Génération d'un client Java enrichi qui utilise un service Web

Table des matières

Générer un client Java enrichi qui utilise un service Web

Ce tutoriel explique comment utiliser l'éditeur visuel Java pour générer un client Java enrichi qui se connecte à un service Web. Le client que vous générez dans le tutoriel est appelé My Company Directory.

My Company Directory est une application Java qui sert à gérer l'annuaire des employés d'une société. L'application se connecte à un exemple de service Web qui fournit des méthodes permettant de créer, d'extraire, de mettre à jour et de supprimer des enregistrements d'employés.

Le client est créé visuellement dans l'éditeur visuel Java à l'aide de composants Swing. L'éditeur visuel Java offre un ensemble de classes auxiliaires (sources de données, objets de données et lieurs, ou ″binders″) permettant de se connecter au service Web et de l'utiliser. Le service Web est déployé localement sur votre installation d'IBM WebSphere Application Server version 6.0, et les outils vous aident à générer un proxy Java pour votre client, fondé sur un fichier WSDL (Web Services Description Language).

Voir le produit fini

Objectifs d'apprentissage

Dans ce tutoriel, vous allez apprendre à accomplir les tâches suivantes :

- Comment utiliser l'éditeur visuel Java pour concevoir et agencer une interface utilisateur
- Comment lier les éléments de l'interface à des objets de données et à un service Web

Durée prévue

2 heure et 15 minutes

Information associée

Afficher la version PDF

Tutoriel : Hello World Java

Introduction : Génération d'un client Java enrichi qui utilise un service Web

L'interface graphique du client a, en grande partie, été générée préalablement de façon visuelle à l'aide de composants Swing. Le premier module permet de mettre en page les composants essentiels de l'interface graphique à l'aide de l'éditeur visuel Java. Dans le second module, vous allez associer les composants de l'interface graphique à la source de données du service Web, aux services et aux objets renvoyés par la source de données. Lors de la génération de l'application dans l'éditeur visuel Java, vous allez utiliser des sources de données, des objets de données et des lieurs (binders), qui sont des instances de classes auxiliaires générées par l'éditeur visuel Java et utilisées par votre application.

Image du produit fini :

Objectifs d'apprentissage

Dans ce tutoriel, vous allez apprendre à accomplir les tâches suivantes :

- Comment utiliser l'éditeur visuel Java pour concevoir et agencer une interface utilisateur
- Comment lier les éléments de l'interface à des objets de données et à un service Web

Durée prévue

Pour exécuter la totalité du tutoriel, vous avez besoin d'environ 2 heures et 30 minutes.

Configuration requise

- WebSphere Application Server, version 6.1. Ce serveur est peut-être déjà en place sur votre machine si vous avez choisi de l'installer en même temps que le produit. Le cas échéant, vous pouvez aussi utiliser votre propre installation autonome. Dans le présent tutoriel, il vous sera demandé de déployer un exemple de service Web sur le serveur WebSphere Application Server exécuté localement sur votre machine.

Il est possible que l'exemple de service Web fonctionne sur d'autres serveurs, mis ce tutoriel n'a été testé qu'avec WebSphere Application Server versions 6.0 et 6.1.

Conditions préalables

Vous devez être familiarisé avec les concepts suivants :

- Principes élémentaires du développement Java
- Principes de fonctionnement des services Web
- Principes d'utilisation du plan de travail, tels que le travail avec des projets et la navigation dans les perspectives et les vues

Module 1 : Conception de l'interface graphique du client dans l'éditeur visuel

Ce module vous apprendra à utiliser l'éditeur visuel Java pour ajouter un composant visuel à une application, puis à l'agencer visuellement et définir ses contraintes de présentation. La dernière leçon de ce module vous fera exécuter le fichier Java pour voir comment il se présentera en tant qu'application.

A faire : Avant de commencer ce module, vous devez posséder les connaissances requises stipulées dans l'introduction du tutoriel.

Objectifs d'apprentissage

Au terme des leçons de ce module, vous connaîtrez les concepts des opérations suivantes et saurez comment les exécuter :

- Ajouter et agencer un composant JTable dans une interface Java
- Exécuter une classe visuelle pour tester votre travail

Durée prévue

Ce module vous prendra environ 15 minutes.

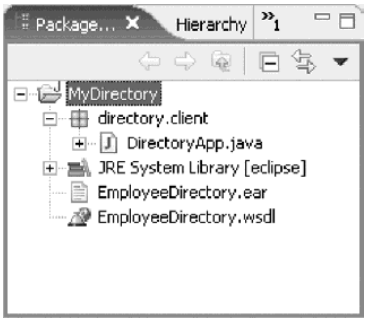
Leçon 1.1 : Mise en place du projet Java

Dans cette leçon, vous allez mettre en place le projet MyDirectory en l'important dans votre espace de travail. Ce projet inclut une seule classe Java ainsi que d'autres fichiers qui seront utilisés plus tard.

Comme l'objectif principal de ce tutoriel est d'illustrer la liaison de composants visuels à un service Web, l'interface graphique Java de l'application "My Company Directory" est en grande partie déjà créée pour vous.

Le projet MyDirectory est le principal projet Java que vous allez utiliser dans ce tutoriel. Il contient votre fichier DirectoryApp.java qui est le fichier Java contenant l'application Java principale que vous créez. Plusieurs versions du projet MyDirectory sont incluses avec ce tutoriel : une pour le démarrage de chaque module et une version finie du projet.

1. Importer le projet MyDirectory.
2. Dans la vue Explorateur de package de la perspective Java, vérifiez que votre projet MyDirectory ressemble à l'image suivante :



Récapitulatif de la leçon

Dans cette leçon, vous avez importé l'exemple de projet MyDirectory, qui constitue le point de départ du présent tutoriel.

Le projet MyDirectory contient les ressources suivantes :

- DirectoryApp.java : Fichier Java contenant l'application que vous développez dans ce tutoriel. Le fichier DirectoryApp.java se trouve dans un package Java intitulé directory.client.
- EmployeeDirectory.ear : Application d'entreprise contenant l'exemple de service Web. Dans le module 2, vous allez déployer ce service Web sur une installation locale de WebSphere Application Server, version 6.0.
- EmployeeDirectory.wsdl : Fichier XML qui utilise le langage WSDL (Web Services Description Language) pour décrire l'exemple de service Web à déployer. Dans le module 2, vous allez utiliser ce fichier WSDL pour générer un proxy Java à l'usage de votre application.

Leçon 1.2 : Ajout et agencement de la table des employés

Dans cette leçon, vous allez utiliser l'éditeur visuel Java pour ajouter un JScrollPane et une JTable à l'application. Dans des exercices ultérieurs, vous programmerez la JTable de sorte qu'elle obtienne ses données d'un service Web qui renvoie la liste de tous les employés enregistrés dans l'annuaire de la société.

Après avoir ajouté la JTable, vous utiliserez la vue de conception de l'éditeur visuel Java pour personnaliser l'agencement de la JTable conformément aux spécifications suivantes :

- Etendre la JTable à trois cellules horizontalement et deux cellules verticalement
- Ajouter un encart gauche de 15 pixels
- Renommer la JTable en employeesTable.

Afficher une démonstration

Ouvrir le fichier DirectoryApp.java dans l'éditeur visuel Java

Pour ouvrir le fichier DirectoryApp.java dans l'éditeur visuel Java, procédez comme suit :

1. Dans la vue Packages de la perspective Java, développez la branche du projet MyDirectory et celle du package directory.client.
2. Cliquez avec le bouton droit sur le fichier DirectoryApp.java et sélectionnez **Ouvrir avec → Editeur visuel**. L'éditeur visuel Java charge la classe Java et affiche la conception dans la zone graphique du canevas.

Conseil :

- Pour changer l'apparence utilisée par l'éditeur visuel Java, sélectionnez **Fenêtre → Préférences → Java → Editeur visuel** et spécifiez une apparence pour Swing. La nouvelle préférence prendra effet à la prochaine ouverture de la classe. Le présent tutoriel utilise l'apparence Windows.
- Pour que l'éditeur visuel soit l'éditeur par défaut de tous les fichiers Java, vous pouvez cliquer sur **Fenêtre → Préférences** et accéder à la page **Plan de travail → Associations de fichiers** pour définir votre préférence.

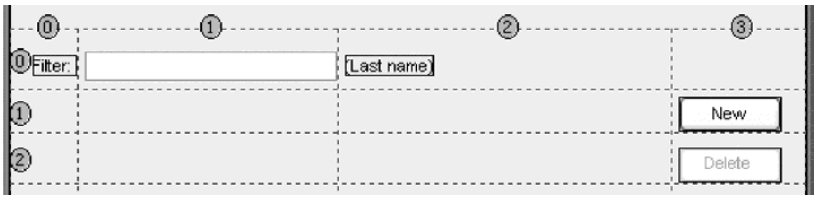
Ajouter une JTable sur un JScrollPane

La fenêtre principale de DirectoryApp.java utilise un JFrame doté d'un JPanel pour son panneau de contenu principal. Le JPanel de notre application est intitulé jContentPane. Le jContentPane a été configuré pour l'utilisation d'un type de gestionnaire d'agencement appelé GridBagLayout. Le GridBagLayout est un puissant schéma d'agencement basé sur une grille de cellules qui peuvent être occupées par des composants visuels. Pour faciliter l'utilisation de GridBagLayout, l'éditeur visuel java affiche les bordures de grille. Il affiche également des marqueurs de positionnement lorsque de nouveaux composants sont placés sur la grille, ainsi que des poignées sur les composants que vous redimensionnez ou déplacez dans le GridBagLayout.

Pour ajouter la table des employés (javax.swing.JTable) à l'interface utilisateur DirectoryApp.java, procédez comme suit :

1. Cliquez avec le bouton droit sur le jContentPane dans la vue de conception ou dans la vue Beans Java et sélectionnez **Afficher la grille**. Un trait pointillé rouge affiche la bordure de la grille et des cercles

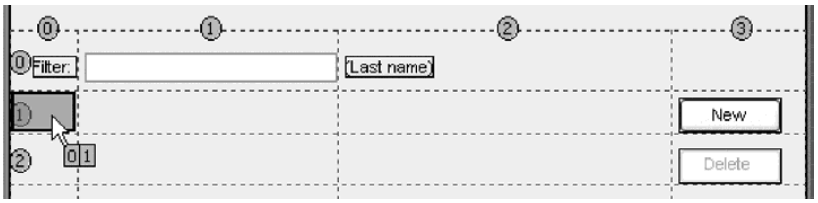
numérotés de couleur bleue indiquent les numéros de ligne et de colonne. Par exemple, vous remarquerez que le bouton **New** occupe la cellule située à l'intersection de la ligne 1 (axe y de la grille) et de la colonne 3 (axe x de la grille).



2. Dans la palette de l'éditeur visuel Java, sélectionnez le composant Swing **JTable dans JScrollPane** . Vous le trouverez dans le tiroir **Composants Swing** de la palette.

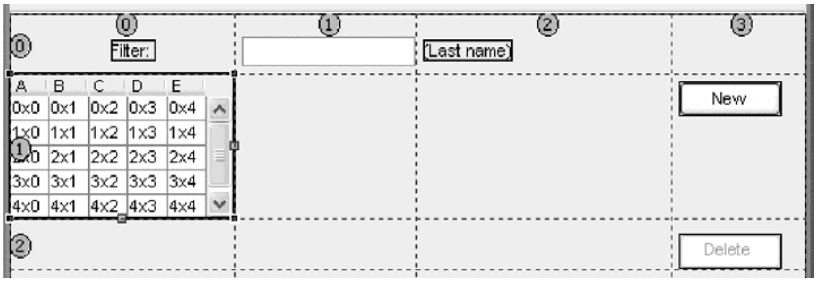
   Conseil : Par défaut, la palette est réduite sur le côté droit de la zone de conception. Vous pouvez la redimensionner et la déplacer.

3. Amenez le pointeur de la souris sur la cellule de la colonne 0 et de la ligne 1 de la grille :



   - Lors du déplacement du pointeur de la souris sur la grille, le pointeur affiche deux carrés numérotés qui indiquent les coordonnées x et y de la grille en fonction de l'emplacement du pointeur.
   - Si le pointeur est placé directement sur une bordure de grille, il se peut que de nouvelles lignes et colonnes soient créées ; les lignes et colonnes existantes sont donc renumérotées. Dans ce cas, des carrés jaunes sur le pointeur, des barres jaunes entre les lignes de quadrillage et des intitulés de colonne ou de ligne jaunes indiquent ce comportement et révèlent l'impact qu'aura le positionnement de l'objet à cet endroit.

