



Компоновка расширенного клиента Java, применяющего Web-службу

Содержание

Создание расширенного клиента Java, использующего Web-службу 1

Введение: создание расширенного клиента Java,
использующего Web-службу 1

Модуль 1: Проектирование клиентского GUI в
визуальном редакторе 3

Урок 1.1: Установка проекта Java. 3

Урок 1.2: Добавление и создание макета таблицы
сотрудников 4

Урок 1.3: Запуск визуального класса. 8

Модуль 2: Создание визуальных компонентов для
Web-службы 10

Урок 2.1: Установка и развертывание Web-службы 10

Урок 2.2: Связывание таблицы сотрудников с
источником данных Web-службы 12

Урок 2.3: Связывание полей подробной
информации с выделением таблицы 18

Урок 2.4: Связывание кнопки Обновить с
редактором связей действий 22

Урок 2.5: Связывание кнопки Удалить с окном
подтверждения 24

Урок 2.6: Настройка действий и привязок для
добавления нового сотрудника 26

Урок 2.7: Программирование поведения кнопки
Отмена 31

Урок 2.8: Настройка фильтра в таблице
сотрудников 32

Обзор: создание расширенного клиента Java,
использующего Web-службу 34

Создание расширенного клиента Java, использующего Web-службу

В этом учебнике описано создание расширенного клиента Java, подключающегося к Web-службе, с помощью визуального редактора для Java. Клиент, который создается в этом учебнике, называется My Company Directory.

My Company Directory - это приложение на Java, применяемое для поддержки каталога сотрудников компании. Приложение подключается к примеру Web-службы, предоставляющей методы для создания, извлечения, обновления и удаления записей о сотрудниках

Клиент создается визуально в визуальном редакторе для Java с помощью компонентов Swing. В визуальном редакторе для Java предусмотрен набор классов вспомогательного приложения (источники данных, объекты данных и редакторы связей) или возможность подключения к Web-службе и работы с ней. Web-служба развернута локально в собственной установке IBM WebSphere Application Server версии 6.0, а инструменты позволяют создавать проху Java на основе файла на языке описания Web-служб (WSDL).

См. готовый продукт

Цель обучения

Учебник состоит из следующих уроков:

- Проектирование и макетирование пользовательского интерфейса с помощью визуального редактора для Java
- Связывание элементов интерфейса с объектами данных и Web-службой

Требуемое время

2 часа 15 минут

Информация, связанная с данной



Версия в формате PDF

Учебник: Здравствуй, мир Java

Введение: создание расширенного клиента Java, использующего Web-службу

Графический пользовательский интерфейс (GUI) для клиента большей частью создан предварительно с помощью компонентов Swing. В первом модуле вы завершите макетирование важнейших компонентов GUI с помощью визуального редактора для Java. Во втором модуле вам предстоит связать компоненты GUI с источником данных Web-службы, службами и объектами, возвращаемыми источником данных. При создании приложения в визуальном редакторе для Java используются источники данных, объекты данных и редакторы связей, которые являются экземплярами вспомогательных классов, создаваемых визуальным редактором для Java и применяемых в приложении.

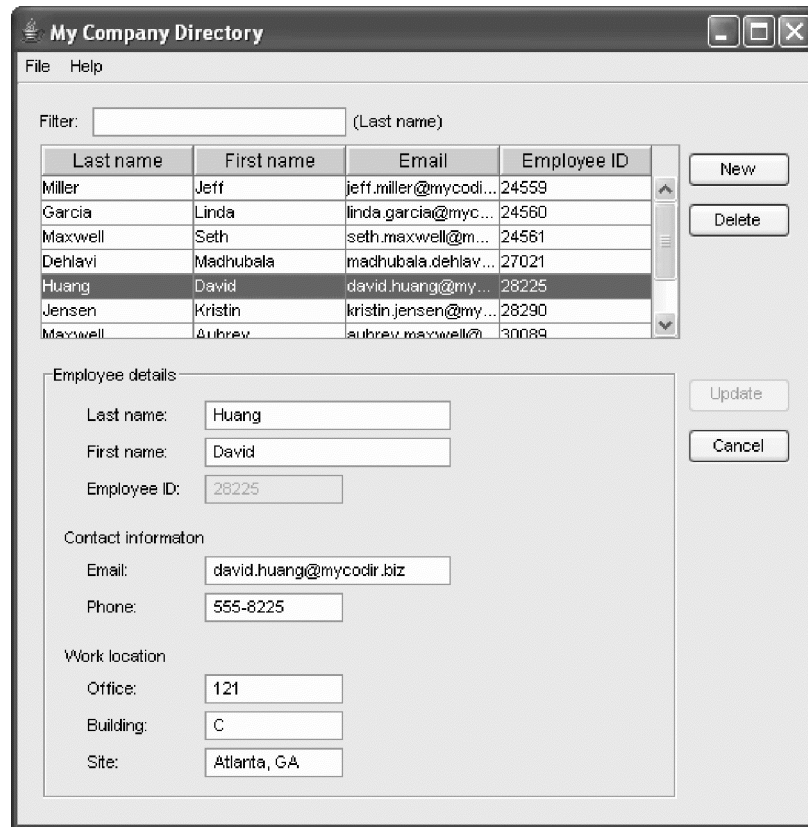


Рисунок заверщенного продукта:

Цель обучения

Учебник состоит из следующих уроков:

- Проектирование и макетирование пользовательского интерфейса с помощью визуального редактора для Java
- Связывание элементов интерфейса с объектами данных и Web-службой

Требуемое время

На изучение всего учебника потребуется приблизительно 2 часа 30 минут.

Системные требования

- WebSphere Application Server v6.1. Можно использовать сервер, установленный вместе с продуктом, или автономный экземпляр. Сценарий в этом учебнике предлагает развернуть пример Web-службы на локальном сервере WebSphere Application Server.

Пример Web-службы можно запустить и на других серверах, однако тестирование этого учебника выполнялось только на серверах WebSphere Application Server v6.0 и v6.1.

Предварительные требования

Необходимо ознакомиться со следующими концепциями:

- Основы разработки на Java
- Основные принципы Web-служб
- Основные навыки работы с рабочей средой, такие как работа с проектами и навигация по проекциям и панелям

Модуль 1: Проектирование клиентского GUI в визуальном редакторе

В этом модуле описано добавление визуального компонента в приложение с помощью визуального редактора для Java, а также последующее визуальное макетирование и настройка ограничений расположения. В последнем уроке этого модуля показан запуск файла Java для проверки его работы в качестве настоящего приложения.

Напоминание: Перед тем как приступить к этому модулю, необходимо ознакомиться с предварительными требованиями, перечисленными во введении.

Цель обучения

В результате изучения этого модуля вы должны освоите следующие действия:

- Добавление и макетирование JTable в интерфейсе Java
- Проверку работы класса путем его запуска

Требуемое время

Изучение этого модуля займет примерно 15 минут.

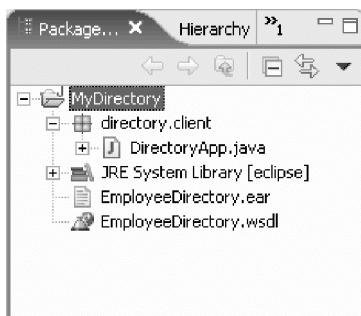
Урок 1.1: Установка проекта Java

В ходе этого урока в рабочую область будет импортирован проект MyDirectory. В состав проекта входит класс Java, а также дополнительные файлы, которые потребуются позднее.

Поскольку в этом учебнике основное внимание уделяется связыванию визуальных компонентов с Web-службой, большая часть GUI Java для приложения "My Company Directory" была разработана предварительно.

Проект MyDirectory - это основной проект, с которым предстоит работать в рамках этого учебника. Она содержит файл DirectoryApp.java. В этом файле Java находится главное приложение на Java, которое вы создаете. Вместе с учебником в архиве ZIP поставляется несколько версий проекта MyDirectory: по одной начальной версии каждого модуля и законченная версия проекта.

1. Импортируйте проект MyDirectory.
2. Убедитесь, что в панели Структура пакетов проекции Java проект MyDirectory выглядит следующим образом:



Полученные знания и навыки

В этом уроке был импортирован пример проекта MyDirectory - начальная точка этого учебника.

В состав проекта MyDirectory входят следующие ресурсы:

- DirectoryApp.java: файл Java, в котором находится приложение, разрабатываемое в этом учебнике. Файл DirectoryApp.java находится в группе Java с именем directory.client.

- EmployeeDirectory.ear: приложение J2EE, которое содержит пример Web-службы. В модуле 2 эта Web-служба будет развертываться в локальной установке WebSphere Application Server версии 6.0.
- EmployeeDirectory.wsdl: файл XML, где на языке описания Web-служб (WSDL) описана Web-служба, которую планируется развертывать. В модуле 2 с помощью этого файла WSDL будет создаваться прокси Java для приложения.

Урок 1.2: Добавление и создание макета таблицы сотрудников

В этом уроке вам предстоит добавлять в приложение JScrollPane и JTable с помощью визуального редактора для Java. В следующих упражнениях нужно будет запрограммировать в JTable получение данных от Web-службы, которая возвращает список всех сотрудников в каталоге компании.

После добавления JTable нужно будет с помощью панели эскиза в визуальном редакторе для Java настроить макет JTable в соответствии со следующими спецификациями:

- Растянуть JTable на три ячейки по горизонтали и на две по вертикали
- Добавить левый отступ в 15 пикселей
- Переименовать JTable в employeesTable.

Показать

Открытие файла DirectoryApp.java в визуальном редакторе Java

Для того чтобы открыть файл DirectoryApp.java в визуальном редакторе для Java, выполните следующие действия:

1. В панели Структура пакетов проекции Java разверните проект MyDirectory и пакет directory.client.
2. Щелкните правой кнопкой мыши на файле DirectoryApp.java и выберите **Открыть с помощью → Визуальный редактор**. Визуальный редактор для Java загрузит класс Java и отобразит эскиз области графического холста.

Совет:

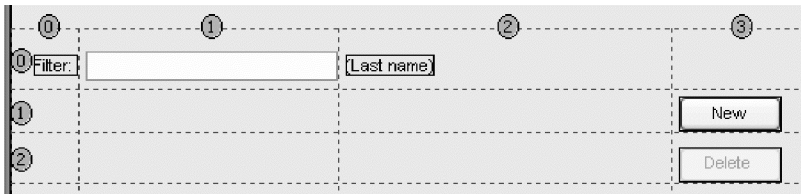
- Для того чтобы изменить внешний вид, используемый визуальным редактором для Java, перейдите к пункту меню **Окно → Параметры → Java → Визуальный редактор** и задайте внешний вид Swing. Параметры вступят в силу при следующей загрузке класса. В этом учебнике используется внешний вид Windows.
- Для того чтобы сделать визуальный редактор редактором для всех файлов Java по умолчанию, выберите **Окно → Параметры** и задайте нужные параметры на странице **Рабочая среда → Типы файлов**.

Добавление JTable в панель JScrollPane

Главное окно DirectoryApp.java использует JFrame с JPanel в качестве основной правой панели. JPanel в нашем приложении называется jContentPane. jContentPane использует тип администратора макетов с именем GridBagLayout. GridBagLayout - это мощная схема макета на основе сетки из ячеек, которые можно заполнять визуальными компонентами. Визуальный редактор для Java позволяет удобно работать с GridBagLayout, показывая границы сетки. Кроме того, при размещении новых компонентов на сетке он отображает маркеры размещения и снабжает компоненты управляющими элементами для изменения размера или перемещения в GridBagLayout.

Для того чтобы добавить таблицу сотрудников (javax.swing.JTable) в пользовательский интерфейс DirectoryApp.java, выполните следующие действия:

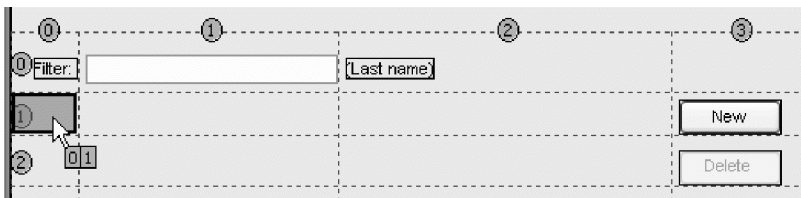
1. Щелкните правой кнопкой мыши на jContentPane в панели эскиза или панели объектов JavaBean и выберите **Показать сетку**. Граница сетки будет показана красным пунктиром, а синие круги с цифрами будут обозначать номера строк и столбцов. Например, обратите внимание, что кнопка **Создать** занимает ячейку в строке 1 (ось y) и столбце 3 (ось x).



- В палитре визуального редактора для Java выберите компонент Swing **JTable** в **JScrollPane** , который по категории находится в лотке палитры **Компоненты Swing**.

