Desenvolver um Java Rich Client que Utiliza um Serviço da Web

Índice

Construir um Java Rich Client que Utiliza um Serviço da Web

Este tutorial ensina como utilizar o editor visual Java para construir um cliente rich Java que se conecta a um serviço da Web. O cliente que você constrói no tutorial é chamado de My Company Directory.

My Company Directory é um aplicativo Java utilizado para manter um diretório do funcionário da empresa. O aplicativo se conecta a um serviço da Web de amostra que fornece métodos para a criação, recuperação, atualização e exclusão de registros de funcionários.

O cliente é construído visualmente no Java Visual Editor, utilizando componentes Swing. O Java Visual Editor fornece um conjunto de classes auxiliares (origens de dados, objetos de dados e binders) que se conectam ou funcionam com o serviço da Web. O serviço da Web é implementado localmente na instalação do IBM WebSphere Application Server v6.0 e as ferramentas ajudam a gerar um proxy Java para o cliente, com base em um arquivo WSDL (Web Services Description Language).

Consulte o Produto Final

Objetivos do Aprendizado

Neste tutorial, você aprenderá as seguintes lições:

- Como utilizar o editor visual Java para projetar e definir o layout de uma interface com o usuário
- Como ligar elementos da interface a objetos de dados e a um serviço da Web

Tempo Necessário

2 horas e 15 minutos

Informações relacionadas

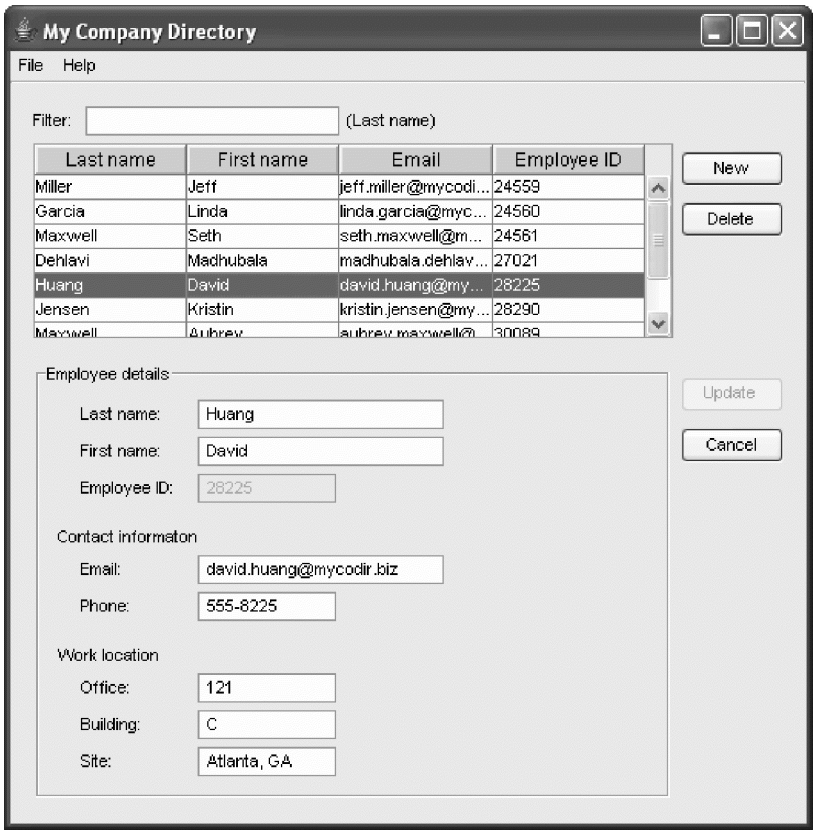
Visualizar a versão em PDF

Tutorial: Hello World Java

Introdução: Construir um Java Rich Client que Utiliza um Serviço da Web

A GUI (Interface Gráfica com o Usuário) do cliente foi praticamente pré-construída visualmente, utilizando os componentes Swing. No primeiro módulo, você concluirá com os componentes GUI essenciais, utilizando o Java Visual Editor. No segundo módulo, você ligará os componentes GUI à origem de dados do serviço da Web, serviços e objetos retornados pela origem de dados. Na construção do aplicativo no Java Visual Editor, você utilizará origens de dados, objetos de dados e binders de dados, nos quais são geradas as instâncias de classes de auxílio, geradas pelo Java Visual Editor e utilizadas pelo seu aplicativo.

Veja uma imagem do produto final:



Objetivos do Aprendizado

Neste tutorial, você aprenderá as seguintes lições:

- Como utilizar o editor visual Java para projetar e definir o layout de uma interface com o usuário
- Como ligar elementos da interface a objetos de dados e a um serviço da Web

Tempo Necessário

Para concluir todo o tutorial, você precisará de, aproximadamente, 2 horas e 30 minutos.

Requisitos do Sistema

- WebSphere Application Server v6.1. Esse servidor pode já ter sido instalado com seu produto, ou você pode utilizar sua própria instalação independente. O cenário nesse tutorial solicita a implementação de um serviço da Web de amostra no WebSphere Application Server que está em execução localmente.

O serviço da Web de amostra pode ser executado em outros servidores, mas esse tutorial foi testado somente com o WebSphere Application Server v6.0 e v6.1.

Pré-requisitos

Você deve estar familiarizado com os seguintes conceitos:

- Desenvolvimento básico em Java
- Princípios básicos do serviço da Web
- Habilidades básicas com o workbench, tais como trabalhar com projetos e navegar em perspectivas e visualizações

Módulo 1: Design da GUI Cliente no Visual Editor

Esse módulo ensina como utilizar o Java Visual Editor para incluir um componente visual a um aplicativo e como organizá-lo visualmente e definir as restrições de disposição. A lição final desse módulo mostra como executar o arquivo Java para consultar que aparência terá como um aplicativo real.

Lembre-se: Antes de iniciar este módulo, você deve ter o conhecimento de pré-requisito descrito na introdução ao tutorial.

Objetivos do Aprendizado

Após concluir as lições desse módulo, você compreenderá os conceitos e saberá como fazer o seguinte:

- Incluir e definir o layout de uma JTable em uma interface Java
- Executar uma classe visual para testar seu trabalho

Tempo Necessário

Este módulo levará aproximadamente 15 minutos para ser concluído.

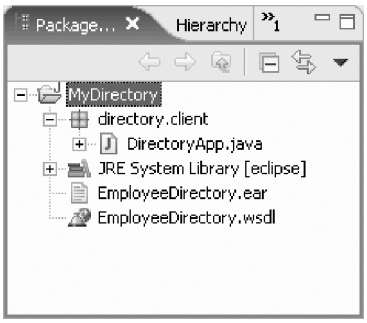
Lição 1.1: Configurar o Projeto Java

Nesta lição, você irá configura o projeto MyDirectory importando um projeto para seu espaço de trabalho. O projeto inclui uma classe Java única, bem como outros arquivos que serão utilizados posteriormente.

Como o foco principal desse tutorial é ligar componentes visuais a um serviço da Web, a maior parte da GUI Java do aplicativo "My Company Directory" já foi designado para você.

O projeto MyDirectory é o principal projeto Java com o qual você trabalhará neste tutorial. Ele contém o arquivo DirectoryApp.java, que é o arquivo Java que contém o principal aplicativo Java em construção. Esse tutorial inclui diversas versões do projeto MyDirectory para ajudá-lo: um para o início de cada módulo e uma versão finalizada do projeto concluído.

1. Importar o Projeto MyDirectory.
2. No Explorador de Pacotes (Package Explorer) da perspectiva Java, certifique-se de que o projeto MyDirectory esteja semelhante à seguinte imagem:



Ponto de Verificação da Lição

Nessa lição, você importou o projeto de exemplo MyDirectory, que é o ponto inicial desse tutorial.

O projeto MyDirectory inclui os seguintes recursos:

- DirectoryApp.java: arquivo Java que contém o aplicativo em desenvolvimento neste tutorial. O arquivo DirectoryApp.java está em um pacote Java chamado directory.client.
- EmployeeDirectory.ear: Um aplicativo corporativo que contém o serviço da Web de amostra. No Módulo 2, você implementará este serviço da Web em uma instalação local do WebSphere Application Server v6.0.

- EmployeeDirectory.wsdl: Um arquivo XML que utiliza WSDL (Web Services Description Language) para descrever o serviço da Web de amostra que você implementará. No Módulo 2, você utilizará este arquivo WSDL para gerar um proxy Java para seu aplicativo utilizar.

Lição 1.2: Incluir e Definir o Layout da Tabela de Funcionários

Nessa lição, você utilizará o editor visual Java para incluir um JScrollPane e um JTable no aplicativo. Em exercícios futuros, você programará o JTable para obter seus dados a partir de um serviço da Web que retorna uma lista de todos os funcionários no diretório da empresa.

Após incluir o JTable, você utilizará a visualização de design do editor visual Java para personalizar o layout do JTable para corresponder às especificações a seguir:

- Estender JTable por três células horizontalmente e duas células verticalmente
- Incluir uma inserção à esquerda de 15 pixels
- Renomear JTable para employeesTable.

Mostre-me

Abrir o Arquivo DirectoryApp.java no Editor Visual Java

Para abrir o arquivo DirectoryApp.java no Java Visual Editor:

1. Na visualização Explorador de Pacotes da perspectiva Java, expanda o projeto MyDirectory e o pacote directory.client.
2. Clique com o botão direito do mouse no arquivo DirectoryApp.java e selecione **Abrir Com (Open With)** → **Visual Editor**. O editor visual Java carrega a classe Java e exibe o design na área de canvas gráfico.

Dica:

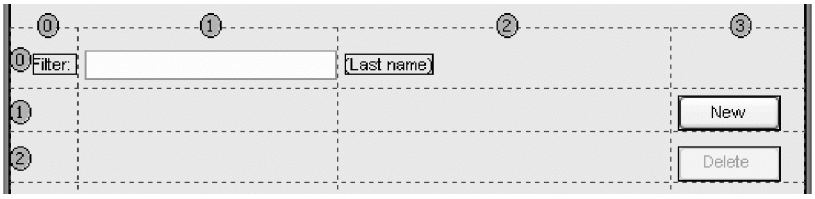
- Para alterar a aparência e o comportamento utilizados pelo editor visual Java, vá para **Janela (Window)** → **Preferências (Preferences)** → **Java** → **Visual Editor** e especifique uma aparência e comportamento Swing. A preferência terá efeito na próxima vez que você abrir a classe. Este tutorial utiliza a aparência e comportamento do Windows.
- Para tornar o Visual Editor o editor padrão para todos os arquivos Java, você pode clicar em **Janela (Window)** → **Preferências (Preferences)** e ir para a página **Workbench** → **Associações de Arquivos (File Associations)**, para definir sua preferência.

Incluir uma JTable em um JScrollPane

A janela principal do DirectoryApp.java utiliza um JFrame com um JPanel para seu painel de conteúdo principal. O JPanel em nosso aplicativo é chamado jContentPane. O jContentPane foi definido para utilizar um tipo de gerenciador de layout chamado GridBagLayout. O GridBagLayout é um poderoso esquema de layout com base em uma grade de células que podem ser ocupadas pelos componentes de visual. O editor visual Java torna fácil o trabalho com o GridBagLayout mostrando as bordas das grades. Ele também mostra os marcadores de localização quando você solta novos componentes na grade e ele mostra alças nos componentes que você está redimensionando ou movendo no GridBagLayout.

Para incluir a tabela de funcionários (javax.swing.JTable) na interface com o usuário DirectoryApp.java:

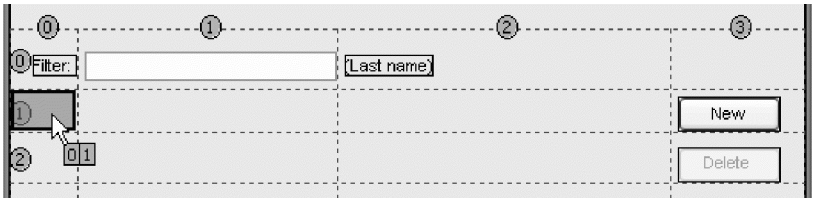
1. Clique com o botão direito do mouse em jContentPane na visualização do design ou dos Java Beans e selecione **Mostrar Grade (Show Grid)**. Uma linha pontilhada vermelha mostra o borda da grade e círculos numerados em azul indicam os números da linha e da coluna. Por exemplo, observe que o botão **Novo (New)** ocupa a célula na linha 1 (grade y) e a coluna 3 (grade x).



2. Na paleta do editor visual Java, selecione o componente Swing **JTable no JScrollPane** , que é categorizado na gaveta **Componentes Swing (Swing components)** da paleta.

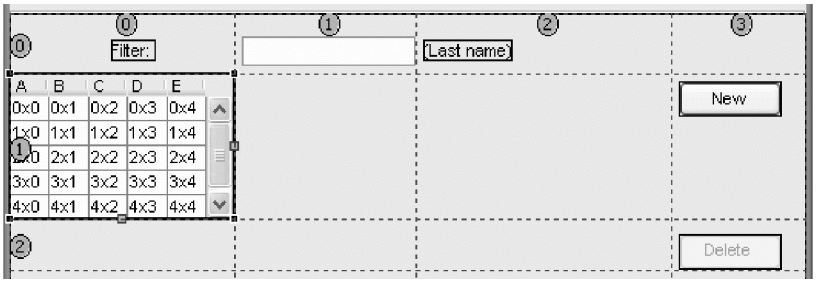
   Dica: Por padrão, a paleta é reduzida no lado direito da área de design. É possível redimensionar e mover a paleta.

3. Mova o ponteiro do mouse sobre a célula na grade na coluna 0, linha 1:



   - Conforme você move o ponteiro do mouse sobre a grade, esse ponteiro mostra dois quadrados numerados que indicam as coordenadas x e y na grade com base no local do ponteiro do mouse.
   - Se você passar o ponteiro do mouse diretamente em uma borda de grade, novas linhas e colunas podem ser criadas e linhas e colunas existentes serão renumeradas. Nesse caso, quadrados amarelos no ponteiro do mouse, barras amarelas entre as grades e etiquetas amarelas de colunas e linhas indicam este comportamento e apontam o impacto que o posicionamento terá.