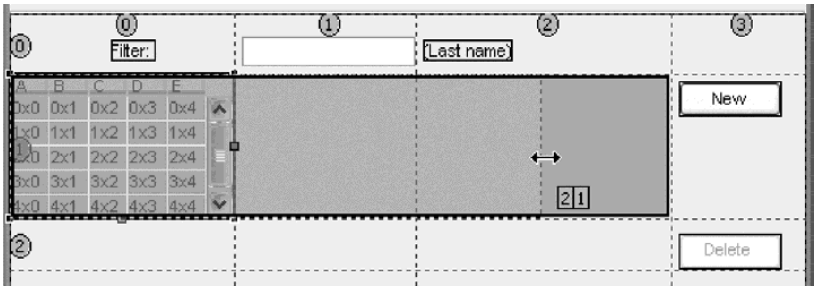
4. Cliquez avec le bouton gauche de la souris pour insérer le JScrollPane et la JTable dans la cellule située à la colonne 0, ligne 1 :



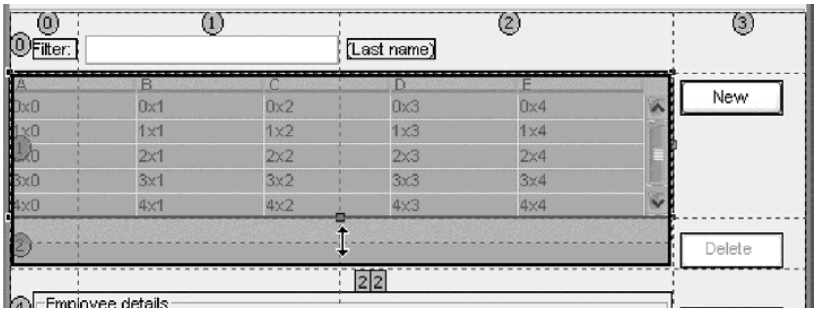
Etendre le JScrollPane et la JTable sur plusieurs colonnes et lignes de la grille

A présent, nous voulons que le JScrollPane (et sa JTable enfant) s'étendent sur trois colonnes et deux lignes de la grille pour améliorer l'espacement et le redimensionnement. Voici comment procéder :

1. Sélectionnez le JScrollPane dans la zone de conception ou la vue Beans Java (il doit encore être sélectionné, car vous venez de l'ajouter). Notez les petits carrés verts sur le côté droit et en bas du JScrollPane. Vous allez utiliser ces poignées de redimensionnement pour étendre le JScrollPane sur plusieurs colonnes et plusieurs lignes de la grille.
2. Cliquez sur la poignée verte de la bordure droite du JScrollPane (ne relâchez pas le bouton de la souris).
3. Faites glisser le pointeur de la souris vers la droite jusqu'à ce que les coordonnées indiquent colonne 2, ligne 1. Un ombrage gris foncé indique également les cellules que le composant occupera lorsque vous relâcherez le bouton de la souris.



4. Relâchez le bouton de la souris. Le JScrollPane s'étend désormais sur trois colonnes.
5. Répétez le même processus pour faire glisser la poignée inférieure du JScrollPane jusqu'à ce que ce dernier s'étende sur la ligne 2 de la grille :



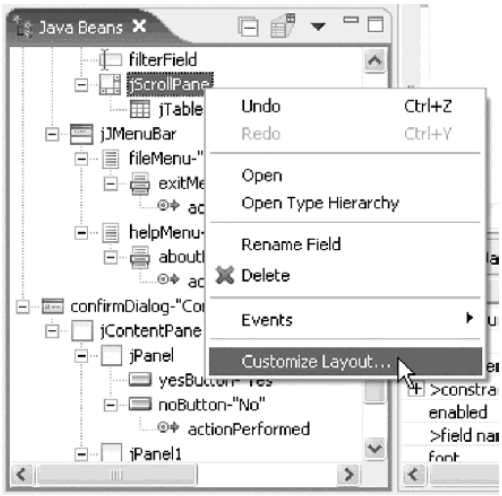
Personnaliser l'espacement du JScrollPane dans le GridBag

Le gestionnaire GridBagLayout permet également de définir diverses contraintes pour personnaliser encore plus la disposition. Vous pouvez par exemple spécifier les contraintes suivantes :

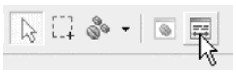
- **ancre :** une orientation de type ancre peut être donnée à un composant dans sa cellule, ce qui a une incidence sur la façon dont le composant se déplace lorsque l'application est redimensionnée par un utilisateur. Par exemple, un composant peut être ancré dans sa partie supérieure gauche, dans sa partie centrale gauche, au centre ou dans sa partie inférieure droite.
- **remplissage :** un composant peut occuper tous l'espace disponible dans sa ou ses cellules horizontalement et/ou verticalement.
- **encarts :** un composant peut recevoir son propre remplissage en haut, en bas, à gauche et à droite afin d'assurer un espacement entre lui-même et le bord de la grille.

Pour personnaliser l'ancre, le remplissage et les encarts du JScrollPane, procédez comme suit :

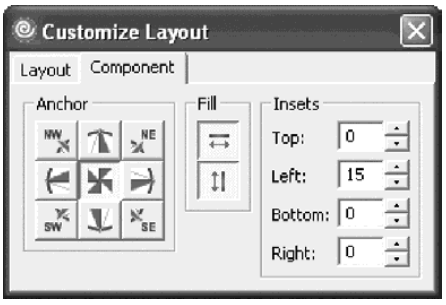
1. Cliquez avec le bouton droit sur le JScrollPane dans la vue de conception ou dans la vue Beans Java et sélectionnez **Personnaliser la disposition**.



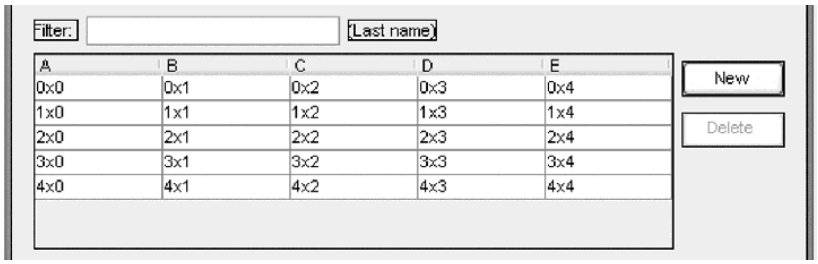
Conseil : La boîte de dialogue Personnalisation de la présentation peut rester ouverte lorsque vous sélectionnez et modifiez l'agencement de différents composants. Vous pouvez ouvrir cette boîte de dialogue à tout moment en cliquant sur le bouton correspondant de la barre d'outils :



2. Sous l'onglet Composant de la boîte de dialogue, vérifiez que le bouton Ancre centre est celui qui est sélectionné.
3. Assurez-vous que les deux boutons **Remplissage horizontal** et **Remplissage vertical** sont actionnés.
4. Ajoutez un encart gauche de 15 (pixels) pour que l'espacement à gauche du JScrollPane soit similaire à celui des autres composants visuels de l'application.



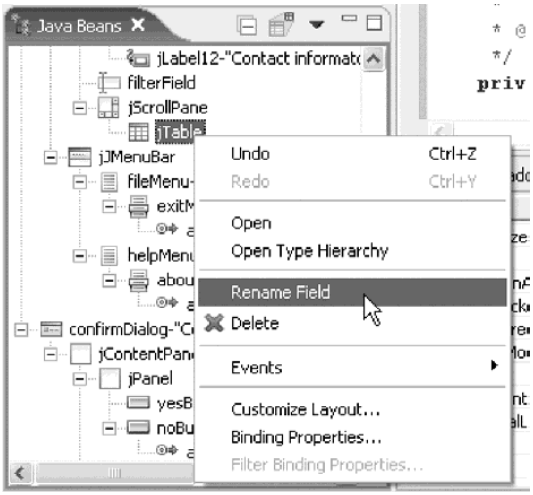
Par exemple, la table est maintenant alignée avec le libellé **Filter**.



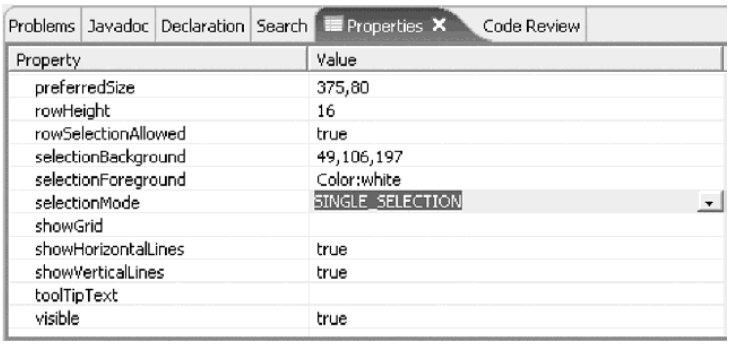
Renommer le nouveau JTable et le configurer pour la sélection d'une seule ligne à la fois

Dans la mesure où vous devrez continuer à travailler sur cette table au cours d'autres leçons, il est utile de renommer l'instance JTable correspondante et sa méthode d'accès get. Pour renommer la table, procédez comme suit :

1. Dans la vue Beans Java, cliquez avec le bouton droit sur le composant jTable et sélectionnez **Renommer la zone**.



2. Entrez `employeesTable` et cliquez sur **OK**. La JTable est désormais intitulée employeesTable et la méthode qui permet de l'instancier est getEmployeesTable.
3. Configurez la table afin qu'elle autorise la sélection d'une seule ligne à la fois :
   a. Sélectionnez employeesTable dans la vue de conception.
   b. Dans la vue Propriétés, sélectionnez la propriété **selectionMode** et choisissez la valeur SELECTION_UNIQUE.



   c. Sauvegardez le fichier DirectoryApp.java.

Récapitulatif de la leçon

Dans cette leçon, vous avez appris à utiliser l'éditeur visuel pour ajouter une table à une interface utilisateur existante. Vous avez ensuite appris à personnaliser son agencement, son positionnement et son espacement.

Leçon 1.3 : Exécution de la classe visuelle

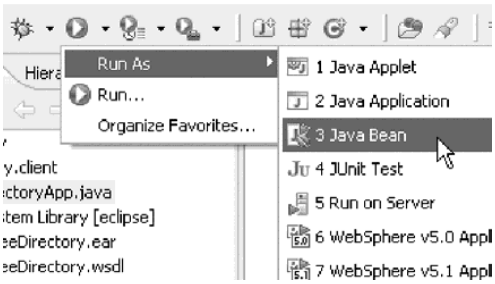
A présent, vous êtes prêt à exécuter l'application Java pour avoir un aperçu de sa présentation. Le plan de travail et l'éditeur visuel facilitent considérablement l'exécution rapide de votre application. Vous pouvez répéter ces étapes à tout moment dans le cadre du développement, afin de tester la présentation réelle de l'environnement d'exécution et le comportement de la classe.

Afficher une démonstration

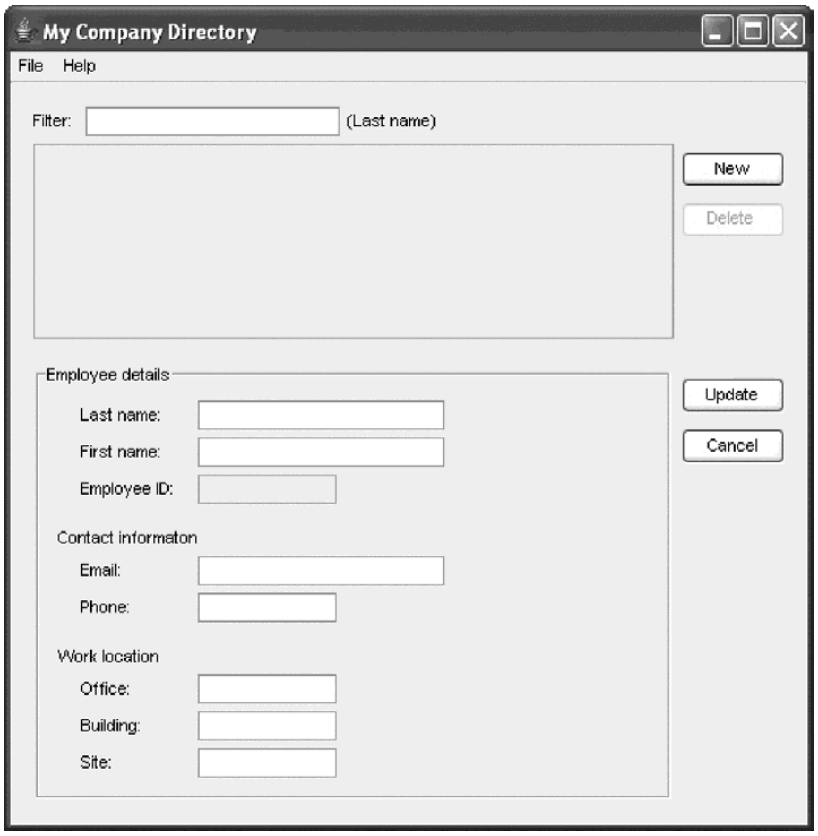
L'éditeur visuel Java fournit un lanceur de beans Java capable d'exécuter des classes dépourvues de méthode main(). Lorsqu'il exécute la classe visuelle, il lance l'application dans une machine virtuelle (VM) distincte. Si vous exécutez la classe visuelle en tant qu'application Java, le lanceur tente d'exécuter la méthode main() de la classe. Pour ce tutoriel, votre application inclut une méthode main() qui appelle et affiche le JFrame DirectoryApp. Vous pouvez donc l'exécuter en tant qu'application Java ou bean Java.

Pour exécuter le fichier DirectoryApp.java comme bean Java, procédez comme suit :

1. Assurez-vous que votre fichier DirectoryApp.java est ouvert dans l'éditeur visuel Java.
2. Dans la barre de menus, cliquez sur **Exécuter → Exécuter en tant que → Bean Java**.



Conseil : L'application s'ouvre sur le bureau en utilisant l'apparence Swing que vous avez définie dans les préférences de l'éditeur visuel (**Fenêtre → Préférences → Java → Editeur visuel**). Vous pouvez également sélectionner **Exécuter → Exécuter...** et définir l'apparence souhaitée dans la configuration de lancement particulière utilisée pour lancer ce bean Java. Si vous exécutez cette application comme application et non comme bean, elle utilisera également l'apparence Windows, car celle-ci est définie dans la méthode main(). Les captures d'écran de ce tutoriel illustrent l'apparence Windows.



