SIMULADOS & TUTORIAIS

TUTORIAIS FLASH

O Componente DataGrid

O Componente DataGrid

Introdução

O componente **DataGrid** é uma espécie de planilha onde os dados são exibidos em linhas e colunas, muito semelhante a alguns aplicativos dessa categoria como a planilha do Excel ou uma tabela de um banco de dados. Ele permite-lhe criar poderosas listagens de dados trazidos das mais diversas fontes. Você pode usar esse componente para instanciar um recordset (recuperado de um banco de dados do ColdFusion, Java ou .NET) usando o Adobe Flash Remoting e mostrá-los através de colunas. Você poderá usar também informações extraídas de uma matriz ou de um arquivo texto externo XML para preencher esse componente.
A nova versão do **DataGrid** foi implementada para incluir scroll horizontal, melhor suporte a eventos, incluindo edição de células, capacidade de classificação avançada, entre outros recursos.
Um componente **DataGrid** pode ser redimensionado e personalizado quanto às suas propriedades, tais como fonte, cor e bordas das colunas. Você poderá usar um movie clip personalizado com um **cell renderer** para qualquer coluna da planilha (um **cell renderer** mostra o conteúdo de uma célula). Podem ser utilizadas também barras de scroll para mover os dados da planilha, desligar a barra de scroll e usar os métodos do **DataGrid** para mostrar um estilo de página, e assim por diante.

Interação do usuário com o componente DataGrid

Para interagir com o componente **DataGrid** você poderá usar tanto o mouse quanto o teclado. Se a propriedade **sortableColumns** e a propriedade **sortable** estiverem configuradas para **true**, clicando-se em um cabeçalho de qualquer coluna, a classificação dos dados é baseada nos valores dessa coluna. Você poderá desabilitar a classificação para uma determinada coluna bastando para isso configurar a propriedade **sortable** para **false**.
Se a propriedade **resizableColumns** for **true**, você poderá redimensionar as colunas arrastando os divisores das mesmas na linha do cabeçalho.
Clicando-se em uma célula editável, o foco ficará com a dita célula. Clicando-se em uma célula não editável, a mesma não terá nenhum foco.

Quando a instância de um componente **DataGrid** estiver com o foco, você poderá utilizar as seguintes teclas para controlá-la:

Tecla	Descrição
Seta para baixo	Quando uma célula estiver sendo editada, o ponto de inserção se deslocará para o final do texto da célula. Se a célula não for editável, essa tecla selecionará a próxima célula.
Seta para cima	Quando uma célula estiver sendo editada, o ponto de inserção se deslocará para o início do texto da célula. Se a célula não for editável, essa tecla selecionará a célula anterior.
Shift+seta para cima/seta para baixo	Se o componente **DataGrid** não for editável, e o parâmetro **allowMultipleSelection** estiver configurado para **true**, serão selecionadas as linhas contíguas (utilizando-se **Shift+seta para cima**). Revertendo a direção com o uso da outra tecla (seta para baixo) serão deselecionadas as linhas selecionadas até você passar pela linha inicial, que, a partir daí serão selecionadas as linhas na direção oposta.
Shift+Click	Se o parâmetro **allowMultipleSelection** estiver configurado para **true**, será selecionado o bloco de linhas contíguas a partir da primeira linha selecionada até a última clicada.
Ctrl+Click	Se o parâmetro **allowMultipleSelection** estiver configurado para

	true, serão selecionadas todas as linhas que forem clicadas sem a necessidade de serem contíguas.
Seta para direita	Quando uma célula está sendo editada, o ponto de inserção se deslocará para o próximo caractere do texto da célula. Se a célula não é editável, essa tecla não terá nenhum efeito.
Seta para esquerda	Quando uma célula estiver sendo editada, o ponto de inserção se deslocará para o caractere anterior do texto da célula. Se a célula não for editável, essa tecla não terá nenhum efeito.
Home	Seleciona a primeira linha em um **DataGrid**.
End	Seleciona a última linha em um **DataGrid**.
PageUp	Seleciona a primeira linha em uma página de um **DataGrid**. Uma página consiste do número de linhas que o **DataGrid** poderá mostrar sem considerar o rolamento.
PageDown	Seleciona a última linha em uma página de um **DataGrid**. Uma página consiste do número de linhas que o **DataGrid** poderá mostrar sem considerar o rolamento.
Return/Enter Shift+Enter	Quando uma célula estiver sendo editada, a alteração será confirmada e o ponto de inserção será movido para a célula da mesma coluna, mas da próxima linha (se usar apenas **Enter** ou **Return**). Se usar **Shift+Enter** ou **Return**, o ponto de inserção será movido para a célula de cima.
Shift+Tab/Tab	Se o **DataGrid** for editável, o foco será movido para o item anterior ou posterior (dependendo do caso) até o final da coluna, e então para a linha anterior ou posterior (dependendo do caso) até que a primeira ou última a célula seja alcançada. Se a primeira célula estiver selecionada, **Shift+Tab** moverá o foco para o controle precedente. Se a última célula estiver selecionada, **Tab** moverá o foco para o próximo controle. Se o **DataGrid** não for editável, essas teclas moverão o foco para o controle anterior ou posterior (um outro componente, por exemplo).