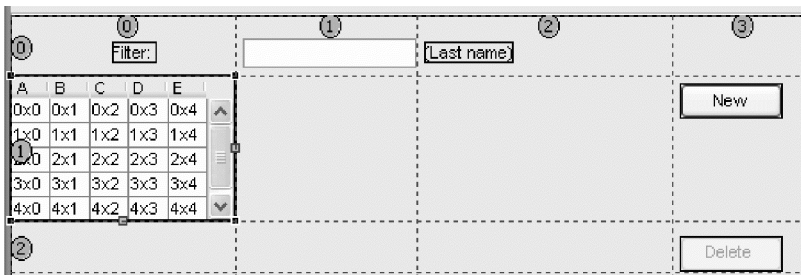
Совет: По умолчанию палитра сворачивается в правой части области эскиза. Можно изменять размеры палитры и перемещать ее.

- Поместите указатель мыши на ячейке сетки в столбце 0, строке 1:



- При перемещении указателя мыши по сетке будут отображаться два пронумерованных квадрата, обозначающие координаты x и y в сетке, в которых находится указатель.
- Если навести указатель мыши непосредственно на границу сетки, можно создать новые столбцы и строки, а номера существующих столбцов и строк будут изменены. В этом случае на данное поведение указывают желтые квадраты на указателе мыши, желтые линии между сетками, а также желтые этикетки столбца и строки. Кроме того, в них отображается воздействие, к которому приведет размещение.

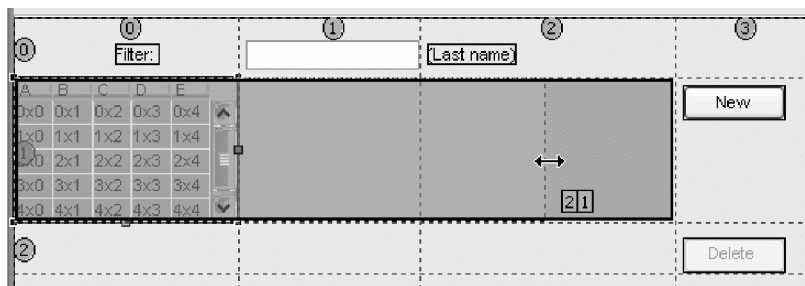
- Щелкните левой кнопкой мыши, чтобы поместить JScrollPane и JTable в ячейку в столбце 0 и строке 1:



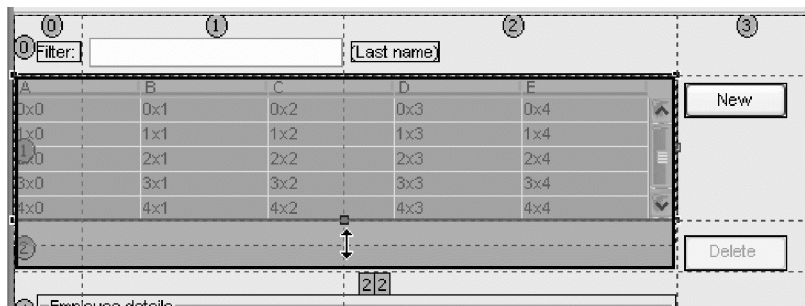
Растягивание JScrollPane и JTable на несколько столбцов сетки

Нужно растянуть JScrollPane (и ее потомок JTable) на три столбца и две строки для того, чтобы получить более удобные отступы и сделать более удобным изменение размеров. Для того чтобы растянуть столбцы и строки, выполните следующие действия:

- Выберите JScrollPane в области эскиза или в области объектов JavaBean (объект должен быть еще выделен, поскольку вы его добавили только что). Обратите внимание на зеленые квадратики в нижней правой части JScrollPane. С помощью этих управляющих элементов для изменения размера можно перетаскивать JScrollPane, чтобы объект растянулся на несколько столбцов и строк.
- Щелкните на зеленом управляющем элементе в правой части JScrollPane и удерживайте левую кнопку мыши нажатой.
- Перетащите указатель мыши вправо до тех пор, пока не отобразится расположение в столбце 2 и строке 1. Кроме того, темно-серая тень будет показывать ячейки, которые займет компонент при отпускании кнопки мыши.



4. Отпустите кнопку мыши. Теперь JScrollPane займет три столбца.
5. Аналогичным образом растяните JScrollPane до строки 2 с помощью нижнего управляющего элемента:



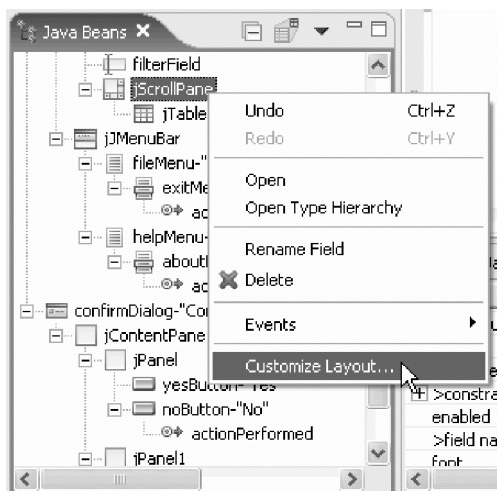
Настройка отступов JScrollPane в GridBag

Кроме того, в администраторе GridBagLayout предусмотрена функция, позволяющая задавать различные ограничения для дальнейшей настройки макета. Например, можно задавать следующие ограничения:

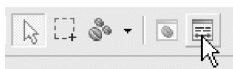
- **метка:** У компонента может быть ориентация на метку внутри ячейки. Это влияет на перемещение компонента, когда пользователь будет изменять размер окна приложения. Например, компонент можно привязать к левому верхнему углу, к центральной верхней части, к центру или к правому нижнему углу.
- **заполнение:** Компонент может занимать все доступное пространство в пределах своих ячеек по горизонтали, по вертикали, либо по обоим осям одновременно.
- **отступы:** Для компонента можно задать отступы сверху, снизу, слева и справа для того, чтобы был промежуток между компонентом и краем сетки.

Для того чтобы настроить метку, заполнение и отступы для JScrollPane, выполните следующие действия:

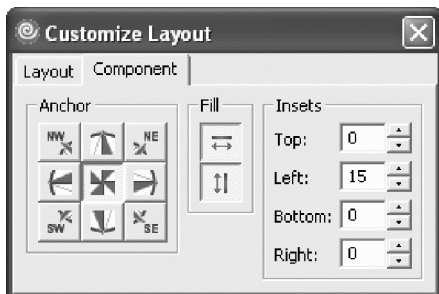
1. Щелкните правой кнопкой мыши на JScrollPane в панели эскиза или в панели объектов JavaBean и выберите **Настройка макета**.



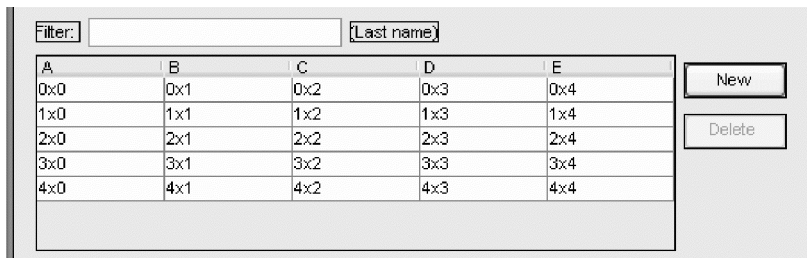
Совет: Окно Настройка макета может оставаться открытым при выборе и изменении макета для различных компонентов. Окно Настройка макета можно открыть в любой момент. Для этого нужно нажать кнопку Настройка макета в строке меню:



2. Убедитесь, что во вкладке Компонент окна Настройка макета нажата кнопка Центр метки.
3. Убедитесь, что нажаты кнопки **Заполнение по горизонтали** и **Заполнение по вертикали**.
4. Добавьте левый отступ в 15 пикселей для того, чтобы отступ слева от JScrollPane был одинаковым с остальными компонентами приложения.



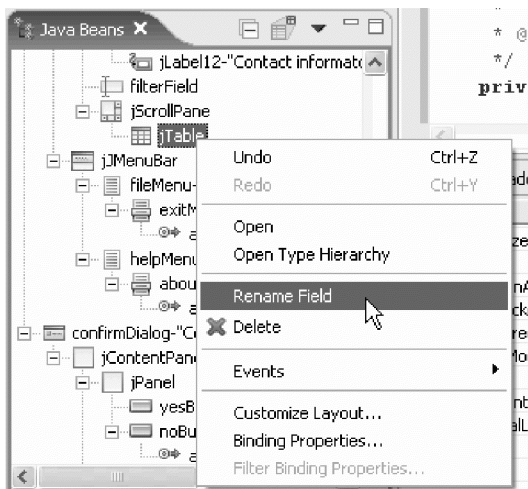
Теперь, например, таблица выравнивается по метке **Фильтр**.



Переименование.JTable на полезное значение и настройка выделения в таблице одной строки

Поскольку впоследствии вам придется работать с таблицей, рекомендуется переименовать экземпляр.JTable и метод.get. Для того чтобы переименовать таблицу:

1. В панели объектов JavaBeans щелкните правой кнопкой мыши на компоненте jTable и выберите во всплывающем меню **Переименовать поле**.



2. Введите `employeesTable` и нажмите кнопку **OK**. Теперь таблица `JTable` называется `employeesTable`, а метод для создания ее экземпляра называется `getEmployeesTable`.
3. Разрешите в таблице выделение только одной строки:
 - a. Выберите `employeesTable` в панели эскиза.
 - b. В панели Свойства выберите свойство `selectionMode` property и задайте значение `SINGLE_SELECTION`.

Property	Value
preferredSize	375,80
rowHeight	16
rowSelectionAllowed	true
selectionBackground	49,106,197
selectionForeground	Color:white
selectionMode	SINGLE_SELECTION
showGrid	
showHorizontalLines	true
showVerticalLines	true
toolTipText	
visible	true

- c. Сохраните файл `DirectoryApp.java`.

Полученные знания и навыки

В этом уроке вы научились добавлять таблицу в пользовательский интерфейс с помощью визуального редактора. После этого вы научились настраивать ее макет, расположение и отступы.

Урок 1.3: Запуск визуального класса

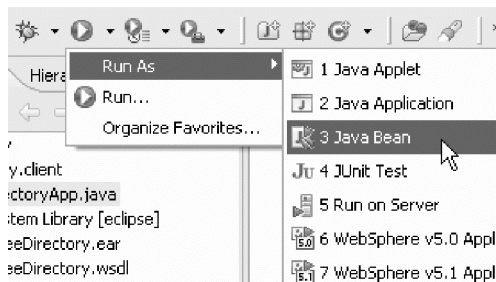
Теперь все готово к запуску приложения на Java для того, чтобы просмотреть его вид. Приложение можно запустить просто и быстро из рабочей среды или визуального редактора. Эти действия можно повторить в любое время для тестирования фактического внешнего вида и поведения класса, какими они будут во время выполнения.

Показать

В визуальном редакторе для Java предусмотрен модуль запуска объектов `JavaBean`, способный запускать классы без метода `main()`. При запуске визуального класса этот модуль запускает приложение в отдельной виртуальной машине (VM). При запуске визуального класса в качестве приложения на Java модуль попытается выполнить метод `main()` класса. В этом учебнике в приложении есть метод `main()`, вызывающий и отображающий `JFrame DirectoryApp`, таким образом, его можно запускать либо как приложение, либо как объект `JavaBean`.

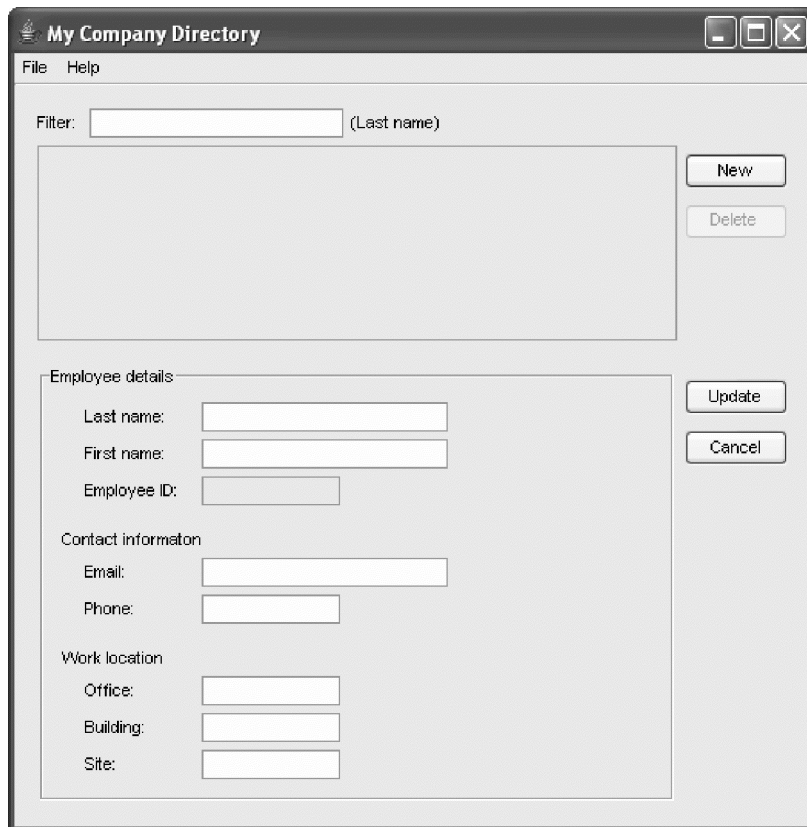
Для того чтобы запустить файл `DirectoryApp.java` как объект `JavaBean`:

1. Убедитесь, что файл `DirectoryApp.java` открыт в визуальном редакторе для Java.
2. В строке меню выберите **Запустить** → **Запустить как** → **Объект JavaBean**.