Récapitulatif de la leçon

Comme vous n'avez conçu que l'interface et que vous n'avez pas encore programmé de fonctionnalité de gestion d'événements ni de connexion aux données, vous ne pouvez rien faire de l'application à ce stade. Toutefois, vous pouvez voir l'agencement de base et l'aspect de votre application telle qu'elle apparaîtra à un utilisateur. Vous pouvez essayer de cliquer sur certains des boutons, mais vous remarquerez qu'ils n'ont pas d'effet. Cependant, les menus File et Help sont déjà implémentés. Vous pouvez les tester pour voir leur fonction. De même, vous pouvez examiner le code Java pour voir comment ils sont implémentés au moyen d'événements actionPerformed.

Leçons étudiées

Ce module vous a montré comment concevoir l'interface d'un client enrichi à l'aide de l'éditeur visuel Java. Cependant, la conception de l'aspect visuel n'est qu'une partie du développement. Pour que le client soit fonctionnel, il vous reste à accomplir beaucoup d'autres tâches. En général, vous avez besoin d'inclure un comportement de gestion des événements ou une autre logique. Dans le cas présent, il convient également de lier les éléments visuels à une source de données.

Dans ce module, vous avez appris à effectuer les tâches suivantes :

- Importer un projet Java sous forme d'informations d'échange de projets
- Ajouter une JTable sur un JScrollPane à votre classe visuelle
- Utiliser le gestionnaire GridBagLayout pour agencer visuellement la table dans l'interface du client enrichi
- Exécuter l'application pour visualiser l'aspect réel du client Java enrichi

Dans le module suivant (Module 2 : Liaison des composants visuels au service Web), vous allez transformer cette interface simple en un puissant client accédant aux méthodes d'un service Web pour créer, extraire, mettre à jour et supprimer des enregistrements d'employé dans un annuaire de société.

Module 2 : Liaison des composants visuels au service Web

Ce module vous apprend à lier les éléments visuels de l'application My Company Directory (boutons, table d'employés, zones et autres actions) à un service Web. Ce service Web fournit la fonctionnalité de traitement proprement dite qui permet de créer, d'extraire, de mettre à jour et de supprimer des employés de l'exemple d'annuaire.

Objectifs d'apprentissage

Au terme des leçons de ce module, vous connaîtrez les concepts des opérations suivantes et saurez comment les exécuter :

- Lier une table à une source de données de service Web
- Lier les champs (ou zones) aux objets
- Programmer les boutons avec des actions

L'étude de ce module vous prendra environ **2 heures**.

Leçon 2.1 : Installation et déploiement du service Web

Dans cette leçon, vous allez installer un exemple d'application d'entreprise (fichier EAR) dans IBM WebSphere Application Server version 6.1 et déployer le service Web EmployeeDirectory. Votre application utilisera ce service Web pour créer, lire, mettre à jour et supprimer des enregistrements d'employé.

Avant de commencer, vous devez effectuer l'*une des opérations suivantes* pour vous assurer que votre projet MyDirectory est au point de départ correct :

- Suivez le «Module 1 : Conception de l'interface graphique du client dans l'éditeur visuel», à la page 3.

  ou
- Importez le projet MyDirectory avec le module 2 comme point de départ

  Conseil : Sauf si vous spécifiez un nom de projet différent au cours de l'importation, cette opération aura pour effet de remplacer le contenu existant de votre projet MyDirectory.

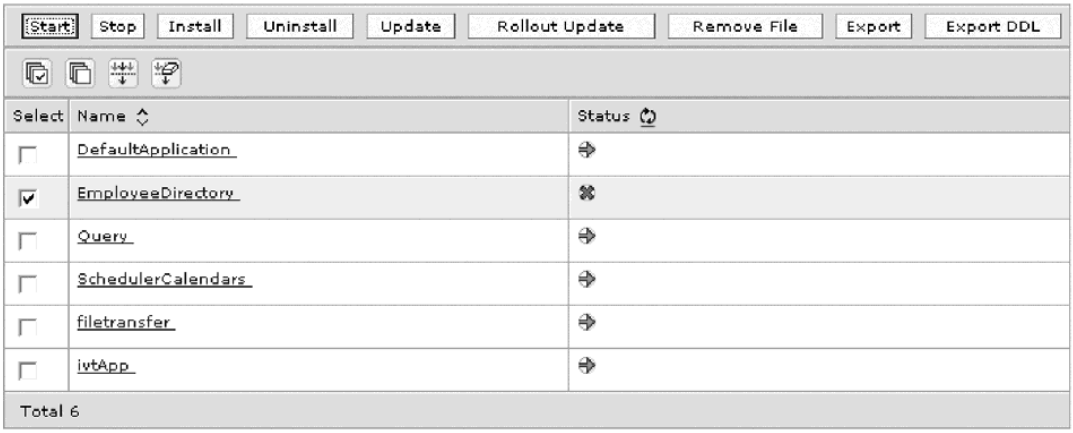
Votre projet Java MyDirectory inclut un fichier EmployeeDirectory.ear. Vous allez utiliser la console d'administration WebSphere pour installer l'application d'entreprise EmployeeDirectory contenue dans le

fichier EAR. Lors de l'installation de l'application, déployez également le service Web inclus dans l'application. L'application My Company Directory terminée utilise ce service Web déployé.

Procédez comme suit pour installer l'exemple d'application EmployeeDirectory et déployer le service Web dans votre environnement WebSphere Application Server v6.1 :

1. Démarrez une instance de votre serveur d'applications à partir du plan de travail. Il existe différentes manières de lancer votre serveur, mais la procédure ci-après décrit comment le faire à partir du plan de travail :
   a. Ouvrez la vue Serveurs. Pour ajouter la vue Serveurs à la perspective Java, sélectionnez **Fenêtre** → **Afficher la vue** → **Autre**, puis choisissez **Serveur** → **Serveurs**.
   b. La vue Serveurs répertorie les serveurs installés et configurés.
   c. Cliquez avec le bouton droit sur votre serveur et sélectionnez **Démarrer**. Vous savez que le serveur est lancé avec succès lorsque son statut affiché dans la vue Serveurs est **Démarré** ou lorsque le message `Serveur server1 ouvert pour e-business` apparaît dans la console. Vous pouvez maintenant exécuter la console d'administration.

   Remarque : Si la vue Serveurs ne contient aucune instance de serveur, créez un serveur :
   a. Cliquez avec le bouton droit dans la vue Serveurs et sélectionnez **Nouveau** → **Serveur**.
   b. Utilisez l'assistant Nouveau serveur pour ajouter WebSphere Application Server v6.1.
2. Exécutez la console d'administration WebSphere. Là encore, il existe différentes manières d'exécuter la console d'administration, mais les instructions ci-après décrivent comment le faire à partir du plan de travail :
   a. Dans la vue Serveurs, cliquez avez le bouton droit sur le serveur que vous venez de démarrer et sélectionnez **Exécuter la console d'administration**. La console d'administration WebSphere s'ouvre dans une fenêtre de navigateur.
   b. Entrez un ID utilisateur et cliquez sur **Connexion**. La page Bienvenue de la console d'administration s'ouvre. L'ID utilisateur que vous entrez n'est utilisé que pour le suivi des modifications apportées par chaque utilisateur aux données de configuration du serveur.
3. Utilisez la console d'administration pour installer l'application d'entreprise EmployeeDirectory.ear de votre projet MyDirectory. La console d'administration utilise un assistant pour vous aider à installer les applications ; vous cliquez sur **Suivant** pour passer d'une page à une autre jusqu'à ce que toutes les options soient définies. Pour installer l'exemple d'application d'entreprise contenant le service Web de ce tutoriel, procédez comme suit :
   a. Dans la partie gauche de la console d'administration, développez l'option de menu **Applications** et cliquez sur **Installation d'une nouvelle application**.
   b. Sélectionnez **Système de fichiers local** et entrez le chemin complet du fichier EmployeeDirectory.ear de votre projet MyDirectory dans la zone **Spécifiez un chemin**. Pour obtenir ce chemin, cliquez avec le bouton droit sur le fichier EmployeeDirectory.ear, dans la vue Packages, et sélectionnez **Propriétés**. Parmi les informations affichées sur la page Propriétés figure l'emplacement du fichier. Vous pouvez le copier et le coller dans la zone **Spécifiez un chemin**.
   c. Cliquez sur **Suivant** jusqu'à atteindre la page **Sélection des options d'installation**.
   d. Sélectionnez **Déploiement de services Web**.
   e. Cliquez sur **Suivant** jusqu'à atteindre la page **Récapitulatif**, puis cliquez sur **Terminer**.
   f. Cliquez sur le lien **Sauvegarde dans la configuration maîtresse** lorsque vous êtes invité à appliquer les modifications que vous avez apportées à votre configuration locale. Vérifiez vos modifications et cliquez sur le bouton **Sauvegarder**.
4. Démarrez l'application EmployeeDirectory à l'aide de la console d'administration :
   a. Cliquez sur **Applications** → **Applications d'entreprise**. L'application EmployeeDirectory est répertoriée comme application installée sur le serveur, mais son état est Arrêté.

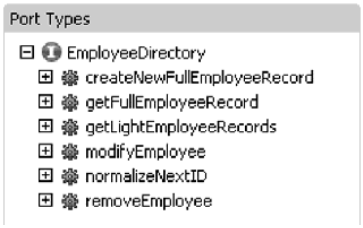


b. Cochez la case en regard d'EmployeeDirectory et cliquez sur **Démarrer**. Un message indique que l'application EmployeeDirectory est démarrée et l'icône d'état se transforme en flèche verte.

L'application EmployeeDirectory est désormais active sur localhost sur le port 9080, et le service Web est accessible. Une fois que vous avez terminé le présent tutoriel, vous pouvez retourner à la console d'administration, arrêter l'application EmployeeDirectory, puis la désinstaller.

Si vous ouvrez le fichier EmployeeDirectory.wsdl de votre projet MyDirectory (il doit s'ouvrir dans l'éditeur graphique WSDL par défaut), vous pouvez examiner le service Web que vous venez de déployer. Si le fichier WSDL ne s'ouvre pas dans l'éditeur WSDL, cela signifie peut-être que la capacité Développeur de services Web n'est pas activée dans le plan de travail. Vous pouvez spécifier les capacités à activer dans les préférences (**Fenêtre** → **Préférences** → **Plan de travail** → **Capacités**).

L'image suivante de l'éditeur WSDL indique les opérations disponibles dans le service EmployeeDirectory :



Vous pouvez utiliser l'éditeur WSDL pour examiner chaque opération et ses messages de demande et de réponse correspondants. Cela vous permettra de vous familiariser avec le service Web et de comprendre comment il est utilisé dans les exercices restants.

Leçon 2.2 : Liaison de la table des employés à la source de données du service Web

L'application My Company Directory affiche la liste de tous les enregistrements d'employés actuellement présents dans l'annuaire. Les enregistrements s'affichent dans une JTable (employeesTable) dont les colonnes peuvent servir de clés de tri et qui comprennent le nom, le prénom, l'adresse e-mail et l'ID de l'employé. Pour que la table employeesTable reçoive les enregistrements à afficher, vous devez la lier à un objet de données qui est renvoyé par la source de données de l'exemple de service Web.

Afficher une démonstration

Aperçu des objets de données, des sources de données et des lieurs (binders)

Afin de procurer à la table employeesTable un objet de données local avec lequel elle puisse fonctionner, vous allez ajouter une source de données à l'application en utilisant l'éditeur visuel. Cette source de

données aura pour rôle de se connecter au proxy du service Web et de découvrir les méthodes disponibles pour votre application. Vous choisirez ensuite la méthode de service getLightEmployeeRecord accessible à partir de la source de données. Enfin, vous établirez un lien entre la table employeesTable de votre application et les champs renvoyés dans l'objet de données de lignes (lightEmployeeRecordRows).

Vous pouvez rapidement et facilement créer tous ces objets de données et sources de données à l'aide des classes de lieur (binder) intégrées à l'éditeur visuel Java. L'éditeur visuel offre un ensemble d'interfaces et de classes génériques qui sont générées dans votre projet à mesure que vous liez les composants visuels aux fabriques de données. Les classes de lieur sont générées, par défaut, dans un package nommé jve.generated. L'éditeur visuel fournit les classes de lieur sous forme d'implémentation générique que vous pouvez personnaliser et étendre pour répondre aux besoins spécifiques de votre application. Le présent tutoriel illustre la puissance et la souplesse d'une utilisation simple des classes de lieur par défaut.

Important : Avant de commencer cet exercice, il est vivement recommandé de lire les rubriques d'aide suivantes. Ces rubriques vous permettent d'en savoir plus sur la fonctionnalité et la logique qui se cachent derrière les objets de données, les sources de données et les lieurs (binders) fournis par l'éditeur visuel Java :

- Généralités sur les lieurs de données
- Référence de l'API du lieur (Binder)

Pour ce tutoriel, vous allez utiliser une source de données de service Web, plusieurs types d'objet de données et plusieurs types de lieurs dans votre application. Lorsque vous ajouterez des instances de ces objets à votre application, l'éditeur visuel ajoutera les classes nécessaires dans le package jve.generated du projet, où vous pourriez étendre, remplacer et réécrire la logique de liaison des données. L'éditeur visuel Java offre une représentation graphique des objets de liaison en affichant, dans la zone à format libre de la vue de conception, les objets de données, les sources de données et les lieurs qui sont utilisés par votre application. Il trace des lignes entre d'une part les composants visuels et d'autre part les objets de données et les sources de données pour afficher les liaisons actuelles d'un objet sélectionné.