Conhecendo o componente DataGrid

Esse componente poderá ser utilizado para vários tipos de aplicações onde os dados precisem ser mostrados em uma espécie de planilha, ou seja, de uma forma tabular. Algumas sugestões para o uso desse componente seriam os seguintes:

- Uma pesquisa feita em um banco de dados sobre algum assunto.
- Uma planilha de custos.
- Um orçamento doméstico.
- O resultado de uma pesquisa na Internet.
- Uma cesta de compras de produtos adquiridos através da Internet.
- Um placar para algum jogo mostrando os melhores resultados.

Conceitualmente, um **DataGrid** é composto de um modelo de dados e um **View** (visualizador) que mostrará esses dados. O modelo de dados consiste de três partes principais:

- **DataProvider** – é a lista de itens com a qual os dados da planilha serão preenchidos. Esses itens poderão ser extraídos de um banco de dados, uma matriz ou um arquivo texto. Veja o exemplo a seguir que cria um **DataProvider** chamado ***meuDP***.

  meuDP = new Array[{Nome:"Maria", Salário: 5000},{Nome:"Carol", Salário: 3000}];

- **Item** – este é um objeto do ActionScript usado para armazenar as unidades de informação nas células de uma coluna. Em um **DataGrid** poderá ser mostrada uma lista de mais de uma coluna de dados. Essa lista poderá ser proveniente de um banco de dados, de uma matriz ou de um arquivo texto. Para o **DataGrid** cada item

é composto de campos de uma linha. No código abaixo, o conteúdo entre as chaves (**{}**) é considerado um item:

```
meuDP = new Array[{Nome:"Carol", Idade: "21"},{Nome:"Maria", Idade: "50"}];
```

- **Campo** – são identificadores que indicam os nomes das colunas dentro dos itens. Corresponde à propriedade **columnNames** na lista de colunas. No componente **List**, os campos são geralmente **label** e **data**, mas no componente **DataGrid** os campos podem ser quaisquer identificadores. Veja no código a seguir um exemplo mostrando os campos **Nome** e **Preço**:

```
meuDP = new Array[{Nome:"Batata", Preço: "0.50"},{Nome:"Cerveja", Preço: "0.97"}];
```

O **View** também consiste de três partes principais:

1. **Row** (linha)– representa cada linha da planilha com suas respectivas células.

2. **Col** (coluna)– as colunas são campos que são mostrados na planilha. Os campos correspondem à propriedade **columnName** de cada coluna. Cada coluna é um objeto **View** (uma instância da classe **DataGridColumn**) responsável por mostrar cada coluna. Por exemplo, largura, cor, tamanho, e assim por diante. Há três formas de adicionar colunas a um **DataGrid**:

    - Associando um objeto **DataProvider** ao **DataGrid.dataProvider**. Isso gera automaticamente uma coluna para cada campo no primeiro item.
    - Configurando o **DataGrid.columnNames** para especificar que campos serão mostrados.
    - Usando um construtor da classe **DataGrid.addColumn()** para adicioná-los à planilha.

3. **Cell** (célula) – é o objeto responsável por exibir os campos individuais de cada item para se comunicar com o **DataGrid**. Esses componentes devem implementar o **CellRendererAPI**.

Ao adicionar um componente **DataGrid** em sua aplicação, você poderá torná-lo acessível aos leitores de tela adicionando as seguintes linhas no seu código ActionScript:

```
import fl.accessibility.DataGridAccImpl;
DataGridAccImpl.enableAccessibility();
```

Essas linhas só serão necessárias somente uma vez por cada componente, independente do número de instâncias que possam ser utilizadas do mesmo.