Совет: Приложение откроется на рабочем столе и будет иметь вид Swing, заданный с помощью параметров визуального редактора (**Окно** → **Параметры** → **Java** → **Визуальный редактор**). Кроме того, можно выбрать **Запустить** → **Запустить**, и определить внешний вид для конкретной конфигурации запуска при запуске этого объекта `JavaBean`. Если это приложение запускается не как объект `JavaBean`, а как

приложение, то оно также будет иметь внешний вид Windows, поскольку это задано в методе main(). На иллюстрациях учебника показан внешний вид Windows.



Полученные знания и навыки

Поскольку интерфейс спроектирован пока только внешне, но еще не запрограммирована связь данных или функции событий, то с его помощью нельзя выполнять никаких действий. Однако можно увидеть основной макет и вид, в котором приложение увидит пользователь. Попробуйте нажимать кнопки, но ничего не произойдет. Впрочем, меню Файл и Справка уже реализованы. Можно проверить их работу и изучить код Java, на котором они реализованы с помощью событий actionPerformed.

Пройденные уроки

В этом модуле вы ознакомились с проектированием интерфейса расширенного клиента с помощью визуального редактора для Java. Впрочем, для того чтобы клиент действительно заработал, кроме проектирования его внешнего вида нужно сделать гораздо больше. Как правило, потребуется реализовать поведение событий или другую логику и, в данном случае, связывание визуальных элементов с каким-либо источником данных.

В этом модуле вы научились выполнять следующие задачи:

- Импортировать проект Java с помощью импорта обмена проектами
- Добавлять таблицу JTable в панель JScrollPane визуального класса
- Визуально макетировать таблицу расширенного клиента с помощью диспетчера GridBagLayout
- Запускать приложение для того, чтобы увидеть фактический внешний вид расширенного клиента Java

В следующем модуле, модуле 2: Связывание визуальных компонентов с Web-службой. На основе простого интерфейса My Company Directory вы создадите мощный расширенный клиент, способный создавать,

извлекать, обновлять и удалять записи о сотрудниках в каталоге компании, путем подключения к Web-службе.

Модуль 2: Создание визуальных компонентов для Web-службы

В этом модуле описано связывание визуальных элементов My Company Directory (кнопок, таблицы сотрудников, полей и других действий) с Web-службой. В Web-службе предусмотрены полноценные возможности для создания, извлечения, обновления и удаления сотрудников из примера каталога.

Цель обучения

В результате изучения этого модуля вы должны освоите следующие действия:

- Связывание таблицы с источником данных Web-службы
- Связывание полей с объектами
- Программирование действий для кнопок

Изучение этого модуля займет примерно **2 часа**.

Урок 2.1: Установка и развертывание Web-службы

В этом уроке вам предстоит установить пример файла приложения J2EE (EAR) на сервере WebSphere Application Server v6.1 и развернуть Web-службу EmployeeDirectory. С помощью этой службы приложение будет создавать, считывать, обновлять и удалять записи о сотрудниках.

Подготовительные действия: с помощью *одной из следующих опций* убедитесь, что проект MyDirectory находится в нужной исходной точке:

- Выполните “Модуль 1: Проектирование клиентского GUI в визуальном редакторе” на стр. 3.
или
- Импортируйте проект MyDirectory в исходную точку Module 2

Совет: Если в ходе импорта не указать другой проект, то содержимое проекта MyDirectory будет заменено.

Проект Java MyDirectory содержит файл EmployeeDirectory.ear. Приложение J2EE EmployeeDirectory, которое находится в файле EAR, устанавливается с помощью административной консоли WebSphere. При установке приложения будет также развернута Web-служба, поставляемая вместе с приложением. Завершенное приложение My Company Directory использует именно эту развернутую Web-службу.

Для того чтобы установить пример приложения EmployeeDirectory и развернуть Web-службу на сервере WebSphere Application Server v6.1, выполните следующие действия:

1. В рабочей среде запустите экземпляр сервера приложений. Это можно сделать несколькими способами, но ниже описан запуск именно из рабочей области:
 - a. Откройте панель Серверы. Для того чтобы добавить панель Серверы в проекцию Java, выберите **Окно → Показать панель → Другую и выбрать сервер → Серверы**.
 - b. На панели Серверы показан список установленных и настроенных серверов.
 - c. Щелкните правой кнопкой мыши на сервере и выберите **Запустить**. Сервер будет запущен, когда в панели Серверы отобразится состояние сервера **Запущен**, либо в консоли появится сообщение **Сервер сервер1 готов для электронного бизнеса**. Теперь можно запустить административную консоль.

Примечание: Если в панели Серверы нет экземпляра сервера, создайте новый сервер:

- a. Щелкните правой кнопкой мыши в панели Серверы и выберите **Создать → Сервер**.
- b. С помощью мастера Создать сервер добавьте новый экземпляр WebSphere Application Server v6.1.

2. Запустите административную консоль WebSphere. Ее можно запускать различными способами, но ниже описан ее запуск именно с помощью рабочей среды:
 - a. В панели Серверы щелкните правой кнопкой мыши на сервере, который только что был запущен, и выберите **Запустить административную консоль**. В окне браузера откроется административная консоль WebSphere.
 - b. Введите ИД пользователя и нажмите **Войти**. Откроется начальная страница административной консоли. ИД пользователя, введенный при входе, используется только для отслеживания изменений данных конфигурации сервера, которые вносит каждый конкретный пользователь.
3. С помощью административной консоли установите приложение J2EE EmployeeDirectory.ear, которое находится в проекте MyDirectory. При установке приложений административная консоль действует аналогично мастеру установки, где пользователь перемещается со страницы на страницу с помощью кнопки **Далее** до тех пор, пока не будут установлены все опции. Для того чтобы установить пример приложения J2EE с Web-службой, выполните следующие действия:
 - a. В левой части административной консоли разверните опцию **Меню приложений** и выберите **Установить новое приложение**.
 - b. Выберите **Локальная файловая система**, а в поле **Укажите путь** введите полный путь к файлу EmployeeDirectory.ear, который находится в проекте MyDirectory. Совет: для того чтобы получить полный путь, щелкните правой кнопкой мыши на файле EmployeeDirectory.ear в панели Структура пакетов и выберите **Свойства**. На странице свойств будет расположение файла, которое можно скопировать и вставить в поле **Укажите путь**.
 - c. Нажимайте кнопку **Далее** до тех пор, пока не откроется страница **Выберите опции установки**.
 - d. Выберите **Развертывание Web-служб**.
 - e. Нажимайте кнопку **Далее** до тех пор, пока не появится страница **Обзор**, затем нажмите кнопку **Готово**.
 - f. Щелкните на ссылке **Сохранить в главном файле конфигурации**, когда появится приглашение применить изменения, внесенные в локальную конфигурацию. Просмотрите изменения и нажмите кнопку **Сохранить**.
4. Запуск приложения EmployeeDirectory с помощью административной консоли:
 - a. Выберите **Приложения** → **Приложения организации**. Приложение EmployeeDirectory будет указано в списке как приложение, установленное на сервере, но его состояние будет "Остановлено".

Start Stop Install Uninstall Update Rollout Update Remove File Export Export DDL		
✓ 📁 🔍 🔄		
Select	Name ↕	Status ↕
☐	DefaultApplication	⇒
☒	EmployeeDirectory	✖
☐	Query	⇒
☐	SchedulerCalendars	⇒
☐	filetransfer	⇒
☐	ivtApp	⇒
Total 6		

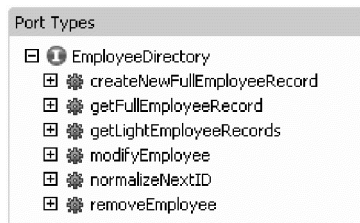
- b. Отметьте переключатель напротив EmployeeDirectory и нажмите кнопку **Запустить**. Появится сообщение об успешном запуске приложения EmployeeDirectory, значок состояния изменится на зеленую стрелку.

Теперь приложение EmployeeDirectory запущено на локальном хосте на порте 9080 и может подключаться к Web-службе. После того как вы пройдете весь учебник, можно вернуться в административную консоль, остановить приложение EmployeeDirectory и удалить его.

В файле EmployeeDirectory.wsdl, который находится в проекте MyDirectory (он должен быть открыт по умолчанию в графическом редакторе для WSDL) можно просмотреть только что развернутую Web-службу.

Если файл WSDL не открывается в редакторе WSDL, то, возможно, функция разработки Web-служб отключена в рабочей среде. Функции рабочей среды задаются в Параметрах (**Окно → Параметры → Рабочая среда → Функции**).

На приведенном ниже рисунке из редактора WSDL показаны операции, доступные для службы EmployeeDirectory:



С помощью редактора WSDL можно изучить любую операцию и соответствующие сообщения-запросы, а также сообщения-ответы. Благодаря этому можно разобраться в работе Web-службы и понять ее применение в оставшихся упражнениях.

Урок 2.2: Связывание таблицы сотрудников с источником данных Web-службы

Приложение My Company Directory отображает список всех текущих записей о сотрудниках в каталоге. Записи отображаются в таблице JTable (employeesTable) в сортируемых столбцах: фамилия, имя, адрес электронной почты и ИД сотрудника. Для получения записей в таблицу необходимо связать таблицу employeesTable с объектом данных, возвращаемым примером источника данных Web-службы.

Показать

Обзор объектов данных, источников данных и редакторов связей

Для передачи локального объекта данных в таблицу employeesTable в приложение нужно будет добавить источник данных с помощью визуального редактора. Источник данных подключается к примеру проху Web-службы и обнаруживает служебные методы, доступные для приложения. Затем вы выбираете служебный метод getLightEmployeeRecord, доступный в источнике данных. Наконец, вы связываете таблицу employeesTable в приложении с полями, возвращаемыми в объекте данных строки (lightEmployeeRecordRows).

Все эти источники и объекты данных можно создавать быстро и легко с помощью встроенных классов редактора связей в визуальном редакторе для Java. В визуальном редакторе предусмотрен набор стандартных интерфейсов и классов, которые создаются в проекте при связывании визуальных компонентов с фабриками данных. Классы редактора связей создаются по умолчанию в пакете с именем jve.generated. Классы редактора связей поставляются в визуальном редакторе в качестве стандартной реализации, которую можно в дальнейшем настраивать и расширять в зависимости от потребностей приложения. В этом учебнике демонстрируется мощность и гибкость использования классов редактора связей по умолчанию даже на самом простом уровне.

Важное замечание: Перед тем как приступить к этому упражнению настоятельно рекомендуется ознакомиться со следующими разделами справки. В них приведена подробная информация о функциях и логике объектов данных, источников данных и редакторов связей, поставляемых с визуальным редактором для Java:

- Обзор редакторов связей данных
- Справочник по API редакторов связей

В рамках этого учебника в приложении предстоит использовать источник данных Web-службы, несколько видов объектов данных и несколько видов редакторов связей. При добавлении экземпляров этих объектов в

приложение визуальный редактор добавляет необходимые классы в пакет `jve.generated` в проекте, где можно расширять, изменять или переписывать логику связывания данных. В визуальном редакторе для Java предусмотрена визуальная поддержка связывания объектов путем отображения объектов данных, источников данных, а также редакторов связей, используемых в приложении, в произвольной области панели эскиза. Визуальный редактор отображает линии между визуальными компонентами и объектами данных, а также источниками данных. Таким образом он показывает текущие связи для любого выбранного объекта.

На следующей диаграмме приведен простой обзор взаимодействия визуальных компонентов, редакторов связей, объектов данных и источников данных. Приложение, которое компонуется в данном учебнике показывает немного более комплексное и творческое использование редакторов связей. На данной диаграмме четко не выражены редакторы связей, объекты данных и источники данных примера приложения.

