构建使用 Web Service 的富 Java 客户机

目录

构建使用 Web Service 的富 Java 客户机

本教程教您如何使用 Java 可视编辑器来构建富 Java 客户机以连接至 Web Service。您在该教程中构建的客户机称为“我的公司目录”。

“我的公司目录”是一个用来维护公司职员目录的 Java 应用程序。该应用程序连接至样本 Web Service，而该样本 Web Service 提供了创建、检索、更新和删除职员记录的方法。

该客户机是在 Java 可视编辑器中使用 Swing 组件以可视方式构建的。Java 可视编辑器提供了一组 helper 类（数据源、数据对象和绑定程序）以用于连接至 Web Service 以及使用 Web Service。Web Service 以本地方式部署在您安装的 IBM WebSphere Application Server V6.0 上，并且这些工具会根据 Web 服务描述语言（WSDL）文件帮助您为客户机生成 Java 代理。

查看完成的产品

学习目标

在本教程中，您将学习下列课程:

- 如何使用 Java 可视编辑器来设计用户界面并对其进行布局
- 如何将界面元素绑定至数据对象和 Web Service

所需时间

2 小时 15 分钟

相关信息

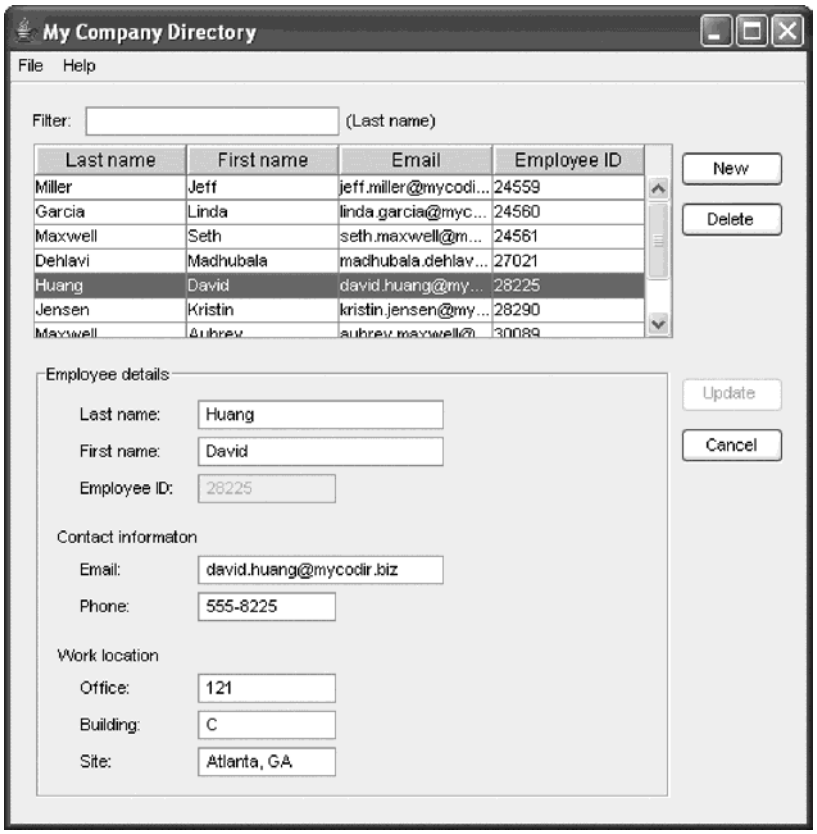
 查看 PDF 版本

教程: Hello World Java

简介: 构建使用 Web Service 的富 Java 客户机

客户机的图形用户界面（GUI）大部分是使用 Swing 组件以可视方式为您预先构建的。在第一个模块中，您将通过使用 Java 可视编辑器来完成基本 GUI 组件的布局。在第二个模块中，您会将 GUI 组件绑定至 Web Service 数据源、服务以及数据源所返回的对象。在 Java 可视编辑器中构建应用程序时，您将使用数据源、数据对象和绑定程序，它们是由 Java 可视编辑器生成并由应用程序使用的 helper 类的实例。

查看已完成产品的图片:



学习目标

在本教程中，您将学习下列课程:

- 如何使用 Java 可视编辑器来设计用户界面并对其进行布局
- 如何将界面元素绑定至数据对象和 Web Service

所需时间

要完成整个教程，您将大约需要 2 小时 30 分钟。

系统要求

- WebSphere Application Server V6.1。此服务器可能已随您的产品安装，或者您可以使用您自己的独立安装。本教程中的方案要求您将样本 Web Service 部署于正在本地运行的 WebSphere Application Server 上。

该样本 Web Service 可能可以在其他服务器上运行，但本教程仅经过 WebSphere Application Server V6.0 和 V6.1 测试。

先决条件

您应该熟悉下列概念:

- 基本 Java 开发
- 基本 Web Service 原理
- 基本的工作台技能，例如，处理项目以及浏览视图和透视图

模块 1: 在可视编辑器中设计客户机 GUI

本模块教您如何使用 Java 可视编辑器来将可视组件添加至应用程序，然后，对它进行可视布局并设置排列约束。本模块的最后一课说明如何运行 Java 文件，以查看它在作为实际应用程序时的外观。

切记: 在开始学习本模块之前，您应该掌握教程简介中概述的必备知识。

学习目标

在完成本模块中的课程之后，您将了解相关概念以及如何执行下列任务:

- 将 JTable 添加到 Java 接口并对 JTable 进行布局
- 运行可视类以测试工作

所需时间

完成本模块大约需要 15 分钟。

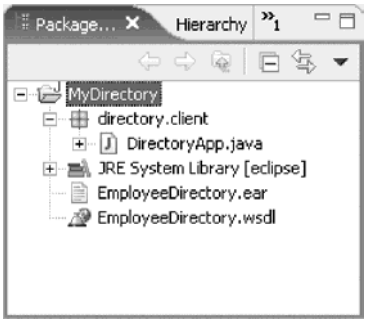
课程 1.1: 设置 Java 项目

在本课中，您会通过将项目导入至工作空间来设置 MyDirectory 项目。该项目包含单个 Java 类以及将在稍后使用的其他文件。

因为本教程的重点是将可视组件绑定至 Web Service，所以“我的公司目录”应用程序的大部分 Java GUI 都已经为您设计好了。

MyDirectory 项目是您将在本教程中处理的主要 Java 项目。它包含 DirectoryApp.java 文件，该文件是包含正在构建的主要 Java 应用程序的 Java 文件。本教程包含 MyDirectory 项目的几个辅助版本: 一个是在各个模块的开始时使用，还有一个已完成项目的完成版本。

1. 导入 MyDirectory 项目。
2. 在 Java 透视图的“包资源管理器”中，确保 MyDirectory 项目看起来类似于下图:



课程要点

在本课中，您已经导入充当本教程的起始点的示例 MyDirectory 项目。

MyDirectory 项目包含下列资源:

- DirectoryApp.java: 包含在本教程中开发的应用程序的 Java 文件。DirectoryApp.java 文件在一个名为 directory.client 的 Java 包中。
- EmployeeDirectory.ear: 包含样本 Web Service 的企业应用程序。在模块 2 中，您会将此 Web Service 部署到 WebSphere Application Server V6.0 的本地安装中。

- EmployeeDirectory.wsdl: 使用 Web 服务描述语言（WSDL）的 XML 文件，该文件用来描述您将部署的样本 Web Service。在模块 2 中，您将使用此 WSDL 文件来生成供应用程序使用的 Java 代理。

课程 1.2: 添加职员表并对其进行布局

在本课中，您将使用 Java 可视编辑器来对应用程序添加 JScrollPane 和 JTable。在稍后的课程中，您将对 JTable 进行编程以从相应的 Web Service（返回公司目录中所有职员的列表）中获取数据。

添加 JTable 之后，您将使用 Java 可视编辑器的设计视图来定制 JTable 的布局，使之符合下列规范:

- 使 JTable 在水平方向上扩展为三个单元格且在垂直方向上扩展为两个单元格。
- 左边添加 15 像素的嵌入。
- 将 JTable 重命名为 employeesTable。

演示

在 Java 可视编辑器中打开 DirectoryApp.java 文件

要在 Java 可视编辑器中打开 DirectoryApp.java 文件:

1. 在 Java 透视图的“包资源管理器”视图中，展开 MyDirectory 项目和 directory.client 包。
2. 右键单击 DirectoryApp.java 文件，然后选择**打开方式 → 可视编辑器**。Java 可视编辑器会装入 Java 类并在图形画布区域上显示设计。

提示:

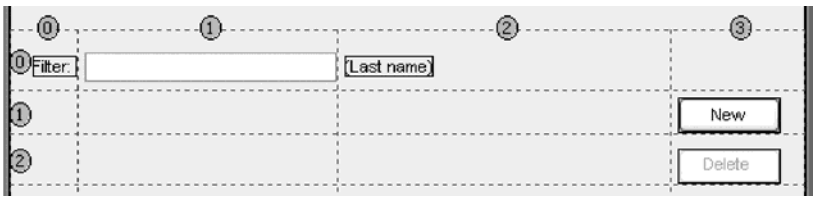
- 要更改 Java 可视编辑器使用的外观，请转至**窗口 → 首选项 → Java → 可视编辑器**，然后指定 Swing 外观。下次打开这个类时，该首选项将生效。本教程使用 Windows 外观。
- 要使“可视编辑器”成为所有 Java 文件的缺省编辑器，可以单击**窗口 → 首选项**，然后转至**工作台 → 文件关联**页以定义首选项。

在 JScrollPane 上添加 JTable

DirectoryApp.java 的主窗口使用具有 JPanel 的 JFrame 作为其主内容窗格。在我们的应用程序中将 JPanel 称为 jContentPane。已将 jContentPane 设置为使用一种称为 GridBagLayout 的布局管理器类型。GridBagLayout 是一种强大布局方案，基于可视组件所占用的单元格网格。Java 可视编辑器通过显示网格边框使您能够便捷地处理 GridBagLayout。当您将新的组件放到网格上时，它还将显示放置标记，并且当您正在使用 GridBagLayout 调整组件大小或移动组件时，它将在这些组件上显示手柄标记。

要将职员表（javax.swing.JTable）添加至 DirectoryApp.java 用户界面:

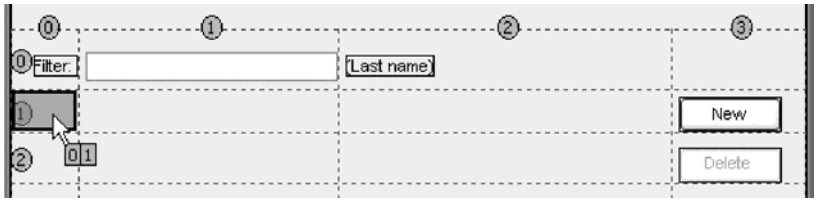
1. 在设计视图或 Java Bean 视图中，右键单击 jContentPane，然后选择**显示网格**。红色的虚线显示网格边框，而带数字的蓝色圆圈指示行号和列号。例如，注意**新建**按钮占用了第 1 行（网格 y）和第 3 列（网格 x）处的单元格。



2. 在 Java 可视编辑器选用板中，选择 **JScrollPane 上的 JTable** Swing 组件，该组件在选用板的 **Swing 组件**抽屉分类下。

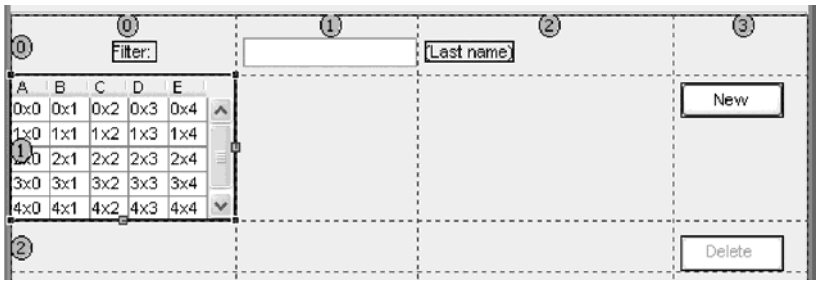
   提示: 缺省情况下，选用板折叠在设计区域的右边。可以调整选用板的大小和移动选用板。

3. 在网格（第 0 列，第 1 行）处的单元格上移动鼠标指针:



- 当您在网格上移动鼠标指针时，鼠标指针将显示两个带数字的方框，根据鼠标指针的位置，它们告诉您鼠标指针在网格中的 x 坐标和 y 坐标。
- 如果将鼠标指针直接悬浮在网格边框上，则可以创建新的行和列并且将对现有的行和列重新编号。在这种情况下，鼠标指针上的黄色方框、网格之间的黄条以及黄色的列标签和行标签将指示此行为并且会指出放置将产生的影响。

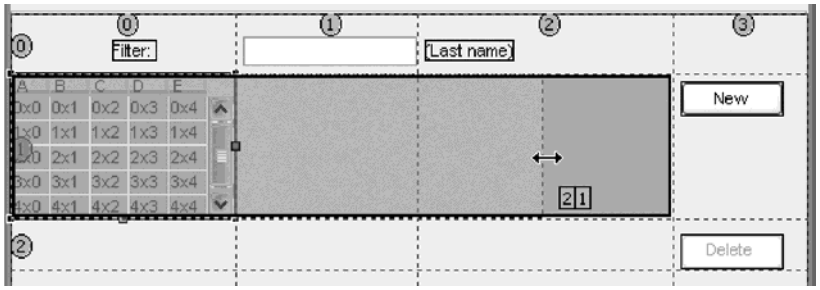
4. 左键单击以将 JScrollPane 和 JTable 放入第 0 列、第 1 行处的单元格中:



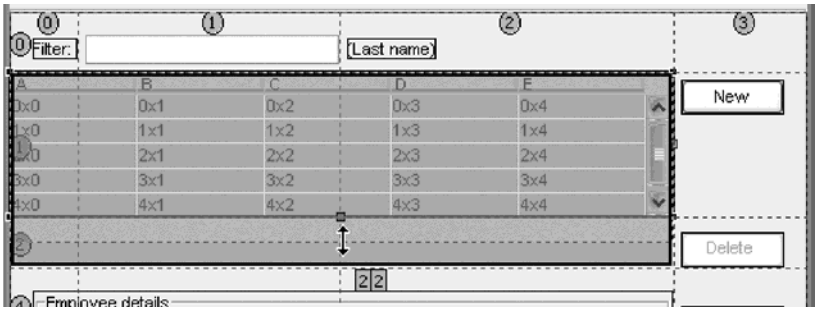
使 JScrollPane 和 JTable 扩展为多个网格列和行

现在，需要使 JScrollPane（及其子 JTable）扩展为三列和两行以更好地调整间隔和大小。要使表扩展这些列和行:

1. 在设计区域或 Java Bean 视图中选择 JScrollPane（因为刚才添加了该项，所以它应该还处于选中状态）。注意 JScrollPane 右边和底部的绿色小方框。您将使用这些调整大小的手柄标记来拖动 JScrollPane 以扩展为多个行和列。
2. 在 JScrollPane 右边的绿色手柄标记上单击并按住鼠标左键。
3. 将鼠标指针向右拖动，直到位置指示第 2 列、第 1 行。当您松开鼠标按键时，深灰色阴影也会指示组件将占用的单元格。



4. 松开鼠标按键。JScrollPane 现在扩展为三列。
5. 重复类似的过程以拖动 JScrollPane 底部的手柄标记，直到 JScrollPane 扩展到第 2 行:



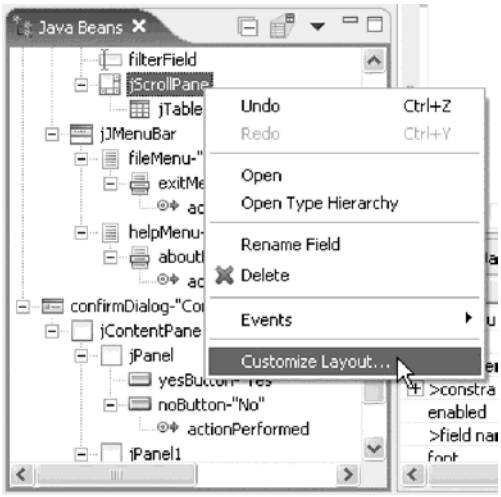
定制 JScrollPane 在网格包中的间隔

GridBagLayout 管理器的另一个功能是您可以指定各种约束以进一步定制布局。例如，可以指定下列约束:

- **锚点**: 可以在单元格内对组件指定锚点的方向，它将影响用户调整应用程序时组件移动的方式。例如，可将组件定位在左上角、中间偏左、居中或右下角。
- **填充**: 可以指示组件在水平和／或垂直方向上占用其单元格内的所有可用空间。
- **嵌入**: 可以在组件的顶部、底部、左边和右边提供边距以在组件与网格边缘之间提供间隔。

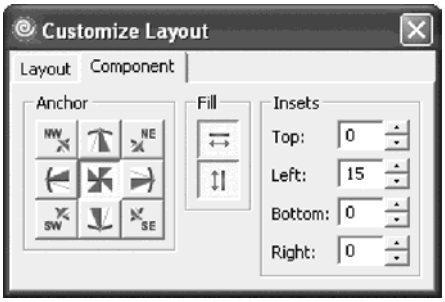
要为 JScrollPane 定制锚点、填充和嵌入:

1. 在设计视图或 Java Bean 视图中，右键单击 JScrollPane 并选择**定制布局**。

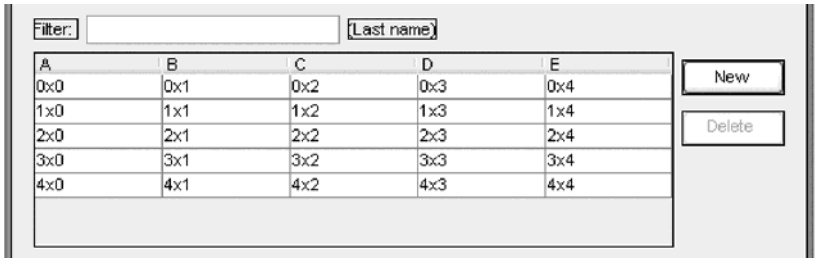


提示: 在为不同的组件选择和更改布局时，“定制布局”对话框可以一直处于打开状态。通过单击菜单栏中的“定制布局”按钮，可以随时打开“定制布局”对话框:

2. 在“定制布局”对话框的“组件”选项卡中，确保按下了锚点居中按钮。
3. 确保按下了**水平填充**和**垂直填充**两个按钮。
4. 左边添加 15（像素）的嵌入以使 JScrollPane 左边的间隔与应用程序上的其他可视组件一样。



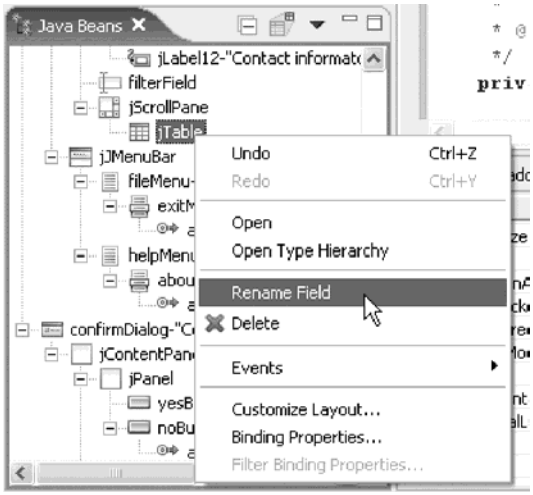
例如，现在表就与**过滤器**标签对齐了。



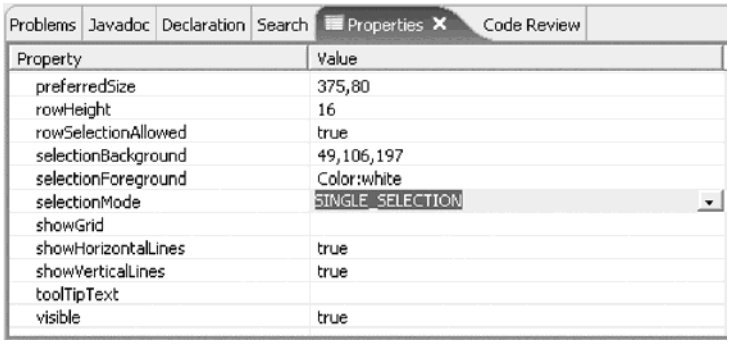
将新的 JTable 重命名为有用的值并设置它以选择单个行

因为稍后您将处理该表，所以重命名 JTable 实例及其 getter 方法将很有用。要重命名该表:

1. 在 Java Bean 视图中，右键单击 jTable 组件并从弹出菜单中选择**重命名字段**。



2. 输入 `employeesTable` 并单击**确定**。JTable 现在的名称是 employeesTable，而用来对它进行实例化的方法为 getEmployeesTable。
3. 设置表以只允许选择单个行:
   a. 在设计视图中选择 employeesTable。
   b. 在“属性”视图中，选择 **selectionMode** 属性并将它设置为 SINGLE_SELECTION。



c. 保存 DirectoryApp.java 文件。

课程要点

在本课中，您已学习如何使用可视编辑器来对现有用户界面添加表。然后学习了如何定制它的布局、位置和间隔。

课程 1.3: 运行可视类

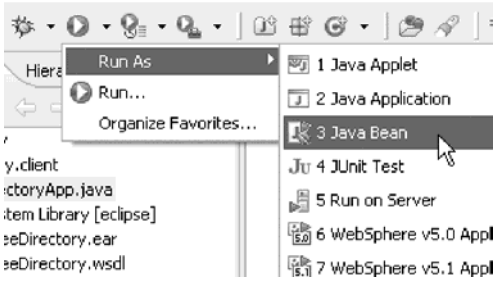
现在，您就可以运行 Java 应用程序来预览它的外观了。工作台和可视编辑器使您能够快速方便地运行应用程序，并且在开发的任何时候都可以重复这些步骤，以测试类的实际运行时外观及行为。

演示

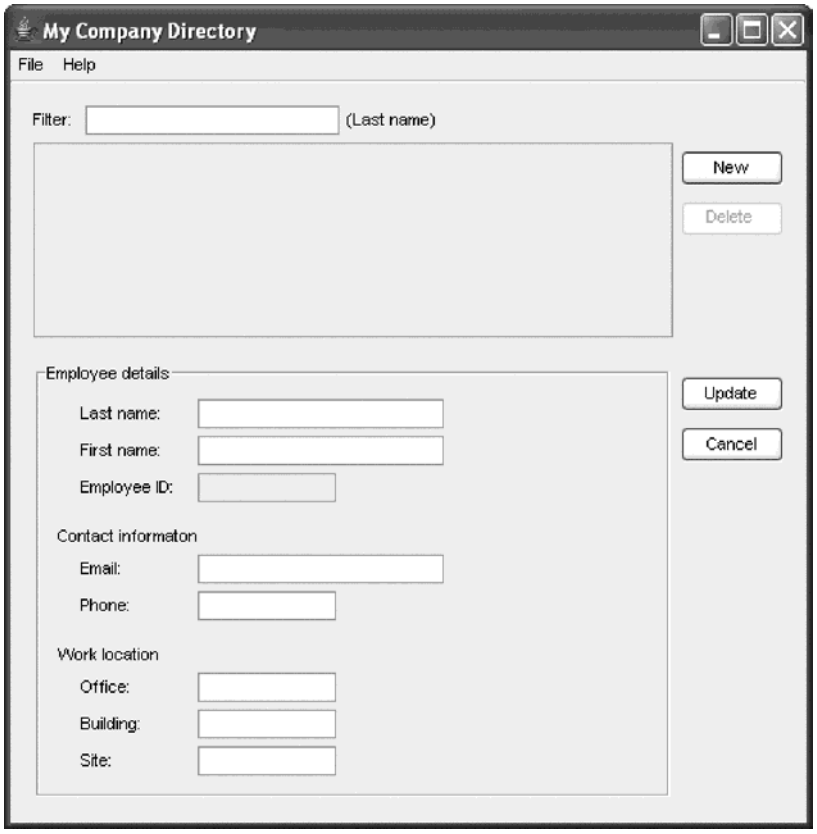
Java 可视编辑器提供了 Java Bean 启动程序，该启动程序能够在不使用 main() 方法的情况下运行类。当运行可视类时，它将在独立的虚拟机（VM）中启动应用程序。如果将可视类作为 Java 应用程序运行，则该启动程序将尝试执行类中的 main() 方法。对于本教程，您的应用程序包括调用并显示 DirectoryApp JFrame 的 main() 方法，因此可以将它作为应用程序或 Java bean 运行。