Os parâmetros do componente DataGrid

Para usar corretamente esse componente é aconselhável conhecer seus parâmetros básicos e como utilizá-los adequadamente para que os resultados sejam conforme o esperado. Mostraremos a seguir os parâmetros disponíveis no **Inspetor de Propriedades** como também no painel **Component Inspector**, que poderão ser alterados diretamente durante o projeto:

Parâmetro	Descrição
allowMultipleSelection	É um valor Booleano que indica se poderão ser selecionados múltiplos itens (**true**) ou não (**false**). O valor padrão é **false**.
editable	É um valor Booleano que indica se a planilha é editável (**true**) ou não (**false**). O valor padrão é **false**.
headerHeight	É um valor numérico utilizado para estabelecer a altura de

	um cabeçalho de um componente **DataGrid**, em pixels. O valor padrão é: **25**.
horizontalLineScrollSize	Indica o número de linhas que serão roladas horizontalmente quando as setas da barra de rolagem são clicadas. O valor padrão é: **4**.
horizontalPageScrollSize	Indica o número de páginas que serão roladas horizontalmente quando as setas da barra de rolagem são clicadas. O valor padrão é: **0**.
horizontalScrollPolicy	É um valor booleano utilizado para indicar se a barra de rolamento horizontal está ativada. Você poderá escolher uma das seguintes opções: **auto**, **on** ou **off**. O valor padrão é: **off**.
resizableColumns	É um valor booleano que indica se o usuário poderá alterar a largura das colunas. O valor padrão é: **true**.
rowHeight	Indica a altura de cada linha, em pixels. Alterando o tamanho da fonte não altera a altura da linha. O valor padrão é **20** pixels.
showHeaders	É um valor booleano utilizado para indicar se os cabeçalhos das colunas de um componente **DataGrid** deverão ser mostrados ou não. O valor padrão é: **true**.
verticalLineScrollSize	Indica o número de linhas que serão roladas verticalmente quando as setas da barra de rolagem são clicadas. O valor padrão é: **4**.
verticalPageScrollSize	Indica o número de páginas que serão roladas verticalmente quando as setas da barra de rolagem são clicadas. O valor padrão é: **0**.
verticalScrollPolicy	É um valor booleano utilizado para indicar se a barra de rolamento vertical está ativada. Você poderá escolher uma das seguintes opções: **auto**, **on** ou **off**. O valor padrão é: **auto**.

Além desses parâmetros você também pode utilizar o ActionScript para controlar outras opções adicionais do componente **DataGrid** usando seus métodos, propriedades e eventos, conforme mostramos a seguir:

As propriedades do componente DataGrid

São as seguintes as propriedades disponíveis para o componente **DataGrid**:

Propriedade	Descrição
columns	Faz referência a uma matriz do objeto **DataGridColumn**, sendo uma para cada coluna a ser mostrada no **DataGrid**.
editedItemPosition	Configura a coluna e o índice da linha do item a ser mostrado para o item do **Data Provider** que está sendo editado.
editedItemRenderer	É uma referência ao item que será exibido no componente **DataGrid** cujo item está sendo atualmente editado.
imeMode	É uma string que permite configurar o modo do método de entrada do editor (IME).
itemEditorInstance	É uma referência à atual instância ativa do item editor, se existir.
labelFunction	É uma função que determina que campos de cada item devem usar o rótulo de texto.
minColumnWidth	Define a largura mínima de uma coluna do **DataGrid**, em pixels.
rowCount	Define o número de linhas que deverão estar parcialmente visíveis na lista de um **DataGrid**.
sortableColumns	Indica se o usuário poderá classificar os itens de uma coluna ao clicar no cabeçalho da respectiva coluna. O valor padrão é: **true**.
sortDescending	Classifica uma coluna em ordem descendente quando o usuário clicar no respectivo cabeçalho. O valor padrão é: **false**.
sortIndex	Resgata o índice da coluna a ser classificada. O valor padrão é: **-1**.

