aktualisieren' erneut binden

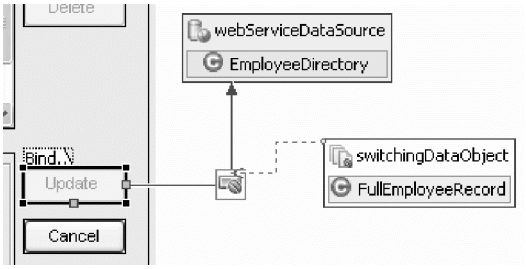
In einer früheren Lerneinheit haben Sie die Schaltfläche **Aktualisieren** für die Verwendung der Methode 'modifyEmployee' für den Web-Service programmiert. Diese Aktion wird als SwingDataServiceAction implementiert. Eine der Eigenschaften der SwingDataServiceAction ist das Quellenobjekt, das als Argument für den Service dient. Das Quellenobjekt für die Änderungsaktion ist momentan auf 'selectedEmployeeRecord' gesetzt. Um die Schaltfläche für das Steuern einer Aktualisierung und einer Hinzufügung zu programmieren, werden Sie die Aktion der Schaltfläche neu konfigurieren, sodass sie 'switchingDataObject' als Argument für den Service 'modifyEmployee' verwendet.

1. Wählen Sie in der Entwurfssicht die Schaltfläche **Aktualisieren** aus. Beachten Sie den rosa gepunkteten Pfeil, der anzeigt, dass der selectedEmployeeRecord das Argument für den Serviceaufruf ist.
2. Klicken Sie auf die Registerkarte **Binden** auf der Schaltfläche **Aktualisieren**.
3. Wählen Sie im Feld **Argument** den Wert 'switchingDataObject' aus.



4. Klicken Sie auf **OK**.

   Beachten Sie, dass die Aktion der Schaltfläche nun für die Verwendung des switchingDataObject als Argument für die Methode 'modifyEmployee' konfiguriert ist:



Dem Binder der Schaltfläche 'Aktualisieren' ein Ereignis hinzufügen, um den Modus zurückzusetzen

Nach dem Klicken auf die Schaltfläche **Aktualisieren** und dem Beenden der Aktion für den Web-Service soll die Anwendung zurück auf ihren Standardmodus und ihr Standardverhalten gesetzt werden. Hierzu fügen Sie dem Aktionsbinder der Schaltfläche einen Ereignislistener hinzu, der den Modus und die Tabelle aktualisiert, nachdem das Aktualisieren oder Hinzufügen abgeschlossen ist.

Fügen Sie den folgenden Code zur Methode 'getModifyEmployeeAction()' für die Schaltfläche 'Aktualisieren' hinzu:

```
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) {}
});
```

Prüfpunkt für die Lerneinheit

Wenn Sie nun die Anwendung 'My Company Directory' ausführen, können Sie auf die Schaltfläche **Neu** klicken und einen neuen Mitarbeiterdatensatz hinzufügen.

Lerneinheit 2.7: Das Verhalten der Schaltfläche 'Abbrechen' programmieren

Beim Verwenden Ihrer Anwendung möchten Sie in der Lage sein, alle an einem Mitarbeiterdatensatz von Ihnen begonnenen Änderungen ohne großen Aufwand zu verlassen, wenn Sie die Änderungen nicht übergeben möchten. Das heißt, Sie müssen die Eingabe in die Felder abbrechen und löschen können, damit Sie noch einmal neu beginnen können. Zum Hinzufügen dieser Funktionalität müssen Sie einige actionPerformed-Ereignisse für die Schaltfläche **Abbrechen** definieren.

In der folgenden Liste ist das erforderliche Verhalten der Schaltfläche **Abbrechen** beschrieben:

- Wenn Sie auf die Schaltfläche **Abbrechen** klicken, während der Modus 'Neu' eingestellt ist, wird die Anwendung aus dem Modus 'Neu' zurückgesetzt.
- Wenn Sie beim Ändern eines Mitarbeiterdatensatzes auf die Schaltfläche **Abbrechen** klicken, werden alle von Ihnen geänderten Werte auf die ursprünglichen Werte zurückgesetzt.