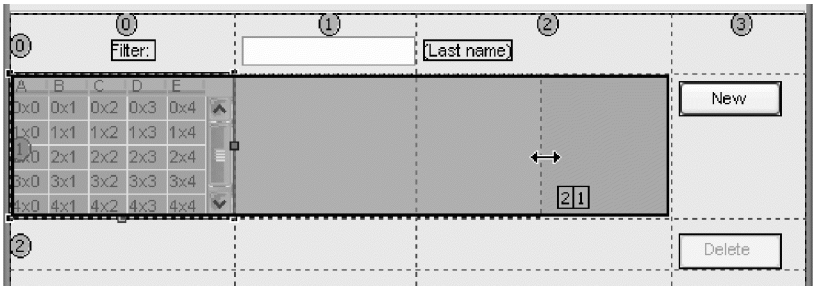
4. Clique com o botão esquerdo em JScrollPane e JTable na célula na coluna 0 e linha 1:



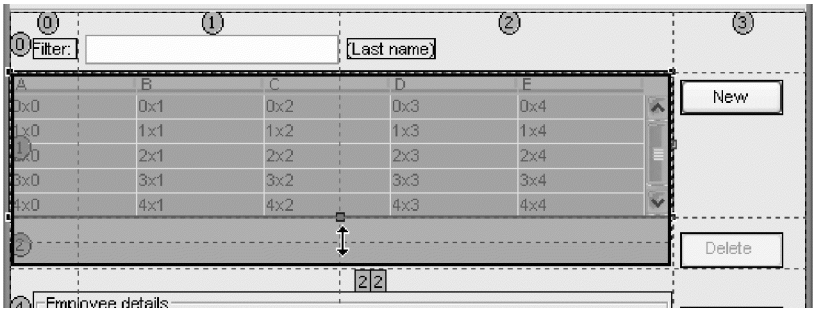
Expandir JScrollPane e JTable em Várias Colunas e Linhas da Grade

Agora é necessário fazer JScrollPane (e seu filho JTable) expandir em 3 colunas e 2 linhas para um melhor comportamento de espaçamento e redimensionamento. Para fazer a tabela estender as colunas e linhas:

1. Selecione JScrollPane na área de design ou exibição Java Beans (ainda deve estar selecionada, pois acabou de ser incluída). Observe os pequenos quadrados verdes no canto inferior direito do JScrollPane. Você utilizará estas alças de redimensionamento para arrastar o JScrollPane para expandir várias colunas e linhas.
2. Clique e segure o botão esquerdo do mouse na alça verde no lado direito de JScrollPane.
3. Arraste o ponteiro do mouse para a direita até que o posicionamento do cursor indique coluna 2, linha 1. Uma sombra cinza escura também indicará as células que o componente ocupará quando você soltar o botão do mouse.



4. Solte o botão do mouse. Agora o JScrollPane foi estendido em 3 colunas.
5. Repita o processo semelhante para arrastar a alça inferior do JScrollPane até que JScrollPane seja estendido na linha 2:



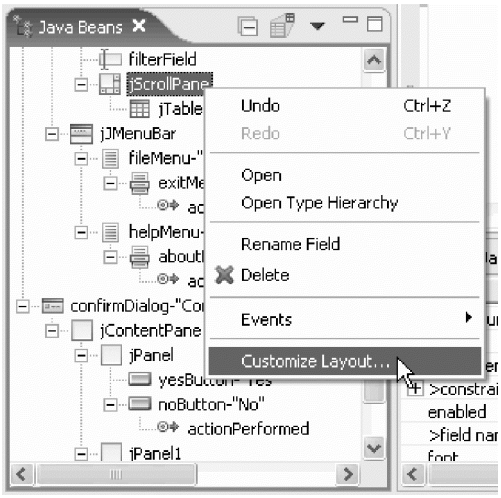
Personalizar o Espaçamento do JScrollPane Dentro de GridBag

Um outro recurso do gerenciador GridBagLayout é que você pode especificar diversos limites para personalização do layout. Por exemplo, é possível especificar as seguintes restrições:

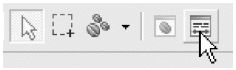
- **âncora: (anchor:)** Um componente pode receber uma orientação de âncora dentro de sua célula, que afetará como o componente se move conforme o aplicativo é redimensionado por um usuário. Por exemplo, um componente poderia ser ancorado no canto superior esquerdo, médio esquerdo, centro ou inferior direito.
- **preencher: (fill:)** Um componente pode ser indicado para ocupar todo o espaço disponível dentro de sua célula ou células, horizontalmente, verticalmente ou ambos.
- **inserção: (insets:)** Um componente pode ser indicado para seu próprio preenchimento superior, inferior, à esquerda e à direita, para fornecer espaçamento entre o componente e a linha da grade.

Para personalizar a âncora, o preenchimento e as inserções para JScrollPane:

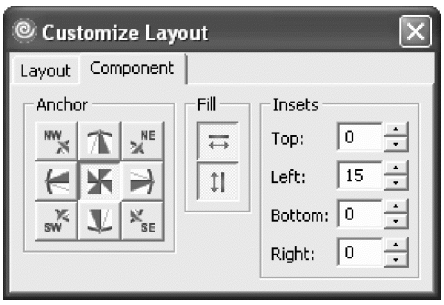
1. Clique com o botão direito do mouse em JScrollPane na visualização de design ou dos Java Beans e selecione **Customizar Layout (Customize Layout)**.



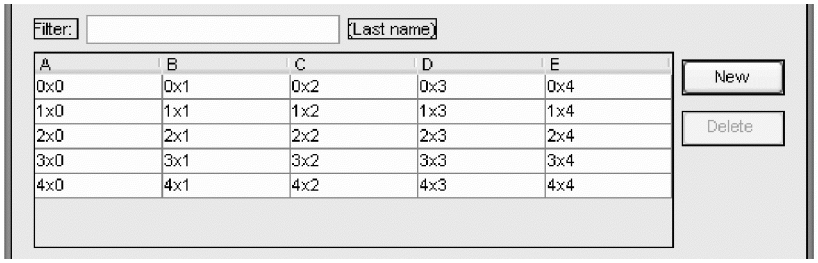
Dica: A caixa de diálogo Customizar Layout (Customize Layout) pode permanecer aberta como selecionada e alterar o layout para diferentes componentes. É possível abrir a caixa de diálogo Customizar Layout (Customize Layout) a qualquer momento, clicando no botão Customizar Layout (Customize Layout) na barra de menus:



2. Na guia Componente (Component) da caixa de diálogo Customizar Layout (Customize Layout), certifique-se de que o botão Centro de Âncora está pressionado.
3. Certifique-se de que os botões **Preenchimento Horizontal (Fill horizontal)** e **Preenchimento Vertical (Fill vertical)** estão pressionados.
4. Inclua uma inserção à esquerda de 15 (pixels) para tornar o espaçamento à esquerda do JScrollPane semelhante aos outros componentes visuais no aplicativo.



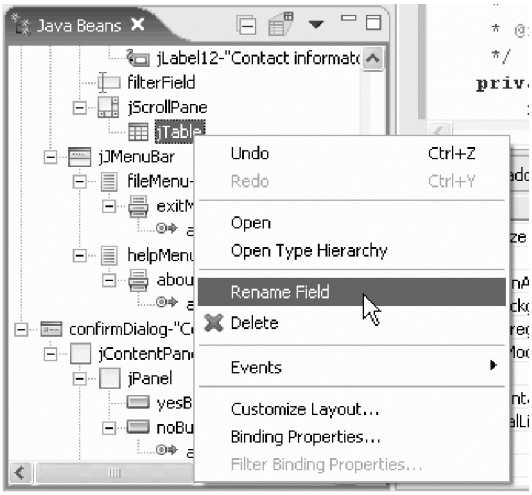
A tabela agora alinha-se com a etiqueta **Filtro (Filter)**, por exemplo.



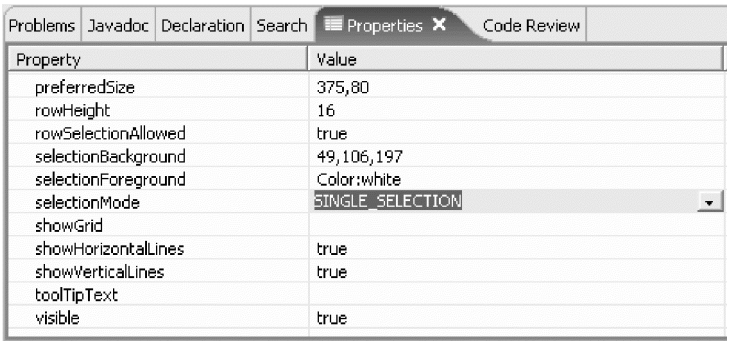
Renomear o Novo JTable para um Valor Útil e Definí-lo para Selecionar uma Única Linha

Como você trabalhará posteriormente com a tabela, será viável renomear a instância do JTable e seu método getter. Para renomear a tabela:

1. Na visualização Java Beans, clique com o botão direito do mouse no componente jTable e selecione **Renomear Campo (Rename field)** no menu pop-up.



2. Digite `employeesTable` e clique em **OK**. O JTable está renomeado para employeesTable, e o método para sua instância é getEmployeesTable.
3. Configure a tabela para permitir apenas uma única linha a ser selecionada:
   a. Selecione employeesTable na visualização de design.
   b. Na visualização Propriedades, selecione a propriedade **selectionMode** e a defina como SINGLE_SELECTION.



   c. Salve o arquivo DirectoryApp.java.

Ponto de Verificação da Lição

Nessa lição, você aprendeu como utilizar o editor visual para incluir uma tabela em uma interface com o usuário existente. Em seguida, você aprendeu como customizar seu layout, posicionamento e espaçamento.

Lição 1.3: Executar a Classe Visual

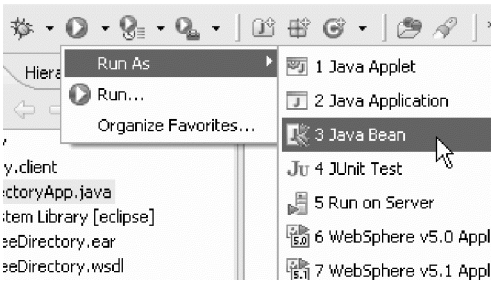
Agora você está pronto para executar o aplicativo Java para visualizar sua aparência. O workbench e o editor visual tornam muito fácil e rápida a execução de seu aplicativo e você pode repetir estas etapas em qualquer momento durante o desenvolvimento para testar a aparência real do tempo de execução e o comportamento da classe.

Mostre-me

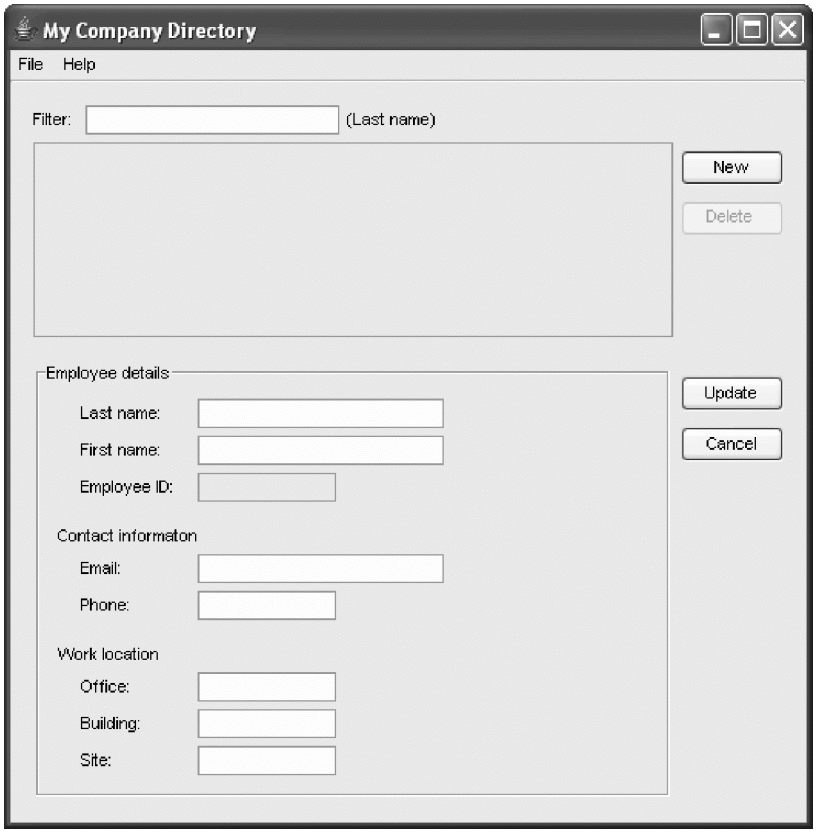
O Java Visual Editor fornece um ativador Java Bean, capaz de executar as classes sem método main(). Quando ele executa a classe visual, ele ativa o aplicativo em uma VM (virtual machine) separada. Se você executar sua classe visual como um aplicativo Java, o ativador tenta executar o método main() na classe. Para este tutorial, seu aplicativo inclui um método main() que chama e mostra o DirectoryApp JFrame, para que seja possível executar como um aplicativo ou um Java bean.

Para executar o arquivo DirectoryApp.java como um Java bean:

1. Certifique-se de que seu arquivo DirectoryApp.java esteja aberto no editor visual Java.
2. Na barra de menus, clique em **Executar (Run)** → **Executar Como (Run As)** → **Java Bean**.



Dica: O aplicativo é aberto no desktop utilizando a aparência e comportamento Swing, definido nas preferências do Visual Editor (**Janela (Window)** → **Preferências (Preferences)** → **Java** → **Visual Editor**). Como alternativa, é possível clicar em **Executar (Run)** → **Executar (Run)** e definir a aparência e comportamento para a configuração de ativação particular para ativar esse Java bean. Se você executar esse aplicativo como tal, em vez de como um bean, também utilizará a aparência e comportamento do Windows, pois isso é definido no método main(). As capturas de tela utilizadas neste tutorial mostram a aparência e comportamento do Windows.