Le diagramme ci-après schématise la manière dont les composants visuels, les lieurs, les objets de données et les sources de données interagissent. L'application que vous créez dans le présent tutoriel illustre une utilisation légèrement plus complexe et créative des lieurs. Ce diagramme ne représente pas de manière exacte les lieurs, les objets de données et les sources de données de l'exemple d'application que vous créez.

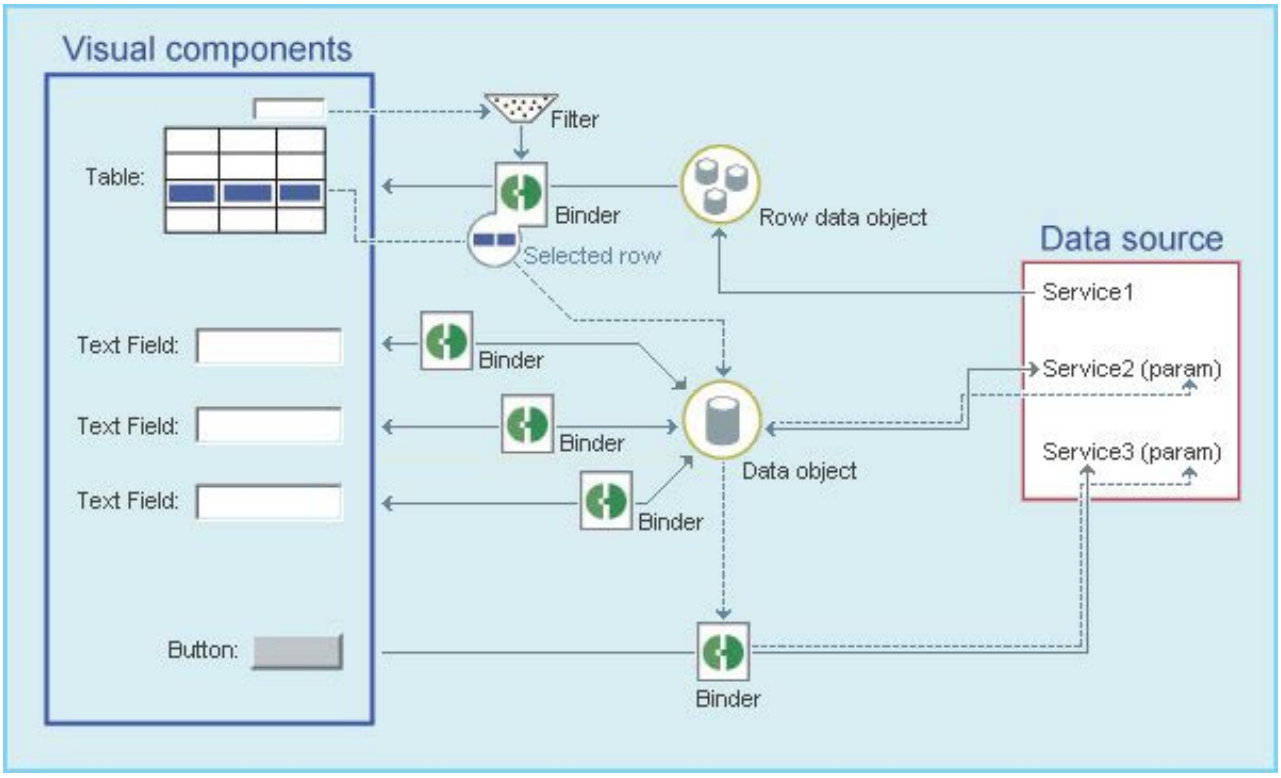


Figure 1. Ce diagramme illustre les relations entre composants visuels, lieurs (binders), objets de données et sources de données.

Dans la figure 1, chaque composant visuel possède son propre lieur qui l'associe à un objet de données ou, dans le cas du bouton, à une source de données. Les lieurs des zones de texte lient chaque zone à une propriété spécifique de l'objet de données. L'objet de données de ligne et l'objet de données de ce diagramme reçoivent leurs données d'appels directs à un service de la source de données. L'objet de données des zones de texte utilise une valeur de clé de la ligne sélectionnée dans la table comme argument pour appeler Service2, qui renvoie un enregistrement complet censé contenir davantage d'informations sur cette ligne. Cet enregistrement complet est utilisé à son tour comme argument du lieur d'action du bouton lorsqu'il appelle Service3, qui peut être une méthode mettant à jour les valeurs entrées dans les zones. Pour des explications plus détaillées sur les objets de données, les lieurs de données et les sources de données, cliquez sur les liens fournis précédemment.

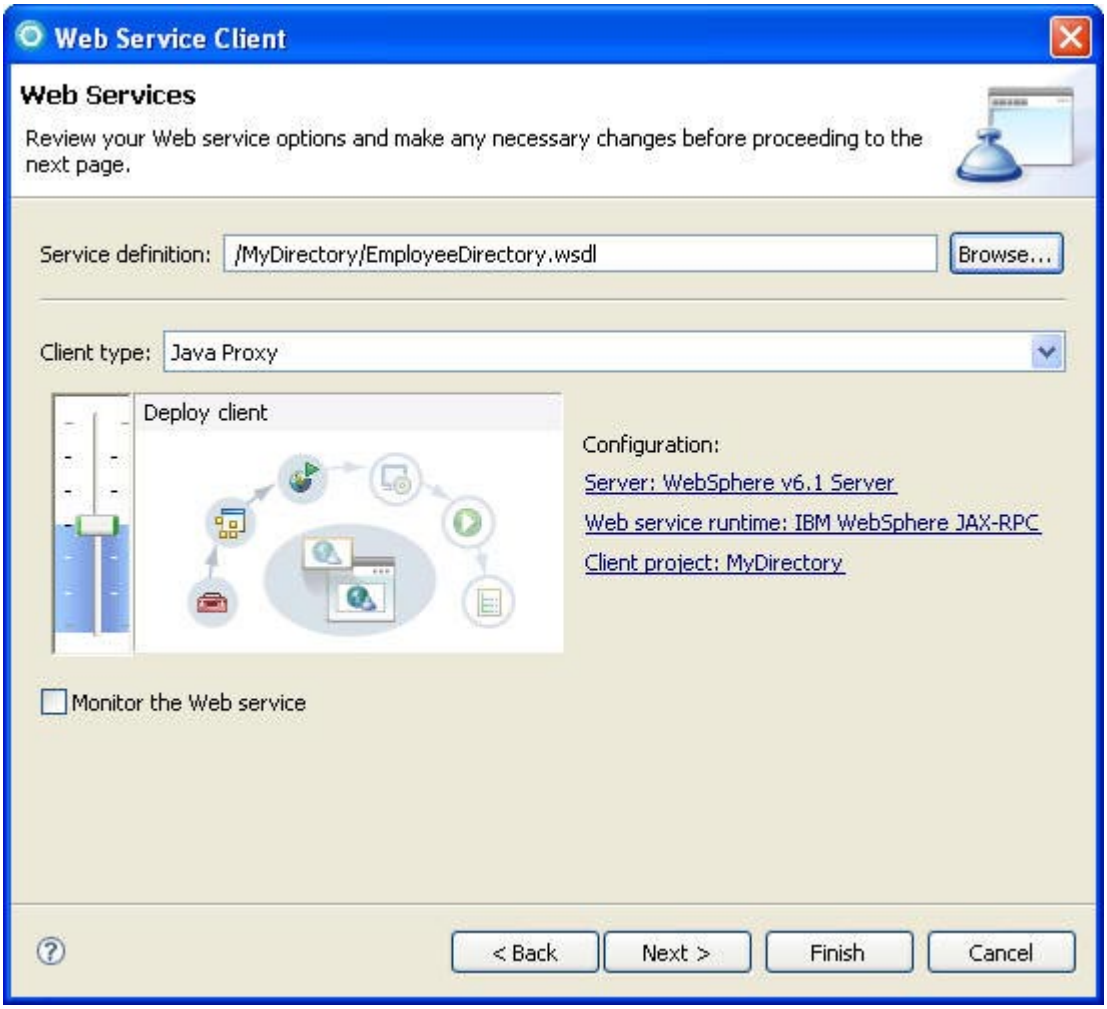
Générer un proxy Java de service Web dans votre projet à l'aide du fichier WSDL fourni

Pour pouvoir exploiter le service Web exécuté sur un serveur, votre application Java requiert un proxy (ou client) Java avec lequel elle puisse interagir. En utilisant un fichier WSDL, vous pouvez générer un proxy Java dans votre projet Java à l'aide de l'assistant Client du service Web. Votre projet MyDirectory inclut le fichier EmployeeDirectory.wsdl que vous utiliserez pour générer ce proxy. Une fois celui-ci généré, vous pourrez créer une source de données représentant le service Web et commencer à lier les composants visuels de l'application aux objets de données.

Important : Dans cet exercice, on suppose que vous avez déployé le service Web décrit par le fichier WSDL sur une installation locale de WebSphere Application Server et utilisé le port par défaut de l'hôte local (http://localhost:9080). Si vous avez déployé le fichier EAR de manière différente, vous devez éditer le fichier WSDL en conséquence avant de commencer.

Pour générer le proxy Java du service Web dans votre projet, procédez comme suit :

1. Sur la barre de menus principale, sélectionnez **Fichier** → **Nouveau** → **Autre**, puis choisissez **Services Web** → **Client du service Web**. Si la catégorie Services Web n'est pas proposée dans la liste, sélectionnez **Afficher tous les assistants**.
2. Utilisez l'assistant pour définir le client de service Web :
   a. Pour la **définition du service**, entrez le fichier WSDL qui se trouve dans votre projet MyDirectory : `/MyDirectory/EmployeeDirectory.wsdl`
   b. Dans la zone **Type de client**, sélectionnez **Proxy Java**.
   c. Réglez le curseur sur l'étape de déploiement du client.
   d. Vérifiez que l'environnement d'exécution du service Web est défini correctement compte tenu du serveur que vous exécutez. Ce tutoriel a été testé avec les serveurs WebSphere versions 6.0 et 6.1 et l'environnement d'exécution IBM WebSphere JAX-RPC.
   e. Assurez-vous que le client de type proxy Java aura comme dossier de sortie le projet MyDirectory.



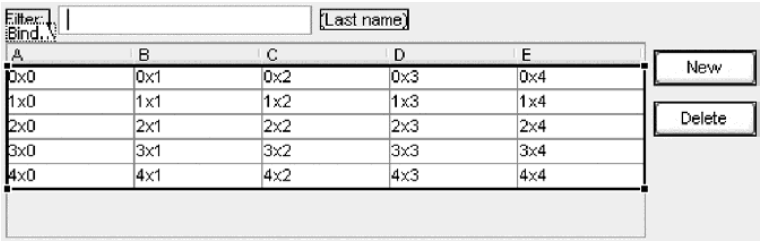
3. Cliquez sur **Terminer**. L'assistant Client du service Web génère le proxy Java dans un nouveau package (directory.service) qu'il crée dans votre projet.

Lier le composant employeesTable à un objet de données de lignes renvoyé par le service Web

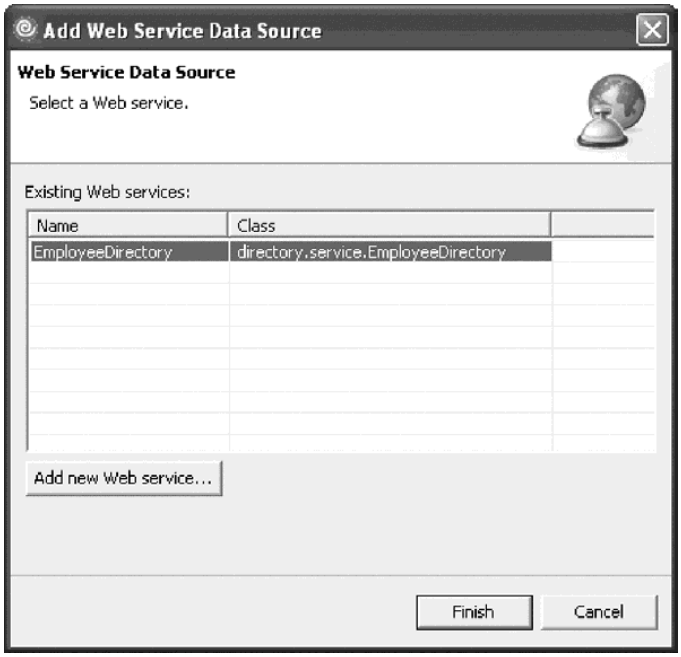
Le composant employeesTable étant le premier composant visuel que vous liez dans cette application, vous devez créer une source de données pointant sur le proxy de service Web que vous venez d'ajouter à votre projet. Vous la réutiliserez dans les exercices à venir, lorsque vous devrez lier les autres composants visuels de l'application. Dans la présente étape, vous allez ajouter la source de données du service Web et l'objet de données lightEmployeeRecordRows.