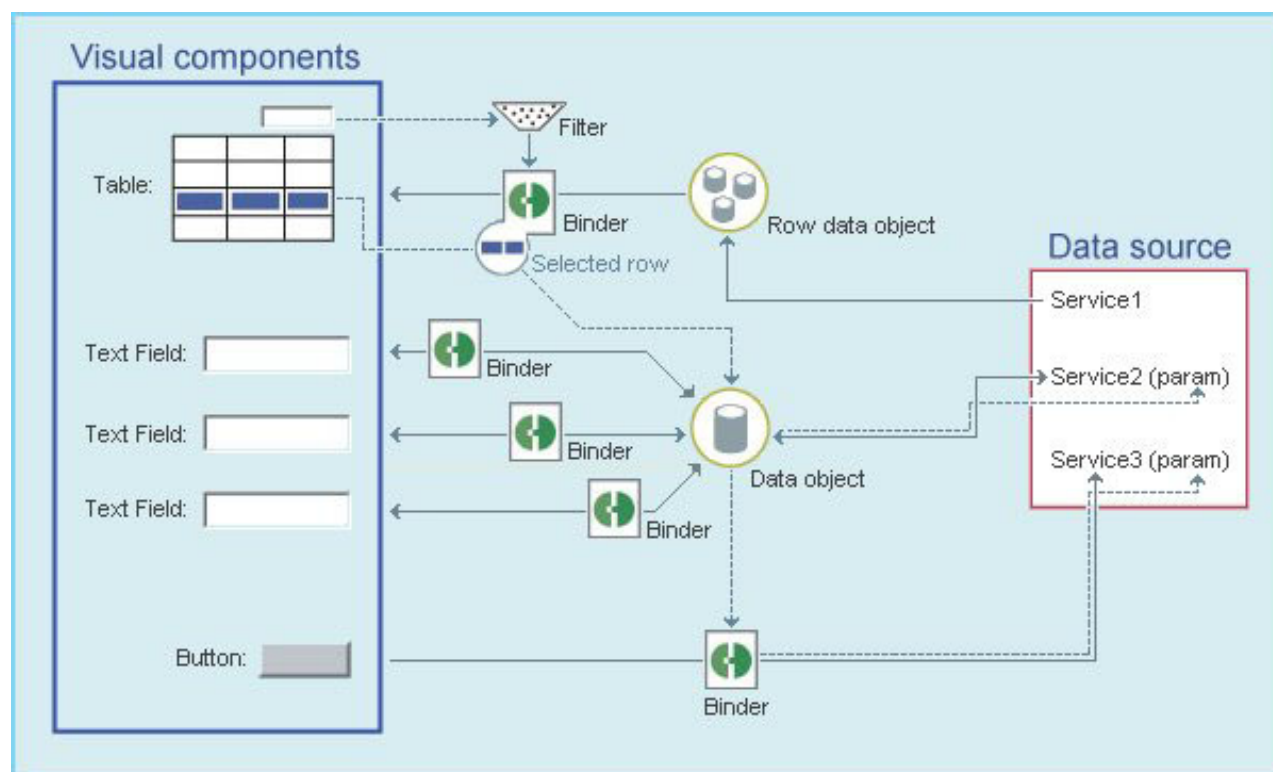


Рисунок 1. На этой диаграмме показан пример взаимодействия между визуальными компонентами, редакторами связей, объектами данных и источниками данных

На рисунке 1 каждому визуальному компоненту соответствует свой собственный редактор связей, связывающий его с объектом данных или с источником данных, если это кнопка. Редакторы связей для текстовых полей, связывают поле с отдельным свойством объекта данных. Строчный объект данных, также как и объект данных на данной диаграмме получают данные из непосредственных вызовов службы источника данных. Объект данных для текстовых полей использует значение ключа из выбранной строки таблицы в качестве аргумента при вызове службы `Service2`, которая возвращает полную запись, предположительно включающую более подробную информацию о выбранной строке таблицы. Эта полная запись, в свою очередь, используется в качестве аргумента для редактора связей действия кнопок при вызове службы `Service3`, которая, как метод, может обновить значения, введенные в поля. Более подробная информация об объектах данных, редакторах связей данных и источниках данных приведена в ссылках, указанных выше.

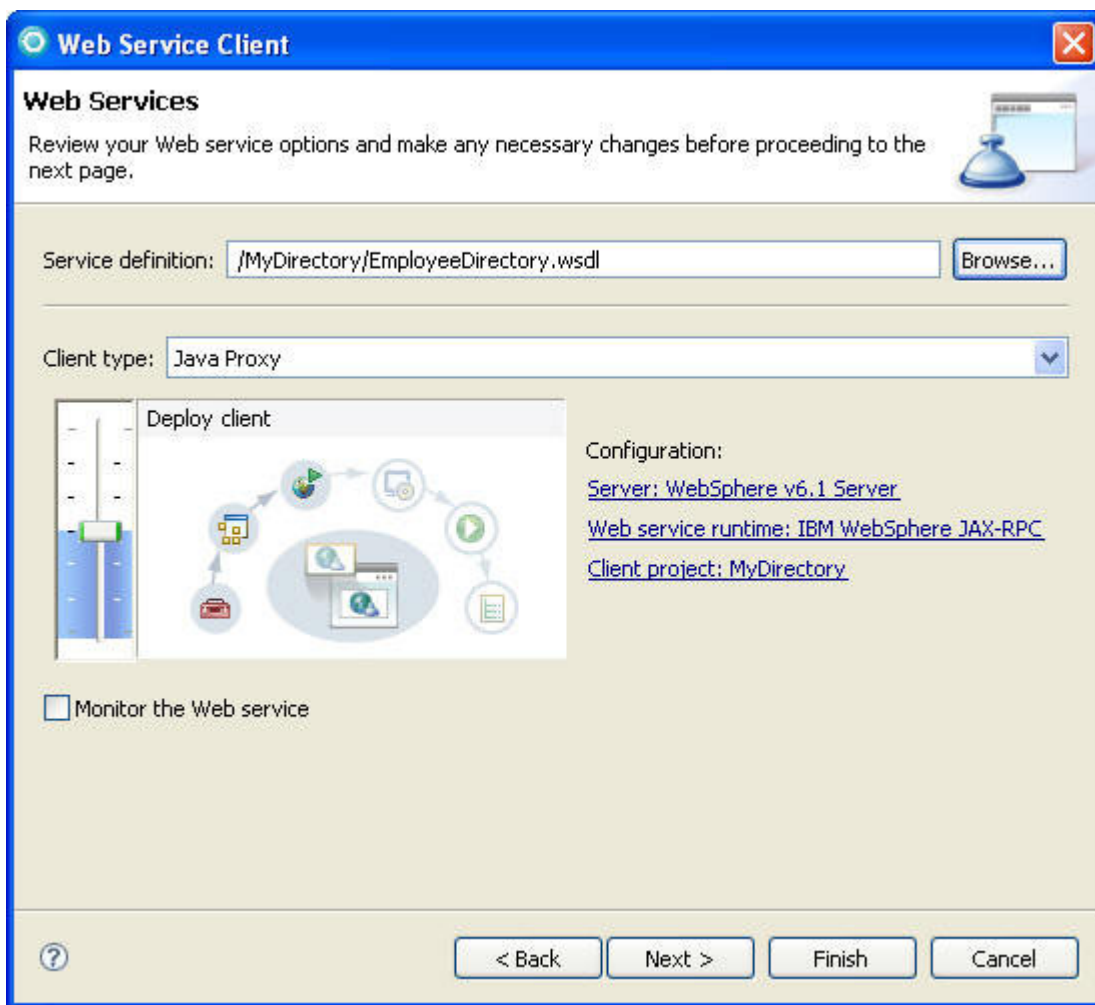
Создание Proxy Java Web-службы в проекте с помощью поставляемого файла WSDL

Для работы с Web-службой, работающей на сервере, приложению на Java требуется проху Java, либо клиент для взаимодействия с ним. Файл WSDL позволяет создавать проху Java в проекте Java с помощью мастера клиента Web-службы. Проект MyDirectory содержит файл EmployeeDirectory.wsdl, который используется для создания этого проху. После создания проху Java можно создать источник данных, представляющий Web-службу, и начать связывание визуальных компонентов.

Важное замечание: В файле WSDL, используемом в этом упражнении, подразумевается, что Web-служба развернута в локальной установке WebSphere Application Server и для локального хоста используется порт по умолчанию (<http://localhost:9080>). Если файл EAR был развернут иначе, то перед тем как продолжать работу, необходимо соответствующим образом отредактировать файл WSDL.

Для того чтобы создать проху Java Web-службы в проекте:

1. В главном меню выберите **Файл → Создать → Другой** и выберите мастер **Web-службы → Клиент Web-службы**. Если категория Web-службы не отображается, выберите **Показать все мастера**.
2. С помощью мастера задайте параметры клиента Web-службы:
 - a. В поле **Определение службы** укажите файл WSDL, поставляемый вместе с проектом MyDirectory: `/MyDirectory/EmployeeDirectory.wsdl`
 - b. В поле **Тип клиента** выберите **Proxy Java**.
 - c. Переместите ползунок в положение **Развернуть клиента**.
 - d. Убедитесь, что для применяемого сервера правильным образом задана среда выполнения Web-службы и сервер. Тестирование этого учебника выполнялось для WebSphere v6.0 и WebSphere v6.1 со средой выполнения IBM WebSphere JAX-RPC.
 - e. Убедитесь, что проект MyDirectory выбран для вывода клиента Proxy Java.



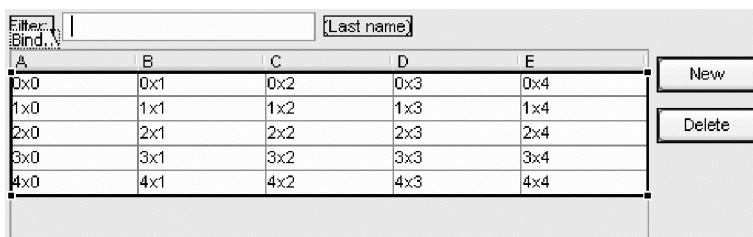
3. Нажмите кнопку **Готово**. Мастер клиента Web-службы создает Proxy Java в новом пакете (directory.service) проекта.

Связывание таблицы employeesTable со строковым объектом данных, возвращаемым Web-службой

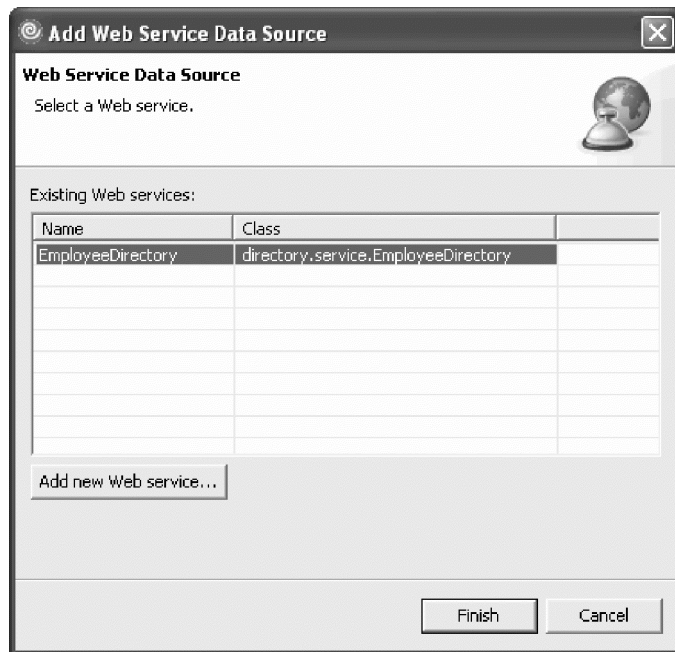
Поскольку таблица employeesTable - это первый визуальный компонент, который вы связываете в этом приложении, нужно создать источник данных, который указывает на тот же проxy Web-службы, который только что был добавлен в проект. При связывании других визуальных компонентов в следующих упражнениях этот источник данных будет использован повторно. В этом шаге добавляется источник данных Web-службы и объект данных lightEmployeeRecordRows.

Для связывания таблицы сотрудников:

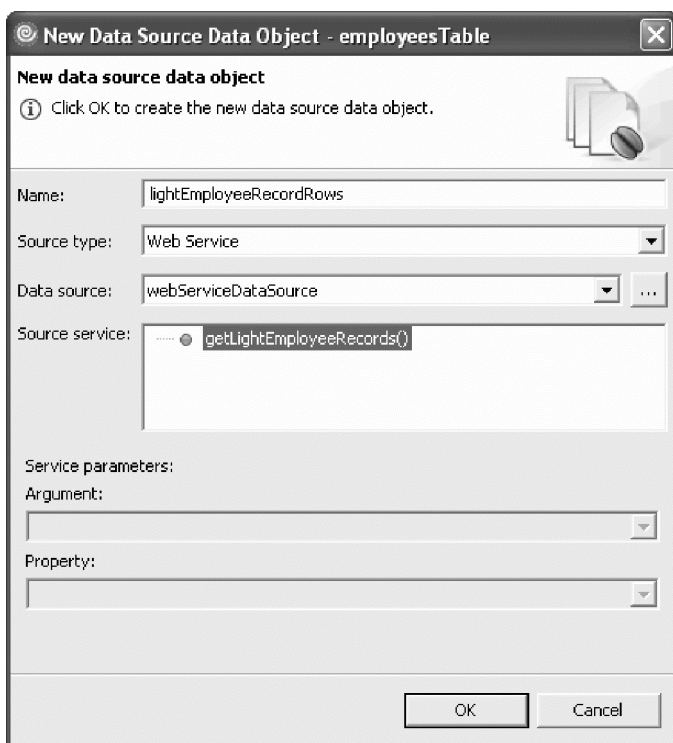
1. В панели объектов JavaBean или в панели эскиза выберите employeesTable. (Убедитесь, что вы не выбрали ее родитель - JScrollPane.) В верхней части таблицы employeesTable в области эскиза появится небольшая надпись **Связать**.