Ponto de Verificação da Lição

Como você projetou apenas a interface, mas não programou nenhuma conexão de dados ou funcionalidade de evento, não é possível fazer nada com o aplicativo. No entanto, é possível consultar o layout básico e a aparência que aparecerá ao usuário. Você pode experimentar clicar alguns do botões, mas será informado que eles não possuem funcionalidade. Os menus Arquivo (File) e Ajuda (Help), no entanto, já estão implementados para você. Você pode experimentá-los para ver o que eles fazem, e você pode inspecionar o código Java para ver como eles são implementados com os eventos actionPerformed.

Lições Aprendidas

Este módulo apresentou o design de uma interface para um rich client utilizando o Java Visual Editor. Além do design da aparência visual de um cliente, entretanto, há muito mais que você precisa fazer para realmente tornar o cliente útil. Geralmente, você precisará incluir comportamento de evento ou outra lógica e, neste caso, a ligação dos elementos visuais com algum tipo de origem de dados.

Neste módulo, você aprendeu como executar as seguintes tarefas:

- Importar um projeto Java utilizando a importação Intercâmbio
- Incluir uma JTable em um JScrollPane para a classe visual
- Utilizar o gerenciador de GridBagLayout para definir visualmente o layout da tabela no rich client
- Executar o aplicativo para ver a aparência real do cliente rich Java

No próximo módulo, Módulo 2: Ligar Componentes Visuais ao Serviço da Web, você tomará a interface simples My Company Directory e a tornará em um rich client poderoso que acessa métodos de serviços da Web para criação, recuperação, atualização e exclusão de registros de funcionários a partir de um diretório da empresa.

Módulo 2: Ligar Componentes Visuais ao Serviço da Web

Esse módulo ensina como ligar os elementos visuais do My Company Directory (os botões, a tabela de funcionários, campos e outras ações) a um serviço da Web. O serviço da Web fornece a funcionalidade real para criar, recuperar, atualizar e excluir funcionários do diretório de amostra.

Objetivos do Aprendizado

Após concluir as lições desse módulo, você compreenderá os conceitos e saberá como fazer o seguinte:

- Ligar uma tabela a uma origem de dados do serviço da Web de dados
- Ligar campos a objetos
- Programar botões com ações

Este módulo levará aproximadamente **2 horas** para ser concluído.

Lição 2.1: Instalar e Implementar o Serviço da Web

Nesse exercício, você instalará um arquivo do aplicativo corporativo de amostra (EAR) no WebSphere Application Server v6.1 e implementará o serviço da Web EmployeeDirectory. Seu aplicativo utilizará este serviço da Web para criar, ler, atualizar e excluir registros de funcionários.

Antes de iniciar, você deve concluir *uma das seguintes opções* para certificar-se de que o seu projeto MyDirectory esteja no ponto inicial adequado:

- Conclua "Módulo 1: Design da GUI Cliente no Visual Editor" na página 3.

  ou

- Importar o projeto MyDirectory no Ponto de Partida do Módulo 2

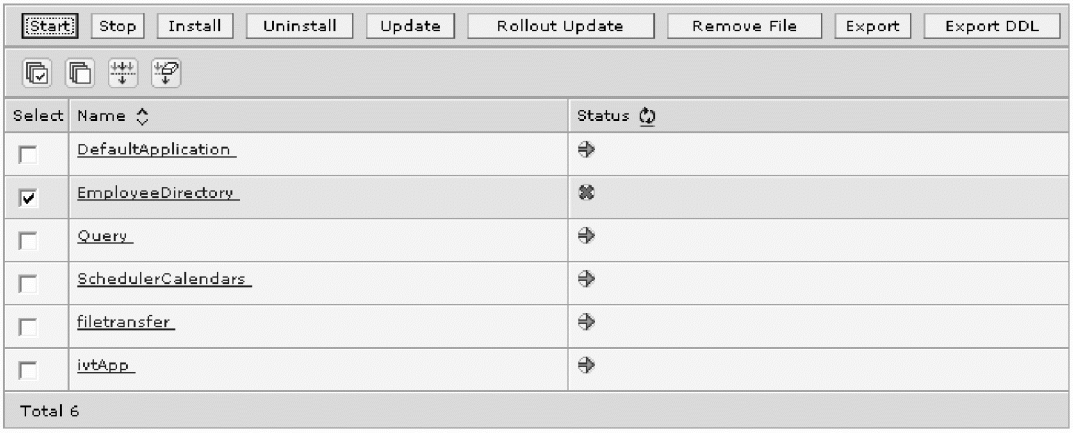
  Dica: A menos que você especifique um nome de projeto diferente durante a importação, esse projeto sobrescreverá o conteúdo de seu projeto MyDirectory.

Seu projeto Java MyDirectory inclui um arquivo EmployeeDirectory.ear. Você utilizará o WebSphere Administrative Console para instalar o aplicativo corporativo EmployeeDirectory contido no arquivo EAR. Ao instalar o aplicativo, você também implementa o serviço da Web incluído no aplicativo. O aplicativo My Company Directory concluído utiliza esse serviço da Web implementado.

Para instalar o aplicativo EmployeeDirectory de amostra e implementar o serviço da Web no ambiente do WebSphere Application Server v6.1:

1. Inicie uma instância do servidor de aplicativos no ambiente de trabalho. Há várias maneiras diferentes que possibilitam ativar o servidor, mas estas etapas descrevem como fazer isso do workbench:
   a. Abra a exibição Servidores. Para incluir a visualização Servidores na perspectiva Java, clique em **Janela (Window)** → **Mostrar Visualização (Show view)** → **Outra e Selecionar Servidor (Other and select Server)** → **Servidores (Servers)**.
   b. A visualização Servidores lista os servidores instalados e configurados.
   c. Clique com o botão direito do mouse em seu servidor e selecione **Iniciar (Start)**. Quando a visualização Servidores mostra o status do servidor como **Iniciado (Started)** ou o console afirma `Servidor server1 aberto para e-business (Server server1 open for e-business)`, o servidor é iniciado com êxito. É possível agora executar o Administrative Console.

   Nota: Se não houver instância do servidor na exibição Servidores, crie um novo servidor:
   a. Clique com o botão direito na exibição Servidores e selecione **Novo (New)** → **Servidor (Server)**.
   b. Utilize o assistente Novo Servidor para incluir o WebSphere Application Server v6.1.
2. Execute o WebSphere Administrative Console. Novamente, há outras maneiras de executar o Administrative Console, mas essas instruções descrevem como fazer isso a partir do workbench:
   a. Na visualização Servidores, clique com o botão direito no servidor iniciado e selecione **Executar o Console Administrativo (Run administrative console)**. O WebSphere Administrative Console é aberto em uma janela do navegador.
   b. Digite um ID do usuário e clique em **Efetuar Login (Log in)**. A página Bem-vindo (Welcome) do Administrative Console é aberta. O ID do usuário digitado é utilizado apenas para monitorar as alterações específicas do usuário para os dados de configuração do servidor.
3. Utilize o Administrative Console para instalar o aplicativo corporativo EmployeeDirectory.ear localizado no projeto MyDirectory. O Administrative Console utiliza uma abordagem do assistente para ajudá-lo a instalar aplicativos, em que você clica em **Avançar (Next)** para mover de página em página até que todas as opções sejam definidas. Para instalar o aplicativo corporativo de amostra que contém o serviço da Web para esse tutorial:
   a. À esquerda do Administrative Console, expanda a opção de menu **Aplicativos (Applications)** e clique em **Instalar Novo Aplicativo (Install New Application)**.
   b. Selecione **Sistema de Arquivos Local (Local file system)** e no campo **Especificar Caminho (Specify path)**, digite o caminho completo para o arquivo EmployeeDirectory.ear que está no projeto MyDirectory. Dica: Para obter o caminho completo, clique com o botão direito no arquivo EmployeeDirectory.ear no Explorador de Pacotes (Package Explorer) e selecione **Propriedades (Properties)**. A página Propriedades (Properties) lista o local do arquivo, que você pode copiar e colar no campo **Especificar Caminho (Specify path)**.
   c. Clique em **Avançar (Next)** até alcançar a página **Selecionar as opções de instalação (Select installation options)**.
   d. Selecione **Implementar Serviços da Web (Deploy Web services)**.
   e. Clique em **Avançar (Next)** até alcançar a página **Resumo (Summary)** e, em seguida, clique em **Concluir (Finish)**.
   f. Clique no link **Salvar na Configuração Master (Save to Master Configuration)** ao ser solicitado a aplicar as alterações feitas na configuração local. Reveja as alterações e clique no botão **Salvar (Save)**.
4. Utilize o Administrative Console para iniciar o aplicativo EmployeeDirectory:
   a. Clique em **Aplicativos (Applications)** → **Aplicativos Corporativos (Enterprise Applications)**. O aplicativo EmployeeDirectory está listado como um aplicativo instalado no servidor, mas seu status é Parado (Stopped).

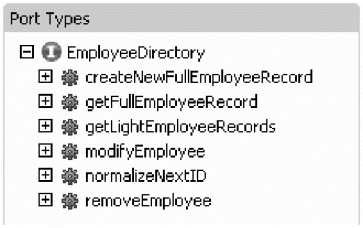


b. Selecione a caixa de opções próxima a EmployeeDirectory e clique em **Iniciar (Start)**. Uma mensagem indica que o aplicativo EmployeeDirectory foi iniciado com êxito e o ícone Status é alterado para a seta verde.

Agora, o aplicativo EmployeeDirectory está em execução no host local na porta 9080 e o serviço da Web pode ser acessado. Após concluir esse tutorial, é possível voltar ao Administrative Console, parar o aplicativo EmployeeDirectory e, em seguida, desinstalá-lo.

Se você abrir o arquivo EmployeeDirectory.wsdl localizado no projeto MyDirectory (deve ser aberto no WSDL Editor gráfico, por padrão), é possível examinar o serviço da Web implementado. Se o arquivo WSDL não for aberto no WSDL Editor, o recurso Web Service Developer pode não estar ligado no workbench. Você pode especificar os recursos de workbench em Preferências (Preferences) (**Janela (Window)** → **Preferências (Preferences)** → **Workbench** → **Recursos (Capabilities)**).

A imagem a seguir do editor WSDL mostra as operações disponíveis no serviço EmployeeDirectory:



É possível utilizar o editor WSDL para examinar cada operação e as mensagens de pedido e mensagens de retorno correspondentes. Isso pode ajudá-lo a compreender o serviço da Web e como é utilizado nos exercícios restantes.

Lição 2.2: Ligar a Tabela de Funcionários à Origem de Dados do Serviço da Web

O aplicativo My Company Directory exibe uma lista de todos os registros de funcionários atuais no diretório. Os registros são exibidos em um JTable (employeesTable) como colunas ordenadas, incluindo o último nome, primeiro nome, e-mail e ID do funcionário. Para obter os registros para a tabela, você precisa ligar a employeesTable em um objeto de dados retornado pela origem de dados do serviço da Web de amostra.

Mostre-me

Visão Geral dos Objetos de Dados, Origens de Dados e Binder

Para obter o objeto de dados local para employeesTable trabalhar, você utilizará o editor visual para incluir uma origem de dados ao aplicativo. A origem de dados conecta-se a um proxy de serviço da Web

de amostra e descobre os métodos de serviço disponíveis para o aplicativo. Em seguida, você escolherá o método de serviço getLightEmployeeRecord, disponível da origem de dados. Finalmente, você ligará employeesTable em seu aplicativo aos campos retornados no objeto de dados da linha (lightEmployeeRecordRows).

É possível criar todas essas origens de dados e objetos de dados rapidamente e com facilidade, utilizando as classes de binder internas do Java Visual Editor. O visual editor fornece um conjunto de interfaces e classes genéricas, geradas no projeto ao ligar os componentes visuais aos depósitos de informações do provedor de dados. As classes de binder são geradas por padrão em um pacote denominado jve.generated. O visual editor fornece as classes de binder como uma implementação genérica, que pode ser personalizada e aprimorada adicionalmente para corresponder às necessidades do aplicativo. Este tutorial demonstra a força e a flexibilidade da utilização, mesmo que básica e simples, das classes de binder padrão.

Importante: Antes de iniciar este exercício, é altamente recomendável que você leia os seguintes tópicos da ajuda. Esses tópicos podem ajudá-lo a saber mais sobre a funcionalidade e a lógica por trás dos objetos de dados, origens de dados e binders fornecidos pelo Java Visual Editor:

- Visão Geral dos Binders de Dados
- Referência de API do Binder

Para este tutorial, você utilizará uma origem de dados do serviço da Web, diversos tipos de objetos de dados e vários binders em seu aplicativo. Ao incluir instâncias desses objetos em seu aplicativo, o editor visual inclui classes necessárias no pacote jve.generated em seu projeto, onde você possa estender, substituir ou reescrever a lógica da ligação de dados. O Java Visual Editor fornece suporte visual a objetos de ligação, mostrando na área livre da visualização de design os objetos e as origens de dados e conectores que seu aplicativo está utilizando. O visual editor traça linhas entre os componentes, os objetos de dados e as origens de dados para mostrar as ligações atuais a qualquer objeto selecionado.

O diagrama a seguir é uma visão geral simples de como os componentes visuais, binders, objetos de dados e origens de dados interagem. O aplicativo construído neste tutorial ilustra uma utilização um pouco mais complexa e criativa dos binders. Este diagrama não representa exatamente os binders, objetos e origens de dados no aplicativo de amostra em construção.