Pour lier la table des employés, procédez comme suit :

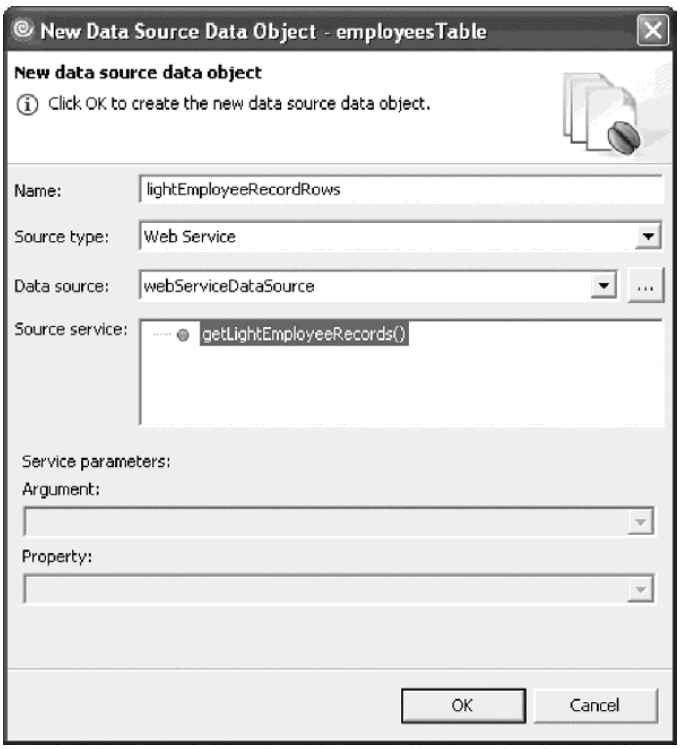
1. Sélectionnez le composant employeesTable dans la vue Beans Java ou dans la vue de conception. (Veillez à ne pas sélectionner son composant parent JScrollPane.) Un petit onglet intitulé **Lier** apparaît en haut du composant employeesTable, dans la zone de conception.



2. Cliquez sur l'onglet **Lier** du composant employeesTable. Vous pouvez également cliquer avec le bouton droit sur le composant employeesTable et sélectionner **Propriétés de liaison**.
3. Dans la mesure où l'application ne contient pas d'objet de données, vous devez en ajouter un. Cliquez sur **Nouvel objet de données de source de données**.
4. Dans la zone **Type de la source**, sélectionnez **Service Web**.
5. Dans la mesure où vous n'avez pas encore ajouté la source de données du service Web à votre application, vous devez l'ajouter maintenant. En regard de la zone **Source de données**, cliquez sur le bouton **Nouveau...** pour ouvrir la boîte de dialogue Ajout d'une source de données de service Web, qui répertorie les clients ou proxies de service Web présents dans votre projet.
6. Sélectionnez le service Web EmployeeDirectory et cliquez sur Terminer. Une nouvelle source de données est ajoutée au fichier DirectoryApp.java.

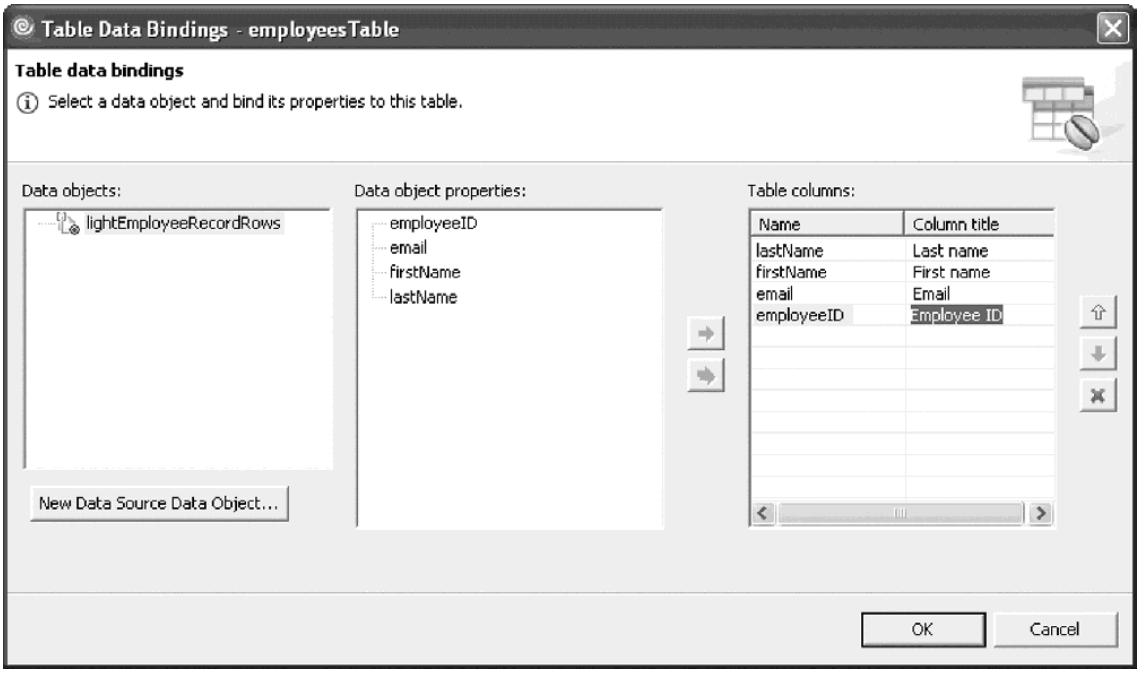


7. Dans la zone Service source de la boîte de dialogue Nouvel objet de données de source de données, sélectionnez getLightEmployeeRecords() et acceptez le nom par défaut du nouvel objet de données : lightEmployeeRecordRows. Aucun paramètre n'est requis pour cette méthode de service. Cliquez sur OK. Le nouvel objet de données est créé et affiché dans la zone à format libre de la vue de conception.



Conseil : Comme le composant que vous êtes en train de lier est une table, la boîte de dialogue Nouvel objet de données de source de données affiche uniquement les services qui renvoient des objets de données de lignes. Dans le cas présent, la méthode getLightEmployeeRecords() est le seul service disponible qui renvoie un tableau d'objets.

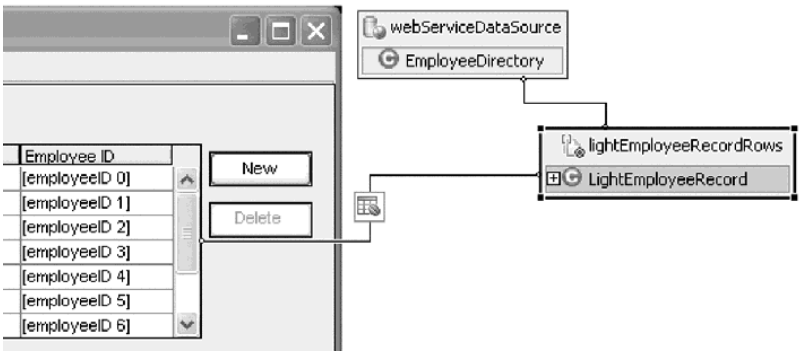
8. Dans la boîte de dialogue Liaisons de données de table, sélectionnez l'objet de données lightEmployeeRecordRows.
9. A présent, vous devez sélectionner les propriétés de l'objet de données lightEmployeeRecordRows que vous souhaitez afficher dans la table employeesTable :



a. Cliquez sur le bouton pour ajouter toutes les propriétés de l'objet à la liste **Colonnes de la table**.
b. Utilisez les boutons fléchés pour organiser les colonnes dans l'ordre suivant, de haut en bas : lastName, firstName, email, employeeID
c. Renommez les titres de colonne : `Last name`, `First name`, `Email`, `Employee ID`

   Conseil : Une fois la table liée, vous avez toujours la possibilité de retourner aux propriétés de liaison pour renommer et réorganiser les colonnes à votre guise.
d. Cliquez sur **OK**.

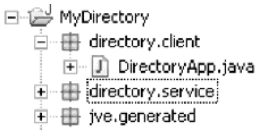
La table employeesTable est désormais liée à l'objet de données lightEmployeeRecordRows à l'aide d'un JRowTableBinder. Si vous cliquez sur l'objet de données lightEmployeeRecordRows dans la zone à format libre, l'éditeur visuel trace une ligne entre l'objet de données et la table. Sur cette ligne, le JRowTableBinder est représenté par l'icône de lieur de table . Une autre ligne indique que l'objet de données utilise webServiceDataSource comme source de données.



Récapitulatif de la leçon

Examinez les modifications apportées à votre projet et à l'application. Au cours de cette leçon, vous avez ajouté la source de données du service Web, un objet de données de lignes et un lieur (binder) établissant la liaison entre cet objet et le composant visuel employeesTable.

Examinez le nouveau package (jve.generated) qui a été créé dans le projet afin de contenir les classes de lieur générées par l'éditeur visuel Java. Notez également le nouveau package (directory.service) qui contient les classes du proxy Java du service Web.



A présent, lorsque vous exécutez l'application My Company Directory, la table des employés est alimentée par le service Web avec les enregistrements d'employé existants.

Leçon 2.3 : Liaison des zones de détails à la ligne sélectionnée dans la table

Dans l'exercice précédent, vous avez lié le composant visuel employeesTable à l'objet de données lightEmployeeRecordRows renvoyé par la méthode getLightEmployeeRecords() du service Web. Vous devez maintenant remplir les zones de détails de l'application en les liant aux données de l'employé sélectionné dans la table.

Pour obtenir les détails supplémentaires de chaque employé sélectionné, on utilise un autre objet de données. L'objet de données selectedEmployeeRecord que vous allez ajouter sera renvoyé par la méthode

de service getFullEmployeeRecord(). Cette méthode prend comme paramètre d'entrée l'ID de l'employé sélectionné dans la table et récupère les autres détails de cet employé, notamment son numéro de téléphone et son lieu de travail.

Le JRowTableBinder qui a été utilisé lorsque vous avez lié la table à l'objet de données de lignes simplifie cette étape. Le JRowTableBinder expose l'élément sélectionné dans la table comme objet de données distinct, pouvant être utilisé comme paramètre de la méthode getFullEmployeeRecord(java.lang.Integer). Vous pouvez dès lors lier aisément chacune des zones de texte à la propriété correspondante dans l'objet de données selectedEmployeeRecord.

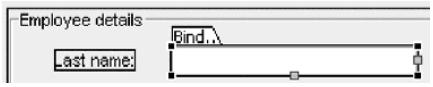
Informations complémentaires sur ce service Web : Le service Web utilisé dans ce tutoriel inclut deux méthodes chargées de récupérer les détails de chaque employé. La table répertorie tous les employés, mais elle n'affiche qu'un sous-ensemble de leurs données. Lorsque l'utilisateur sélectionne un employé dans la table, le reste des données concernant cet employé seul est récupéré pour alimenter les zones de détails. Si le service Web envoyait toutes les données de chaque employé à la table, cela augmenterait inutilement le trafic Web et nuirait aux performances de l'application.

Par exemple, si l'enregistrement d'employé comprenait une photo ou un fichier joint, vous ne souhaiteriez certainement pas récupérer toutes les photos lors de l'extraction de la liste complète des employés pour alimenter la table. La méthode de service getLightEmployeeRecord est donc utilisée pour remplir la table, tandis que la méthode getFullEmployeeRecord sert à extraire l'enregistrement complet de l'employé sélectionné dans la table.

Lier la zone Last name

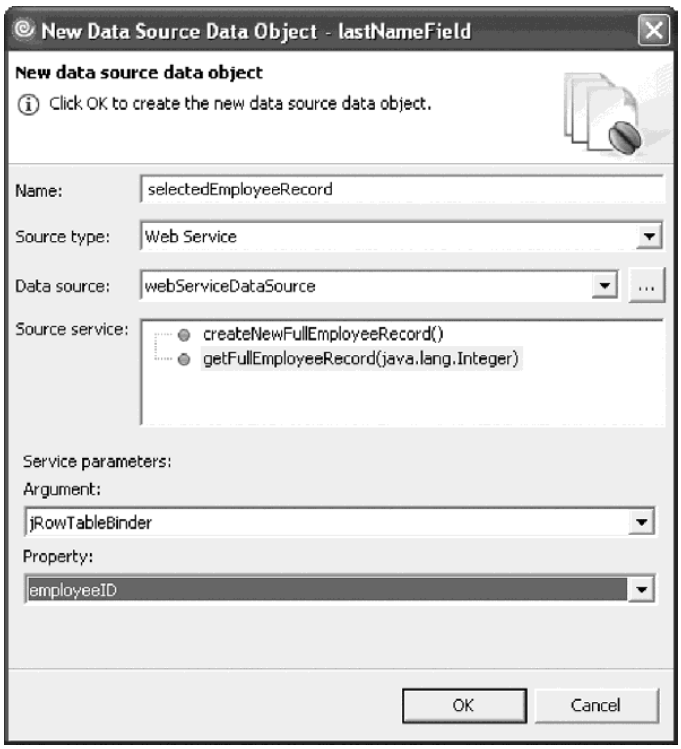
Procédez comme suit pour lier la zone **Last name** à la propriété lastName de l'objet de données selectedEmployeeRecord :

1. Dans la vue Beans Java ou dans la vue de conception, sélectionnez le composant JTextField lastNameField. Un onglet **Lier** apparaît en haut de la zone de texte, dans la zone de conception.