- Щелкните на вкладке **Связать** в employeesTable. Кроме того, можно щелкнуть правой кнопкой мыши на employeesTable и выбрать **Свойства связывания**.
- Поскольку в приложении нет объектов данных, нужно добавить новый. Выберите **Создать объект Источник данных**.
- В поле **Исходный тип** выберите **Web-служба**.
- Поскольку источник данных Web-службы еще не добавлен в приложение, его необходимо добавить сейчас. Нажмите кнопку ... рядом с полем **Источник данных**. Откроется окно Добавить источник данных Web-службы, которое попытается найти доступные клиенты или ргоху Web-служб в проекте.
- Выберите Web-службу EmployeeDirectory и нажмите кнопку Готово. Новый источник данных будет добавлен в файл DirectoryApp.java.

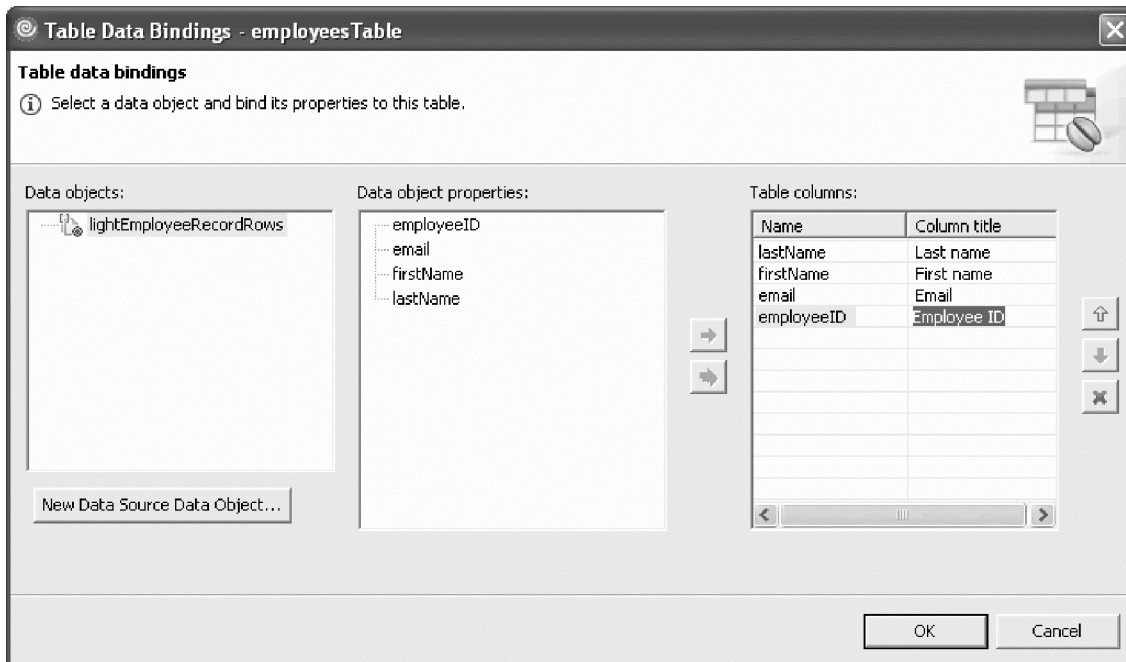


- В окне Создать объект данных источника данных выберите getLightEmployeeRecords() в поле исходной службы и примите значение имени по умолчанию для нового объекта данных: lightEmployeeRecordRows. Для этого служебного метода параметры не нужны. Нажмите кнопку ОК. Новый объект данных будет создан и отобразится в произвольной области панели эскиза.



Совет: Поскольку вы связываете таблицу, в окне Создать объект данных источника данных отображаются только те службы, которые возвращают строковые объекты данных. В этом случае метод `getLightEmployeeRecords()` - это единственная доступная служба, возвращающая массив объектов.

8. В окне Привязки данных таблицы выберите объект данных `lightEmployeeRecordRows`.
9. Теперь нужно выбрать свойства объекта данных `lightEmployeeRecordRows`, которые должны отображаться в `employeesTable`:



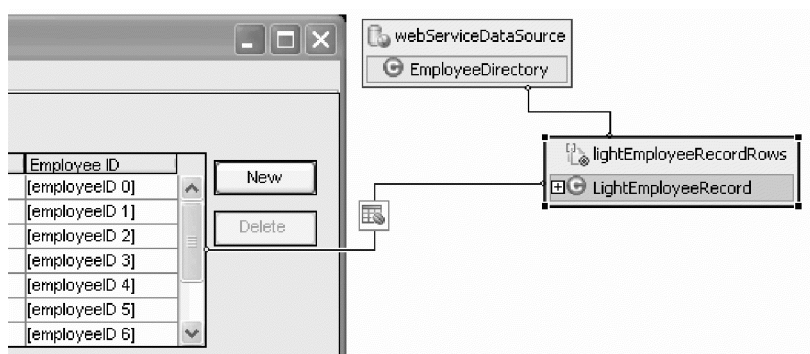
- a. Нажмите кнопку  с двойной стрелкой для добавления всех свойств объекта в список **Столбцы таблицы**.

- b. С помощью стрелок вверх и вниз расположите столбцы в следующем порядке сверху вниз: lastName, firstName, email, employeeID
- c. Переименуйте заголовки столбцов: Фамилия, Имя, Электронная почта, ИД сотрудника

Совет: После того как вы закончили связывание таблицы, вы в любой момент можете вернуться к свойствам связывания, переименовать и изменить порядок столбцов.

- d. Нажмите кнопку **ОК**.

Теперь таблица employeesTable связана с объектом данных lightEmployeeRecordRows с помощью JRowTableBinder. Если щелкнуть на объекте данных lightEmployeeRecordRows в произвольной области, визуальный редактор отобразит линию от объекта данных к таблице. На этой линии редактор связей JRowTableBinder будет представлен значком . Еще одна линия обозначает, что объект данных использует в качестве источника данных webServiceDataSource.



Полученные знания и навыки

Отметьте изменения в проекте и приложении. В этом уроке вы добавляли источник данных Web-службы, строковый объект данных, а также редактор связей, связывающий таблицу employeesTable со строковым объектом данных.

Проверьте в новом пакете (jve.generated), созданном в проекте, наличие всех классов редактора связей, созданных визуальным редактором для Java. Кроме того, обратите внимание на пакет (directory.service), в котором находится прокси Java для Web-службы. Опишите пройденное в этом уроке и подведите итог.



Теперь при запуске приложения My Company Directory Web-служба заполняет таблицу существующими записями о сотрудниках.

Урок 2.3: Связывание полей подробной информации с выделением таблицы

В предыдущем упражнении вы связывали таблицу employeesTable с объектом данных lightEmployeeRecordRows, возвращаемым службой getLightEmployeeRecords() в Web-службе. Теперь нужно заполнить поля подробной информации на основе сотрудника, выделенного в таблице.

Для получения дополнительных сведений для каждого выделенного сотрудника понадобится еще один объект данных. Будет добавлен объект selectedEmployeeRecord, который возвращает служба getFullEmployeeRecord(). Эта служба получает ИД выбранного сотрудника в таблице в качестве параметра и загружает дополнительные сведения о сотруднике, включая его номер телефона и место работы.

Объект `JRowTableBinder`, который использовался при связывании таблицы с объектом данных строки, упрощает этот шаг. `JRowTableBinder` помещает выбранный элемент в таблицу как отдельный объект данных, который можно использовать в качестве параметра для метода `getFullEmployeeRecord(java.lang.Integer)`. Затем можно легко связать все текстовые поля с соответствующими свойствами в объекте данных `selectedEmployeeRecord`.

Дополнительная информация о Web-службе: Web-служба состоит из двух служб, получающих подробные сведения о каждом сотруднике. В таблице перечислены все сотрудники, и в ней отображается только подмножество данных. Таким образом, при выборе одного сотрудника можно получить остальную информацию только по выбранному сотруднику. Если бы Web-служба отправляла все данные по каждому сотруднику, когда их запрашивает таблица, поток данных в сети был бы слишком велик и от этого падало бы быстродействие приложения.

Например, если бы запись сотрудника содержала бы фотографию или вложение, то для получения полного списка сотрудников не было бы необходимости получать также и все фотографии. Таким образом, служба `getLightEmployeeRecord` используется для заполнения таблицы, а `getFullEmployeeRecord` получает целиком запись сотрудника, выбранного в таблице.

Связывание поля Фамилия

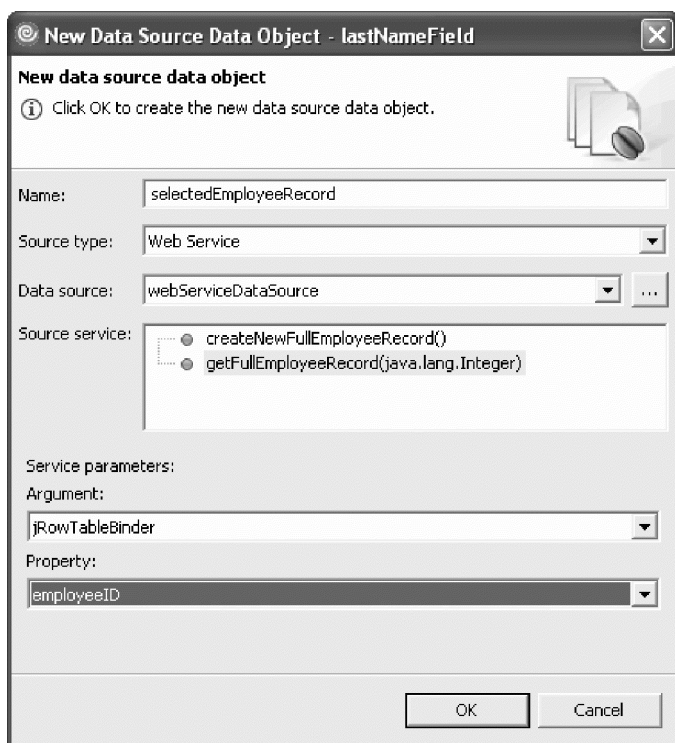
На этом шаге мы рассмотрим связывание поля **Фамилия** со свойством `lastName` в объекте данных `selectedEmployeeRecord`:

1. В панели объектов `JavaBean` или в панели эскиза выберите `TextField` для фамилии (`lastNameField`). В области эскиза на текстовом поле появится вкладка **Связать**.

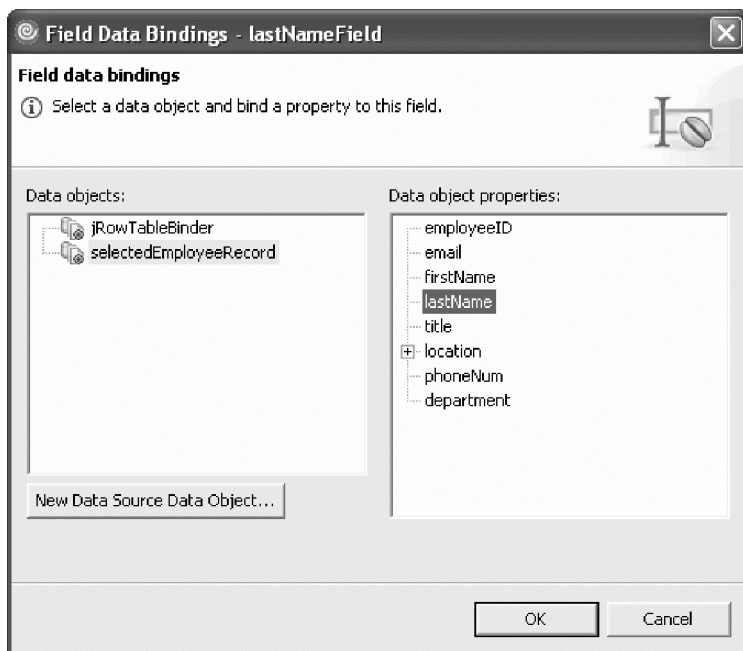


2. Щелкните на вкладке **Связать**. Откроется окно связывания данных поля.
3. Выберите **Создать объект Источник данных**. Хотя существующий объект данных `jRowTableBinder` возвращает правильную фамилию, он не возвращает запись сотрудника целиком. Необходимо создать новый объект данных, который будет представлять ее полностью.
4. Убедитесь, что в поле **Тип источника** выбрано значение **Web-служба**, а в поле **Источник данных** - **webServiceDataSource**.
5. В списке **Исходная служба** выберите `getFullEmployeeRecord(java.lang.Integer)`. В окне создания нового объекта **Источник данных** будет список служб, которые возвращают объекты данных, совместимые с текстовым полем.
6. В поле **Имя** введите `selectedEmployeeRecord`.
7. В поле **Аргумент** выберите `jRowTableBinder`, в поле **Свойство** - `employeeID`. ИД сотрудника для выбранной строки задается в качестве аргумента для служебного метода `getFullEmployeeRecord()`.