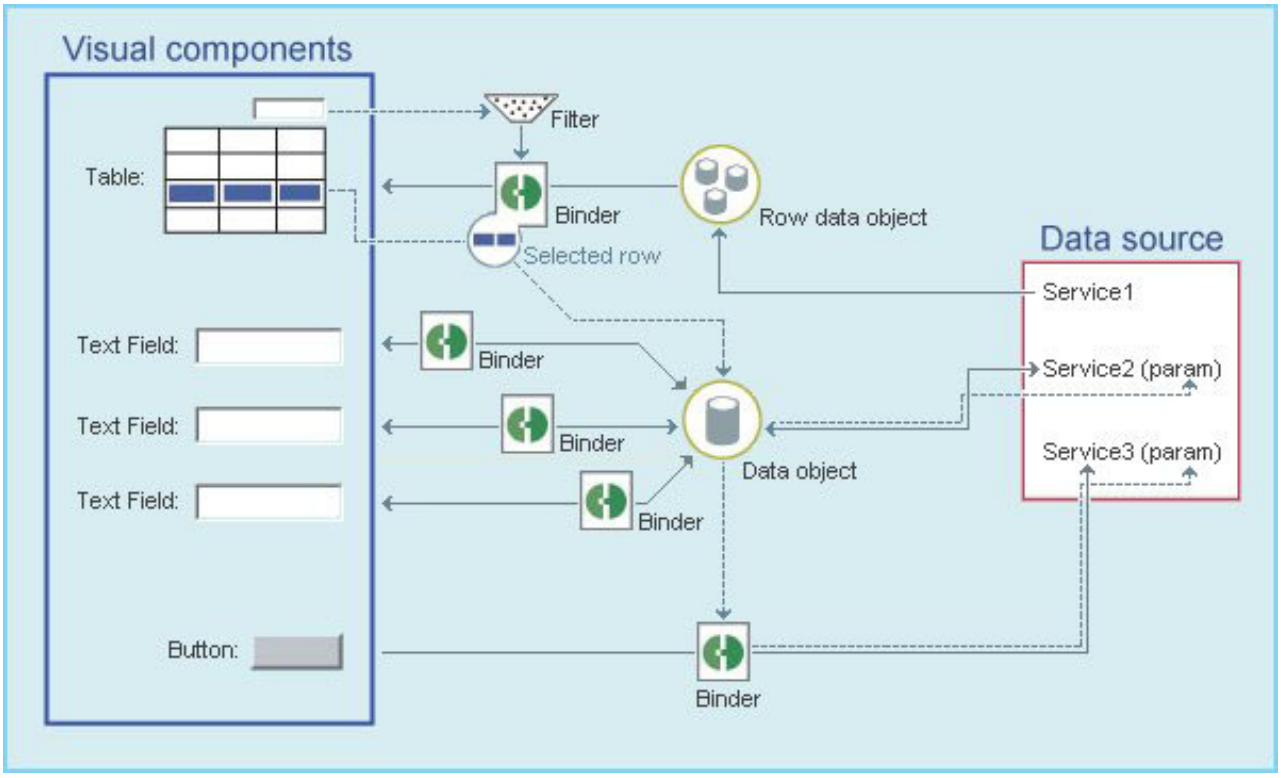


Figura 1. Esse Diagrama Ilustra um Relacionamento de Amostra entre Componentes Visuais, Conectores, Objetos de Dados e Origens de Dados

Na Figura 1, cada componente visual possui seu próprio binder, que o associa a um objeto de dados ou, no caso do botão, a uma origem de dados. Os binders para os campos de texto ligam o campo a uma propriedade específica do objeto de dados. Tanto o objeto de dados de linha quanto o objeto de dados neste diagrama obtêm seus dados de chamadas diretas a um serviço na origem de dados. O objeto de dados para os campos de texto utiliza um valor chave a partir da linha selecionada na tabela como seu argumento para chamar Service2, o que retorna um registro completo que, presumidamente, inclui informações adicionais sobre a linha selecionada na tabela. Esse registro completo, por sua vez, é utilizado como argumento para o binder de ação do botão ao chamar Service3, que pode ser um método que atualiza os valores digitados nos campos. Para obter explicações detalhadas dos objetos, binders e origens de dados, siga os links fornecidos anteriormente.

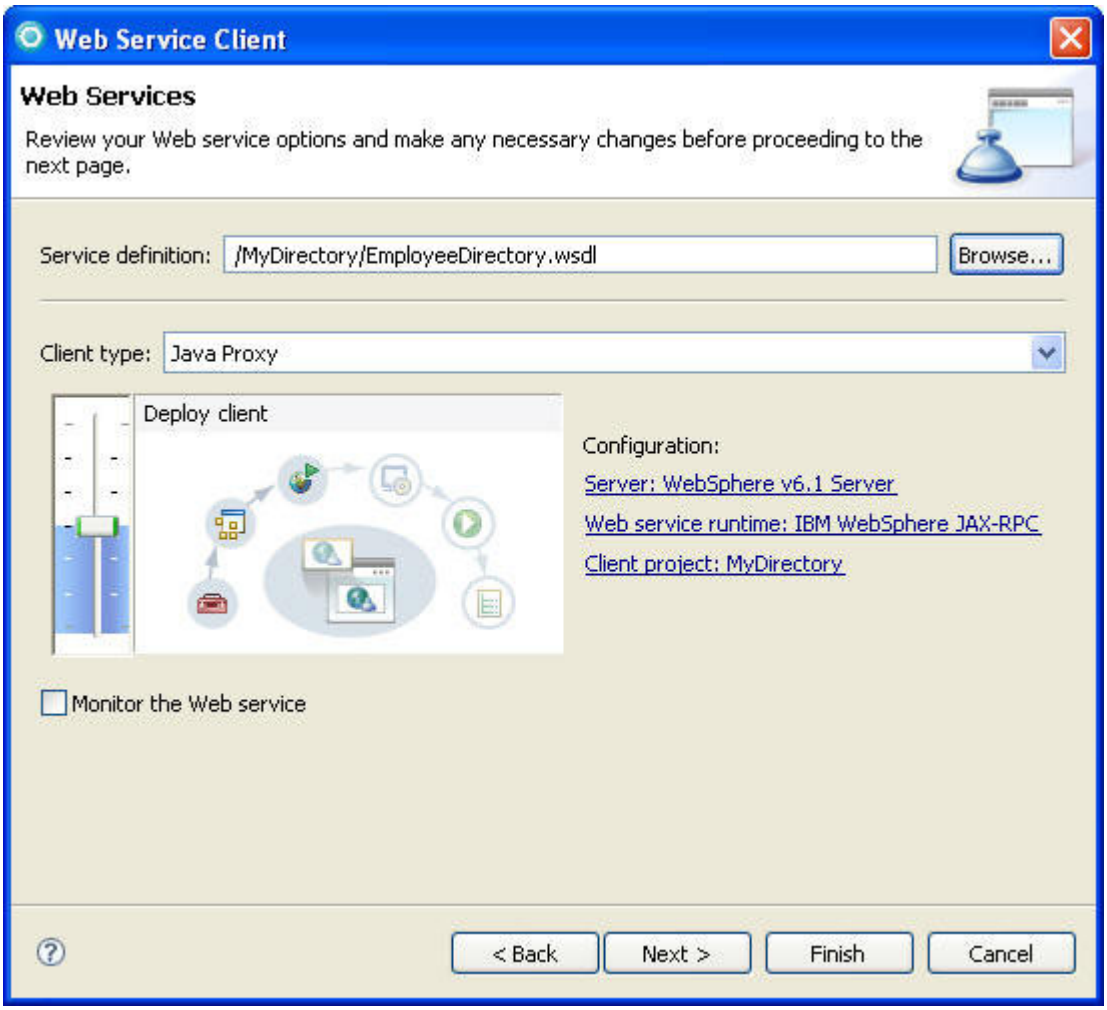
Gerar um Proxy do Serviço da Web em seu Projeto Utilizando o Arquivo WSDL Fornecido

Para trabalhar com o serviço da Web em execução em um servidor, o aplicativo Java requer um proxy Java ou cliente com o qual interagir. Utilizando um arquivo WSDL, é possível gerar um proxy Java no projeto Java, utilizando o assistente Cliente de Serviço da Web. O projeto MyDirectory inclui o arquivo EmployeeDirectory.wsdl, que será utilizado para gerar esse proxy. Após gerar o proxy Java, é possível criar uma origem de dados que representa o serviço da Web e inicia a ligação dos componentes visuais.

Importante: O arquivo WSDL utilizado no exercício assume que você implementou o serviço da Web em uma instalação local do WebSphere Application Server e utilizou a porta padrão para host local (http://localhost:9080). Se você implementou o arquivo EAR de maneira diferente, é necessário editar o arquivo WSDL adequadamente, antes de prosseguir.

Para gerar o proxy Java de serviço da Web no projeto:

1. No menu principal, clique em **Arquivo (File)** → **Novo (New)** → **Outro (Other)** e selecione o assistente **Serviços da Web (Web Services)** → **Cliente de Serviço da Web (Web Service Client)**. Se a categoria Serviços da Web não estiver sendo mostrada, selecione **Mostrar Todos os Assistentes (Show all wizards)**.
2. Utilize o assistente para definir o cliente de serviço da Web:
   a. Na **Definição de Serviço (Service definition)**, digite o arquivo WSDL fornecido em seu projetp MyDirectory: `/MyDirectory/EmployeeDirectory.wsdl`
   b. No campo **Tipo de Cliente (Client type)**, selecione **Java proxy**.
   c. Configure a barra deslizante para **Implementar cliente (Deploy client)**.
   d. Certifique-se de que o servidor e o tempo de execução do serviço da Web estejam configurados adequadamente no servidor que estiver sendo executado. Esse tutorial foi testado de acordo com WebSphere v6.0 e WebSphere v6.1 com o tempo de execução IBM WebSphere JAX-RPC.
   e. Certifique-se de que o cliente proxy Java seja a saída para o projeto MyDirectory.



3. Clique em **Concluir (Finish)**. O assistente Cliente de Serviço da Web gera o proxy Java em um novo pacote (directory.service) no projeto.

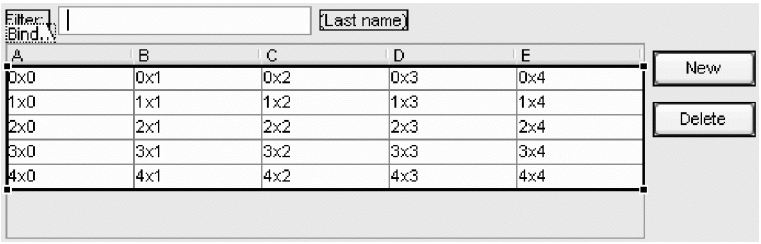
Ligar a employeesTable a um Objeto de Dados de Linha Retornado pelo Serviço da Web

Como employeesTable é o primeiro componente visual sendo ligado neste aplicativo, é necessário criar uma origem de dados que aponte ao proxy do serviço da Web de amostra, incluído no projeto. Quando

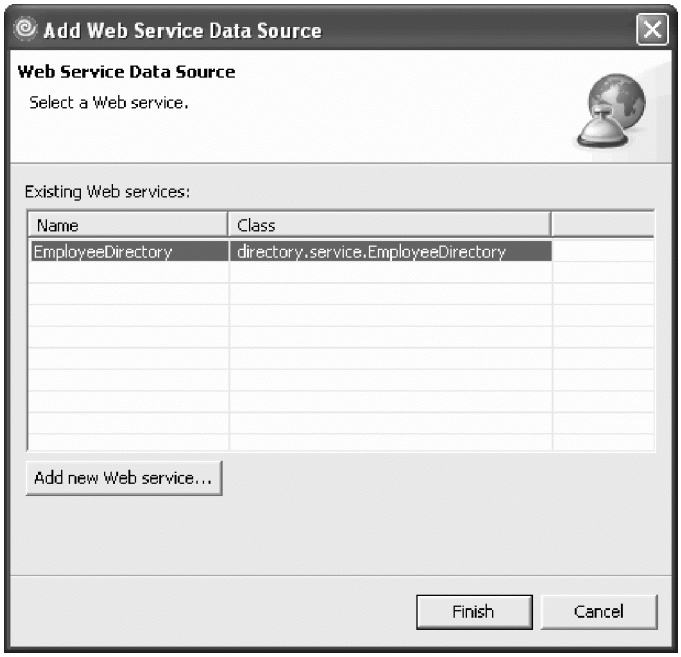
você ligar outros componentes visuais em exercícios posteriores, você reutilizará a origem de dados. Nesta etapa, você inclui a origem de dados do serviço da Web e o objeto de dados lightEmployeeRecordRows.

Para ligar a tabela de funcionários:

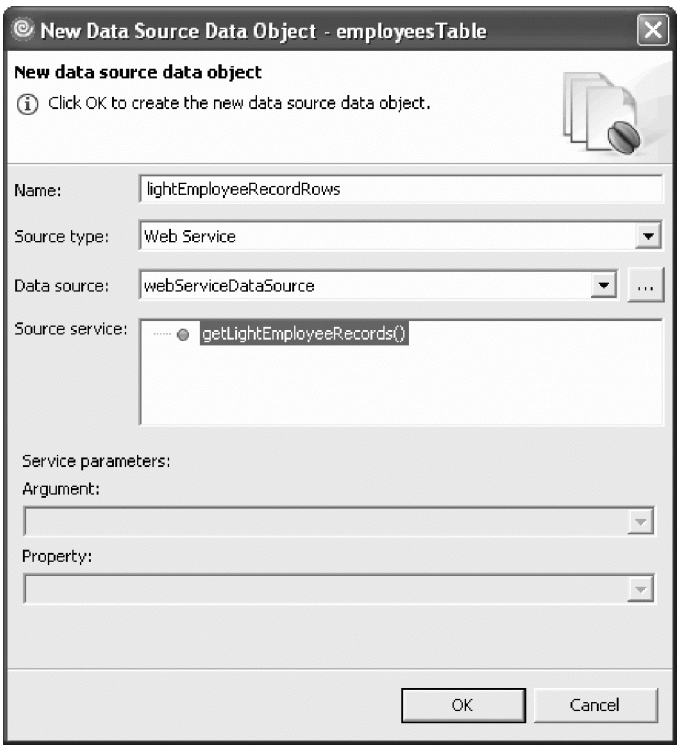
1. Na visualização de Java Beans ou de design, selecione employeesTable. (Certifique-se de não selecionar o pai JScrollPane dele). Uma pequena guia etiquetada **Ligação (Bind)** é mostrada no topo de employeesTable na área de design.