Gehen Sie wie folgt vor, um der Schaltfläche **Abbrechen** ein actionPerformed-Ereignis hinzuzufügen, damit das erforderliche Verhalten erzielt wird:

1. Klicken Sie in der Entwurfssicht mit der rechten Maustaste auf die Schaltfläche **Abbrechen**, und wählen Sie **Ereignisse → actionPerformed** aus. Der folgende Code wird in der Methode 'getCancelButton()' generiert:

   ```
   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. Ersetzen Sie den generierten Ereignis-Stub durch den folgenden Code:

   ```
   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();
         }
      }
   });
   ```

Prüfpunkt für die Lerneinheit

In dieser Lerneinheit haben Sie gelernt, die Schaltfläche **Abbrechen** mit actionPerformed-Ereignissen zu programmieren.

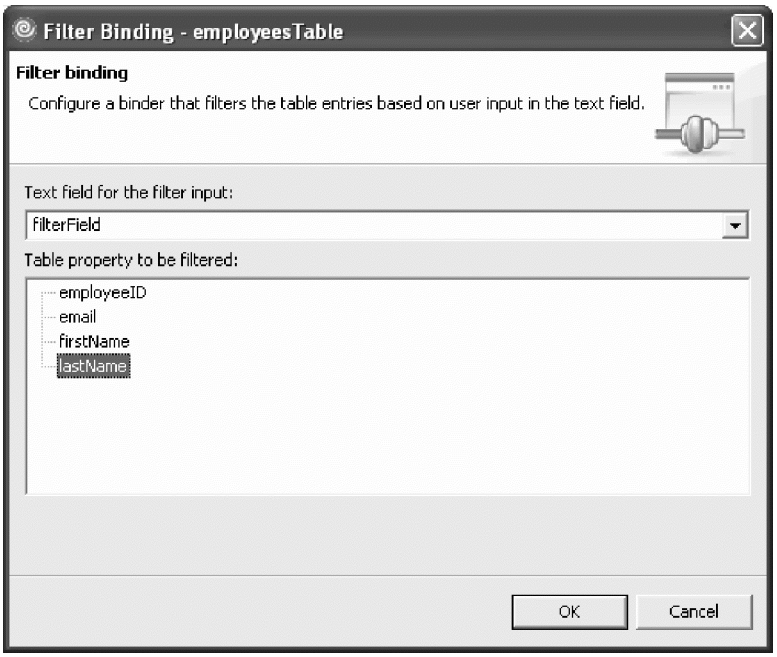
Lerneinheit 2.8: Einen Filter für die Mitarbeitertabelle konfigurieren

Sie können einen Textfilterbinder verwenden, um den Inhalt der Mitarbeitertabelle zu filtern. Der Filter erhält Eingabe von einem Textfeld und filtert die Tabelle auf Basis einer bestimmten Eigenschaft oder Spalte in der Tabelle.

In der Anwendung verwenden Sie die im Feld **Filter** eingegebenen Zeichen, um nach dem Nachnamen von Mitarbeitern zu filtern. Wenn genau die im Feld **Filter** eingegebenen Werte im Nachnamen eines Mitarbeiterdatensatzes vorkommen, wird dieser Mitarbeiterdatensatz in der Tabelle angezeigt.

Gehen Sie wie folgt vor, um einen Filter für die Tabelle zu erstellen:

1. Wählen Sie das Binder-Symbol für employeesTable aus, und wählen Sie **Filter-Binding-Eigenschaften** aus. Das Dialogfeld 'Filter-Binding' wird geöffnet.
2. Wählen Sie in der Liste **Textfeld für die Filtereingabe** den Eintrag **filterField** aus.
3. Wählen Sie in der Liste **Zu filternde Tabelleneigenschaft** den Eintrag **lastName** aus.



4. Klicken Sie auf **OK**.

Es wird ein neuer SwingPropertyFilter generiert. Die Filtereigenschaft für den Binder der Tabelle wird für das Verwenden des neuen Filters konfiguriert. Der neue Filter wird für die Verwendung des Felds **Filter** als Eingabe konfiguriert und zum Filtern nach der Eigenschaft lastName der Tabelle.

Prüfpunkt für die Lerneinheit

In dieser Lerneinheit haben Sie gelernt, einen Filter für eine Tabelle zu konfigurieren.

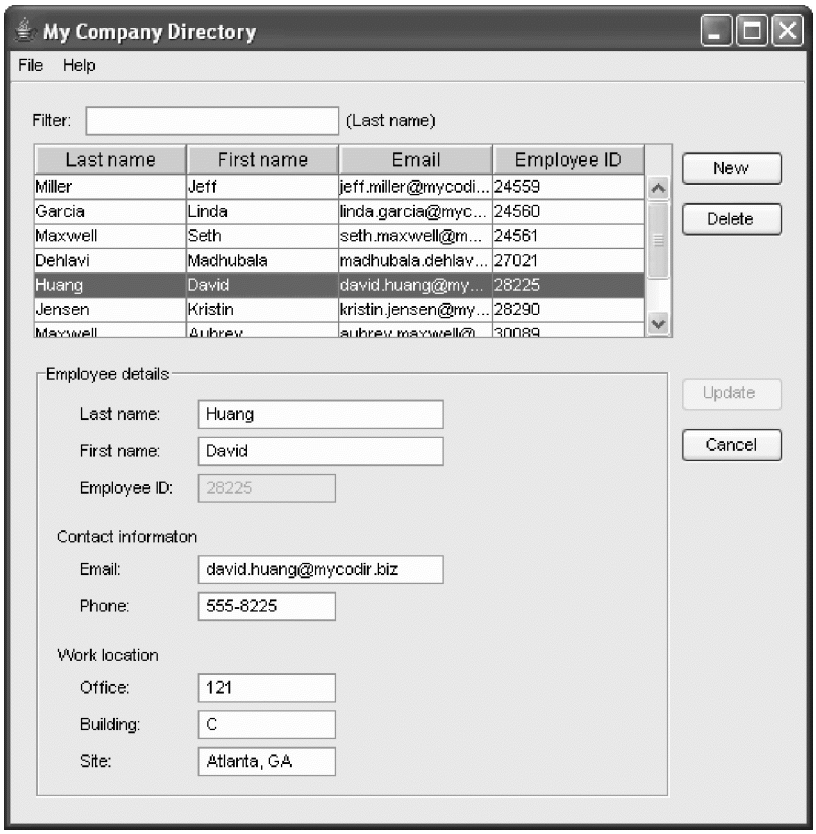
Wenn Sie nun die Anwendung 'My Company Directory' ausführen, können Sie im Feld **Filter** Zeichen eingeben. Die Tabelle wird daraufhin so gefiltert, dass die Zeilen angezeigt werden, in denen der Nachname die eingegebenen Zeichen enthält.