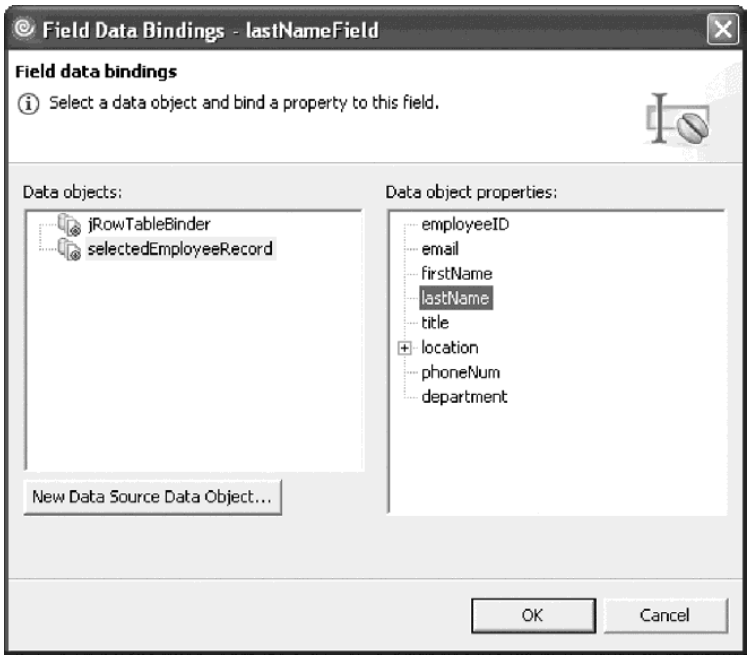
   

2. Cliquez sur l'onglet **Lier** pour ouvrir la boîte de dialogue Liaisons de données de zone.
3. Cliquez sur **Nouvel objet de données de source de données**. L'objet de données jRowTableBinder existant renvoie le nom d'employé correct, mais il n'inclut pas l'enregistrement d'employé complet. Vous devez donc créer un nouvel objet de données représentant l'enregistrement d'employé complet.
4. Vérifiez que **Service Web** est sélectionné dans la zone **Type de la source** et que **webServiceDataSource** est sélectionné dans la zone **Source de données**.
5. Dans la zone **Service source**, sélectionnez getFullEmployeeRecord(java.lang.Integer). La boîte de dialogue Nouvel objet de données de source de données répertorie les services qui renvoient des objets de données compatibles avec une zone de texte.
6. Dans la zone **Nom**, entrez `selectedEmployeeRecord`.
7. Dans la zone **Argument**, sélectionnez jRowTableBinder et dans la zone **Propriété**, sélectionnez employeeID. L'ID employé de la ligne sélectionnée est désormais défini comme argument de la méthode de service getFullEmployeeRecord().

   Remarque : La méthode getFullEmployeeRecord(java.lang.Integer) requiert un entier comme argument. Pour extraire un enregistrement complet, vous souhaitez utiliser l'ID employé de la ligne sélectionnée dans la table des employés. Lorsque vous avez lié la table, l'éditeur visuel a généré automatiquement l'objet jRowTableBinder. Or, cet objet possède une propriété, employeeID, dont la valeur identifie la sélection courante dans la table. Pour l'entier à passer en argument à la méthode, vous allez donc utiliser la propriété employeeID de jRowTableBinder.



8. Cliquez sur **OK**.
9. Dans la boîte de dialogue Liaisons de données de zone, vérifiez que l'objet selectedEmployeeRecord est sélectionné dans la liste **Objets de données**. Notez que les propriétés disponibles pour l'objet de données selectedEmployeeRecord sont plus nombreuses que celles de l'objet jRowTableBinder.
10. Dans la liste **Propriétés des objets de données**, sélectionnez la propriété lastName.



11. Cliquez sur **OK**. La zone Last name de votre application est maintenant liée à la propriété lastName de l'objet de données selectedEmployeeRecord, qui est renvoyé par getFullEmployeeRecord().

    Un nouvel objet de données nommé selectedEmployeeRecord est créé et ajouté à votre application. Une représentation visuelle de cet objet est ajoutée à la zone à format libre de la vue de conception, comme l'illustre l'image suivante :

A présent, lorsque vous sélectionnez la zone lastName dans la zone de conception, une ligne indique que cette zone est liée à l'objet selectedEmployeeRecord. Cette ligne comporte en son milieu une icône qui représente le lieur de texte SwingTextComponentBinder utilisé pour cette liaison. Si vous sélectionnez la ligne ou l'icône représentant le lieur dans la zone de conception, vous pouvez en consulter les propriétés dans la vue Propriétés.

Lier les zones de détails restantes

Pour lier les autres zones de détails de l'enregistrement d'employé, vous devez appliquer une procédure similaire à celle utilisée pour la zone Last name, mais vous n'avez pas besoin d'ajouter l'objet de données. Comme vous avez déjà ajouté l'objet de données selectedEmployeeRecord, vous n'avez plus qu'à lier chaque autre zone à la propriété correspondante de cet objet.

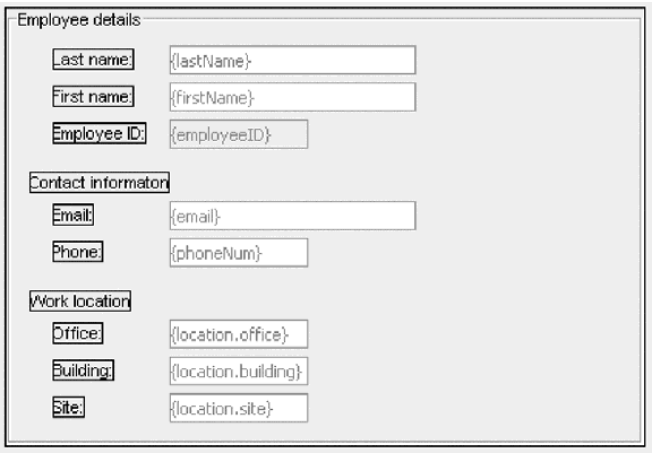
Procédez comme suit pour lier chaque autre zone de la section Employee details de l'application :

1. Sélectionnez la zone concernée dans la vue de conception, puis cliquez sur son onglet **Lier**.
2. Dans la boîte de dialogue Liaisons de données de zone, sélectionnez l'objet selectedEmployeeRecord dans la liste **Objets de données**.
3. Dans la liste **Propriétés des objets de données**, sélectionnez la propriété appropriée à la zone que vous liez. Le tableau suivant indique la propriété à laquelle chaque zone de texte doit être liée :

Zone	**Propriété de l'objet de données selectedEmployeeRecord**
lastNameField	lastName
firstNameField	firstName
idField	employeeID
emailField	email
phoneField	phoneNum
officeField	location.office
buildingField	location.building
siteField	location.site

4. Cliquez sur **OK**.

Lorsque vous avez fini de lier les zones de texte, la zone de conception doit ressembler à l'image suivante :



Configurer la zone Employee ID en lecture seule

La zone Employee ID est désactivée (grisée) car sa propriété 'editable' a la valeur false. Cependant, par défaut, le lieur de zone de texte attribue la valeur true à cette propriété lorsque l'objet de données contient une valeur. Vous pouvez empêcher ce comportement afin que la zone demeure en lecture seule.

Pour empêcher le lieur de changer automatiquement la propriété 'editable' :

1. Sélectionnez la zone Employee ID. Une ligne apparaît dans la zone de conception avec une icône qui représente le lieur de la zone.
2. Cliquez sur l'icône du lieur de la zone Employee ID.
3. Dans la vue Propriétés, affectez la valeur **false** à la propriété autoEditable. Appuyez sur **Entrée**.

Récapitulatif de la leçon

A présent, lorsque vous exécutez l'application et sélectionnez un employé dans la table, les détails de l'enregistrement de cet employé sont affichés dans les zones de détails.

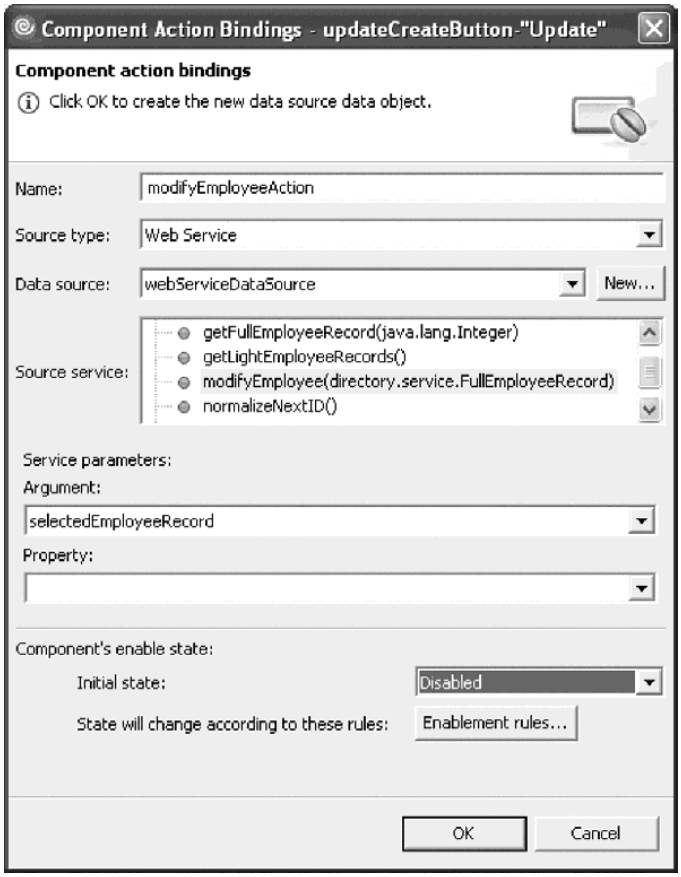
Leçon 2.4 : Liaison du bouton Update à un lieur d'action

L'éditeur visuel Java fournit des lieurs (binders) d'action qui permettent d'appeler un service sur une source de données lorsque l'utilisateur clique sur un bouton. Par exemple, lorsque le bouton Update est actionné, l'application doit exécuter la méthode modifyEmployee() du service Web en appliquant les modifications que l'utilisateur a entrées dans les zones de détails. Dans cette leçon, vous allez lier le bouton Update à un lieur d'action.

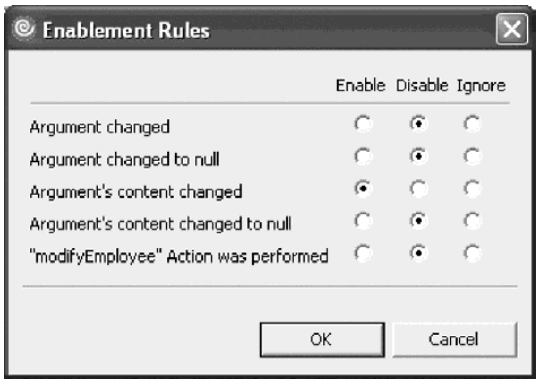
Pour lier le bouton Update :

1. Sélectionnez le bouton **Update** dans la zone de conception et cliquez sur son onglet **Lier** afin d'ouvrir la boîte de dialogue Liaisons d'action du composant.

2. Dans la zone **Type de la source**, sélectionnez **Service Web**.
3. Dans la zone **Source de données**, sélectionnez **webServiceDataSource**.
4. Dans la liste **Service source**, sélectionnez la méthode **modifyEmployee(directory.service.FullEmployeeRecord)**.
5. La zone **Nom** prend automatiquement la valeur **modifyEmployeeAction**. Acceptez cette valeur par défaut.
6. Dans la zone **Argument**, sélectionnez **selectedEmployeeRecord**.
7. Comme la méthode modifyEmployee() reçoit pour argument un enregistrement d'employé complet, vous devez laisser la zone **Propriété** vide.
8. Choisissez **Désactivé** pour l'**état initial** du bouton.



9. Pour définir les modalités de changement d'état du bouton, cliquez sur **Règles d'activation**. Spécifiez que le bouton est activé uniquement lorsque le contenu de l'argument change, et désactivé dans tous les autres cas. Cliquez sur **OK**.



Cela signifie que le bouton **Update** est désactivé (grisé) jusqu'à ce que le contenu de l'objet selectedEmployeeRecord soit modifié. En d'autres termes, dès qu'une nouvelle valeur est entrée dans l'une des zones de détails, qui sont liées à selectedEmployeeRecord, le lieur active le bouton. Si l'utilisateur sélectionne un nouvel enregistrement dans la table ou clique sur le bouton **Update**, celui-ci est à nouveau désactivé.

10. Cliquez sur **OK**.

Un nouveau lieur SwingDataServiceAction est ajouté pour le bouton **Update**. Si vous sélectionnez ce bouton dans la zone de conception, l'éditeur visuel trace une ligne indiquant que le bouton est lié à la source de données du service Web. Une flèche en pointillés de couleur rose part de l'objet selectedEmployeeRecord vers la ligne. Cette flèche indique que selectedEmployeeRecord est l'argument passé dans l'appel à la méthode de service.

Récapitulatif de la leçon

A présent, lorsque vous exécutez l'application, vous pouvez mettre à jour l'enregistrement d'un employé.

Sélectionnez un employé dans la table et changez son nom. Dès que vous commencez à éditer le contenu de la zone, le bouton **Update** est activé. Lorsque vous cliquez sur **Update**, la méthode de service modifyEmployee est appelée et l'enregistrement de l'employé est mis à jour. Le nouveau nom est reflété dans la table des employés.

Leçon 2.5 : Activation du bouton Delete et de la boîte de dialogue Confirm Delete

Dans cet exercice, vous allez programmer l'application My Company Directory afin qu'elle permette à l'utilisateur de supprimer un enregistrement d'employé.

La liste suivante décrit le comportement que vous souhaitez donner à l'application :

- Lorsqu'un employé est sélectionné dans la table, le bouton **Delete** est activé.
- Lorsque vous cliquez sur le bouton **Delete**, la boîte de dialogue Confirm Delete s'affiche et vous invite à confirmer la suppression.
- Si vous cliquez sur le bouton **Yes**, l'enregistrement d'employé est supprimé, la boîte de dialogue Confirm Delete se ferme et la liste des employés est actualisée.
- Si vous cliquez sur **No**, la suppression est annulée et la boîte de dialogue Confirm Delete se ferme.

Programmer le bouton Delete de sorte qu'il soit activé ou désactivé selon qu'une ligne est sélectionnée ou non dans la table

Pour programmer l'état d'activation du bouton Delete, vous allez ajouter à la table un écouteur (listener) qui activera ce bouton lorsqu'une ligne sera sélectionnée.