2. Clique na guia **Ligação (Bind)** no employeesTable. Também é possível clicar com o botão direito em employeesTable e selecionar **Propriedades de Ligação (Binding Properties)**.
3. Como ainda não há objetos de dados em seu aplicativo, é necessário incluir um novo. Clique em **Novo Objeto de Dados da Origem de Dados (New Data Source Data Object)**.
4. No campo **Tipo de Origem (Source type)**, selecione **Serviço da Web (Web service)**.
5. Como você ainda não incluiu a origem de dados do serviço da Web em seu aplicativo, é necessário inclui-la agora. Próximo ao campo **Origem de Dados (Data source)**, clique no botão **...**, para abrir a caixa de diálogo Incluir Origem de Dados do Serviço da Web (Add Web Service Data Source), que procura por clientes de serviços da Web ou proxies disponíveis no projeto.
6. Selecione o serviço da Web EmployeeDirectory e clique em Concluir (Finish). Uma nova origem de dados é incluída no arquivo DirectoryApp.java.

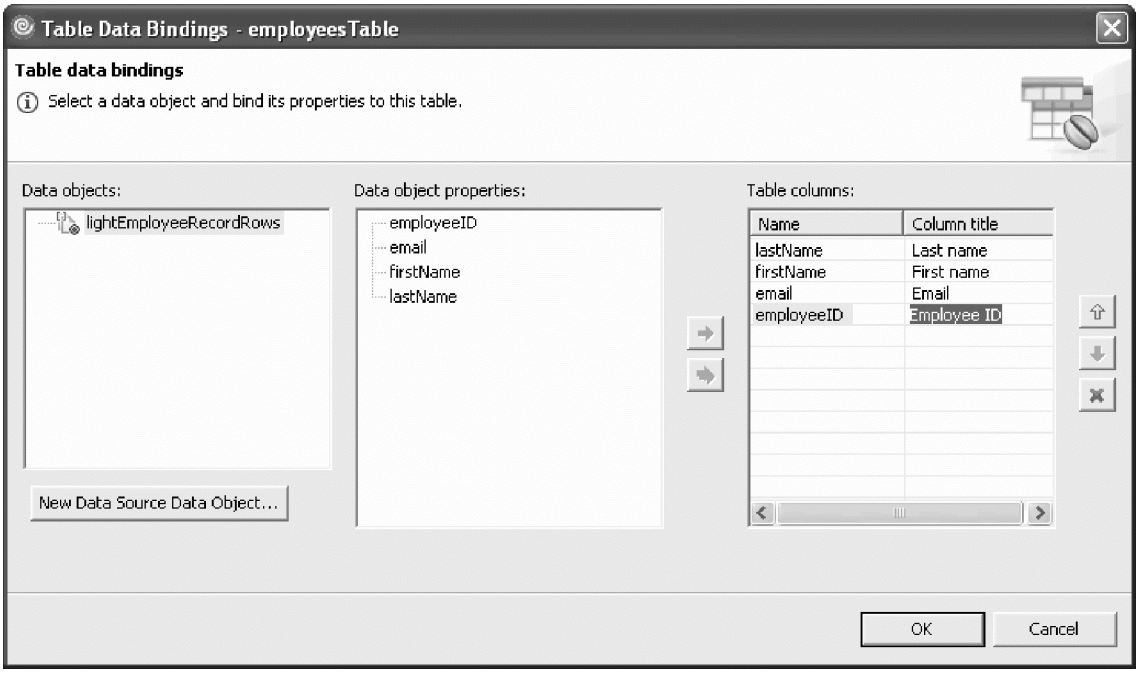


7. Na caixa de diálogo Novo Objeto de Dados de Origem de Dados (New Data Source Data Object), selecione getLightEmployeeRecords() no campo Serviço de Origem (Source service) e aceite o nome padrão para o novo objeto de dados: lightEmployeeRecordRows. Nenhum parâmetro é necessário para esse método de serviço. Clique em OK. O novo objeto de dados é criado e exibido na área de formato livre da visualização design.



Dica: Como você está ligando uma tabela, a caixa de diálogo Novo Objeto de Dados da Origem de Dados (New Data Source Data Object) exibe apenas os serviços que retornam objetos de dados de linha. Nesse caso, o método getLightEmployeeRecords() é o único serviço disponível que retorna uma matriz de objetos.

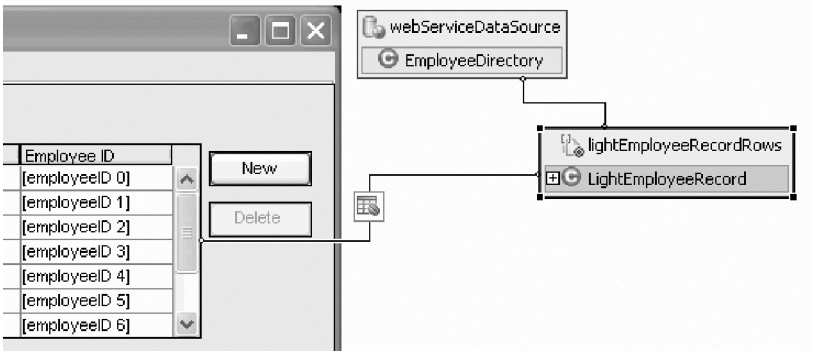
8. Na caixa de diálogo Ligações de Dados da Tabela (Table Data Bindings), selecione o objeto de dados lightEmployeeRecordRows.
9. Agora, é necessário selecionar as propriedades do objeto de dados lightEmployeeRecordRows que você deseja exibir em employeesTable:



a. Clique no botão de seta dupla ⇨, para incluir todas as propriedades de objetos na lista **Colunas da Tabela (Table columns)**.
b. Utilize as setas para cima e para baixo para ajustar as colunas na seguinte ordem, de cima para baixo: lastName, firstName, email, employeeID
c. Renomeie os títulos das colunas: `Sobrenome (Last name)`, `Primeiro Nome (First name)`, `E-mail`, `ID do Funcionário (Employee ID)`

   Dica: Após concluir a ligação da tabela, você pode sempre voltar, a qualquer momento, às propriedades de ligação e renomear e reordenar as colunas.
d. Clique em **OK**.

Agora o employeesTable está ligado no objeto de dados lightEmployeeRecordRows utilizando um JRowTableBinder. Se você clicar no objeto de dados lightEmployeeRecordRows na área livre, o visual editor traça uma linha do objeto de dados à tabela. Na linha, o JRowTableBinder é representado pelo ícone do binder de tabela . Outra linha indica que o objeto de dados utiliza o webServiceDataSource como origem de dados.



Ponto de Verificação da Lição

Avisa as alterações ao projeto e aplicativo. Durante essa lição, você incluiu a origem de dados do serviço da Web, um objeto de dados de linha e um binder que liga employeesTable ao objeto de dados de linha.

Examine o novo pacote (jve.generated), criado no projeto para manter todas as classes de binder geradas pelo editor visual de Java. Observe também o novo pacote (directory.service), que mantém o proxy Java para o serviço da Web. Descreva ou resuma o que foi aprendido nessa lição.

Agora, quando você executa o aplicativo My Company Directory, a tabela de funcionários é preenchida pelo serviço da Web com os registros de funcionário existentes.

Lição 2.3: Ligar os Campos de Detalhes à Seleção da Tabela

No exercício anterior, você ligou employeesTable ao objeto de dados lightEmployeeRecordRows retornado pelo serviço getLightEmployeeRecords() no serviço da Web. É necessário, agora, ocupar os campos de detalhes com base no funcionário selecionado na tabela.

Para obter detalhes extras de cada funcionário selecionado, é utilizado um outro objeto de dados. O objeto de dados selectedEmployeeRecord que será incluído é retornado pelo serviço

getFullEmployeeRecord(). Este serviço toma o ID do funcionário selecionado na tabela como um parâmetro e pega os detalhes adicionais sobre o funcionário, incluindo número de telefone e local de trabalho.

O JRowTableBinder, utilizado ao ligar a tabela para o objeto de dados da linha, simplifica esta etapa. JRowTableBinder expõe o elemento selecionado na tabela como um objeto de dados separado, que pode ser utilizado como parâmetro para o método getFullEmployeeRecord(java.lang.Integer). Então, você pode ligar facilmente cada um dos campos de texto em sua propriedade correspondente no objeto de dados selectedEmployeeRecord.

Aprenda Mais Sobre esse Serviço da Web: O serviço da Web inclui dois serviços para obter todos os detalhes de cada funcionário. A tabela lista todos os funcionários e apenas um subconjunto de dados é exibido na tabela. Portanto, quando um único funcionário é selecionado, é possível recuperar o restante das informações de funcionário apenas daquele selecionado. Se o serviço da Web enviou todos os dados de cada funcionário quando a tabela solicitou dados, o tráfego da Web pode ficar pesado e causar um baixo desempenho do aplicativo.

Por exemplo, se o registro do funcionário possui uma foto ou um anexo, você não deseja recuperar as fotos quando está simplesmente obtendo a lista completa de funcionários. Assim, o serviço getLightEmployeeRecord é utilizado para preencher a tabela e o getFullEmployeeRecord obtém o registro completo para o funcionário selecionado na tabela.

Ligar o Campo Sobrenome

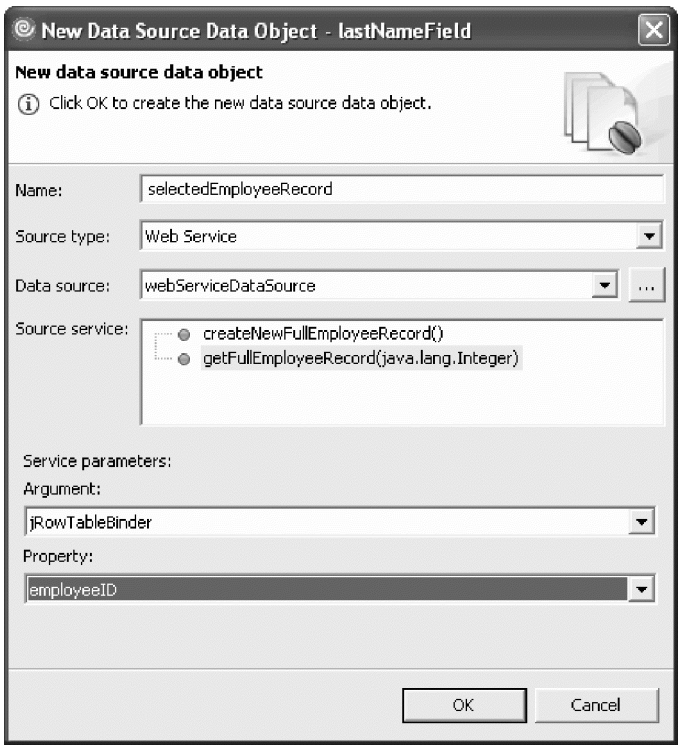
Nessa etapa, você ligará o campo **Sobrenome (Last name)** à propriedade lastName no objeto de dados selectedEmployeeRecord:

1. Na exibição Java Beans ou na exibição de design, selecione JTextField para o sobrenome (lastNameField). A área de design mostra uma guia **Ligação (Bind)** no campo de texto.

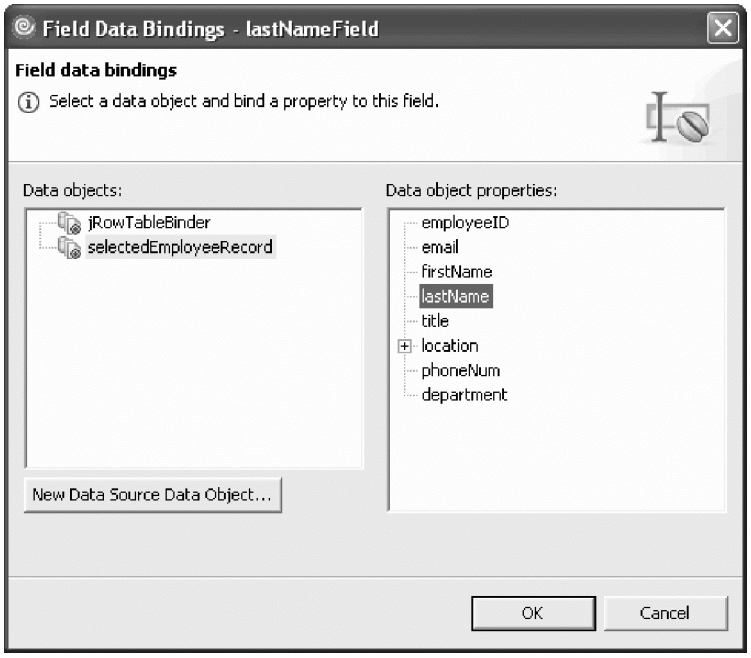
   

2. Clique na guia **Ligação (Bind)**, para abrir a caixa de diálogo Ligações de Dados do Campo (Field Data Bindings).
3. Clique em **Novo Objeto de Dados da Origem de Dados (New Data Source Data Object)**. Embora o objeto de dados jRowTableBinder existente retorne o último nome correto, ele não inclui o registro completo do funcionário. É necessário criar um novo objeto de dados que representa o registro completo do funcionário.
4. No campo **Tipo de Origem (Source type)**, certifique-se de que **Serviço da Web (Web Service)** está selecionado e, para **Origem de Dados (Data source)**, certifique-se de que **webServiceDataSource** está selecionado.
5. Na lista **Serviço de Origem (Source service)**, selecione getFullEmployeeRecord(java.lang.Integer). A caixa de diálogo Novo Objeto de Dados da Origem de Dados (New Data Source Data Object) lista os serviços que retornam objetos de dados compatíveis com um campo de texto.
6. No campo **Nome (Name)**, digite `selectedEmployeeRecord`.
7. No campo **Argumento (Argument)**, selecione jRowTableBinder e no campo **Propriedade (Property)**, selecione employeeID. O ID do funcionário da linha selecionada agora é definido para ser o argumento para o método do serviço getFullEmployeeRecord().

   Nota: O getFullEmployeeRecord(java.lang.Integer) requer um inteiro como um argumento. Você deseja utilizar o ID de funcionário da seleção atual na tabela de funcionários para recuperar um registro completo. Quando você liga a tabela, o Visual Editor gera automaticamente o jRowTableBinder, o qual também atende a seleção atual na tabela de funcionários. Para o parâmetro inteiro, você utilizará o employeeID da linha selecionada em jRowTableBinder.