Herzlichen Glückwunsch! Die Anwendung 'My Company Directory' ist fertig.

Zusammenfassung: Einen Rich-Java-Client erstellen, der einen Web-Service verwendet

Herzlichen Glückwunsch! Sie haben gelernt, den Java Visual Editor zum Erstellen der Anwendung 'My Company Directory' zu verwenden. Dabei handelt es sich um einen Rich-Java-Client, der mit einem Beispiel-Web-Service verbunden ist, um ein Mitarbeiterverzeichnis zu verwalten.

Ein Bild des fertigen Produkts anzeigen:



Durchgeführte Lerneinheiten

Sie haben den Visual Editor verwendet, um die grafische Benutzerschnittstelle unter Verwendung von GridBagLayout zum Anordnen der Mitarbeitertabelle zu erstellen. Anschließend haben Sie die Tabelle, die Felder und Schaltflächen an entsprechende Datenobjekte gebunden, damit die Anwendung mit einem von Ihnen generierten Web-Service-Java-Proxy arbeiten kann. Ferner haben Sie einige komplexe Codierungen vorgenommen, damit die Anwendung das richtige Verhalten aufweist und einfach und intuitiv verwendet werden kann. Außerdem haben Sie gelernt, eine Unternehmensanwendung auf WebSphere Application Server v6.0 zu installieren und einen Web-Service zu implementieren.

Als wichtigstes Ziel haben Sie alles über die leistungsfähige Binderklasse erfahren, die vom Java Visual Editor für die Arbeit mit Daten bereitgestellt wird. Nun können Sie eigenständig experimentieren und die Binder für neue und interessante Aufgaben verwenden.

Sie sollten nun in der Lage sein, die folgenden Aufgaben auszuführen:

- Verwenden des Java Visual Editor zum Entwerfen von Komponenten in einem GridBagLayout.
- Ausführen einer visuellen Klasse als Java-Bean.

- Binden der visuellen Komponenten einer Java-Anwendung an die Methoden und Datenobjekte, die von einem Web-Service zurückgegeben werden.
- Hinzufügen von Ereignissen zu visuellen Komponenten.

Zusätzliche Ressourcen

Eine fertige Version der Anwendung 'My Directory' importieren

Dieses Projekt enthält die fertige Anwendung, das Paket 'jve.generated' mit Binder-Klassen, sowie den Web-Service-Java-Client, der für WebSphere Application Server v6.1 konfiguriert ist. Wenn Sie dieses fertige Projekt importieren, ohne das Lernprogramm durchzuarbeiten, müssen Sie unter Umständen Ihre Java-Java-Erstellungspfadvariable konfigurieren. Sie muss auf Ihre Thin Client-JAR-Datei für WebSphere v6.1 Web-Services zeigen.

Tipp: Sofern Sie beim Import keinen anderen Projektnamen angeben, werden hierdurch die Inhalte Ihres Projekts 'MyDirectory' überschrieben.