1. Sélectionnez employeesTable dans la vue Beans Java. La ligne suivante est mise en évidence dans la vue du code source :

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

2. Juste après cette ligne, ajoutez un nouvel écouteur ListSelectionListener et un événement valueChanged au composant employeesTable :

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

3. Une fois ces lignes de code ajoutées, l'éditeur de code source les signale comme étant des erreurs tant que ListSelectListener et ListSelectionEvent ne sont pas importés. Pour ajouter les importations requises, sélectionnez **Source → Organiser les importations** sur la barre de menus principale. Les lignes suivantes sont ajoutées à la section des importations de la classe :

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

Maintenant, lorsqu'une ligne est sélectionnée dans la table, le bouton **Delete** est activé.

Programmer l'ouverture de la boîte de dialogue Confirm Delete lorsque le bouton Delete est actionné

Vous allez ajouter un événement actionPerformed au bouton Delete et programmer cet événement pour qu'il ouvre la boîte de dialogue Confirm Delete.

1. Cliquez avec le bouton droit sur le bouton **Delete** et sélectionnez **Evénements → actionPerformed**. La souche d'événement suivante est ajoutée à la méthode 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()
   }
});
```

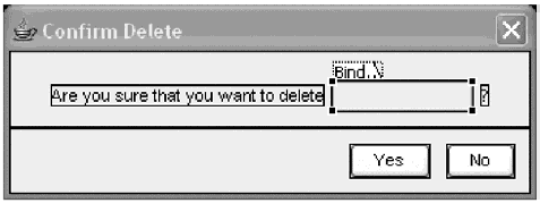
2. Remplacez cette souche générée par le code suivant, dont le but est d'afficher la boîte de dialogue Confirm Delete lorsque l'utilisateur clique sur le bouton :

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

Lier la zone de texte dans la boîte de dialogue Confirm Delete

Vous allez maintenant lier la zone de texte de la boîte de dialogue Confirm Delete afin qu'elle affiche le prénom de l'employé à supprimer.

1. Dans la vue Beans Java ou dans la zone de conception, sélectionnez la zone de texte employeeToDeleteField et cliquez sur son onglet **Lier**.



2. Dans la boîte de dialogue Liaisons de données de zone, sélectionnez l'objet de données selectedEmployeeRecord et la propriété firstName, puis cliquez sur **OK**.

   La zone de texte est désormais liée à la colonne firstName de la ligne sélectionnée dans la table employeesTable.

3. Pour que cette zone soit en lecture seule, affectez la valeur **false** à la propriété **autoEditable** du lieur de la zone.

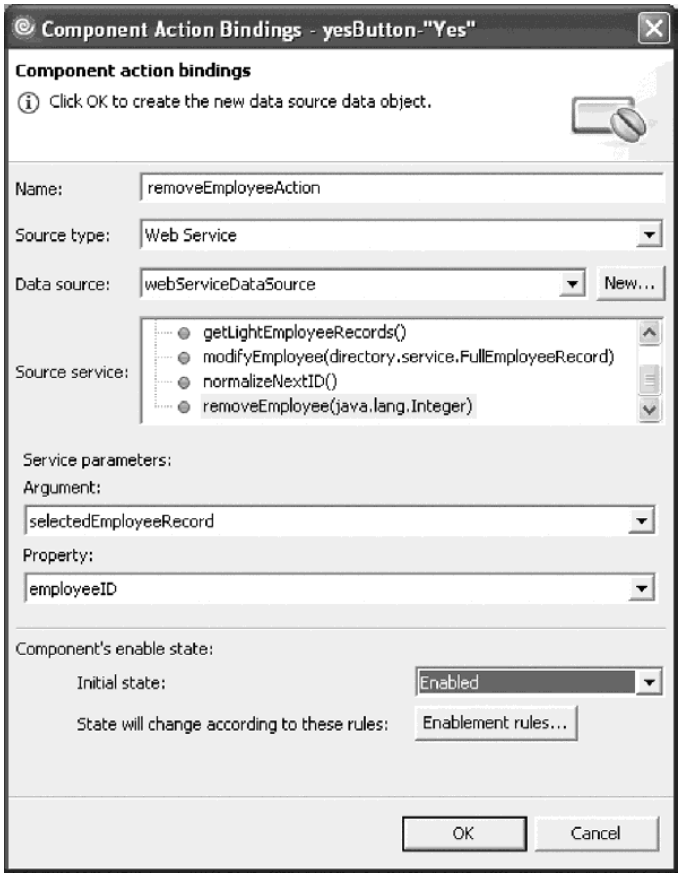
Lier le bouton Yes pour l'exécution de la suppression

Vous allez lier le bouton **Yes** afin qu'il appelle la méthode removeEmployee(java.lang.Integer) du service Web.

1. Sélectionnez le bouton **Yes** et cliquez sur son onglet **Lier** afin d'ouvrir la boîte de dialogue Liaisons d'action du composant.
2. Dans la zone **Type de la source**, sélectionnez **Service Web**.
3. Dans la zone **Source de données**, sélectionnez **webServiceDataSource**.
4. Dans la liste **Service source**, sélectionnez **removeEmployee(java.lang.Integer)**.
5. La zone **Nom** prend automatiquement la valeur **removeEmployeeAction**. Acceptez cette valeur par défaut.
6. Dans la zone **Argument**, sélectionnez **selectedEmployeeRecord**.
7. Dans la zone **Propriété**, sélectionnez **employeeID**. Comme la méthode removeEmployee() prend un entier comme argument d'entrée, vous utilisez l'ID d'employé (employeeID) contenu dans l'objet selectedEmployeeRecord.

8. Affectez la valeur **Activé** à l'**état initial** du bouton.
9. En ce qui concerne les **règles d'activation**, sélectionnez **Ignorer** pour chacune des conditions.

   Cet état de composant signifie que le bouton Yes sera toujours activé, car son état n'a pas besoin de changer.



10. Cliquez sur **OK**.

Ajouter un événement pour masquer la boîte de dialogue Confirm Delete une fois l'employé supprimé

Dans cette étape, vous allez ajouter un événement au *lieur* du bouton **Yes** (et non au bouton **Yes** lui-même). Vous souhaitez que la boîte de dialogue Confirm Delete se ferme après le retrait de l'employé, c'est à dire une fois que le lieur a appelé le service sur la source de données.

Ajoutez le code suivant à la méthode 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) {}
});
```

Ce code d'événement masque (ferme) la boîte de dialogue Confirm Delete une fois l'action du lieur effectuée.

Récapitulatif de la leçon

A présent, lorsque vous exécutez l'application My Company Directory, vous pouvez sélectionner un employé dans la table, cliquer sur le bouton **Delete**, puis sur le bouton **Yes** pour confirmer la suppression. L'enregistrement d'employé sera supprimé de l'annuaire et la liste des employés reflétera cette suppression.

Leçon 2.6 : Configuration des actions et des liaisons pour l'ajout d'un nouvel employé

Dans cette leçon, vous allez compléter l'application My Company Directory afin de permettre à l'utilisateur d'ajouter un nouvel enregistrement d'employé.

Comme l'ajout d'un nouvel employé exige de l'application un comportement plus compliqué et plus dynamique, cet exercice est plus complexe et nécessite que vous apportiez quelques modifications manuelles au code source. Il illustre également certaines fonctions avancées des objets de données et donne un exemple créatif des différentes techniques d'utilisation des lieurs (binders) et objets de données pour répondre à des besoins spécifiques.

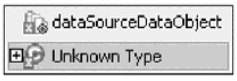
La liste suivante décrit le comportement requis de l'application :

- Lorsque vous cliquez sur le bouton **New**, l'application se comporte de la manière suivante :
  - La sélection est effacée de la table des employés et la table est désactivée.
  - L'effacement de la sélection entraîne la désactivation du bouton **Delete**.
  - La zone **Filter** est désactivée.
  - Toutes les valeurs des zones de détails sont effacées, excepté celle de l'ID d'employé, qui reçoit une nouvelle valeur.
  - Le texte du bouton **Update** est remplacé par **Add**.
- Lorsque vous cliquez sur le bouton **Add**, l'application se comporte de la manière suivante :
  - Les valeurs entrées dans les zones de détails sont ajoutées à l'annuaire pour former un nouvel enregistrement d'employé.
  - La table est activée et les valeurs qu'elle contient sont actualisées.
  - La zone **Filter** est activée.
  - Le texte du bouton **Add** redevient **Update**.

Ajouter un nouvel objet de données de source de données appelant la méthode createNewFullEmployeeRecord()

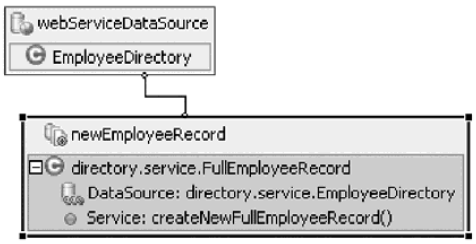
L'exemple de service Web comporte une méthode createNewFullEmployeeRecord dont le rôle est de fournir un nouvel enregistrement d'employé vide, recevant le premier ID d'employé disponible. L'enregistrement vide peut alors être alimenté par les données d'un nouvel employé et renvoyé au service Web.

1. Dans la palette de l'éditeur visuel Java, ouvrez le tiroir Objets de données et sélectionnez **Objet de données de la source de données**.
2. Amenez le pointeur de la souris sur une partie vide de la vue de conception (ou zone à format libre) et cliquez avec le bouton gauche pour déposer l'objet de données. Un nouvel objet de données de source de données est ajouté et affiché dans la zone à format libre :



3. Cliquez avec le bouton droit sur l'objet de données et sélectionnez **Renommer la zone**. Renommez l'objet de données en `newEmployeeRecord`.
4. Cliquez avec le bouton droit sur l'objet de données newEmployeeRecord et sélectionnez **Propriétés de liaison**. La boîte de dialogue Liaison de données s'affiche.
5. Dans la zone **Source de données**, sélectionnez webServiceDataSource.
6. Dans la zone **Service**, sélectionnez la méthode createNewFullEmployeeRecord().
7. Cliquez sur **OK**.

Dans la zone à format libre, vous pouvez voir que l'objet de données newEmployeeRecord est maintenant lié au service Web.

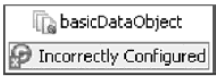


Ajouter un objet de données de base pour faciliter la commutation des objets de données

Comme les zones de détails et le bouton Update doivent pouvoir alterner entre deux modes (un pour la mise à jour d'un enregistrement d'employé existant et un autre pour l'ajout d'un nouvel enregistement), ils doivent être liés à deux objets de données différents, correspondant chacun à l'un de ces deux modes. Pour faciliter cette étape, vous allez ajouter un objet de données de base appelé switchingDataObject. Cet objet permettra de commuter la liaison des zones de texte entre les objets selectedEmployeeRecord et newEmployeeRecord.

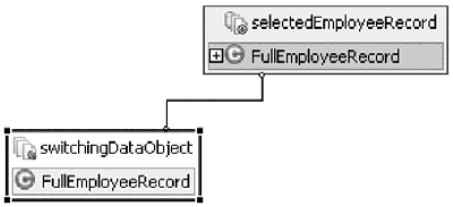
Le nouvel objet de données de base pointe simplement sur un autre objet de données (selectedEmployeeRecord) que vous avez défini au cours d'un exercice précédent. Il deviendra utile lorsque vous créerez une méthode lui indiquant d'utiliser l'objet newEmployeeRecord que vous avez créé précédemment. En d'autres termes, il servira d'objet intermédiaire alternant entre les objets de données selectedEmployeeRecord et newEmployeeRecord. Ce mécanisme permettra aux composants visuels de l'application de fonctionner avec deux objets de données différents.

1. Dans la palette de l'éditeur visuel, sélectionnez **Objet de données de base** et déposez le nouvel objet sur une partie vide de la zone à format libre. Un objet basicDataObject est ajouté.

2. Renommez l'objet de données en `switchingDataObject`
3. Dans la vue Propriétés de l'objet switchingDataObject, affectez la valeur **selectedEmployeeRecord** à la propriété **sourceObject**. Vous pouvez sélectionner selectedEmployeeRecord dans le menu déroulant de la colonne Valeur de la propriété.

   A présent, switchingDataObject fait référence à selectedEmployeeRecord et reflète les mêmes valeurs :

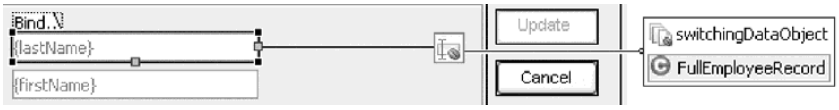
Redéfinir la liaison de chaque zone de l'enregistrement d'employé à l'objet switchingDataObject

Les zones de détails de l'employé sont toutes déjà liées à selectedEmployeeRecord, mais vous devez maintenant les lier à switchingDataObject. Une fois ces liens établis, vous pourrez alterner dynamiquement entre les deux objets de données, selon que vous modifiez un enregistrement d'employé existant ou que vous ajoutez un nouvel enregistrement d'employé.

Pour chacune des zones de la section Employee details, effectuez les étapes suivantes :

1. Sélectionnez la zone et cliquez sur son onglet **Lier**.
2. Dans la boîte de dialogue Liaisons de données de zone, sélectionnez l'objet switchingDataObject. Vous avez précédemment lié les zones à l'objet selectedEmployeeRecord.

3. Assurez-vous que la zone est toujours associée à la propriété correcte de l'objet de données, puis cliquez sur **OK**. Si vous sélectionnez la zone dans la vue de conception, vous pouvez voir que les lignes de liaison pointent à présent sur l'objet switchingDataObject.