8. Clique em **OK**.
9. No dialogo Ligações de Dados do Campo (Field Data Bindings), certifique-se de que selectedEmployeeRecord esteja selecionado na lista **Objetos de Dados (Data objects)**. Observe que há mais propriedades disponíveis para o objeto de dados selectedEmployeeRecord do que para o objeto de dados jRowTableBinder.
10. Na lista **Propriedades do Objeto de Dados (Data object properties)**, selecione a propriedade lastName.



11. Clique em **OK**. O campo de sobrenome no aplicativo agora é ligado à propriedade lastName do objeto de dados selectedEmployeeRecord, retornado por getFullEmployeeRecord().

O novo objeto de dados chamado selectedEmployeeRecord é criado e incluído em seu aplicativo. Uma representação visual do objeto de dados é incluída na área livre da exibição de design, como mostrada na imagem a seguir:

Agora, ao selecionar o campo lastName na área de design, uma linha indica que ele está ligado ao selectedEmployeeRecord. Na metade da linha, o ícone do binder de texto representa SwingTextComponentBinder, utilizado para essa ligação. Se você selecionar a linha ou ícone representando o binder na área de design, você pode examinar suas propriedades na exibição Propriedades.

Ligar os Campos de Detalhes Restantes

Para ligar cada um dos campos de detalhes restantes para um funcionário, você seguirá um processo semelhante ao do campo de sobrenome, mas não será necessário incluir o objeto de dados. Como você já incluiu o objeto de dados selectedEmployeeRecord, você pode, facilmente, ligar cada um dos campos de texto em sua propriedade correspondente no objeto de dados selectedEmployeeRecord.

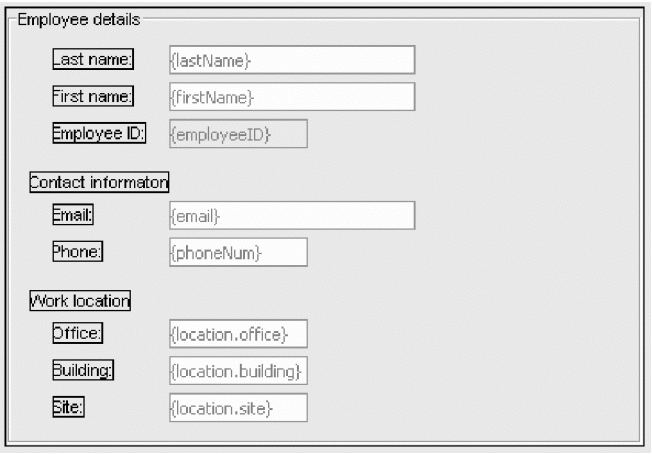
Para ligar os campos, conclua as etapas a seguir para cada um dos campos na seção Detalhes do Funcionário do aplicativo:

1. Selecione o campo na exibição de design e clique na guia **Ligação (Bind)**.
2. No diálogo Ligações de Dados do Campo (Field Data Bindings), selecione selectedEmployeeRecord a partir da lista **Objetos de Dados (Data objects)**.
3. Na lista **Propriedades do Objeto de Dados (Data object properties)**, selecione a propriedade apropriada para o campo que você está ligando. O gráfico a seguir mostra a propriedade que cada campo de texto precisa ser ligado a:

Campo	**Propriedade no Objeto de Dados selectedEmployeeRecord**
lastNameField	lastName
firstNameField	firstName
idField	employeeID
emailField	email
phoneField	phoneNum
officeField	location.office
buildingField	location.building
siteField	location.site

4. Clique em **OK**.

Ao concluir a ligação dos campos de texto, a área de design deve ter a aparência da imagem a seguir:



Transformar o Campo ID do Funcionário em Campo de Leitura

O campo do ID do Funcionário (employee ID) está desativado, pois a propriedade editável no campo está definida como false. No entanto, o comportamento padrão do binder do campo de texto altera o estado ativado no campo quando o objeto de dados contém um valor. É possível desligar esse comportamento do binder para que o campo permaneça no estado inicial de leitura.

Para evitar que o binder alterne automaticamente a propriedade editável:

1. Selecione o campo ID do Funcionário (Employee ID). É exibida uma linha na área de design com um ícone representando o binder para o campo.
2. Clique no ícone do binder para o campo ID do Funcionário (Employee ID).
3. Na exibição Propriedades (Properties), altere a propriedade autoEditable para **false**. Pressione **Enter**.

Ponto de Verificação da Lição

Agora, ao executar o aplicativo e selecionar um funcionário da tabela, os detalhes do registro do funcionário são exibidos nos campos de detalhes.

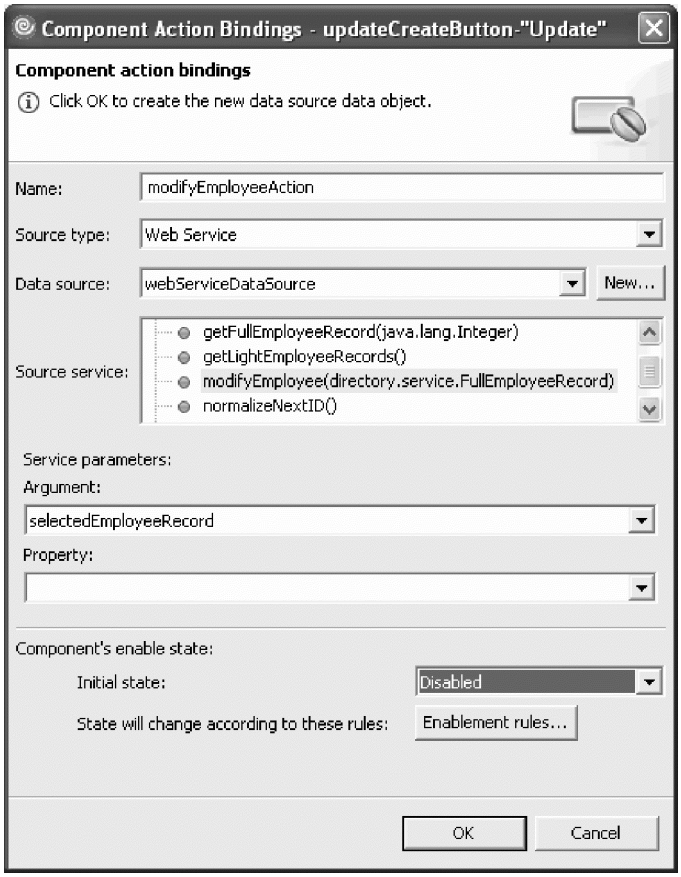
Lição 2.4: Ligar o Botão Atualizar a um Conector de Ação

O Java visual editor fornece binders de ações para chamar um serviço em uma origem de dados, quando um botão é clicado. Por exemplo, ao clicar no botão Atualizar (Update), o aplicativo deve executar um método modifyEmployee() no serviço da Web com as alterações digitadas nos campos de detalhes. Nessa lição, você ligará o botão Atualizar (Update) a um binder de ação.

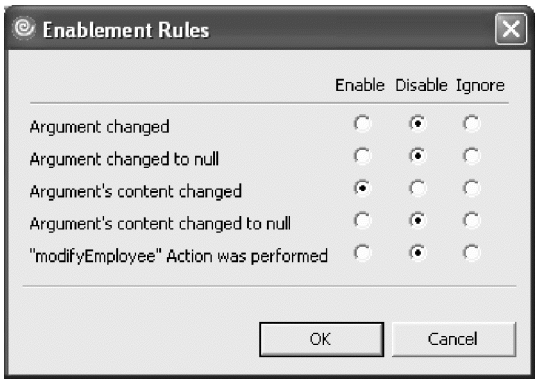
Para ligar o botão Atualizar (Update):

1. Selecione o botão **Atualizar (Update)** na área de design e clique na guia **Ligação (Bind)**, para abrir a caixa de diálogo Ligações de Ação do Componente (Component Action Bindings).

2. No campo **Tipo de Origem (Source type)**, selecione **Serviço da Web (Web Service)**.
3. No campo **Origem de Dados (Data source)**, selecione **webServiceDataSource**.
4. Na lista **Serviço de Origem (Source service)**, selecione **modifyEmployee(directory.service.FullEmployeeRecord)**.
5. O campo **Nome (Name)** é alterado automaticamente para **modifyEmployeeAction**. Aceite este padrão.
6. No campo **Argumento (Argument)**, selecione **selectedEmployeeRecord**.
7. Como o método modifyEmployee() toma um registro completo do funcionário como argumento, você deve deixar o campo **Propriedade (Property)** em branco.
8. Configure o **Estado Inicial (Initial state)** do botão como **Desativado (Disabled)**.



9. Para definir como o botão altera seu estado, clique em **Regras de Ativação (Enablement rules)**. Especifique se o botão deve ser ativado apenas quando o conteúdo do argumento é alterado e desativado em todos os outros casos. Clique em **OK**.



   Isto significa que o botão **Atualizar (Update)** é desativado até que o conteúdo de selectedEmployeeRecord seja alterado. Em outras palavras, assim que você digitar um novo valor em um dos campos de detalhes, ligados a selectedEmployeeRecord, o binder ativa o botão. Se você selecionar um novo registro ou clicar em **Atualizar (Update)**, o botão será desativado novamente.

10. Clique em **OK**.

Um novo binder SwingDataServiceAction é incluído para o botão **Atualizar (Update)**. Se você selecionar o botão na área de design, o visual editor traça uma linha que indica que o botão está ligado à origem de dados do serviço da Web. Uma seta pontilhada rosa aponta do objeto selectedEmployeeRecord à linha. Esta seta indica que selectedEmployeeRecord é o argumento para a chamada ao serviço.

Ponto de Verificação da Lição

Agora, ao executar o aplicativo, é possível atualizar um registro de funcionário.

Selecione um funcionário na tabela e altere o sobrenome. Assim que você modificar o sobrenome, o botão **Atualizar (Update)** será ativado. Ao clicar em **Atualizar (Update)**, o serviço modifyEmployee é chamado e o funcionário é atualizado. O novo sobrenome é refletido na tabela de funcionários.

Lição 2.5: Ativar o Botão Excluir e a Caixa de Diálogo de Confirmação

Neste exercício, você programará o aplicativo My Company Directory para excluir um registro de funcionário.

A lista a seguir descreve o comportamento que deseja que o aplicativo utilize:

- Ao selecionar um funcionário na tabela, o botão **Excluir (Delete)** é ativado.
- Ao clicar no botão **Excluir (Delete)**, a caixa de diálogo Confirmar Exclusão (Confirm Delete) é aberto e solicita uma confirmação de exclusão.
- Se você clicar no botão **Sim (Yes)** na caixa de diálogo Confirmar Exclusão (Confirm Delete), o registro do funcionário é excluído, a caixa de diálogo é fechada e a lista de funcionários é atualizada.
- Se você clicar em **Não (No)**, a exclusão é cancelada e a caixa de diálogo Confirmar Exclusão (Confirm Delete) é fechada.

Programar o Botão Excluir para Ser Ativado ou Desativado Dependendo da Seleção de uma Linha na Tabela

Para programar o botão Excluir (Delete) para ser ativado ou desativado, inclua um listener na tabela que ativa o botão quando uma linha é selecionada.

1. Selecione a employeesTable na visualização Java Beans. A exibição de origem destaca a seguinte linha:
   ```
   employeesTable = new JTable();
   ```
2. Imediatamente após esta linha, inclua um novo evento ListSelectionListener e valueChanged em employeesTable:
   ```
   employeesTable.getSelectionModel().addListSelectionListener(new ListSelectionListener() {
       public void valueChanged(ListSelectionEvent e) {
          getDeleteButton().setEnabled(getEmployeesTable().getSelectedRowCount() != 0);
      }
   });
   ```
3. Após incluir estas linhas de código, o editor de origem marca-as como erros até que você importe ListSelectListener e ListSelectionEvent. Para incluir as importações requeridas, clique em **Origem (Source)** → **Organizar Importações (Organize Imports)** no menu principal. As linhas a seguir são incluídas na seção de importações da classe:
   ```
   import javax.swing.event.ListSelectionEvent;
   import javax.swing.event.ListSelectionListener;
   ```

Agora, ao selecionar uma linha na tabela, o botão **Excluir (Delete)** é ativado.

Programar a Caixa de Diálogo Confirmar Exclusão para Abrir ao Clicar em Excluir

Inclua um evento actionPerformed no botão Excluir (Delete) e programe o evento para abrir a caixa de diálogo Confirmar Exclusão (Confirm Delete):

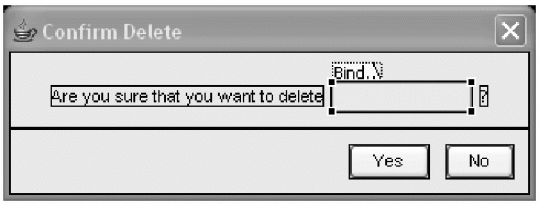
1. Clique com o botão direito do mouse no botão **Excluir (Delete)** e selecione **Eventos (Events)** → **actionPerformed**. O stub de evento a seguir está incluído no método getDeleteButton():
   ```
   deleteButton.addActionListener(new java.awt.event.ActionListener() {
       public void actionPerformed(java.awt.event.ActionEvent e) {
        System.out.println("actionPerformed()");
        // TODO Auto-generated Event stub actionPerformed()
      }
   });
   ```
2. Substitua esse stub gerado pelo código a seguir, que configura a caixa de diálogo Confirmar Exclusão (Confirm Delete) para ser visível quando o botão é clicado:

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

Ligar o Campo de Texto na Caixa de Diálogo Confirmar Exclusão

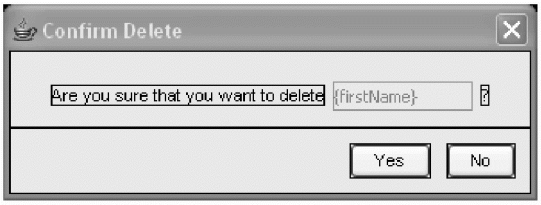
Ligue o campo de texto na caixa de diálogo Confirmar Exclusão (Confirm Delete) para exibir o primeiro nome do funcionário a ser excluído.