要将 DirectoryApp.java 文件作为 Java bean 运行:

1. 确保在 Java 可视编辑器中打开了 DirectoryApp.java 文件。
2. 在菜单栏中，单击**运行 → 运行方式 → Java bean**。



提示: 将在桌面上打开使用 Swing 外观的应用程序，该外观已在“可视编辑器”首选项（**窗口 → 首选项 → Java → 可视编辑器**）中定义。或者，可以单击**运行 → 运行**，然后定义用于启动此 Java bean 的特定启动配置的外观。如果将此应用程序作为应用程序而不是 bean 运行，它还将使用 Windows 外观，原因是在 main() 方法中定义了该外观。本教程中使用的屏幕快照显示了 Windows 外观。



课程要点

因为您只设计了界面，但尚未对任何数据连接或事件功能进行编程，所以不能使用应用程序来完成任何任务。然而，可以查看用户将看到的基本布局和外观。您可以尝试单击某些按钮，但是会注意到它们不会执行任何操作。不过，已经为您实现了“文件”菜单和“帮助”菜单。您可以尝试查看这些菜单的功能，并且可以检查 Java 代码以了解如何使用 actionPerformed 事件来实现这些菜单。

学习的课程

此模块向您介绍了使用 Java 可视编辑器为富客户机设计界面的过程。然而，除了设计客户机的可视外观外，您还需要执行更多的操作才能使客户机有用。通常，您将需要包括事件行为或其他逻辑，在该示例中，需要将可视元素绑定至某种数据源。

在此模块中，您已学习如何执行下列任务:

- 使用“项目交换”导入来导入 Java 项目
- 在 JScrollPane 中将 JTable 添加至可视类
- 使用 GridBagLayout 管理器来在富客户机上对表进行可视布局
- 运行应用程序以查看富 Java 客户机的实际外观

在下一模块“模块 2: 将可视组件绑定至 Web Service”中，您将实现“我的公司目录”的简单界面并将它转变成功能强大的富客户机，以便访问从公司目录中创建、检索、更新和删除职员记录的 Web Service 方法。

模块 2: 将可视组件绑定至 Web Service

此模块将教您如何将“我的公司目录”的可视元素（按钮、职员表、字段和其他操作）绑定至 Web Service。Web Service 提供了实际的功能来从样本目录创建、检索、更新和删除职员。

学习目标

在完成本模块中的课程之后，您将了解相关概念以及如何执行下列任务:

- 将表绑定至数据 Web Service 数据源
- 将字段绑定至对象
- 使用操作对按钮进行编程

完成此模块大约需要 **2 小时**。

课程 2.1: 安装并部署 Web Service

在本课中，您会将样本企业应用程序（EAR）文件安装到 WebSphere Application Server V6.1 并部署 EmployeeDirectory Web Service。您的应用程序将使用此 Web Service 来创建、读取、更新和删除职员记录。

开始之前，*必须完成下列其中一个选项*，以确保 MyDirectory 项目处于正确的起始点:

- 完成第 3 页的『模块 1: 在可视编辑器中设计客户机 GUI』。

  或者

- 在开始模块 2 时导入 MyDirectory 项目。

  提示: 如果在导入期间未指定另一项目名称，则将覆盖 MyDirectory 项目内容。

MyDirectory Java 项目包括一个 EmployeeDirectory.ear 文件。您将使用 WebSphere 管理控制台来安装包含在 EAR 文件中的 EmployeeDirectory 企业应用程序。当您安装应用程序时，还将部署包括在应用程序中的 Web Service。完成的“我的公司目录”应用程序使用此已部署的 Web Service。

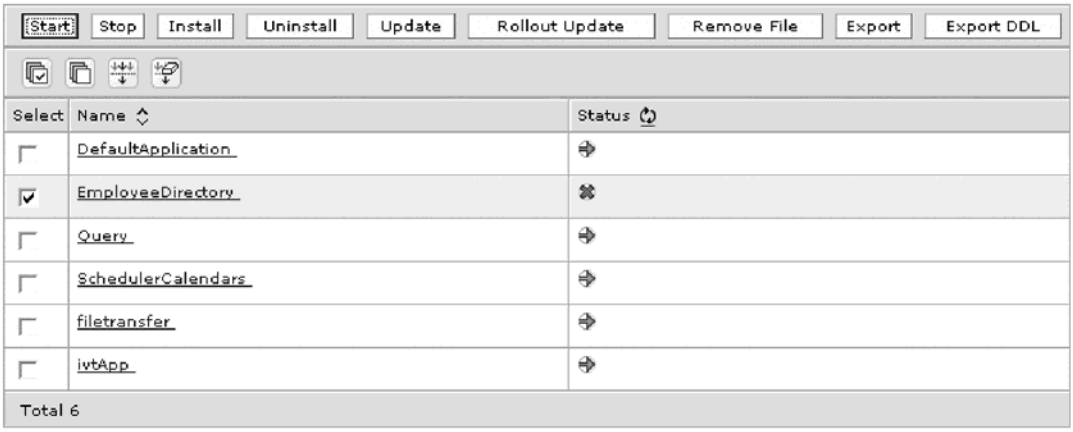
要在 WebSphere Application Server V6.0 环境上安装样本 EmployeeDirectory 应用程序并部署 Web Service:

1. 从工作台启动一个应用程序服务器实例。以下步骤描述了如何从工作台启动服务器，此外，还可以使用其他几种方法来完成该任务:
   a. 打开“服务器”视图。要将“服务器”视图添加至 Java 透视图，请单击**窗口 → 显示视图 → 其他 → 服务器 → 服务器**。
   b. “服务器”视图会列示已安装并设置的服务器。
   c. 右键单击服务器并选择**启动**。当“服务器”视图显示服务器的状态为**已启动**或控制台提示打开服务器 `server1` 以进行电子商务时，就成功启动了该服务器。现在，可以运行管理控制台了。

   注: 如果“服务器”视图中不存在服务器实例，则创建新的服务器:
   a. 右键单击“服务器”视图并选择**新建 → 服务器**。
   b. 使用“新建服务器”向导来添加 WebSphere Application Server V6.1。
2. 运行 WebSphere 管理控制台。以下指示信息描述如何从工作台运行管理控制台，此外，还可以使用其他方法来完成该任务:
   a. 在“服务器”视图中，右键单击刚启动的服务器并选择**运行管理控制台**。WebSphere 管理控制台将在浏览器窗口中打开。

b. 输入用户标识并单击**登录**。管理控制台的“欢迎”页打开。输入的用户标识只用来跟踪对服务器配置数据所做的特定于用户的更改。

3. 使用管理控制台来安装 MyDirectory 项目中的 EmployeeDirectory.ear 企业应用程序。管理控制台使用向导方法来帮助您安装应用程序，在该向导中单击**下一步**以按顺序从一个页面移至下一个页面，直到设置了所有选项为止。在本教程中，要安装包含 Web Service 的样本企业应用程序:
   a. 在管理控制台的左边，展开**应用程序菜单**选项并单击**安装新的应用程序**。
   b. 选择**本地文件系统**并在**指定路径**字段中输入 MyDirectory 项目中的 EmployeeDirectory.ear 文件的完整路径。提示: 要获取完整路径，请在“包资源管理器”中右键单击 EmployeeDirectory.ear 文件并选择**属性**。“属性”页会列示该文件的位置，可以复制该位置并将它粘贴到**指定路径**字段中。
   c. 单击**下一步**直到进入**选择安装选项**页。
   d. 选择**部署 Web Service**。
   e. 单击**下一步**直到进入**摘要**页，然后单击**完成**。
   f. 当提示您应用对本地配置所做的更改时，请单击**保存至主配置**链接。检查这些更改并单击**保存**按钮。
4. 使用管理控制台来启动 EmployeeDirectory 应用程序:
   a. 单击**应用程序 → 企业应用程序**。EmployeeDirectory 应用程序在服务器上列示为已安装应用程序，但是它的状态为“已停止”。

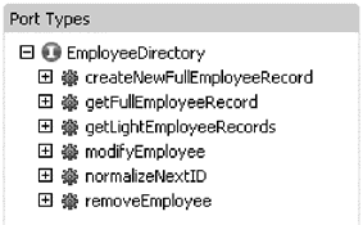


   b. 选择 EmployeeDirectory 旁边的复选框并单击**启动**。将显示一条消息，指示已成功启动 EmployeeDirectory 应用程序，并且“状态”图标会更改为绿色箭头。

EmployeeDirectory 应用程序现在正在本地主机（端口 9080）上运行，并且您现在可以访问 Web Service。在完成本教程之后，可以返回至管理控制台，停止 EmployeeDirectory 应用程序，然后将它卸载。

如果打开 MyDirectory 项目中的 EmployeeDirectory.wsdl 文件（缺省情况下，它在图形 WSDL 编辑器中应该是打开的），则可以检查刚才部署的 Web Service。如果 WSDL 文件没有在 WSDL 编辑器中打开，则在工作台中可能未打开“Web Service 开发者”功能。可以在首选项（**窗口 → 首选项 → 工作台 → 功能**）中指定工作台功能。

WSDL 编辑器的下图显示了 EmployeeDirectory 服务中可用的操作:



可以使用 WSDL 编辑器来检查每个操作及其相应请求消息和返回消息。这可以帮助您了解 Web Service 以及在剩下的课程中如何使用。

课程 2.2: 将职员表绑定至 Web Service 数据源

“我的公司目录”应用程序显示目录中的所有当前职员记录列表。记录显示在具有可排序的列（包括姓、名、电子邮件和职员标识）的 JTable（employeesTable）中。要为该表获取记录，需要将 employeesTable 绑定至样本 Web Service 数据源返回的数据对象。

演示

数据对象、数据源和绑定程序概述

为了获取本地数据对象以供 employeesTable 使用，将使用可视编辑器来将数据源添加至应用程序。该数据源连接至样本 Web Service 代理，并发现可用于应用程序的服务方法。然后，将选择该数据源中可用的 getLightEmployeeRecord 服务方法。最后，您会将应用程序中的 employeesTable 绑定至在行数据对象（lightEmployeeRecordRows）中返回的字段。

通过使用 Java 可视编辑器的内置绑定程序类，可以快捷地创建所有数据源和数据对象。可视编辑器提供了一组类属接口和类，它们在将可视组件绑定至数据工厂时生成到项目中。缺省情况下，绑定程序类生成到名为 jve.generated 的包中。可视编辑器将绑定程序类作为类属实现来提供，可以进一步定制并增强这些类以满足应用程序的需要。本教程演示了只对缺省绑定程序类进行基本和简单的操作时所表现出的强大功能和灵活性。

重要: 在开始本课之前，强烈建议先阅读下列帮助主题。这些主题可以帮助您更多地了解 Java 可视编辑器提供的数据对象、数据源和绑定程序所具有的功能和逻辑:

- 数据绑定程序概述
- 绑定程序 API 参考

对于本教程，您将在应用程序中使用 Web Service 数据源、几种类型的数据对象和几种类型的绑定程序。当您将这些对象的实例添加至应用程序时，可视编辑器会将必需的类添加到项目的 jve.generated 包中，在这里您可以扩展、替换或重写数据绑定逻辑。通过在设计视图的自由格式区域中显示应用程序正在使用的数据对象、数据源和绑定程序，Java 可视编辑器可以提供对绑定对象的可视支持。可视编辑器会在可视组件、数据对象和数据源之间绘制一些线，以显示任何选择的对象的当前绑定。

下图是可视组件、绑定程序、数据对象和数据源交互方式的简要概述。在本教程中构建的应用程序表明了对绑定程序的稍微复杂和更具创意的使用。此图没有准确地表示要构建的样本应用程序中的绑定程序、数据对象和数据源。