Примечание: В качестве аргумента `getFullEmployeeRecord(java.lang.Integer)` должно быть задано целое число. Для получения полной записи сотрудника используется ИД сотрудника, выбранного в таблице в настоящий момент. При связывании таблицы визуальный редактор автоматически создал редактор связей `jRowTableBinder`, который прослушивает текущий выбор в таблице сотрудников. В качестве целочисленного параметра используется `employeeID` выбранной строки в `jRowTableBinder`.



8. Нажмите кнопку **OK**.
9. Убедитесь, что в окне Привязки данных полей в списке **Объекты данных** выбрано значение `selectedEmployeeRecord`. Обратите внимание, что для объекта данных `selectedEmployeeRecord` доступно больше свойств, чем для объекта данных `jRowTableBinder`.
10. В списке **Свойства объекта данных** выберите свойство `lastName`.



11. Нажмите кнопку **OK**. Теперь поле фамилии в приложении связано со свойством `lastName` объекта данных `selectedEmployeeRecord`, который возвращается методом `getFullEmployeeRecord()`.
Новый объект данных с именем `selectedEmployeeRecord` создается и добавляется в приложение. Визуальное представление объекта данных добавляется в произвольную область панели эскиза, как показано на приведенном ниже рисунке:



Теперь при выборе поля lastName в области эскиза привязка к selectedEmployeeRecord будет показана линией. Значок текстового редактора связей посередине линии представляет редактор связей SwingTextComponentBinder, который используется для связывания. При выборе линии или значка, представляющего редактор связей в области эскиза, можно просмотреть свойства редактора связей в панели Свойства.

Связывание остальных полей подробной информации

Для того чтобы связать каждое из оставшихся полей подробной информации о сотруднике, повторите действия с полем фамилии, но не нужно добавлять объект данных. Поскольку объект данных selectedEmployeeRecord уже добавлен, можно просто связать каждое поле с соответствующим свойством в объекте данных selectedEmployeeRecord.

Для того чтобы связать поля, выполните следующие действия для каждого поля в разделе Подробные сведения о сотрудниках в приложении:

1. Выберите поле в панели эскиза и щелкните на вкладке **Связать**.
2. В окне Привязки данных полей из списка **Объекты данных** выберите selectedEmployeeRecord.
3. Из списка **Свойства объектов данных** выберите соответствующее свойство для связываемого поля. В следующей схеме показаны свойства, с каждым из которых следует связать текстовое поле:

Поле	Свойство в объекте данных selectedEmployeeRecord
lastNameField	lastName
firstNameField	firstName
idField	employeeID
emailField	email
phoneField	phoneNum
officeField	location.office
buildingField	location.building
siteField	location.site

4. Нажмите кнопку **ОК**.

Когда вы закончите связывание текстовых полей, область эскиза должна будет выглядеть как на приведенном ниже рисунке:

Employee details

Last name: {lastName}

First name: {firstName}

Employee ID: {employeeID}

Contact information

Email: {email}

Phone: {phoneNum}

Work location

Office: {location.office}

Building: {location.building}

Site: {location.site}

Как сделать поле ИД сотрудника доступным только для чтения

Поле ИД сотрудника неактивно, поскольку его свойство доступности для редактирования задано в значение `false`. Впрочем, по умолчанию редактор связей для текстового поля изменяет состояние на доступное для редактирования, как только в объекте данных появляется значение. Такое поведение редактора связей можно отключить, и поле будет все время оставаться доступным только для чтения.

Для того чтобы редактор связей не переключал свойство доступности для редактирования автоматически, выполните следующие действия:

1. Выберите поле ИД сотрудника. В области эскиза появится линия со значком , представляющая редактор связей для поля.
2. Щелкните на значке  редактора связей для поля ИД сотрудника.
3. В панели Свойства измените свойство `autoEditable` на **false**. Нажмите **Enter**.

Полученные знания и навыки

Теперь, когда при запуске приложения вы будете выбирать пользователя из таблицы, в полях подробной информации будут отображаться подробные сведения о каждом сотруднике.

Урок 2.4: Связывание кнопки Обновить с редактором связей действий

В визуальном редакторе для Java предусмотрены редакторы связей для действий, позволяющие вызывать службу для источника данных при нажатии кнопки. Например, при нажатии кнопки Обновить, приложение должно запускать метод `modifyEmployee()` в Web-службе с изменениями, введенными в полях подробных сведений. В этом уроке вы научитесь связывать кнопку Обновить с редактором связей событий.

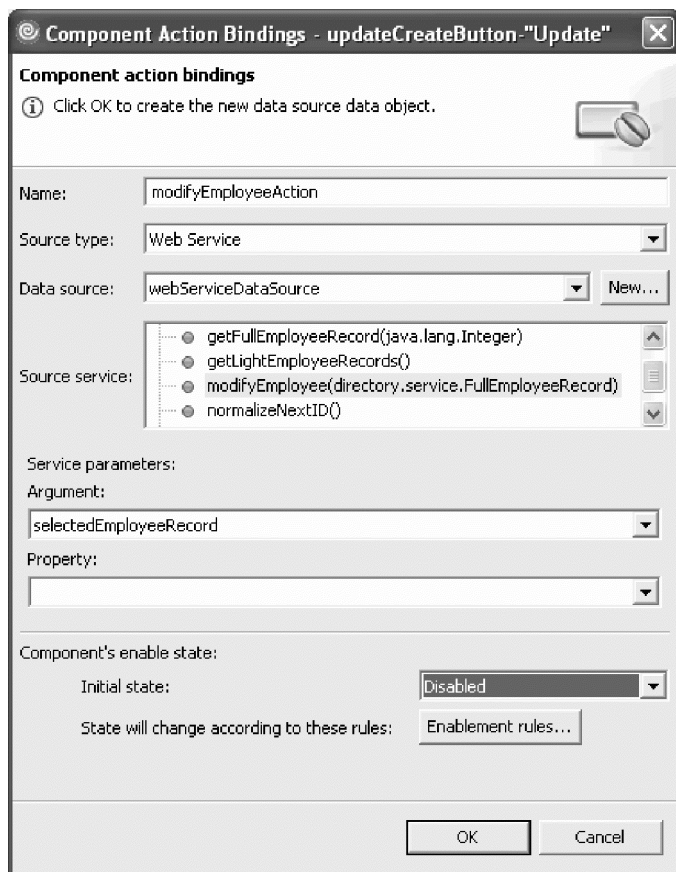
Для связывания кнопки Обновить:

1. Выберите кнопку **Обновить** в области эскиза и щелкните на вкладке **Связать**. Откроется окно связывания действий компонента.

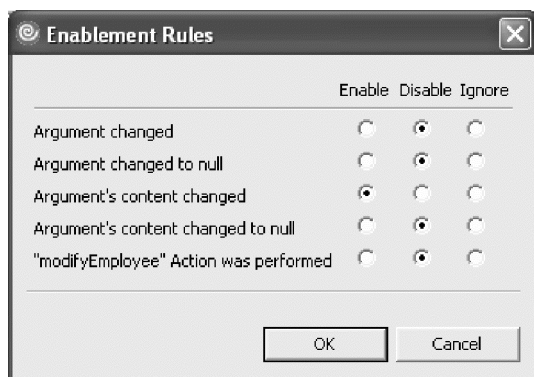


2. В поле **Исходный тип** выберите **Web-служба**.
3. В поле **Источник данных** выберите **webServiceDataSource**.
4. В списке **Исходная служба** выберите **modifyEmployee(directory.service.FullEmployeeRecord)**.
5. В поле **Имя** автоматически отобразится **modifyEmployeeAction**. Примите значение по умолчанию.
6. В поле **Аргумент** выберите **selectedEmployeeRecord**.
7. Поскольку метод `modifyEmployee()` получает в качестве аргумента полную запись о пользователе, поле **Свойство** следует оставить пустым.

8. Задайте Первоначальное состояние кнопки - Отключено.



9. Для того чтобы определить, каким образом будет изменяться состояние кнопки, выберите **Правила активации**. Укажите, что кнопка должна быть активна только при изменении содержимого аргумента и неактивна во всех остальных случаях. Нажмите кнопку **ОК**.



Это означает, что кнопка **Обновить** будет отключена до тех пор, пока не изменится содержимое selectedEmployeeRecord. Другими словами, как только в одном из полей подробных сведений будет введено новое значение, связанное с selectedEmployeeRecord, редактор связей активирует кнопку. При выборе новой записи или нажатии кнопки **Обновить**, эта кнопка снова станет неактивной.

10. Нажмите кнопку **ОК**.

Для кнопки **Обновить** добавляется новый редактор связей SwingDataServiceAction. При выборе кнопки в области эскиза, визуальный редактор отобразит линию, обозначающую связь кнопки с источником данных Web-службы. Розовая пунктирная стрелка указывает от объекта selectedEmployeeRecord на линию. Эта стрелка показывает, что объект selectedEmployeeRecord является аргументом для вызова службы.

Полученные знания и навыки

Теперь при запуске приложения можно обновлять запись сотрудника.

Выберите сотрудника в таблице и измените фамилию. Как только фамилия будет изменена, кнопка **Обновить** станет активной. При нажатии на кнопку **Обновить** будет вызвана служба `modifyEmployee`, а данные о сотруднике обновятся. Новая фамилия отобразится в таблице сотрудников.

Урок 2.5: Связывание кнопки Удалить с окном подтверждения

В этом упражнении мы будем программировать удаление записи о сотруднике в приложении My Company Directory.

В следующем списке приведены функции, которые нужно будет использовать в приложении:

- При выборе сотрудника в таблице активируется кнопка **Удалить**.
- При нажатии кнопки **Удалить** открывается окно Подтвердить удаление, которое запрашивает подтверждение удаления.
- При нажатии кнопки **Да** в окне Подтвердить удаление, запись о сотруднике будет удалена, окно закроется, а список сотрудников будет обновлен.
- При нажатии кнопки **Нет** удаление будет отменено, а окно Подтвердить удаление закроется.

Программирование активности и неактивности кнопки Удалить в зависимости от того, выделена ли строка в таблице

Для того чтобы запрограммировать активацию или деактивацию кнопки Удалить, добавьте в таблицу обработчик событий, который будет активировать кнопку при выборе строки.

1. Выберите таблицу `employeesTable` в панели объектов JavaBean. В панели исходного кода выделится следующая строка исходного кода:

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

2. Сразу после этой строки добавьте в таблицу `employeesTable` новый обработчик событий `ListSelectionListener` и событие `valueChanged`:

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

3. После добавления этих строк в редакторе исходного кода они будут помечены как строки с ошибками до тех пор, пока не будут импортированы `ListSelectListener` и `ListSelectionEvent`. Для добавления нужных объектов выберите в главном меню **Исходный код** → **Организовать импорт**. В раздел импорта текущего класса будут добавлены следующие строки:

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

Теперь при выборе строки в таблице кнопка **Удалить** будет становиться активной.

Программирование окна Подтвердить удаление при нажатии кнопки Удалить

Добавьте событие `actionPerformed` в кнопку Удалить и запрограммируйте в нем открытие окна Подтвердить удаление.

1. Щелкните правой кнопкой мыши на кнопке **Удалить** и выберите **События** → **actionPerformed**. В метод `getDeleteButton()` будет добавлена следующая заготовка события:

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

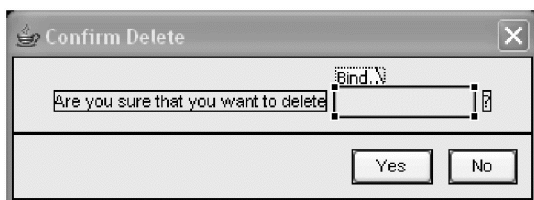

2. Замените эту заготовку приведенным ниже кодом, который отображает окно Подтвердить удаление при нажатии кнопки:

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

Связывание текстового поля в окне Подтвердить удаление

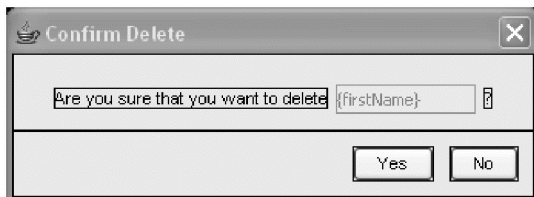
Свяжите текстовое поле в окне Подтвердить удаление таким образом, чтобы в нем отображалось имя удаляемого пользователя.

1. В панели объектов JavaBean или в области эскиза выберите текстовое поле employeeToDeleteField и щелкните на вкладке **Связать**.



2. В окне Привязки данных полей выберите объект данных selectedEmployeeRecord и поле firstName, затем нажмите кнопку **ОК**.

Текстовое поле будет связано со столбцом firstName выбранной строки в таблице employeesTable.