Définir un fanion et une méthode pour la mise à jour et la commutation des modes

La méthode updateMode() présentée ci-après vérifie si le fanion isNewMode est true ou false, puis elle change en conséquence le comportement de l'application. Par défaut, le booléen isNewMode a la valeur false, ce qui signifie que le mode "mise à jour d'enregistrement" est en vigueur. Dans ces conditions, la méthode updateMode() active la table des employés et la zone Filter et affecte le texte "Update" au bouton updateCreateButton. Si le fanion isNewMode prend la valeur true, la table des employés est désactivée et ne comporte plus de sélection, la zone Filter est également désactivée et le texte du bouton devient "Add".

Ajoutez le code suivant à votre classe DirectoryApp.java, juste avant la dernière accolade fermante :

```
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");
   }
}
```

Ajouter un événement actionPerformed au bouton New

Dans cette étape, vous allez ajouter un code d'événement qui sera utilisé lorsque le bouton **New** sera actionné. Cet événement indique à switchingDataObject d'utiliser l'objet de données newEmployeeRecord, affecte la valeur "true" au fanion isNewMode et exécute la méthode updateMode() que vous avez ajoutée à l'étape précédente.

1. Dans la vue de conception, cliquez avec le bouton droit sur le bouton **New** et sélectionnez **Evénements** → **actionPerformed**. Le code suivant est généré dans la méthode getNewButton() :

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

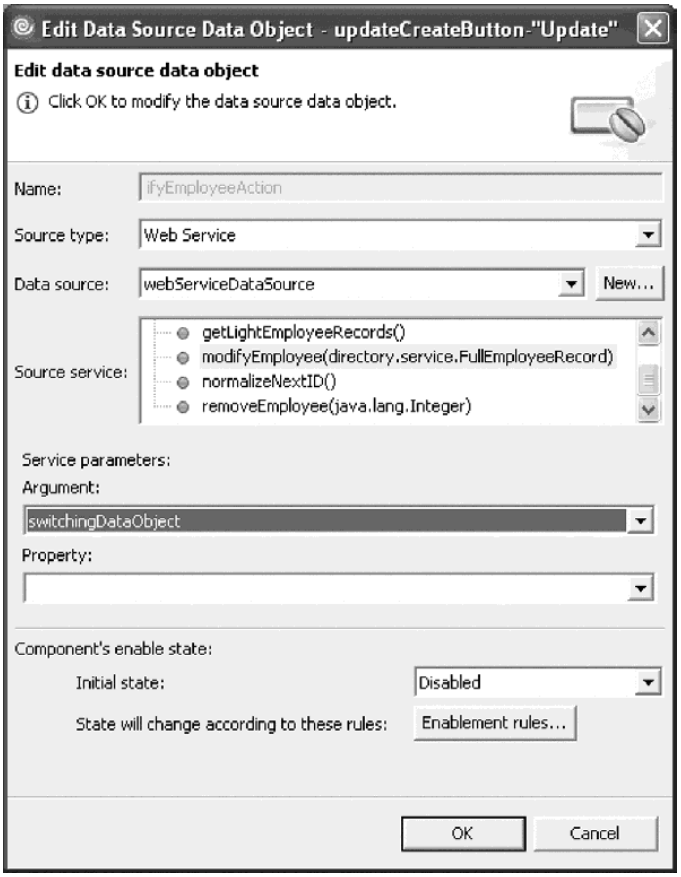
2. Remplacez cette souche générée par le code suivant :

```
newButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent e) {
      getSwitchingDataObject().setSourceObject(getNewEmployeeRecord());
      getNewEmployeeRecord().refresh();
      isNewMode = true; // place l'application en mode 'nouvel enregistrement'
      updateMode(); // change le comportement de l'interface graphique conformément au mode
'nouvel enregistrement' getLastNameField().grabFocus();
   }
});
```

Redéfinir la liaison du bouton Update

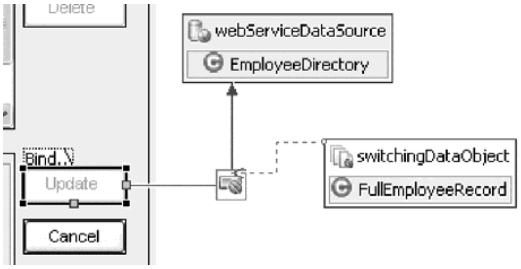
Dans une précédente leçon, vous avez programmé le bouton **Update** afin qu'il utilise la méthode modifyEmployee du service Web. Cette action est implémentée sous forme de SwingDataServiceAction. Une des propriétés du SwingDataServiceAction est l'objet source, qui sert d'argument passé à la méthode du service. Actuellement, l'objet source de l'action de modification (modifyEmployeeAction) est selectedEmployeeRecord. Pour programmer le bouton afin qu'il contrôle alternativement une mise à jour et un ajout d'enregistrement, vous allez reconfigurer son action de sorte qu'elle utilise l'objet switchingDataObject comme argument passé à la méthode de service modifyEmployee.

1. Dans la vue de conception, sélectionnez le bouton **Update**. Notez la flèche en pointillés de couleur rose. Elle indique que selectedEmployeeRecord est l'argument passé dans l'appel à la méthode de service.
2. Cliquez sur l'onglet **Lier** du bouton **Update**.
3. Dans la zone **Argument**, sélectionnez switchingDataObject.



4. Cliquez sur **OK**.

   Notez que l'action du bouton est maintenant configurée pour utiliser switchingDataObject comme argument à passer à la méthode modifyEmployee :



Ajouter un événement au lieu du bouton Update pour réinitialiser le mode

Une fois que l'utilisateur a cliqué sur le bouton **Update** et que l'action correspondante a été effectuée par le service Web, l'application doit retrouver son comportement et son mode par défaut. Pour cela, vous allez ajouter un écouteur d'événements au lieu d'action du bouton, qui changera le mode et actualisera la table une fois la mise à jour ou l'ajout d'enregistrement effectué.

Ajoutez le code suivant à la méthode getModifyEmployeeAction() pour le bouton Update :

```
modifyEmployeeAction.addActionBinderListener(new jve.generated.IActionBinder.ActionBinderListener() {
    public void afterActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {
      if (isNewMode) {
        //Rétablir l'utilisation de selectedEmployeeRecord
        getSwitchingDataObject().setSourceObject(getSelectedEmployeeRecord());
        //Sortir du mode 'nouvel enregistrement'
        isNewMode = false;
        updateMode();
      }
      //Actualiser l'objet de données de la table
```

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

Récapitulatif de la leçon

A présent, lorsque vous exécutez l'application My Company Directory, vous pouvez cliquer sur le bouton **New** pour ajouter un nouvel enregistrement d'employé.

Leçon 2.7 : Programmation du comportement du bouton Cancel

Lors de l'utilisation de l'application, vous voulez avoir la possibilité d'annuler facilement les modifications que vous commencez à effectuer dans un enregistrement d'employé si vous décidez de ne pas soumettre ces modifications. En d'autres termes, vous devez être en mesure d'annuler et d'effacer les zones afin de pouvoir recommencer dès le début. Pour ajouter cette fonctionnalité, vous devez définir certains événements actionPerformed sur le bouton **Cancel**.

La liste suivante décrit le comportement requis du bouton **Cancel** :

- Si l'utilisateur clique sur le bouton **Cancel** alors que le mode ″nouvel enregistrement″ est actif, l'application sort de ce mode.
- Si l'utilisateur clique sur le bouton **Cancel** alors qu'il est en train de modifier un enregistrement d'employé existant, toutes les modifications sont annulées et les valeurs initiales sont rétablies dans les zones de détails.

Pour ajouter un événement actionPerformed au bouton **Cancel** afin d'obtenir le comportement requis, procédez comme suit :

1. Dans la vue de conception, cliquez avec le bouton droit sur le bouton **Cancel** et sélectionnez **Evénements → actionPerformed**. Le code suivant est généré dans la méthode getCancelButton() :

   ```
   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. Remplacez la souche d'événement générée par le code suivant :

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

Récapitulatif de la leçon

Dans cette leçon, vous avez appris à programmer le bouton **Cancel** pour lui associer des événements actionPerformed.

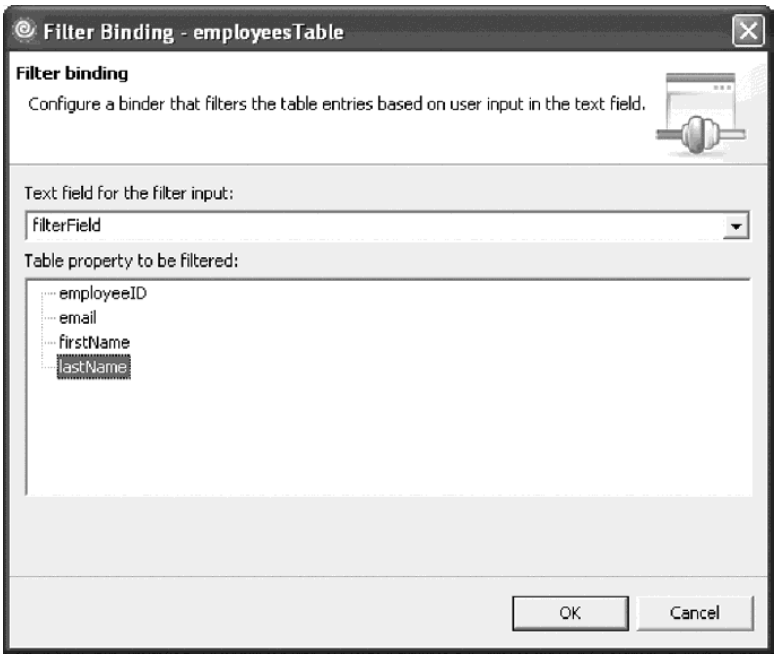
Leçon 2.8 : Configuration d'un filtre sur la table des employés

Vous pouvez utiliser un lieur avec filtrage de texte pour filtrer le contenu de la table des employés. Le filtre utilise les données entrées dans une zone de texte, les confronte à une propriété (ou colonne) particulière de la table et filtre le contenu de cette dernière en conséquence.

Dans l'application, vous utiliserez les caractères entrés dans la zone **Filter** pour effectuer un filtrage par nom d'employé. Si la chaîne exacte entrée dans la zone **Filter** est présente dans le nom d'un employé, l'enregistrement correspondant sera affiché dans la table.

Pour créer un filtre sur la table, procédez comme suit :

1. Cliquez sur le lieur du composant employeesTable et sélectionnez **Filtrer les propriétés de liaison**. La boîte de dialogue Liaison du filtre s'affiche.
2. Dans la liste **Zone de texte pour la saisie du filtre**, sélectionnez **filterField**.
3. Dans la liste **Propriété de table à filtrer**, sélectionnez **lastName**.



4. Cliquez sur **OK**.

Un nouveau filtre SwingPropertyFilter est généré. La propriété 'filter' du lieur de la table reçoit pour valeur le nouveau filtre. Ce dernier est quant à lui configuré pour utiliser la zone **Filter** en entrée et filtrer le contenu de la table en fonction de sa propriété lastName.

Récapitulatif de la leçon

Dans cette leçon, vous avez appris à configurer un filtre sur une table.

A présent, lorsque vous exécutez l'application My Company Directory, vous pouvez entrer des caractères dans la zone **Filter**. Le contenu de la table est instantanément filtré pour n'afficher que les lignes dont la colonne Last name contient les caractères entrés (peu importe où ils se situent dans le nom).

Félicitations ! L'application My Company Directory est terminée.

Récapitulatif : Génération d'un client Java enrichi qui utilise un service Web

Félicitations ! Vous avez appris à utiliser l'éditeur visuel Java pour construire et générer l'application My Company Directory, client Java enrichi qui se connecte à un exemple de service Web pour gérer un annuaire d'employés.

Voir une image du produit fini :

Leçons étudiées

A l'aide de l'éditeur visuel, vous avez complété l'interface graphique en utilisant le gestionnaire d'agencement GridBagLayout pour organiser la table des employés. Vous avez ensuite lié la table, les zones et les boutons à une source de données et aux objets de données appropriés pour que l'application puisse fonctionner avec le service Web par l'intermédiaire d'un proxy Java que vous avez généré. Vous avez également codé quelques parties complexes pour donner à l'application le comportement souhaité et la rendre à la fois simple d'emploi et intuitive. Parallèlement, vous avez appris à installer une application d'entreprise sur WebSphere Application Server version 6.0 et à déployer un service Web.

Plus important encore, vous connaissez maintenant tout de la puissante classe de lieur (binder) fournie par l'éditeur visuel Java et qui permet de manipuler des données. Vous êtes désormais prêt à faire vos propres expériences et à trouver de nouvelles applications aux lieurs.

Vous devriez maintenant être en mesure d'accomplir les tâches suivantes :

- Disposer des composants dans un GridBagLayout à l'aide de l'éditeur visuel Java.
- Exécuter une classe visuelle en tant que bean Java.
- Lier les composants visuels d'une application Java aux méthodes exposées par un service Web et aux objets de données qu'il renvoie.
- Ajouter des événements à des composants visuels.

Ressources supplémentaires

Importer une version finie de l'application My Directory

Ce projet inclut l'application terminée, le package jve.generated avec les classes de lieur et le client Java de service Web configuré pour WebSphere Application Server v6.1. Si vous importez le projet terminé sans suivre les leçons du tutoriel, il se peut que vous deviez configurer votre variable de chemin de génération Java. Elle doit en effet pointer sur le fichier JAR du client léger de services Web WebSphere v6.1.

Conseil : Sauf si vous spécifiez un nom de projet différent au cours de l'importation, cette opération aura pour effet de remplacer le contenu existant de votre projet MyDirectory.