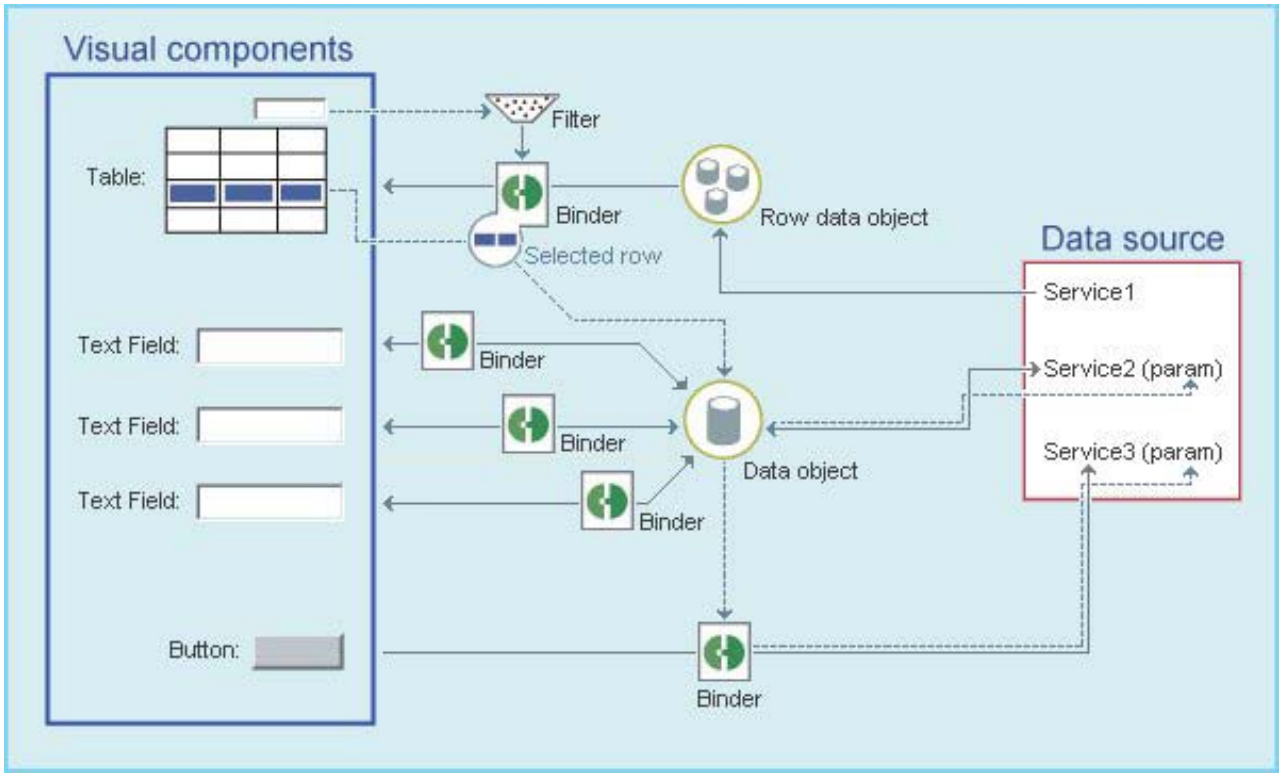


图 1. 下图说明了可视组件、绑定程序、数据对象与数据源之间的样本关系

在图 1 中，每个可视组件都有它自己的绑定程序，该绑定程序将可视组件与数据对象相关联，如果可视组件为按钮，则将可视组件与数据源相关联。文本字段的绑定程序将字段绑定至数据对象的特定属性。此图中的行数据对象和数据对象的数据都来自对数据源中的服务的直接调用。文本字段的数据对象使用表中所选行的键值作为它用来调用 Service2 的自变量，Service2 会返回可能包括有关表中所选行的更多信息的完整记录。同样，此完整记录又用作按钮的操作绑定程序在调用 Service3 时的自变量，Service3 可能是更新在字段中输入的值的方法。有关数据对象、数据绑定程序和数据源的更详细说明，访问先前提供的链接。

使用提供的 WSDL 文件在项目中生成 Web Service Java 代理

为了使用在服务器上运行的 Web Service，Java 应用程序要求 Java 代理或客户机与它交互。通过使用 WSDL 文件，可以使用“Web Service 客户机”向导来将 Java 代理生成到 Java 项目中。MyDirectory 项目包括将用来生成此代理的 EmployeeDirectory.wsdl 文件。在生成 Java 代理之后，可以创建表示 Web Service 的数据源并开始对可视组件进行绑定。

重要: 本课中使用的 WSDL 文件假定您已将 Web Service 部署在 WebSphere Application Server 的本地安装中并且使用了本地主机的缺省端口（http://localhost:9080）。如果以不同方式部署了 EAR 文件，则在继续之前必须相应地编辑该 WSDL 文件。

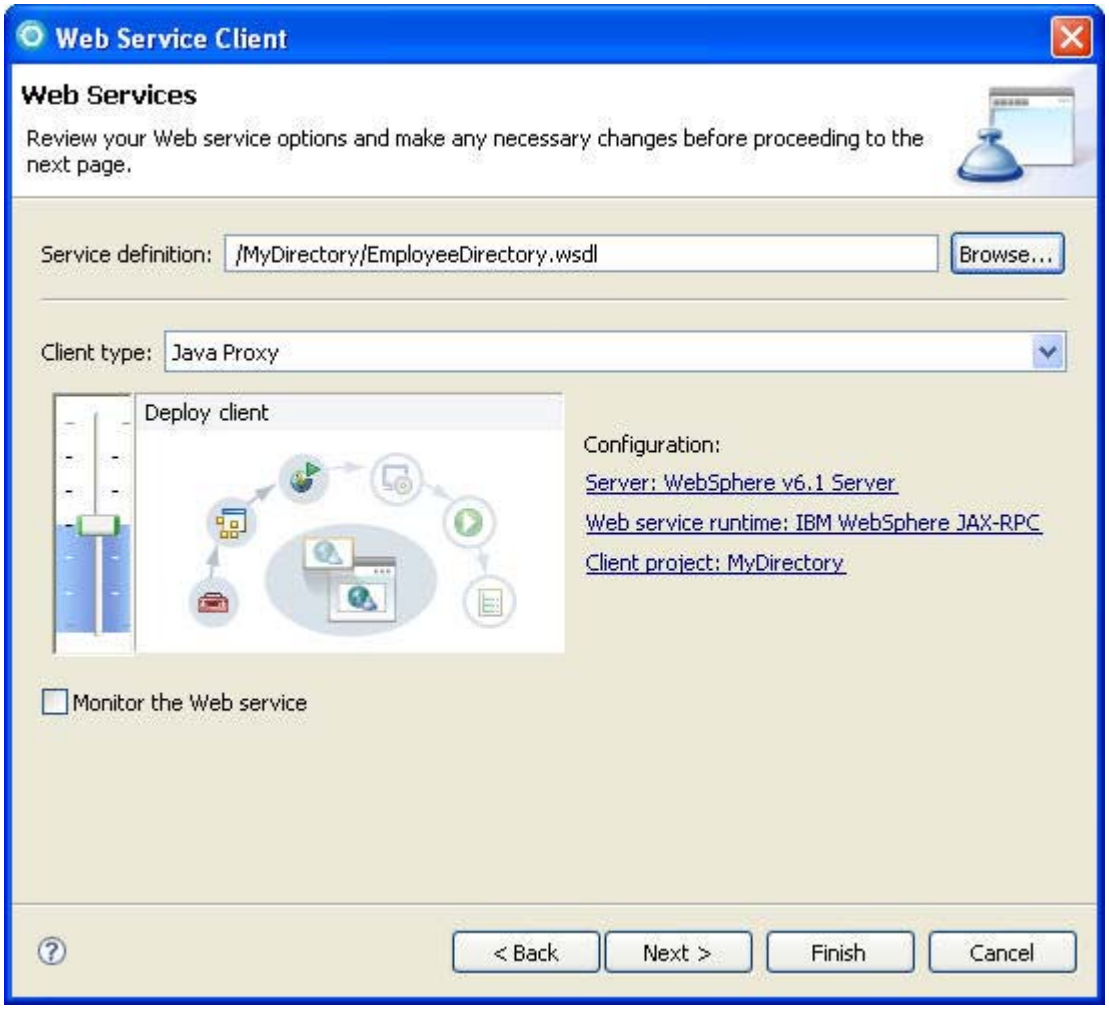
要在项目中生成 Web Service Java 代理:

1. 在主菜单上，单击**文件** → **新建** → **其他**并选择 **Web Service** → **Web Service 客户机**向导。如果没有显示 Web Service 类别，则选择**显示全部向导**。
2. 使用向导定义 Web Service 客户机:
   a. 对于**服务定义**，请输入 MyDirectory 项目中提供的 WSDL 文件: `/MyDirectory/EmployeeDirectory.wsdl`。
   b. 在**客户机类型**字段中，选择 **Java 代理**。

c. 将滑块条设置为**部署客户机**。

d. 确保服务器就是您要运行的服务器，并为此服务器正确设置了 Web Service 运行时。本教程已根据安装了 IBM WebSphere JAX-RPC 运行时的 WebSphere V6.0 和 WebSphere V6.1 进行测试。

e. 确保 Java 代理客户机已输出至 MyDirectory 项目。



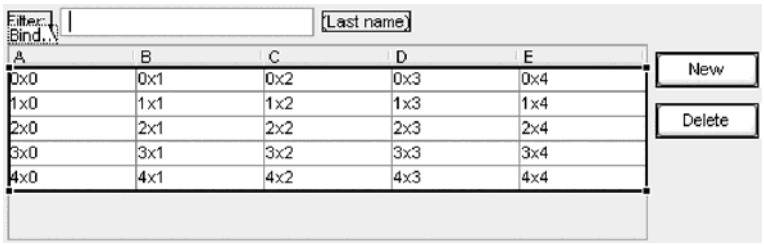
3. 单击**完成**。“Web Service 客户机”向导将在项目的新包（directory.service）中生成 Java 代理。

将 employeesTable 绑定至 Web Service 所返回的行数据对象

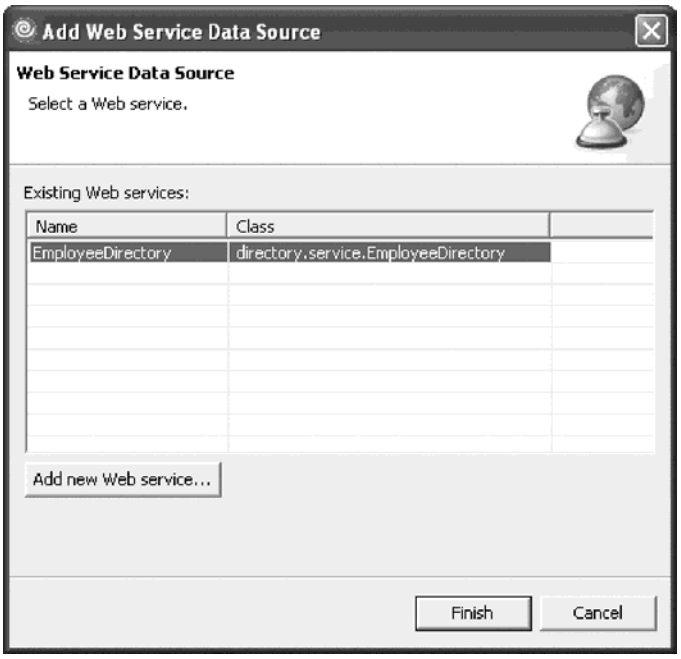
因为 employeesTable 是在此应用程序中要绑定的第一个可视组件，所以需要创建一个数据源以指向刚才添加至项目的样本 Web Service 代理。当在后面的课程中绑定其他可视组件时，您将重用此数据源。在此步骤中，将添加 Web Service 数据源和 lightEmployeeRecordRows 数据对象。

要绑定职员表:

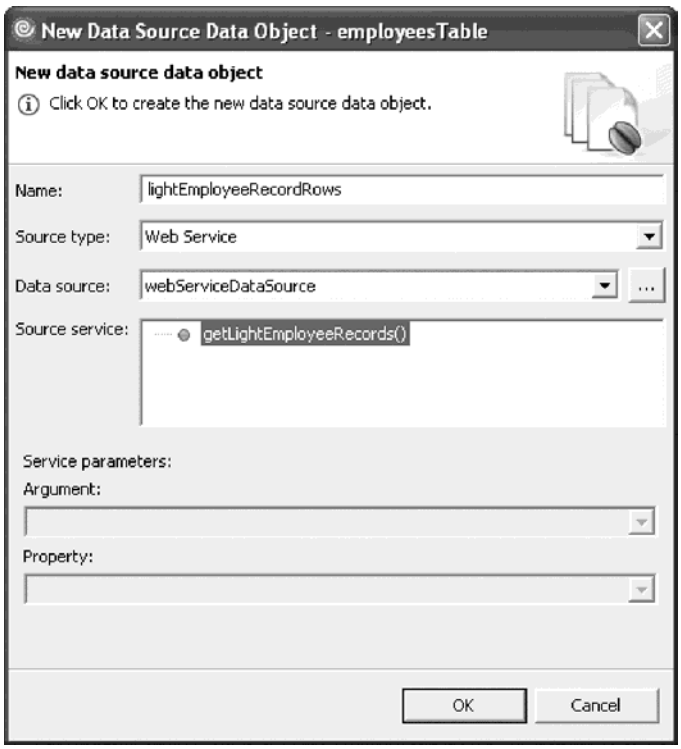
1. 在 Java Bean 视图或设计视图中，选择 employeesTable。（确保没有选择它的 JScrollPane 父代。）标记为**绑定**的小选项卡显示在 employeesTable 的设计区域的顶部。



2. 单击 employeesTable 上的**绑定**选项卡。或者，可以右键单击 employeesTable 并选择**绑定属性**。
3. 因为应用程序中没有数据对象，所以需要添加新的数据对象。单击**新建数据源数据对象**。
4. 在**源类型**字段中，选择 **Web Service**。
5. 因为尚未将 Web Service 数据源添加至应用程序，所以您现在需要添加它。在**数据源**字段的旁边，单击 **...** 按钮以打开“添加 Web Service 数据源”对话框，该对话框会查找项目中可用的 Web Service 客户机或代理。
6. 选择 EmployeeDirectory Web Service 并单击“完成”。新的数据源会添加至 DirectoryApp.java 文件。

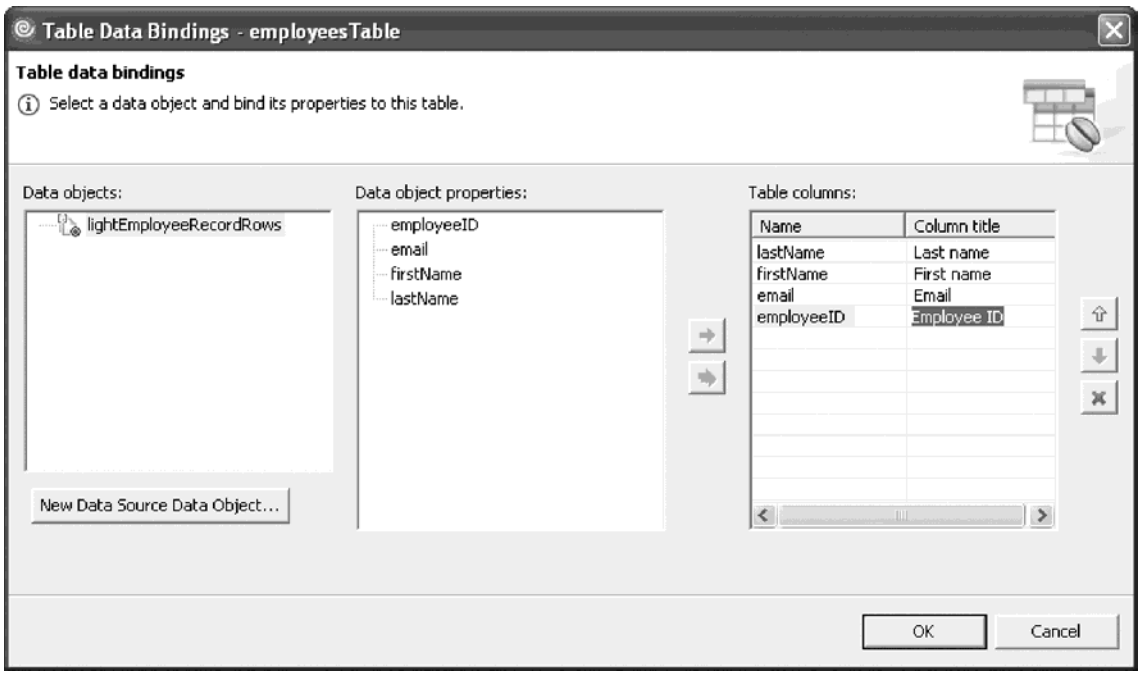


7. 在“新建数据源数据对象”对话框中，在“源服务”字段中选择 getLightEmployeeRecords() 并接受新数据对象的缺省名称: lightEmployeeRecordRows。此服务方法不需要参数。单击“确定”。新的数据对象将创建并显示在设计视图的自由格式区域上。



提示: 因为您正在绑定表，所以“新建数据源数据对象”对话框只显示返回行数据对象的服务。在这种情况下，getLightEmployeeRecords() 方法是返回对象数组的唯一可用服务。

8. 在“表数据绑定”对话框中，选择 lightEmployeeRecordRows 数据对象。
9. 现在，需要选择想要显示在 employeesTable 上的 lightEmployeeRecordRows 数据对象的属性:

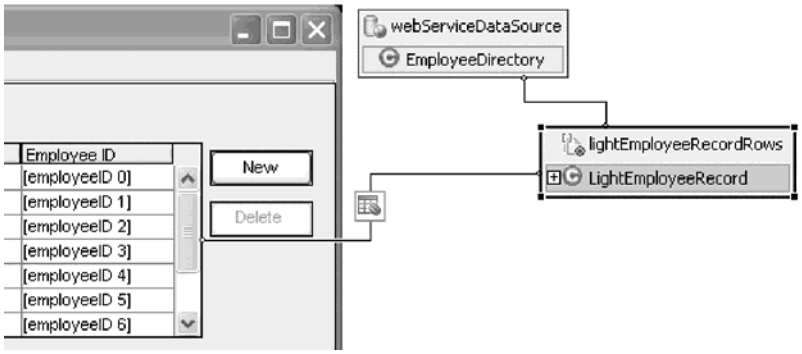


a. 单击双箭头 ⇨ 按钮以将所有对象属性添加至**表列**列表。
b. 使用向上和向下箭头按照从上向下的顺序排列这些列: lastName、firstName、email 和 employeeID。
c. 重命名列标题: 姓、名、电子邮件和职员标识。

提示: 在完成表绑定之后，可以总是返回至绑定属性，并且可以随时重命名这些列和对它们重新排序。

d. 单击**确定**。

现在，通过使用 JRowTableBinder 已将 employeesTable 绑定至 lightEmployeeRecordRows 数据对象。如果在自由格式区域上单击 lightEmployeeRecordRows 数据对象，则可视编辑器会绘制一条从该数据对象至表的线。在这条线上，JRowTableBinder 用表绑定程序  图标表示。将有另一条线指示数据对象使用 webServiceDataSource 作为它的数据源。



课程要点

注意对项目和应用程序所做的更改。在学习本课期间，您已添加 Web Service 数据源、行数据对象以及将 employeesTable 绑定至该行数据对象的绑定程序。

检查新包（jve.generated），它是在项目中创建的以存放 Java 可视编辑器生成的所有绑定程序类。还要注意新包（directory.service），它存放 Web Service 的 Java 代理。描述或总结在本课中学习的内容。

现在，当您运行“我的公司目录”应用程序时，职员表会由 Web Service 使用现有职员记录来填充。

课程 2.3: 将详细信息字段绑定至表选择

在前面的课程中，已将 employeesTable 绑定至 Web Service 中的 getLightEmployeeRecords() 服务返回的 lightEmployeeRecordRows 数据对象。现在，需要根据表中选择的职员来填充详细信息字段。

要获取每个选择的职员的其他详细信息，需要使用另一个数据对象。将要添加的 selectedEmployeeRecord 数据对象是由 getFullEmployeeRecord() 服务返回的。此服务会将表中选择的职员的标识作为参数，并且会访存有关该职员的其他详细信息（包括电话号码和工作地点）。

将表绑定至行数据对象时使用的 JRowTableBinder 可以简化此步骤。JRowTableBinder 将表中选择的元素显示为单独的数据对象，该数据对象可以用作 getFullEmployeeRecord(java.lang.Integer) 方法的参数。然后，您可以很容易地将每个文本字段绑定至 selectedEmployeeRecord 数据对象中的相应属性。

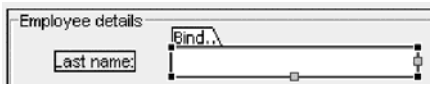
更多地了解此 Web Service: Web Service 包括用于获取每个职员的所有详细信息的两个服务。表中会列示所有职员，但表中只会显示部分数据。这样，当选择了单个职员时，可以只检索该选择职员的其他方面的职员信息。如果在表请求数据时，Web Service 发送每个职员的所有数据，则 Web 流量可能很大并且会使应用程序的性能降低。

例如，如果职员记录包括照片或附件，则您在获取职员的完整列表时，并不想要检索所有照片。因此，使用 getLightEmployeeRecord 服务来填充表，而 getFullEmployeeRecord 会获取表中选择的职员的完整记录。

绑定姓字段

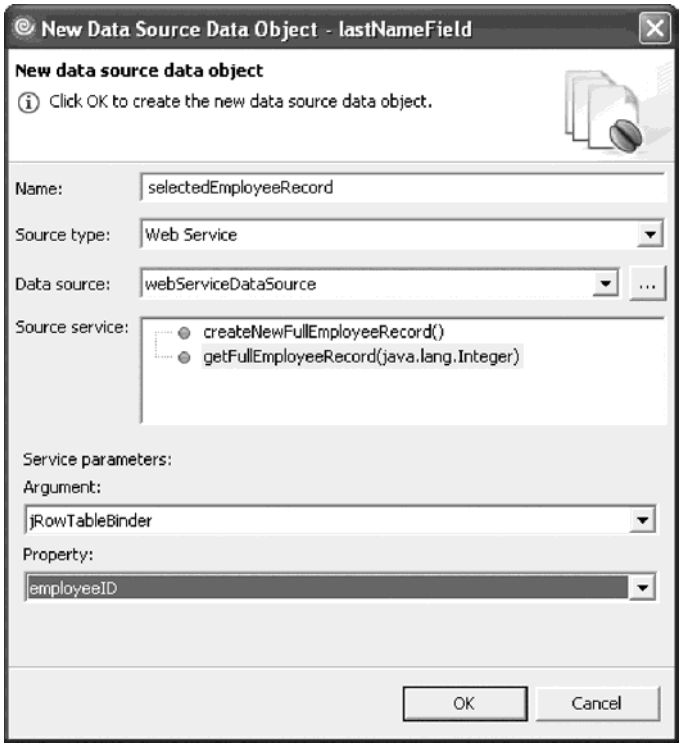
在此步骤中将**姓**字段绑定至 selectedEmployeeRecord 数据对象中的 lastName 属性:

1. 在 Java Bean 视图或设计视图中，为“姓”（lastNameField）选择 JTextField。设计区域显示文本字段上的**绑定**选项卡。



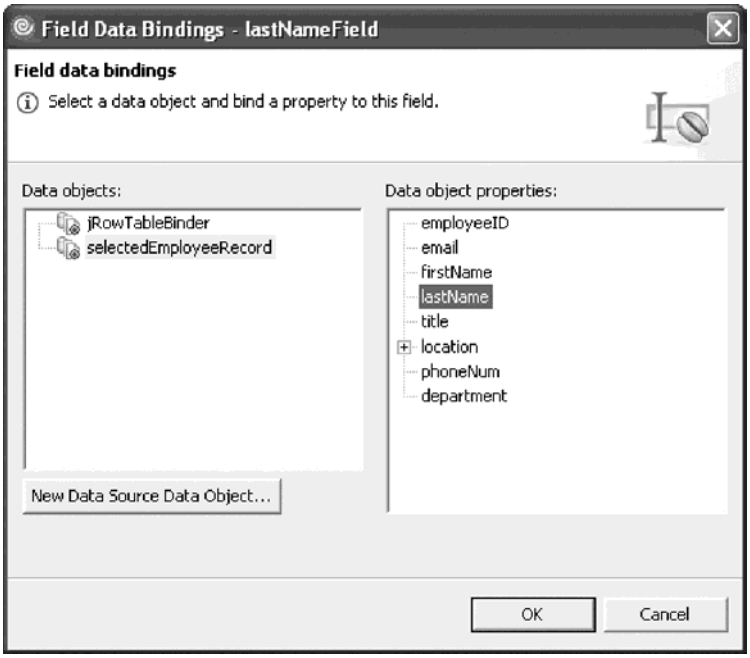
2. 单击**绑定**选项卡以打开“字段数据绑定”对话框。
3. 单击**新建数据源数据对象**。尽管现有 jRowTableBinder 数据对象会返回正确的姓，但是它不包括完整的职员记录。您需要创建表示完整的职员记录的新数据对象。
4. 在**源类型**字段中，确保选择了 **Web Service**，而在**数据源**中，确保选择了 **webServiceDataSource**。
5. 在**源服务**列表中，选择 getFullEmployeeRecord(java.lang.Integer)。“新建数据源数据对象”对话框会列示返回与文本字段兼容的数据对象的服务。
6. 在**名称**字段中，输入 `selectedEmployeeRecord`。
7. 在**自变量**字段中，选择 jRowTableBinder，然后在**属性**字段中，选择 employeeID。选择的行的职员标识现在设置为 getFullEmployeeRecord() 服务方法的自变量。