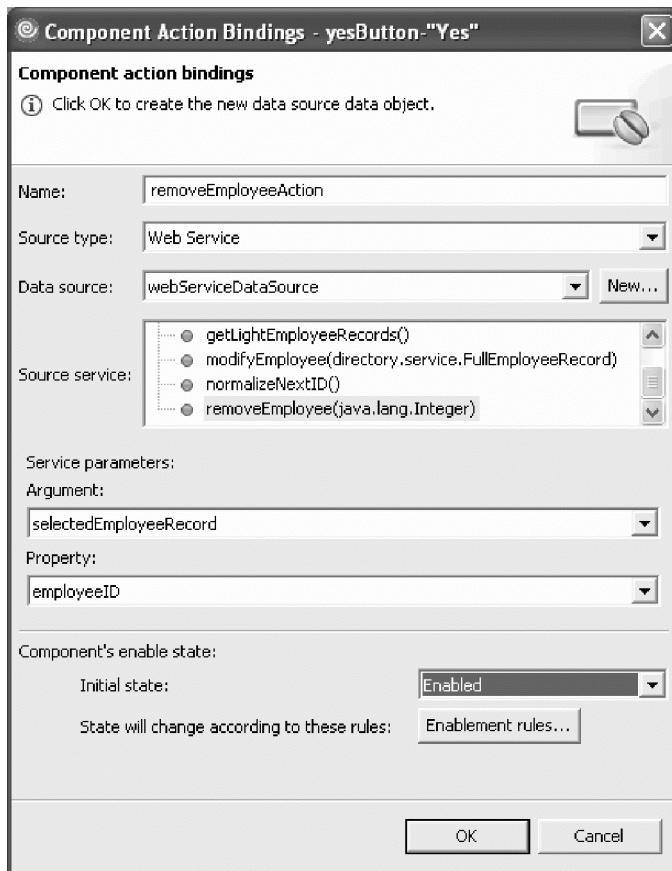
3. Для того чтобы убедиться, что это поле доступно только для чтения, задайте в свойстве **autoEditable** для редактора связей поля значение **false**.

Связывание кнопки Да с выполнением удаления

Свяжите кнопку **Да** с вызовом метода removeEmployee(java.lang.Integer) в Web-службе.

1. Выберите кнопку **Да** и щелкните на вкладке **Связать**. Откроется окно Привязки действий компонентов.
2. В поле **Исходный тип** выберите **Web-служба**.
3. В поле **Источник данных** выберите **webServiceDataSource**.
4. В списке **Исходная служба** выберите **removeEmployee(java.lang.Integer)**.
5. В поле **Имя** автоматически отобразится **removeEmployeeAction**. Примите значение по умолчанию.
6. В поле **Аргумент** выберите **selectedEmployeeRecord**.
7. В поле **Свойство** выберите **employeeID**. Поскольку метод removeEmployee() в качестве аргумента принимает целые числа, в него передается ИД сотрудника из selectedEmployeeRecord.
8. Задайте **Первоначальное состояние** кнопки - **Включено**.
9. В разделе **Правила активации** для каждого условия выберите **Игнорировать**.

Это состояние компонента означает, что кнопка **Да** всегда будет активной, поскольку нет необходимости изменять ее состояние.



10. Нажмите кнопку **OK**.

Добавление события, скрывающего окно Подтвердить удаление после удаления сотрудника

В этом шаге добавляется событие для *редактора связей* кнопки **Да** (но не для самой кнопки **Да**). Окно Подтвердить удаление должно закрываться после удаления сотрудника, то есть после успешного выполнения вызова от редактора связей к службе в источнике данных.

Добавьте следующий код в метод `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) {}
});
```

Этот код события скрывает окно Подтвердить удаление после того, как действие редактора связей выполнено.

Полученные знания и навыки

Теперь при запуске приложения *My Company Directory* можно выбрать сотрудника в таблице, нажать кнопку **Удалить** и выбрать **Да** для подтверждения. Запись о сотруднике будет удалена из каталога, что отобразится в списке сотрудников.

Урок 2.6: Настройка действий и привязок для добавления нового сотрудника

В этом уроке в приложение *My Company Directory* будет реализована возможность добавлять новые записи сотрудников.

Поскольку при добавлении нового сотрудника в приложении выполняются более сложные и динамичные действия, это упражнение также сложнее предыдущих, поскольку вам потребуется вручную вносить некоторые изменения в исходный код. Кроме того, в этом упражнении показаны некоторые расширенные возможности объектов данных. Здесь на творческом примере проиллюстрированы такие способы применения редакторов связей и объектов данных, при которых они наилучшим образом выполняют поставленные задачи.

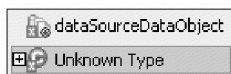
В приведенном ниже списке перечислены обязательные функции приложения:

- При нажатии на кнопку **Создать** выполняются следующие действия:
 - Очищается выделение в таблице сотрудников и таблица становится неактивной.
 - Очистка выделения таблицы делает неактивной кнопку **Удалить**.
 - Поле **Фильтр** становится неактивным.
 - Из полей подробной информации удаляются все значения кроме ИД нового сотрудника.
 - Текст в кнопке **Обновить** заменяется на **Добавить**.
- При нажатии на кнопку **Добавить** выполняются следующие действия:
 - Значения, введенные в полях подробной информации, добавляются в каталог в качестве новой записи о сотруднике.
 - Таблица становится активной и значения обновляются.
 - Поле **Фильтр** становится активным.
 - Текст в кнопке **Добавить** заменяется на **Обновить**.

Добавление нового объекта данных источника данных, вызывающего createNewFullEmployeeRecord()

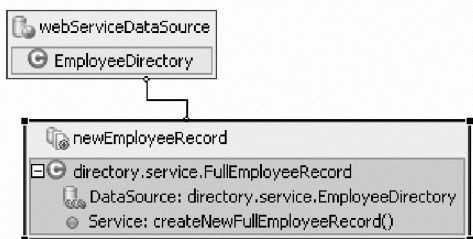
В примере Web-службы предусмотрена служба createNewFullEmployeeRecord, предоставляющая новую, пустую запись о сотруднике, которая заполняется следующим доступным номером ИД сотрудника. Эту пустую запись можно заполнить сведениями о новом сотруднике и отправить обратно в Web-службу.

1. В палитре визуального редактора для Java разверните лоток Объекты данных и выберите **Объект данных источника данных**.
2. Поместите указатель мыши на пустой области в панели эскиза или в произвольной области и щелкните левой кнопкой для того, чтобы поместить объект данных источника данных. Новый объект данных источника данных будет добавлен и отобразится в произвольной области:



3. Щелкните правой кнопкой мыши на объекте данных источника данных и выберите **Переименовать поле**. Переименуйте объект данных в newEmployeeRecord.
4. Щелкните правой кнопкой мыши на объекте данных newEmployeeRecord и выберите **Свойства связывания**. Откроется окно связывания данных.
5. В поле **Источник данных** выберите webServiceDataSource
6. В поле **Служба** выберите createNewFullEmployeeRecord()
7. Нажмите кнопку **ОК**.

В произвольной области видно, что объект данных источника данных newEmployeeRecord связан с Web-службой.

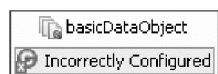


Добавление простого объекта данных, обеспечивающее переключение объектов данных

Поскольку для полей подробной информации и кнопки Обновить требуется переключение режимов (как для обновления, так и для создания нового сотрудника), они должны быть связаны в разное время с двумя различными объектами данных. Для того чтобы обеспечить эту возможность, нужно добавить простой объект данных с именем `switchingDataObject`. Простой объект данных будет применяться для переключения привязки текстовых полей между `selectedEmployeeRecord` и `newEmployeeRecord`.

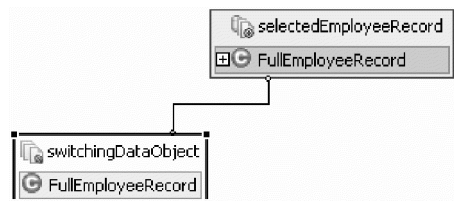
Новый простой объект данных просто указывает на другой объект данных (`selectedEmployeeRecord`), определенный в предыдущем упражнении. Этот новый объект данных применяется при создании метода, с помощью которого простой объект данных использует созданный ранее `newEmployeeRecord`. Другими словами, этот простой объект данных будет выступать в роли промежуточного объекта данных, который переключается между объектами данных `selectedEmployeeRecord` и `newEmployeeRecord`, что позволяет визуальным компонентам в приложении работать с двумя различными объектами данных.

1. В палитре визуального редактора выберите **Простой объект данных**, и поместите его в произвольную область. Будет добавлен объект `basicDataObject`.



2. Переименуйте объект данных в `switchingDataObject`
3. В панели свойств объекта `switchingDataObject` задайте **selectedEmployeeRecord** в качестве значения свойства **sourceObject**. `selectedEmployeeRecord` можно выбрать в выпадающем меню в столбце свойства Значение.

Теперь `switchingDataObject` ссылается на `selectedEmployeeRecord` и отражает такие же значения:

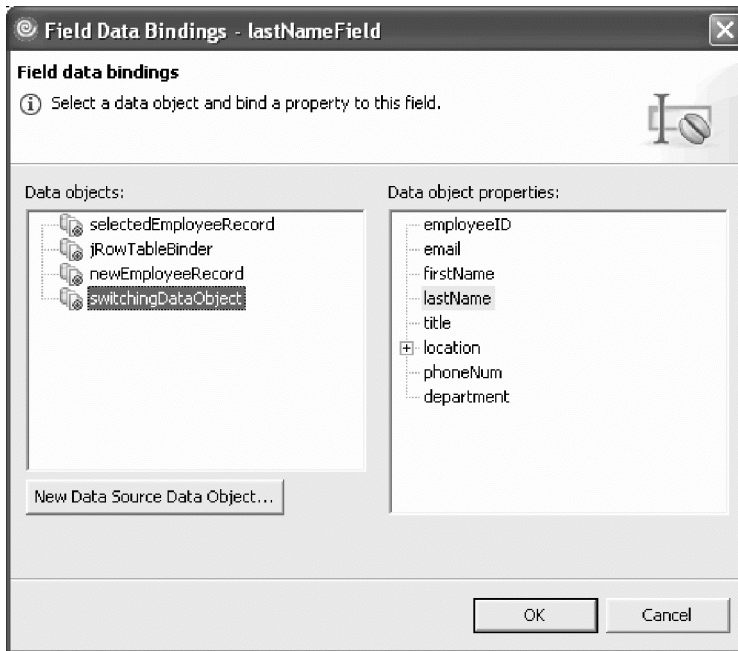


Повторное связывание всех сотрудников с switchingDataObject

Несмотря на то, что все поля подробных сведений о сотрудниках уже связаны с `selectedEmployeeRecord`, теперь их нужно связать с `switchingDataObject`. После связывания полей можно динамически переключаться между объектами данных для полей в зависимости от того, редактируете ли вы существующую запись о сотруднике или добавляете новую запись.

Для каждого из полей в разделе Подробные сведения о сотрудниках выполните следующие действия:

1. Выберите поле и щелкните на вкладке **Связать**.
2. В окне Привязки данных таблицы выберите `switchingDataObject`. Ранее вы связали поля с `selectedEmployeeRecord`.



- Убедитесь, что поле связано с нужным свойством объекта данных и нажмите кнопку **OK**. При выборе поля в панели эскиза вы увидите, что линии редактора связей теперь указывают на `switchingDataObject`.



Определение метода и флага для обновления и переключения режимов

Приведенный ниже метод `updateMode()` проверяет значение флага режима и изменяет соответствующим образом поведение приложения. По умолчанию булевский флаг `isNewMode` принимает значение `false`, а метод `updateMode()` активирует таблицу сотрудников и поле фильтра, а также задает текст "Обновить" в кнопке Обновить. Если `isNewMode` принимает значение `true`, то таблица сотрудников неактивна и выделение очищено, поле фильтра неактивно, а в кнопке Обновить отображается текст "Добавить".

Добавьте приведенный ниже код в класс `DirectoryApp.java` перед последней закрывающей фигурной скобкой:

```
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");
    }
}
```

Добавление события `actionPerformed` в кнопку Создать

В этом шаге добавляется код события, выполняемого при нажатии кнопки **Создать**. С помощью события `switchingDataObject` использует объект данных `newEmployeeRecord`, присваивает флагу режима значение "new" и запускает метод `updateMode()`, добавленный в предыдущем шаге.

- В панели эскиза щелкните правой кнопкой мыши на кнопке **Создать** и выберите **События** → **actionPerformed**. В методе `getNewButton()` будет сформирован следующий код:


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

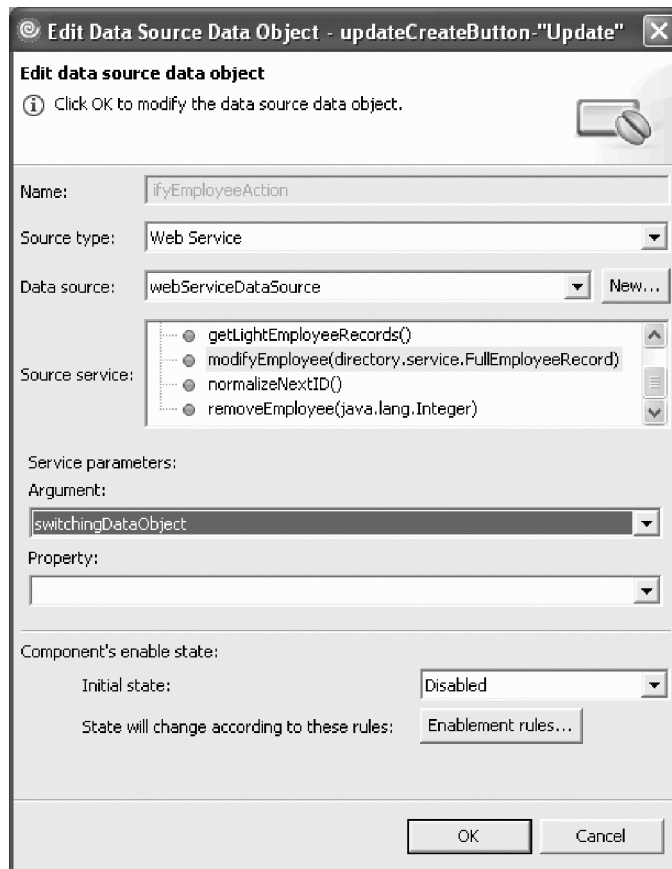
2. Замените эту заготовку следующим кодом:

```
newButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        getSwitchingDataObject().setSourceObject(getNewEmployeeRecord());
        getNewEmployeeRecord().refresh();
        isNewMode = true; //переключение приложения в другой режим
        updateMode(); //изменение пользовательского интерфейса в новом режиме
        getLastNameField().grabFocus();
    }
});
```

Повторное связывание кнопки Обновить

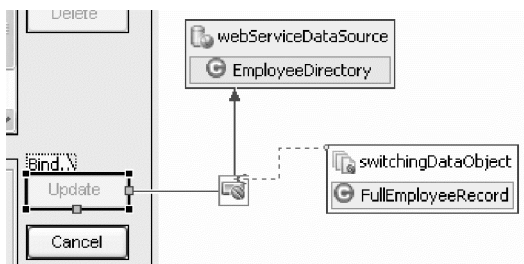
В предыдущем уроке мы программировали применение кнопкой **Обновить** метода `modifyEmployee` в Web-службе. Это действие реализовано как `SwingDataServiceAction`. Одно из свойств действия `SwingDataServiceAction` является исходным объектом, который выступает в роли аргумента для службы. В качестве исходного объекта для действия редактирования в настоящее время задан `selectedEmployeeRecord`. Для того чтобы кнопка управляла как обновлением, так и добавлением, нужно перенастроить ее действие для использования `switchingDataObject` в качестве аргумента в службе `modifyEmployee`.

1. В панели эскиза выберите кнопку **Обновить**. Обратите внимание на розовую пунктирную стрелку, которая показывает, что `selectedEmployeeRecord` является аргументом для вызова службы.
2. Щелкните на вкладке **Связать** для кнопки **Обновить**.
3. В поле **Аргумент** выберите `switchingDataObject`.



4. Нажмите кнопку **OK**.

Теперь обратите внимание, что теперь действие кнопки настроено для использования `switchingDataObject` в качестве аргумента в методе `modifyEmployee`:



Добавление события в редактор связей кнопки Обновить для сброса режима

После того при нажатии кнопки **Обновить** действие в Web-службе будет выполнено, приложение должно переключаться в прежний режим. Для этого в редактор связей действия нужно добавить обработчик событий, который будет обновлять режим и обновлять таблицу после добавления или обновления.

Добавьте следующий код для кнопки Обновить в метод `getModifyEmployeeAction()`:

```
modifyEmployeeAction.addActionBinderListener(
    new jve.generated.IActionBinder.ActionBinderListener() {
        public void afterActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {
            if (isNewMode) {
                //возврат к использованию selectedEmployeeRecord
                getSwitchingDataObject().setSourceObject(getSelectedEmployeeRecord());
                //Возврат из нового режима
                isNewMode = false;
                updateMode();
            }
            // Обновить объект данных таблицы
            getLightEmployeeRecordRows().refresh();
        }
        public void beforeActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {}
    });
```

Полученные знания и навыки

Теперь в приложении My Company Directory при нажатии кнопки **Создать** будет создаваться новая запись о сотруднике.

Урок 2.7: Программирование поведения кнопки Отмена

При работе с приложением иногда приходится отменять изменения, внесенные в запись сотрудника, если их решено не сохранять. Другими словами, должна быть возможность отменить изменения, очистить поля и заполнить их заново. Для добавления такой возможности с кнопкой **Отмена** связываются события, соответствующие выполненным действиям.

В приведенном ниже списке перечислены обязательные функции кнопки **Отмена**:

- При нажатии кнопки **Отмена** в определенном режиме приложение переключается в предыдущий режим.
- При нажатии кнопки **Отмена** во время редактирования записи сотрудника все измененные значения возвращаются к первоначальным.

Для того чтобы добавить событие `actionPerformed` в кнопку **Отмена** и обеспечить выполнение необходимых операций, выполните следующие действия:

1. В панели Эскиз щелкните правой кнопкой мыши на кнопке **Отмена** и выберите **События** → **actionPerformed**. В методе `getCancelButton()` будет сформирован следующий код:

```
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. Замените заготовку созданного события следующим кодом:

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

Полученные знания и навыки

В этом уроке вы изучили программирование событий actionPerformed для кнопки **Отмена**.

Урок 2.8: Настройка фильтра в таблице сотрудников

Компоненты таблицы сотрудников фильтруются с помощью редактора связей Текстовый фильтр. Фильтр получает входные данные из текстового поля и отфильтровывает данные таблицы по определенному свойству или столбцу таблицы.

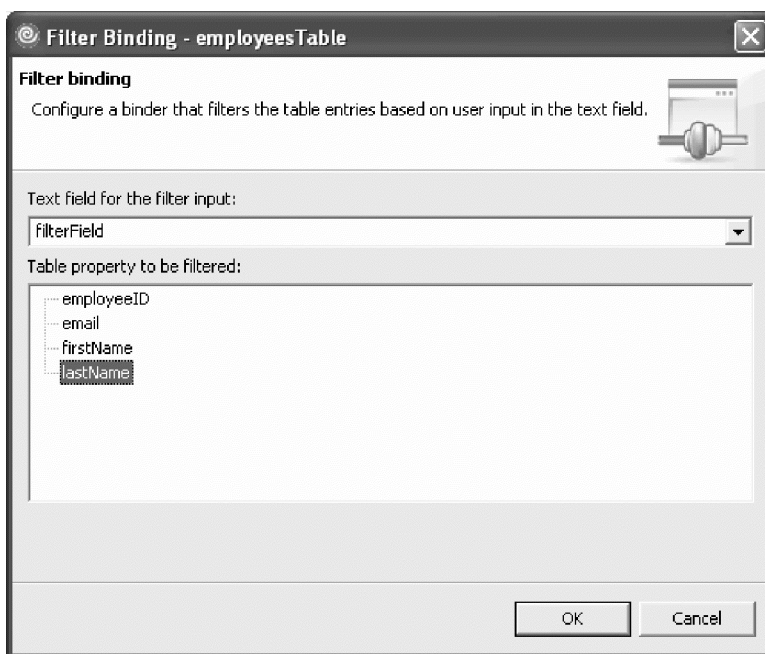
Для фильтрации сотрудников по фамилии в приложении будут использоваться символы, вводимые в поле **Фильтр**. Если точные значения, введенные в поле **Фильтр**, присутствуют записи, соответствующей фамилии сотрудника, то эта запись отобразится в таблице.

The screenshot shows a web interface with a filter input field labeled "Filter: ma (Last name)". Below the input field is a table with four columns: Last name, First name, Email, and Employee ID. The table contains three rows of data. To the right of the table are two buttons: "New" and "Delete".

Last name	First name	Email	Employee ID
Maxwell	Seth	seth.maxwell@my...	24561
Maxwell	Aubrey	aubrey.maxwell@...	30089
Martinez	James	james.martinez@m...	31780

Для того чтобы создать фильтр для таблицы, выполните следующие действия:

1. Щелкните значке редактора связей таблицы employeesTable и выберите **Фильтровать свойства связывания**. Откроется окно фильтрации связывания.
2. В списке **Текстовое поле для ввода фильтра** выберите **Поле фильтрации**.
3. В списке **Свойство таблицы для фильтрации** выберите **Фамилия**.



4. Нажмите кнопку **OK**.

Будет создан фильтр `SwingPropertyFilter`. В свойстве фильтра редактора связей таблицы задается использование нового фильтра. Новый фильтр получает входные данные из поля **Фильтр** и выполняет фильтрацию по свойству `lastName` в таблице.

Полученные знания и навыки

В этом уроке вы изучили настройку фильтра для таблицы.

Теперь при запуске приложения `My Company Directory` можно вводить символы в поле **Фильтр**, и таблица будет отфильтрована таким образом, что отобразятся строки с фамилиями сотрудников, которые содержат введенные символы.

Поздравляем! Приложение `My Company Directory` готово.

Обзор: создание расширенного клиента Java, использующего Web-службу

Поздравляем! Вы научились создавать с помощью визуального редактора для Java приложение My Company Directory, а также расширенный клиент Java, который поддерживает каталог сотрудников путем подключения к примеру Web-службы.

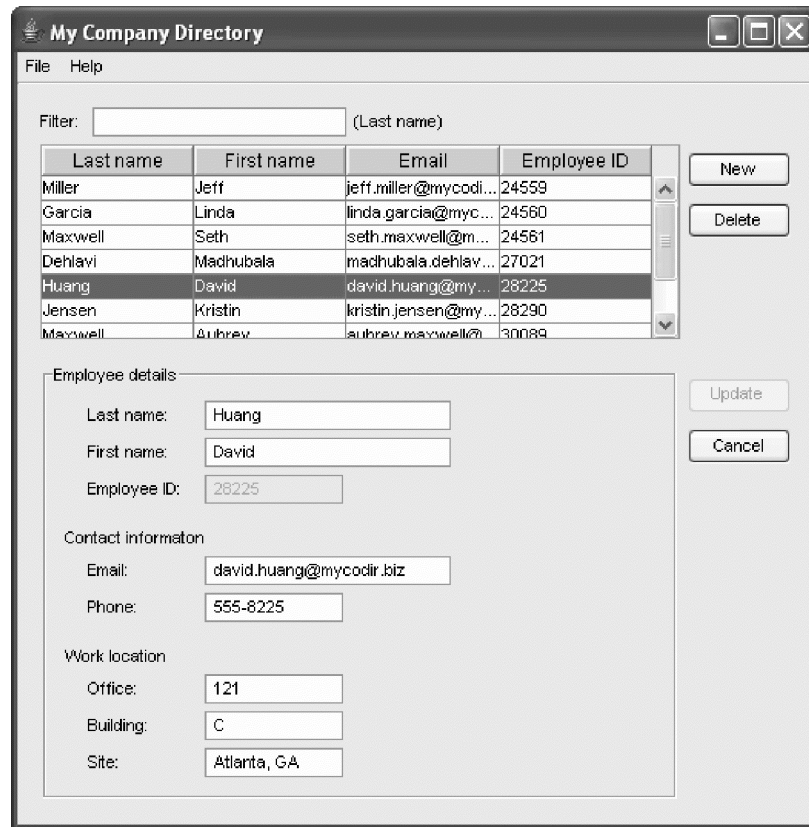


Рисунок завершенного продукта:

Пройденные уроки

С помощью визуального редактора вы создали графический пользовательский интерфейс, а с помощью GridBagLayout - макет таблицы сотрудников. Затем вы связали таблицу, поля и кнопки с соответствующими объектами данных и источникам данных, что позволило приложению работать с созданным проху Java Web-службы. Кроме того, вы проделали сложные манипуляции с исходным кодом, чтобы обеспечить правильную работу приложения и сделать интерфейс интуитивно понятным и простым в использовании. Кроме того, вы научились устанавливать приложение J2EE на WebSphere Application Server версии 6.0 и развертывать Web-службу.

Важнее всего то, что вы полностью изучили мощный класс редактора связей, поставляемый с визуальным редактором для работы с данными. Теперь вы можете начать самостоятельные эксперименты и программировать с помощью редакторов связей новые и интересные функции.

Вы должны были освоить следующие задачи:

- Макетирование компонентов в GridBagLayout с помощью визуального редактора для Java.
- Запуск визуального класса в качестве объекта JavaBean.
- Связывание визуальных компонентов приложения на Java с методами и объектами данных, возвращаемыми Web-службой.
- Добавление событий в визуальные компоненты.

Дополнительные ресурсы

Импортируйте готовую версию приложения My Directory

В состав этого проекта входит готовое приложение, пакет `jve.generated` с классами редактора связей, а также клиент Java Web-службы, настроенный для работы с WebSphere Application Server v6.1. В случае импорта этого проекта независимо от учебника может потребоваться дополнительная настройка переменной пути компоновки Java. Она должна указывать на файл JAR тонкого клиента Web-служб WebSphere v6.1.

Совет: Если в ходе импорта не указать другой проект, то содержимое проекта MyDirectory будет заменено.