1. Na visualização Java Beans ou na área de design, selecione o campo de texto employeeToDeleteField e clique na guia **Ligação (Bind)**.



2. Na caixa de diálogo Ligações de Dados do Campo (Field Data Bindings), selecione o objeto de dados selectedEmployeeRecord e o campo firstName e, em seguida, clique em **OK**.

   Agora o campo de texto está ligado à coluna firstName da linha selecionada na employeesTable.



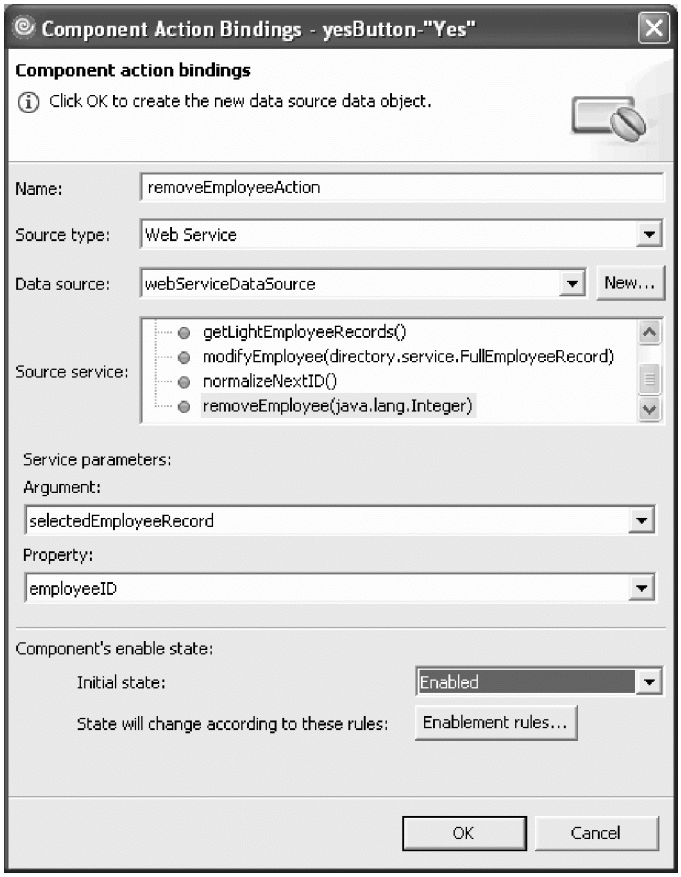
3. Para certificar-se de que este campo é de leitura, defina a propriedade **autoEditable** para o binder do campo como **false**.

Ligar o Botão Sim para Executar a Exclusão

Ligue o botão **Sim (Yes)** para chamar o método removeEmployee(java.lang.Integer) no serviço da Web.

1. Selecione o botão **Sim (Yes)** e clique na guia **Ligação (Bind)**, para abrir o diálogo Ligações de Ação do Componente (Component Action Bindings).
2. No campo **Tipo de Origem (Source type)**, selecione **Serviço da Web (Web Service)**.
3. No campo **Origem de Dados (Data source)**, selecione **webServiceDataSource**.
4. Na lista **Serviço de Origem (Source service)**, selecione **removeEmployee(java.lang.Integer)**.
5. O campo **Nome (Name)** é alterado automaticamente para **removeEmployeeAction**. Aceite este padrão.
6. No campo **Argumento (Argument)**, selecione **selectedEmployeeRecord**.
7. No campo **Propriedade (Property)**, selecione **employeeID**. Como o método removeEmployee() toma um inteiro como argumento, utilize o ID do funcionário de selectedEmployeeRecord.
8. Defina **Estado Inicial (Initial state)** do botão como **Ativado (Enabled)**.
9. Para as **Regras de Ativação (Enablement rules)**, selecione **Ignorar (Ignore)** para cada uma das condições.

   Esse estado do componente significa que o botão Sim (Yes) estará sempre ativado, porque não há necessidade de alterar o estado.



10. Clique em **OK**.

Incluir um Evento para Ocultar o Diálogo Confirmar Exclusão Após Excluir o Funcionário

Nesta etapa, você inclui um evento ao *binder* do botão **Sim (Yes)** (não ao botão **Sim (Yes)** em si). Você deseja que a caixa de diálogo Confirmar Exclusão (Confirm Delete) feche após remover o funcionário, o que significa após o binder ter chamado com êxito o serviço na origem de dados.

Incluir o seguinte código ao método 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) {}
});
```

Este código de evento oculta a caixa de diálogo Confirmar Exclusão (Confirm Delete) após executar a ação do binder.

Ponto de Verificação da Lição

Agora, quando você executar o aplicativo My Company Directory, pode selecionar um funcionário na tabela, clicar no botão **Excluir (Delete)** e clicar em **Sim (Yes)** para confirmar a exclusão. O registro do funcionário será removido do diretório e a lista de funcionários refletirá a remoção.

Lição 2.6: Configurar Ações e Ligações para Incluir um Novo Funcionário

Nessa lição, você ativa o aplicativo My Company Directory para incluir um novo registro de funcionário.

Devido ao comportamento do aplicativo ser mais complicado e dinâmico para incluir um novo funcionário, este exercício é de natureza mais complexa e requer algumas alterações manuais no código fonte. Além disso, este exercício demonstra alguns recursos avançados dos objetos de dados e fornece um exemplo criativo de maneira que você pode utilizar binders e objetos de dados para ajustar às suas necessidades.

A lista a seguir descreve o comportamento requerido do aplicativo:

- Ao clicar no botão **Novo (New)**, o seguinte comportamento ocorre:
  - A seleção é limpa na tabela de funcionários e a tabela está desativada.
  - A limpeza da seleção da tabela causa a desativação do botão **Excluir (Delete)**.
  - O campo **Filtro (Filter)** é desativado.
  - Os campos de detalhes são limpos de quaisquer valores, exceto um novo ID do funcionário.
  - O texto no botão **Atualizar (Update)** é alterado para **Incluir (Add)**.
- Ao clicar no botão **Incluir (Add)**, o seguinte comportamento ocorre:
  - Os valores digitados nos campos de detalhes são incluídos no diretório como um novo registro do funcionário.
  - A tabela é ativada e os valores são atualizados.
  - O campo **Filtro (Filter)** é ativado.
  - O texto no botão **Incluir (Add)** é alterado para **Atualizar (Update)**.

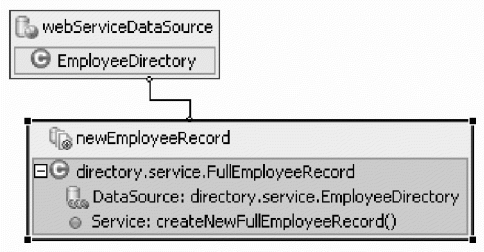
Incluir um Novo Objeto de Dados da Origem de Dados, Que Chama createNewFullEmployeeRecord()

O serviço da Web de amostra fornece um serviço createNewFullEmployeeRecord, que fornece um novo registro de funcionário em branco, ocupado com o próximo número de ID de funcionário disponível. Este registro em branco pode, então, ser ocupado com novas informações do funcionário e enviado de volta ao serviço da Web.

1. Na paleta do Java Visual Editor, expanda a gaveta Objetos de Dados e selecione **Objeto de Dados da Origem de Dados (Data Source Data Object)**.
2. Mova o ponteiro do mouse sobre a área em branco da visualização de design ou área livre e clique com o botão esquerdo para soltar o Objeto de Dados da Origem de Dados (Data Source Data Object). Um novo Objeto de Dados da Origem de Dados (Data Source Data Object) é incluído e mostrado na área livre:

3. Clique com o botão direito do mouse no Objeto de Dados da Origem de Dados (Data Source Data Object) e selecione o campo **Renomear (Rename)**. Renomeie o objeto de dados como `newEmployeeRecord`.
4. Clique com o botão direito no objeto de dados newEmployeeRecord e selecione **Propriedades de Ligação (Binding Properties)**. A caixa de diálogo Ligação dos Dados (Data Binding) é aberta.
5. No campo **Origem de Dados (Data source)**, selecione webServiceDataSource.
6. No campo **Serviço (Service)**, selecione createNewFullEmployeeRecord()
7. Clique em **OK**.

Na área de formato livre, você verá que o objeto de dados da origem de dados newEmployeeRecord está ligado ao serviço da Web.



Incluir um Objeto de Dados Básico para Facilitar a Comutação de Objetos de Dados

Como os campos de detalhes e o botão Atualizar (Update) precisam alternar modos (para executarem uma atualização e criarem um novo funcionário), é necessário que sejam ligados a dois objetos de dados diferentes, em momentos diferentes. Para facilitar esta etapa, você incluirá um Objeto de Dados Básico (Basic Data Object), denominado switchingDataObject. Você utilizará esse Objeto de Dados Básico (Basic Data Object) para alternar a ligação para os campos de texto entre selectedEmployeeRecord e newEmployeeRecord.

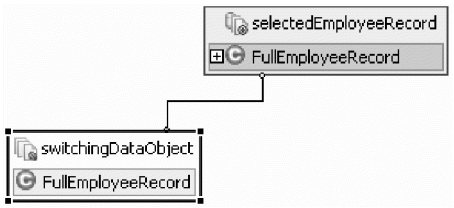
O novo Objeto de Dados Básico (Basic Data Object) simplesmente aponta para outro objeto de dados (selectedEmployeeRecord), definido no exercício anterior. Este novo objeto de dados se tornará útil ao criar um método que manda esse objeto de dados básico utilizar o newEmployeeRecord criado anteriormente. Em outras palavras, esse objeto de dados básico funcionará como um objeto de dados intermediário, que alterna entre o objeto de dados selectedEmployeeRecord e o objeto de dados newEmployeeRecord, permitindo que componentes visuais no aplicativo funcionem com dois objetos de dados diferentes.

1. Na paleta do visual editor, selecione **Objeto de Dados Básico (Basic Data Object)** e solte-o na área livre. Um basicDataObject é incluído.

2. Renomeie o objeto de dados para `switchingDataObject`
3. Na exibição Propriedades para switchingDataObject, defina a propriedade **sourceObject** como **selectedEmployeeRecord**. Você pode selecionar selectedEmployeeRecord no menu drop-down na coluna Valor para a propriedade.

   Agora, switchingDataObject refere-se a selectedEmployeeRecord e reflete os mesmos valores:

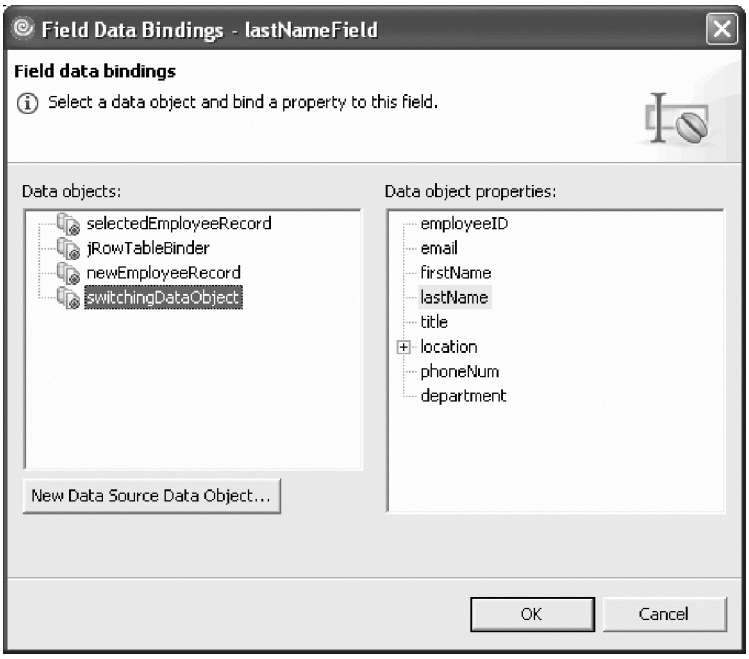
   

Religar Cada Campo do Funcionário ao switchingDataObject

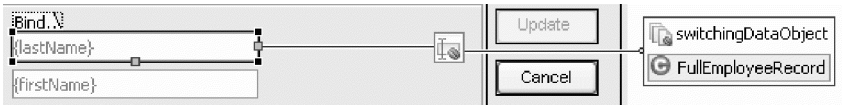
Apesar de cada campo de detalhe do funcionário já estar ligado a selectedEmployeeRecord, você irá ligá-los agora a switchingDataObject. Após ligar os campos, é possível, dinamicamente, alternar entre objetos de dados para os campos, dependendo de você estar modificando um registro de funcionário existente ou incluindo um novo registro de funcionário.

Para cada campo na seção Detalhes do Funcionário (Employee details), conclua as seguintes etapas:

1. Selecione o clique na guia **Ligar (Bind)**.
2. Na caixa de diálogo Ligações de Dados do Campo (Field Data Bindings), selecione switchingDataObject. Você ligou anteriormente os campos a selectedEmployeeRecord.



3. Certifique-se de que o campo ainda está ligado à propriedade do objeto de dados correto e clique em **OK**. Se você selecionar o campo na exibição de design, poderá ver que as linhas do binder apontam, agora, para switchingDataObject.



Definir um Sinalizador e um Método de Atualização e Modos de Comutação

O método updateMode() a seguir verifica se o sinalizador de modo está definido como novo, então altera algum comportamento do aplicativo adequadamente. Por padrão, o sinalizador Booleano isNewMode é definido como false e o método updateMode() ativa a tabela de funcionários e o campo de filtro e define o texto no botão Atualizar (Update) como "Atualizar" ("Update"). Se isNewMode estiver definido como true, a tabela de funcionários será desativada e limpa de qualquer seleção, o campo de filtro será desativado e o texto no botão Atualizar (Update) será definido como "Incluir" ("Add").

Inclua o seguinte código em sua classe DirectoryApp.java antes do último sinal de chave de fechamento:

```
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");
   }
}
```

Incluir um Evento actionPerformed ao Botão Novo

Nesta etapa, você inclui código de evento ao clicar no botão **Novo (New)**. O evento instrui switchingDataObject a utilizar o objeto de dados newEmployeeRecord, define o sinalizador de modo como "novo" e executa o método updateMode() incluído na etapa anterior.

1. Na visualização design, clique com o botão direito do mouse no botão **Novo (New)** e selecione **Eventos (Events)** → **actionPerformed**. O código a seguir é gerado pelo método getNewButton():

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

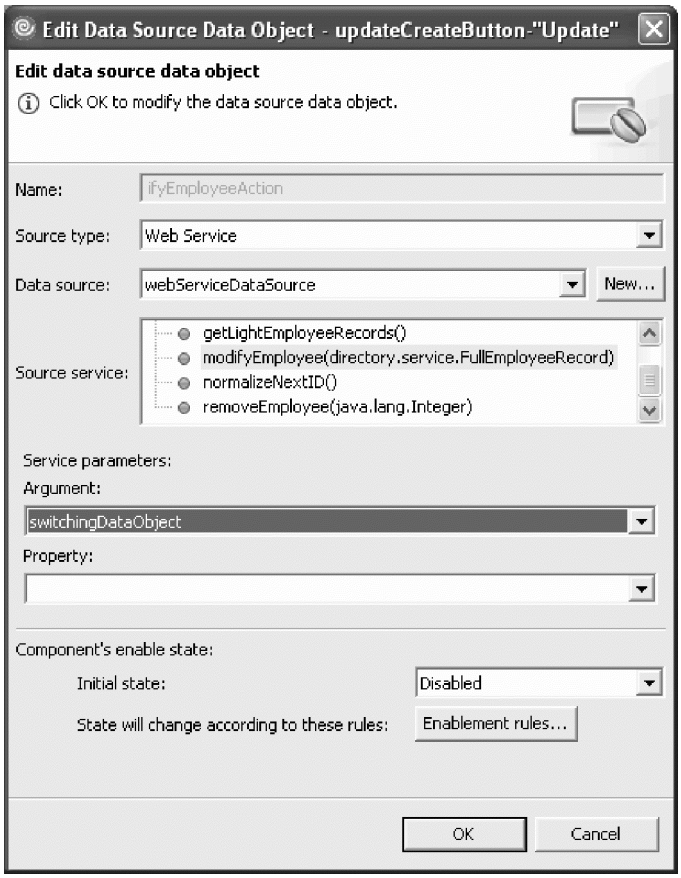
2. Substitua este stub gerado pelo seguinte código:

```
newButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
      getSwitchingDataObject().setSourceObject(getNewEmployeeRecord());
      getNewEmployeeRecord().refresh();
      isNewMode = true; //define o aplicativo como o modo novo
      updateMode(); //altera a UI de acordo com o modo
      getLastNameField().grabFocus();   }
});
```

Religar o Botão Atualizar

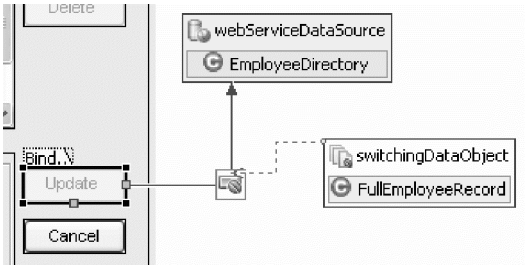
Em uma lição anterior, você programou o botão **Atualizar (Update)** para utilizar o método modifyEmployee no serviço da Web. Essa ação é implementada como um SwingDataServiceAction. Uma das propriedades do SwingDataServiceAction é o objeto de origem, o qual atua como o argumento para o serviço. O objeto de origem para a ação de modificação está atualmente definido como selectedEmployeeRecord. Para programar o botão para controlar uma atualização e uma inclusão, você reconfigurará a ação do botão para utilizar o switchingDataObject como um argumento para o serviço modifyEmployee.

1. Na exibição de design, selecione o botão **Atualizar (Update)**. Observe a seta pintada de rosa, mostrando que selectedEmployeeRecord é o argumento para a chamada de serviço.
2. Clique na guia **Ligação (Bind)** no botão **Atualizar (Update)**.
3. No campo **Argumento (Argument)**, selecione switchingDataObject.



4. Clique em **OK**.

Observe que a ação do botão está configurada para utilizar switchingDataObject como argumento para o método modifyEmployee:



Incluir um Evento ao Binder do Botão Atualizar para Reconfigurar o Modo

Após clicar no botão **Atualizar (Update)** e concluir a ação no serviço da Web, você deseja que o aplicativo retorne ao seu modo e comportamento padrão. Para fazer isso, você inclui um listener de evento no binder de ação do botão, que atualizará o modo e a tabela após executar a atualização ou inclusão.

Incluir o seguinte código ao método getModifyEmployeeAction() no botão Atualizar:

```
modifyEmployeeAction.addActionBinderListener(
  new jve.generated.IActionBinder.ActionBinderListener() {
     public void afterActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {
      if (isNewMode) {
         //Volte a utilizar selectedEmployeeRecord
         getSwitchingDataObject().setSourceObject(getSelectedEmployeeRecord());
         //Reversão do novo modo
         isNewMode = false;
         updateMode();
      }
      // Atualize o objeto de dados da tabela
      getLightEmployeeRecordRows().refresh();    }
     public void beforeActionPerformed(jve.generated.IActionBinder.ActionBinderEvent e) {}
});
```

Ponto de Verificação da Lição

Agora, quando você executa o aplicativo My Company Directory, pode clicar no botão **Novo (New)** e incluir um novo registro do funcionário.

Lição 2.7: Programar o Comportamento do Botão Cancelar

Ao utilizar seu aplicativo, você deseja ser capaz reverter facilmente quaisquer alterações iniciadas para um registro de funcionário, se decidir não submeter as alterações. Em outras palavras, você precisa cancelar e limpar os campos para que você possa iniciar novamente. Para incluir essa funcionalidade, defina alguns eventos actionPerformed no botão **Cancelar (Cancel)**.

A lista a seguir descreve o comportamento requerido do botão **Cancelar (Cancel)**:

- Se você clicar no botão **Cancelar (Cancel)** enquanto estiver no novo modo, o aplicativo será revertido do novo modo.
- Se você clicar no botão **Cancelar (Cancel)** enquanto modifica um registro do funcionário, quaisquer valores modificados são revertidos para os valores originais.

Para incluir um evento actionPerformed no botão **Cancelar (Cancel)** para executar o comportamento requerido:

1. Na visualização de design, clique com o botão direito do mouse no botão **Cancelar (Cancel)** e selecione **Eventos (Events)** → **actionPerformed**. O código a seguir é gerado pelo método getCancelButton():

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

2. Substitua o stub do evento gerado pelo código a seguir:

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

Ponto de Verificação da Lição

Nessa lição, você aprendeu como programar o botão **Cancelar (Cancel)** com eventos actionPerformed.

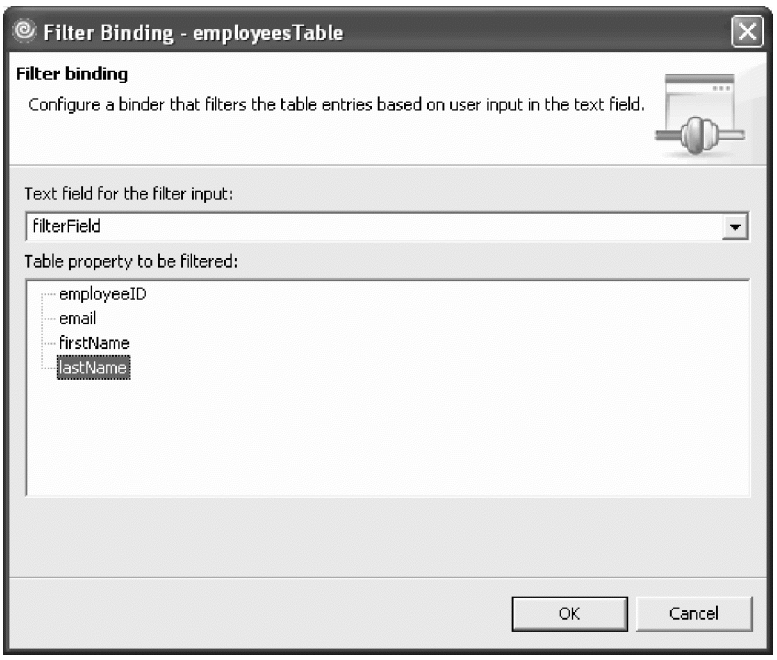
Lição 2.8: Configurar um Filtro na Tabela de Funcionários

É possível utilizar um Binder de Filtro de Texto para filtrar o conteúdo da tabela de funcionários. O filtro pega entrada de um campo de texto e filtra a tabela com base em uma propriedade particular ou coluna na tabela.

No aplicativo, você utilizará os caracteres digitados no campo **Filtro (Filter)** para filtrar pelo sobrenome do funcionário. Se os valores exatos digitados no campo **Filtro (Filter)** estiverem presentes no sobrenome de um registro do funcionário, o registro do funcionário será exibido na tabela.

Para criar um filtro para a tabela:

1. Selecione o ícone employeesTable e selecione **Filtrar Propriedades de Ligação**. A caixa de diálogo Ligação do Filtro é aberta.
2. Na lista **Campo de Texto para a Entrada do Filtro (Text field for the filter input)**, selecione **filterField**.
3. Na lista **Propriedade da Tabela a ser Filtrada (Table property to be filtered)**, selecione **lastName**.



4. Clique em **OK**.

Um novo SwingPropertyFilter é gerado. A propriedade do filtro no binder da tabela é definida para utilizar o novo filtro. O novo filtro é configurado para utilizar o campo **Filtro (Filter)** para a entrada e para filtrar a propriedade lastName da tabela.

Ponto de Verificação da Lição

Nessa lição você aprendeu como configurar um filtro para uma tabela.

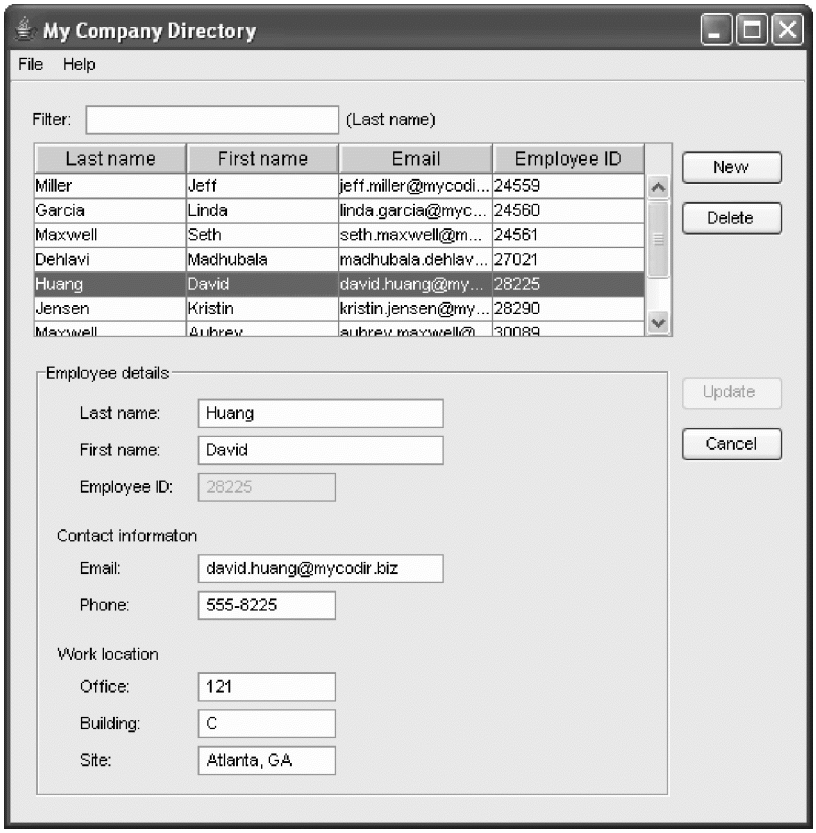
Agora, quando você executa o aplicativo My Company Directory, pode digitar caracteres no campo **Filtro (Filter)** e a tabela será filtrada para mostrar as linhas onde o sobrenome contém os caracteres digitados.

Parabéns! O aplicativo Company Directory foi concluído.

Resumo: Construir um Java Rich Client que Utiliza um Serviço da Web

Parabéns! Você aprendeu como utilizar o editor visual Java para construir o aplicativo My Company Directory, um cliente rich Java que se conecta a um serviço da Web de amostra para manter um diretório de funcionários.

Consulte uma figura do produto final:



Lições Aprendidas

Você utilizou o visual editor para concluir a interface gráfica com o usuário, utilizando GridBagLayout para organizar a tabela de funcionários. Em seguida, você ligou a tabela, campos e botões aos objetos e origens de dados apropriados para fazer o aplicativo funcionar com um proxy Java do serviço da Web gerado. Você também fez alguma codificação complexa para fazer o aplicativo comportar-se apropriadamente para facilitar a utilização e a intuição. E você aprendeu como instalar um aplicativo corporativo no WebSphere Application Server v6.0 e implementou um serviço da Web.

Mais importante, você aprendeu tudo sobre a classe de binder poderosa, fornecida pelo Java visual editor para trabalhar com dados. Agora você está pronto para começar a experimentar sozinho e colocar os binders para novas e excitantes utilizações.

Você agora pode fazer as seguintes tarefas:

- Utilizar o Java visual editor para definir o layout dos componentes em um GridBagLayout.
- Executar uma classe visual como um Java bean.
- Ligar os componentes visuais de um aplicativo Java para os métodos e objetos de dados retornados por um serviço da Web.
- Incluir eventos aos componentes visuais.

Recursos Adicionais

Importar uma versão final do aplicativo My Directory

Esse projeto inclui o aplicativo final, o pacote jve.generated com classes de ligação e o cliente Java do serviço da Web configurado para o WebSphere Application Server v6.1. Se você importar esse projeto

final sem seguir o tutorial, você pode precisar configurar a variável do caminho de construção Java. Ela precisa apontar para o arquivo JAR do cliente thin dos serviços da Web do WebSphere v6.1.

Dica: A menos que você especifique um nome de projeto diferente durante a importação, esse projeto sobrescreverá o conteúdo de seu projeto MyDirectory.