   注: getFullEmployeeRecord(java.lang.Integer) 要求将整数作为自变量。您需要使用职员表中当前选择的职员标识来检索完整的记录。当绑定了表时，可视编辑器自动生成了 jRowTableBinder（它将侦听当前在该职员表中进行的选择）。对于整数参数，将使用 jRowTableBinder 中选择的行的 employeeID。



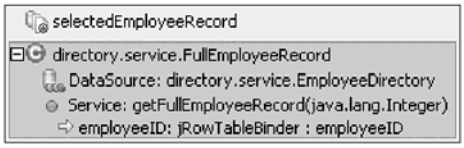
8. 单击**确定**。

9. 在“字段数据绑定”对话框中，确保在“数据对象”列表中选择了 **selectedEmployeeRecord**。注意，与 jRowTableBinder 数据对象相比，selectedEmployeeRecord 数据对象具有更多可用的属性。
10. 在**数据对象属性**列表中，选择 lastName 属性。



11. 单击**确定**。应用程序中的“姓”字段现在已绑定至 selectedEmployeeRecord 数据对象的 lastName 属性，该数据对象由 getFullEmployeeRecord() 返回。

    将创建名为 selectedEmployeeRecord 的新数据对象并且会将它添加至应用程序。该数据对象的可视表示会添加至设计视图的自由格式区域，如下图所示:



    现在，当在设计区域中选择 lastName 字段时，会有一条线指示它已绑定至 selectedEmployeeRecord。在这条线的中间，文本绑定程序“文本绑定程序”图标表示用于此绑定的 SwingTextComponentBinder。如果在设计区域上选择这条线或表示绑定程序的图标，则可以在“属性”视图中检查绑定程序的属性。

绑定剩余的详细信息字段

要绑定职员的其余的每个详细信息字段，将遵循与“姓”字段相似的过程，但是不需要添加数据对象。因为已添加了 selectedEmployeeRecord 数据对象，所以可以只将每个字段绑定至 selectedEmployeeRecord 数据对象中的相应属性。

要绑定字段，请对应用程序的职员详细信息部分中的每个字段完成下列步骤:

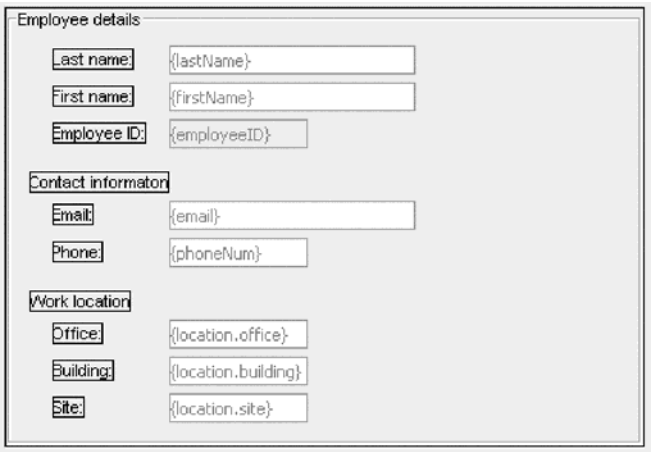
1. 在设计视图中，选择字段并单击**绑定**选项卡。
2. 在“字段数据绑定”对话框的**数据对象**列表中选择 selectedEmployeeRecord。

3. 在**数据对象属性**列表中，选择要绑定的字段的相应属性。以下图表显示每个文本字段需要绑定至的属性:

字段	**selectedEmployeeRecord** 数据对象中的属性
lastNameField	lastName
firstNameField	firstName
idField	employeeID
emailField	email
phoneField	phoneNum
officeField	location.office
buildingField	location.building
siteField	location.site

4. 单击**确定**。

当绑定完文本字段时，设计区域看起来应如下图所示:



使职员标识字段成为只读

因为职员标识字段上的可编辑属性设置为 false，所以该字段被禁用。然而，当数据对象包含值时，文本字段绑定程序的缺省行为会将该字段更改为启用状态。可以关闭此绑定程序行为，这样该字段将保持为它的初始的只读状态。

要防止绑定程序自动切换可编辑属性:

1. 选择“职员标识”字段。设计区域上会显示一条线，它带有表示该字段的绑定程序的图标 。
2. 单击“职员标识”字段的绑定程序 图标。
3. 在“属性”视图中，将 autoEditable 属性更改为 **false**。按 **Enter** 键。

课程要点

现在，当运行应用程序并从表中选择职员时，该职员的记录的详细信息会显示在详细信息字段中。

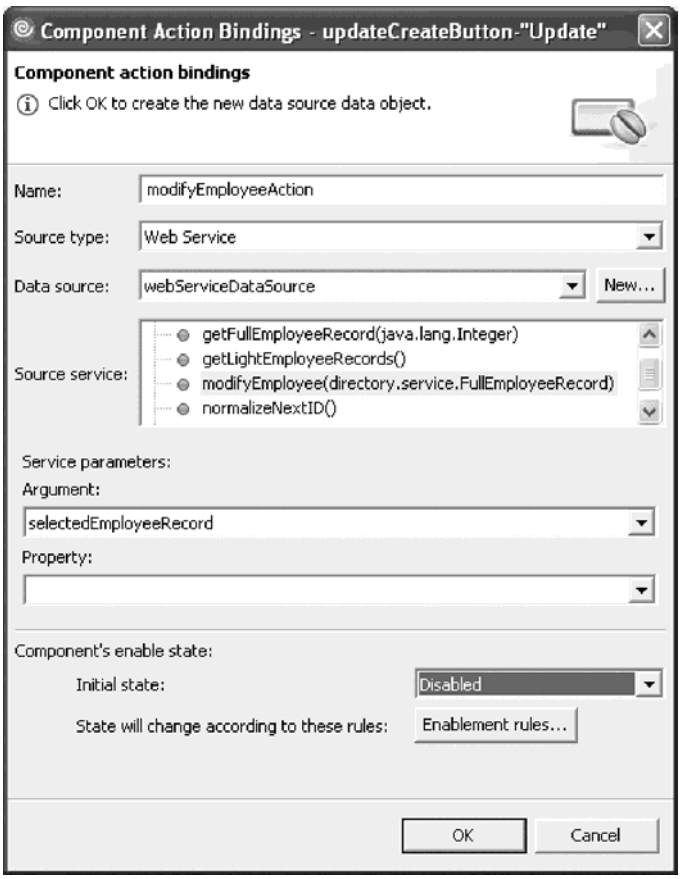
课程 2.4: 将“更新”按钮绑定至操作绑定程序

Java 可视编辑器提供了一些操作绑定程序，这些绑定程序用于在单击按钮时对数据源调用服务。例如，当单击“更新”按钮时，应用程序应该对 Web Service 运行 modifyEmployee() 方法，同时将更改输入到详细信息字段中。在本课中，您会将“更新”按钮绑定至操作绑定程序。

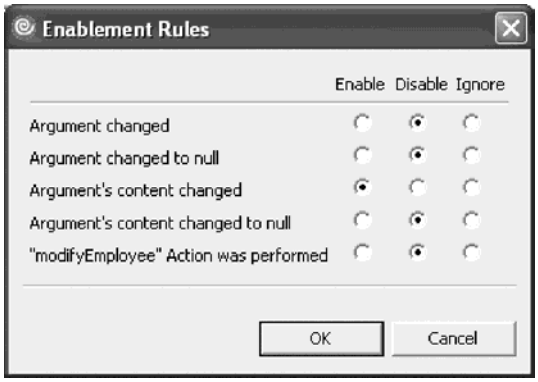
要绑定“更新”按钮:

1. 在设计区域中选择**更新**按钮，然后单击**绑定**选项卡以打开“组件操作绑定”对话框。

2. 在**源类型**字段中，选择 **Web Service**。
3. 在**数据源**字段中，选择 **webServiceDataSource**。
4. 从**源服务**列表中，选择 **modifyEmployee(directory.service.FullEmployeeRecord)**。
5. **名称**字段会自动更改为 **modifyEmployeeAction**。接受此缺省值。
6. 在**自变量**字段中，选择 **selectedEmployeeRecord**。
7. 因为 modifyEmployee() 方法将完整的职员记录作为它的自变量，所以必须将**属性**字段保留为空白。
8. 将该按钮的**初始状态**设置为**禁用**。



9. 要定义按钮如何更改其状态，单击**启用规则**。指定在自变量的内容已更改时按钮的状态为已启用，并且在其他所有实例中为已禁用。单击**确定**。



这意味着，在 selectedEmployeeRecord 的内容更改之前，**更新**按钮已被禁用。换句话说，一旦在其中一个已绑定至 selectedEmployeeRecord 的详细信息字段中输入新值，绑定程序就会启用该按钮。如果选择新记录或单击**更新**，则该按钮将再次变为禁用。

10. 单击**确定**。

会为**更新**按钮添加新的 SwingDataServiceAction 绑定程序。如果在设计区域中选择该按钮，则可视编辑器会绘制一条线，这条线指示该按钮已绑定至 Web Service 数据源。会有一个粉红色的虚线箭头从 selectedEmployeeRecord 对象指向这条线。此箭头指示 selectedEmployeeRecord 是调用服务的自变量。

课程要点

现在，当您运行应用程序时，可以更新职员的记录。

选择表中的职员并更改姓。一旦更改了姓，就会启用**更新**按钮。当单击**更新**时，会调用 modifyEmployee 服务并更新职员。新的姓会反映在职员表中。

课程 2.5: 启用“删除”按钮和确认对话框

在本课中，将对“我的公司目录”应用程序进行编程以删除职员记录。

以下列表描述了想要应用程序使用的行为:

- 当选择表中的职员时，将启用**删除**按钮。
- 当单击**删除**按钮时，“确认删除”对话框将打开并要求您确认删除。
- 如果在“确认删除”对话框中单击**是**按钮，则将删除职员记录，然后关闭“确认删除”对话框并刷新职员列表。
- 当单击**否**时，将取消删除并关闭“确认删除”对话框。

根据是否在表中选择了行，对“删除”按钮进行编程以将其启用或禁用

要对删除按钮进行编程以将其启用或禁用，请将侦听器添加至表，该侦听器在选择了行时会启用该按钮。

1. 在 Java Bean 视图中选择 employeesTable。“源代码”视图会突出显示以下行:

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

2. 紧接此行，将新的 ListSelectionListener 和 valueChanged 事件添加至 employeesTable:

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

3. 在添加这些代码行之后，源代码编辑器会将它们标记为错误，直到导入 ListSelectListener 和 ListSelectionEvent 为止。要添加必需的导入，请在主菜单上单击**源代码 → 组织导入**。下列各行会添加至类的导入部分:

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

现在，当选择表中的某行时，会启用**删除**按钮。

对“确认删除”对话框进行编程以在单击“删除”时将该对话框打开

将 actionPerformed 事件添加至“删除”按钮并对该事件进行编程以打开“确认删除”对话框。

1. 右键单击**删除**按钮并选择**事件 → actionPerformed**。以下事件存根会添加至 getDeleteButton() 方法:

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

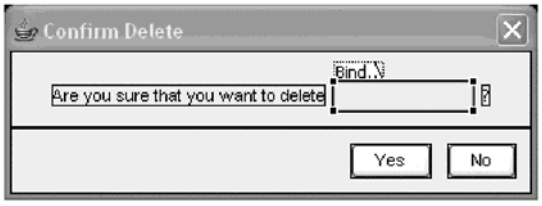
2. 将此生成的存根替换为以下代码，这样将设置为当单击该按钮时出现“确认删除”对话框:

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

绑定“确认删除”对话框中的文本字段

绑定“确认删除”对话框中的文本字段以显示要删除的职员的名。

1. 在 Java Bean 视图或设计区域上，选择 employeeToDeleteField 文本字段并单击**绑定**选项卡。



2. 在“字段数据绑定”对话框中，选择 selectedEmployeeRecord 数据对象和 firstName 字段，然后单击**确定**。

   现在，文本字段已绑定至 employeesTable 中所选行的 firstName 列。

3. 要确保此字段是只读的，将该字段的绑定程序的 **autoEditable** 属性设置为 **false**。

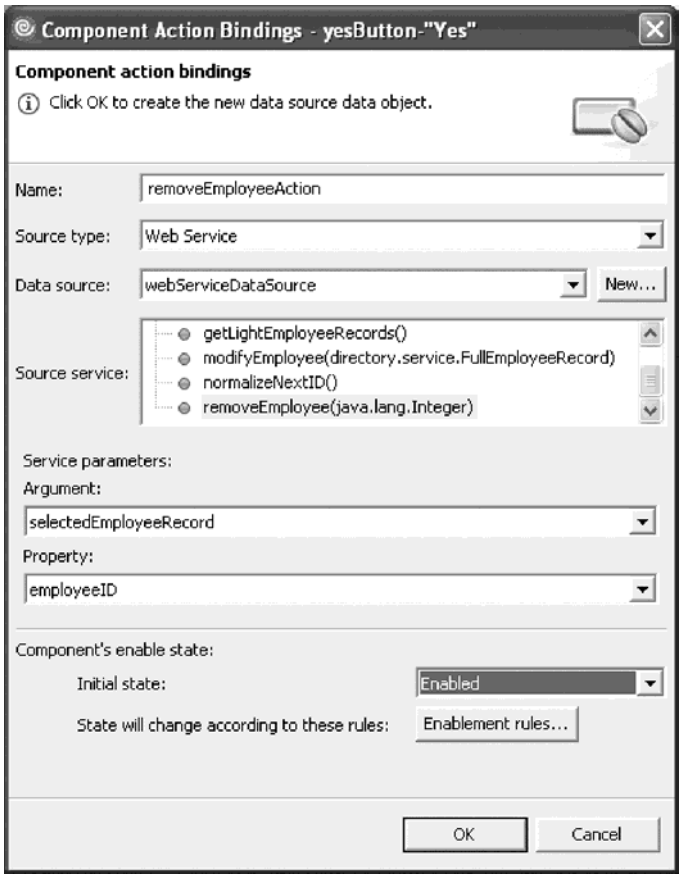
绑定“是”按钮以执行删除

绑定**是**按钮以对 Web Service 调用 removeEmployee(java.lang.Integer) 方法。

1. 选择**是**按钮，然后单击**绑定**选项卡以打开“组件操作绑定”对话框。
2. 在**源类型**字段中，选择 **Web Service**。
3. 在**数据源**字段中，选择 **webServiceDataSource**。
4. 从**源服务**列表中，选择 **removeEmployee(java.lang.Integer)**。
5. **名称**字段会自动更改为 **removeEmployeeAction**。接受此缺省值。

6. 在**自变量**字段中，选择 **selectedEmployeeRecord**。
7. 在**属性**字段中，选择 **employeeID**。因为 removeEmployee() 方法将整数作为它的自变量，所以使用 selectedEmployeeRecord 的职员标识。
8. 将该按钮的**初始状态**设置为**启用**。
9. 对于**启用规则**，对每个条件选择**忽略**。

   此组件状态意味着将总是启用“是”按钮，原因是不需要更改它的状态。



10. 单击**确定**。

添加事件以在删除职员之后隐藏“确认删除”对话框

在此步骤中，将事件添加至**是**按钮的*绑定程序*（而不是**是**按钮本身）。您想要在除去职员之后关闭“确认删除”对话框，这意味着绑定程序已成功地对数据源调用服务。

对 getRemoveEmployeeAction() 方法添加以下代码:

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

此事件代码会在执行绑定程序的操作之后隐藏“确认删除”对话框。

课程要点

现在，当您运行“我的公司目录”应用程序时，可以在表中选择职员，单击**删除**按钮，然后单击**是**以确认删除。将从目录中除去职员记录，并且职员的列表中将反映该操作。

课程 2.6: 设置操作和绑定以添加新职员

在本课中，将启用“我的公司目录”应用程序来添加新的职员记录。

因为对于添加新职员来说，应用程序的行为比较复杂并且是动态的，所以本课具有更高的内在复杂度并且将要求您对源代码进行一些手工更改。另外，本课将演示数据对象的一些高级功能，并且提供了一个具有创意的方法示例，可以借助这些方法使用绑定程序和数据对象来满足您的需要。

以下列表描述应用程序的必需行为:

- 当您单击**新建**按钮时，将发生以下行为:
  - 清除对职员表的选择并且禁用该表。
  - 清除选择表会导致**删除**按钮被禁用。
  - 禁用**过滤器**字段。
  - 清除详细信息字段的任何值（除了新的职员标识之外）。
  - **更新**按钮上的文本将切换为**添加**。
- 当您单击**添加**按钮时，将发生以下行为:
  - 将输入到详细信息字段中的值添加至目录作为新的职员记录。
  - 启用表并刷新值。
  - 启用**过滤器**字段。
  - **添加**按钮上的文本将切换回**更新**。

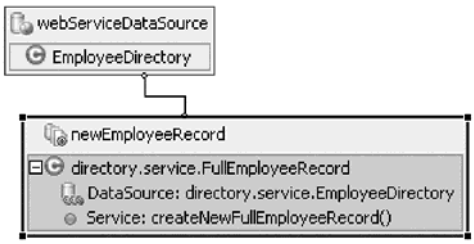
添加调用 createNewFullEmployeeRecord() 的新的数据源数据对象

样本 Web Service 提供了 createNewFullEmployeeRecord 服务，而该服务会提供一个可以使用下一个可用职员标识号填充的新的空白职员记录。然后，可以使用新职员的信息填充此空白记录并将此记录提交回 Web Service。

1. 在 Java 可视编辑器的选用板上，展开“数据对象”抽屉并选择**数据源数据对象**。
2. 将鼠标指针移至设计视图的空白区域或自由格式区域上并左键单击以放下数据源数据对象。将添加新的数据源数据对象并且将在自由格式区域上显示它。

3. 右键单击数据源数据对象并选择**重命名**字段。将该数据对象重命名为 `newEmployeeRecord`。
4. 右键单击 newEmployeeRecord 数据对象并选择**绑定属性**。“数据绑定”对话框打开。
5. 在**数据源**字段中，选择 webServiceDataSource。
6. 在**服务**字段中，选择 createNewFullEmployeeRecord()。
7. 单击**确定**。

在自由格式区域上，您可以看到 newEmployeeRecord 数据源数据对象已绑定至 Web Service。

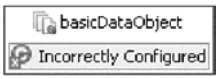


添加基本数据对象以便于切换数据对象

因为详细信息字段和“更新”按钮需要切换方式（用于执行更新和创建新职员），所以在不同的时候它们需要绑定至两个不同的数据对象。为了便于完成此步骤，将添加名为 switchingDataObject 的基本数据对象。将使用此基本数据对象来切换 selectedEmployeeRecord 与 newEmployeeRecord 之间文本字段的绑定。

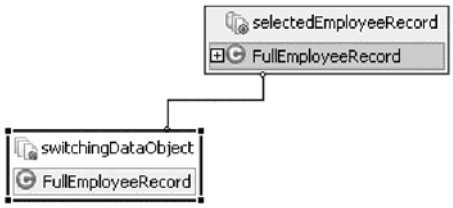
新的基本数据对象只指向在前面的课程中定义的另一个数据对象（selectedEmployeeRecord）。当您创建一个方法来告知此基本数据对象使用先前创建的 newEmployeeRecord 的方法时，这个新的数据对象将很有用。换句话说，此基本数据对象将用作中间数据对象，它可以在 selectedEmployeeRecord 数据对象与 newEmployeeRecord 数据对象之间进行切换，从而使应用程序中的可视组件能够使用两个不同的数据对象。

1. 在可视编辑器选用板上，选择**基本数据对象**并将它放到自由格式区域上。这就添加了 basicDataObject。



2. 将该数据对象重命名为 `switchingDataObject`。
3. 在 switchingDataObject 的“属性”视图中，将 **sourceObject** 属性设置为 **selectedEmployeeRecord**。可以从该属性的“值”列的下拉菜单中，选择 selectedEmployeeRecord。

   现在，switchingDataObject 指的是 selectedEmployeeRecord 并且反映相同的值:

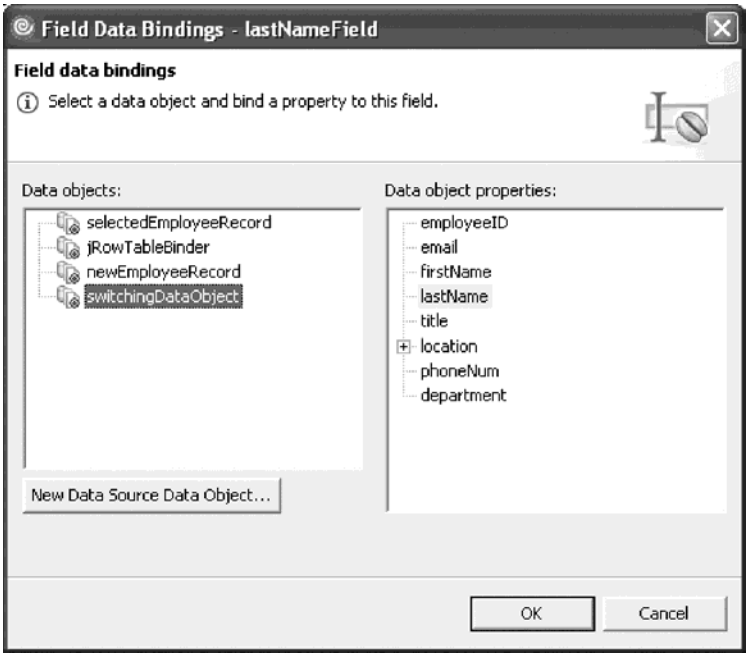


将每个职员字段重新绑定至 switchingDataObject

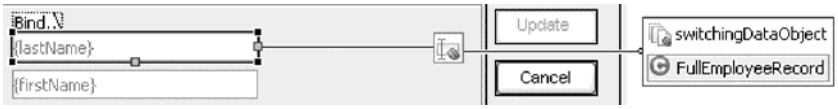
即使每个职员详细信息字段已绑定至 selectedEmployeeRecord，现在您也需要将它们绑定至 switchingDataObject。在绑定这些字段之后，根据您是正在修改现有的职员记录还是在添加新的职员记录，可以在字段的数据对象之间动态地进行切换。

对于职员详细信息部分中的每个字段，请完成下列步骤:

1. 选择字段并单击**绑定**选项卡。
2. 在“字段数据绑定”对话框中，选择 switchingDataObject。前面已将这些字段绑定至 selectedEmployeeRecord。



3. 确保该字段仍然绑定至正确的数据对象属性并单击**确定**。如果在设计视图上选择该字段，则可以看到绑定程序线现在指向 switchingDataObject。



定义更新方式和切换方式的标志和方法

以下 updateMode() 方法将检查是否将方式标志设置为新建，然后相应地更改应用程序行为。缺省情况下，布尔值标志 isNewMode 设置为 false，updateMode() 方法会启用职员表、启用过滤器字段并将更新按钮上的文本设置为“更新”。如果 isNewMode 设置为 true，则禁用职员表并清除任何选择、禁用过滤器字段以及将“更新”按钮上的文本设置为“添加”。

将以下代码添加至 DirectoryApp.java 类中最后一个右花括号的前面:

```
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");
   }
}
```

将 actionPerformed 事件添加至“新建”按钮

在此步骤中，添加单击**新建**按钮时的事件代码。该事件告知 switchingDataObject 使用 newEmployeeRecord 数据对象、将方式标志设置为“新建”和运行在先前步骤中添加的 updateMode() 方法。

1. 在设计视图中，右键单击**新建**按键，并选择**事件 → actionPerformed**。以下代码是在 getNewButton() 方法中生成的:

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

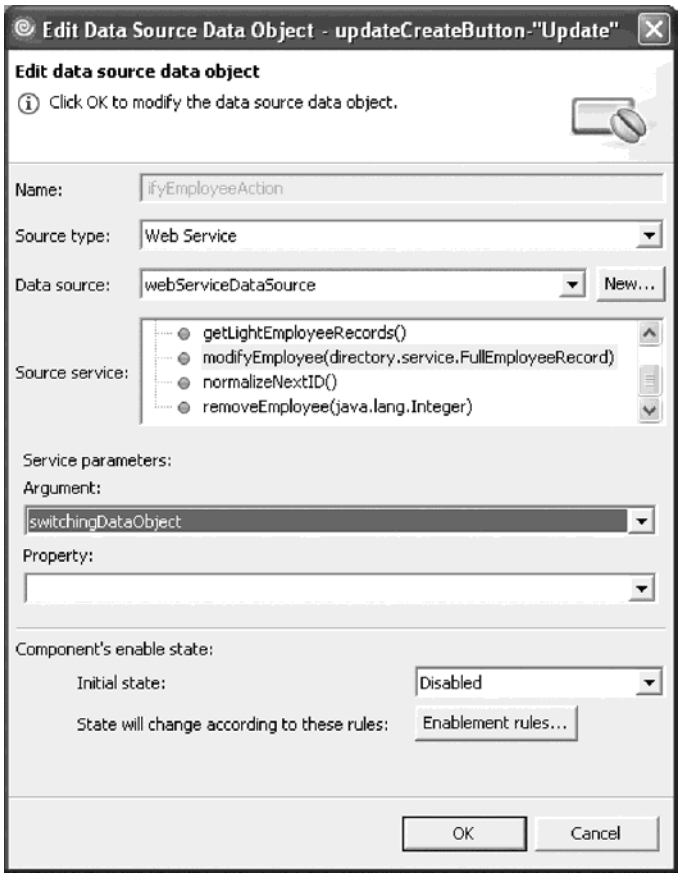
2. 将此生成的存根替换为下面的代码:

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

重新绑定“更新”按钮

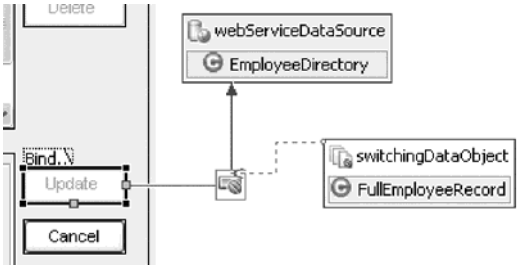
在上一课中，对**更新**按钮进行了编程以对 Web Service 使用 modifyEmployee 方法。该操作是作为 SwingDataServiceAction 实现的。SwingDataServiceAction 的其中一个属性是源对象，它充当服务的自变量。修改操作的源对象当前设置为 selectedEmployeeRecord。为了对该按钮进行编程以控制更新和添加，将重新配置该按钮的操作以将 switchingDataObject 用作 modifyEmployee 服务的自变量。

1. 在设计视图中，选择**更新**按钮。注意，粉红的虚线箭头显示 selectedEmployeeRecord 是服务调用的自变量。
2. 单击**更新**按钮上的**绑定**选项卡。
3. 在**自变量**字段中，选择 switchingDataObject。



4. 单击**确定**。

现在，注意该按钮的操作已配置为将 switchingDataObject 用作 modifyEmployee 方法的自变量:



将事件添加至“更新”按钮的绑定程序以复位方式

在单击**更新**按钮并对 Web Service 完成操作之后，您想要将应用程序返回到它的缺省方式和行为。为此，将事件侦听器添加在该按钮的操作绑定程序上，该绑定程序将在执行更新或添加之后更新方式并刷新表。

对“更新”按钮的 getModifyEmployeeAction() 方法添加以下代码:

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

课程要点

现在，当您运行“我的公司目录”应用程序时，可以单击**新建**按钮并添加新的职员记录。

课程 2.7: 对“取消”按钮行为进行编程

当使用应用程序时，如果您决定不提交对职员记录所做的任何更改，则您想要可以很容易地撤销这些更改。换句话说，您需要能够取消并清除这些字段，以便可以重新开始。要添加此功能，对**取消**按钮设置一些 actionPerformed 事件。

以下列表描述**取消**按钮的必需行为:

- 如果在新建方式下单击**取消**按钮，则应用程序会从新建方式还原。
- 如果修改职员记录时单击**取消**按钮，则已更改的任何值都会还原为最初的值。

要将 actionPerformed 事件添加至**取消**按钮以执行必需的行为:

1. 在设计视图中，右键单击**取消**按钮，然后选择**事件 → actionPerformed**。以下代码是在 getCancelButton() 方法中生成的:

   ```
   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. 将生成的事件存根替换为以下代码:

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

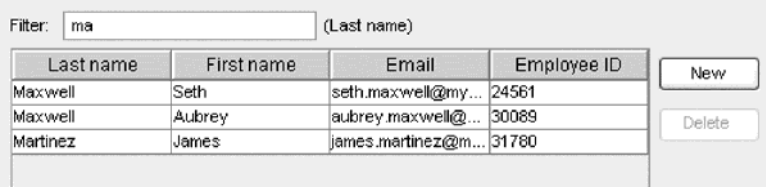
课程要点

在本课中，您已学习如何使用 actionPerformed 事件对**取消**按钮进行编程。

课程 2.8: 对职员表设置过滤器

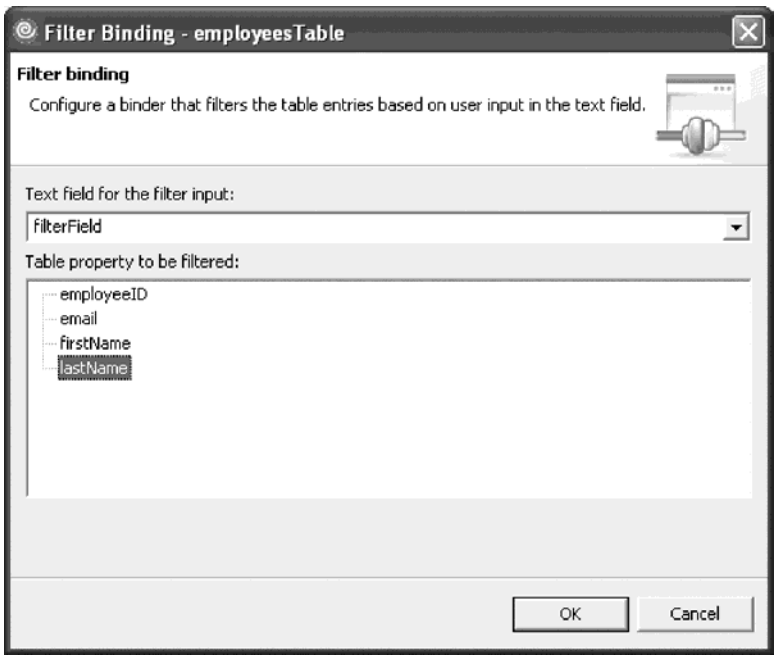
可以使用“文本过滤器绑定程序”来过滤职员表的内容。过滤器从文本字段获取输入并根据表中的特定属性或列对表进行过滤。

在应用程序中，将使用在**过滤器**字段中输入的字符来按职员的姓进行过滤。如果在**过滤器**字段中输入的精确值出现在职员记录的姓中，则表中将显示该职员记录。



要为表创建过滤器:

1. 选择 employeesTable 的绑定程序图标并选择**过滤器绑定属性**。“过滤器绑定”对话框打开。
2. 在**过滤器输入的文本字段**列表中，选择 **filterField**。
3. 在**要过滤的表属性**列表中，选择 **lastName**。



4. 单击**确定**。

将生成新的 SwingPropertyFilter。表的绑定程序的过滤器属性设置为使用新的过滤器。新的过滤器配置为将**过滤器**字段用于它的输入并根据表的 lastName 属性进行过滤。

课程要点

在本课中，您已学习如何为表设置过滤器。

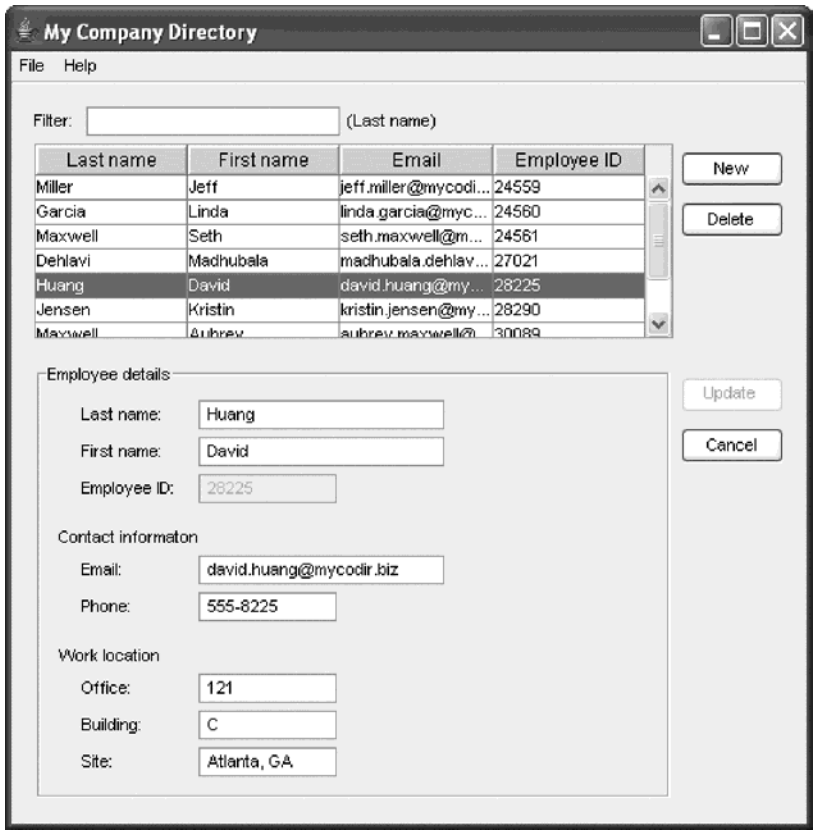
现在，当您运行“我的公司目录”应用程序时，可以在**过滤器**字段中输入字符，并且将对表进行过滤以显示姓中包含输入字符的那些行。

祝贺您！您已完成“我的公司目录”应用程序。

总结: 构建使用 Web Service 的富 Java 客户机

祝贺您！您已学习如何使用 Java 可视编辑器来构建“我的公司目录”应用程序，该应用程序是一种连接至样本 Web Service 以维护职员目录的富 Java 客户机。

查看已完成产品的图片:



学习的课程

使用了可视编辑器来完成图形用户界面，并使用 GridBagLayout 来排列职员表。然后，将表、字段和按钮绑定至相应的数据对象和数据源，以使应用程序使用生成的 Web Service Java 代理。还完成了一些复杂的编码工作来使应用程序正常运行，以使它易于使用且直观。而且，学习了如何将企业应用程序安装在 WebSphere Application Server V6.0 上和部署 Web Service。

最重要的是，您全面地了解了 Java 可视编辑器在处理数据方面提供的功能强大的绑定程序类。现在，您可以自己开始实验，对绑定程序进行一些新的有创意的使用。

现在，您应该能够执行下列任务:

- 使用 Java 可视编辑器来在 GridBagLayout 中布局组件。
- 将可视类作为 Java bean 来运行。
- 将 Java 应用程序的可视组件绑定至 Web Service 返回的方法和数据对象。
- 将事件添加至可视组件。

其他资源

导入“我的目录”应用程序的已完成版本

此项目包含已完成的应用程序、带有绑定程序类的 jve.generated 包以及为 WebSphere Application Server V6.1 配置的 Web Service Java 客户机。如果您没有完成本教程就导入此已完成项目，则可能需要配置 Java 构建路径变量。它需要指向您的 WebSphere V6.1 Web Service 瘦客户机 JAR 文件。

提示: 如果在导入期间未指定另一项目名称，则将覆盖 MyDirectory 项目内容。