Os métodos do componente DataGrid

São os seguintes os métodos disponíveis para o componente **DataGrid**:

Método	Descrição
DataGrid	Cria uma nova instância do componente **DataGrid**.
addColumn	Adiciona uma coluna no final de uma matriz de colunas.
addColumnAt	Insere uma coluna no índice especificado em uma matriz de colunas.
createItemEditor	Usa o editor especificado pela propriedade **itemEditor** para criar um item editor para o item exibidor na coluna e índice da linha identificado pela propriedade **editedItemPosition**.
destroyItemEditor	Fecha um item editor que está atualmente aberto sobre um item exibidor.
editField	Edita um campo dado ou propriedade no componente **DataGrid**.
getCellRendererAt	Obtém a instância de uma célula exibidora na posição especificada no **DataGrid**.
getColumnAt	Recupera a coluna que está localizada no índice especificado na matriz de colunas.
getColumnCount	Recupera o número de colunas no componente **DataGrid**.
getColumnIndex	Recupera o índice da coluna do nome especificado, ou retorna **-1** se nenhum for encontrado.
getStyleDefinition	Recupera o estilo de mapa padrão para o componente atual.
itemToCellRenderer	O componente **DataGrid** possui múltiplas células para qualquer item dado, portanto, o método **itemToCellRenderer** sempre retorna **null**.
removeAllColumn	Remove todas as colunas de um componente **DataGrid**.
removeColumnAt	Remove a coluna que está localizada no índice especificado em uma matriz de colunas.
scrollToIndex	Rola a lista para o item do índice especificado.
spaceColumnsEqually	Estabelece a mesma largura para todas as colunas que se encontram visíveis.

Os eventos do componente DataGrid

Eventos são ações utilizadas pelos componentes para executar tarefas específicas, de acordo com a finalidade de cada situação. Veja a seguir os eventos disponíveis para esse componente e para que servem.

Evento	Descrição
columnStretch	É executado após o usuário expandir uma coluna horizontalmente.
headerRelease	É executado após o usuário clicar no cabeçalho de uma coluna.
itemEditBegin	É executado após a propriedade **editedItemPosition** for configurada e o item poder ser editado.
itemEditBeginning	É executado após o usuário editar um item, por exemplo, liberando o botão do mouse sobre o item.
itemEditEnd	É executado quando uma sessão de edição de um item termina por qualquer razão.
itemFocusIn	É executado após um item receber o foco.
itemFocusOut	É executado após um item perder o foco.

Veja como utilizar esses recursos nos exemplos mais adiante neste capítulo e no DVD, que acompanha a presente obra.

Criando aplicações com o componente DataGrid

Para criar uma aplicação com esse componente você primeiro deverá determinar qual a fonte de seus dados. Por exemplo, se ela é proveniente de um banco de dados, de um arquivo XML ou de uma matriz. Para inserir os dados em um **DataGrid** você deverá configurar a propriedade **DataGrid.dataProvider** para a fonte de dados. Você também poderá usar os métodos da classe **DataGrid** e **DataGridColumn** para criar os dados localmente.

Para entender melhor como funciona esse componente, mostraremos a seguir alguns exemplos práticos, inclusive em conjunto com outros componentes.

Exemplo 1

Esse exemplo mostra como preencher um componente **DataGrid** com apenas duas colunas de dados provenientes de uma matriz. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **DataGrid** para o palco. Veja na **Figura 7.1** abaixo a aparência desse componente quando arrastado para o palco:

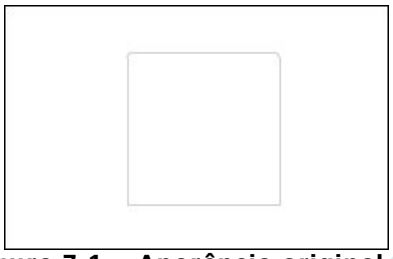


Figura 7.1 – Aparência original do componente DataGrid quando arrastado para o palco.

3. Abra o painel **Properties**, altere sua largura para **120**, e crie uma instância para ele. Digamos: **planilha**;
4. Nomeie a camada atual para: **componentes**;
5. Crie uma nova camada e chame-a de: **ações**;
6. Clique no primeiro frame dessa camada e digite o seguinte código:

```
stop();
import fl.data.DataProvider;
//
var meuDP:Array = new Array();
meuDP=[{Nome: "Maria", Idade: "50"}, {Nome: "Joana", Idade: "17"}, {Nome: "Joyce", Idade: "25"}];
planilha.dataProvider = new DataProvider(meuDP);
planilha.rowCount=planilha.length;
```

Vejamos os comentários sobre o código:

OBS.: Em todos os exemplos do presente capítulo incluiremos o comando **stop()** na primeira linha do código. Esse comando tem como objetivo interromper a aplicação quando a mesma for executada, mesmo que a aplicação não possua mais de um frame na sua linha do tempo. Consideramos um bom hábito essa prática. Portanto, não comentaremos mais essa linha nos próximos exemplos.

Na linha:

import fl.data.DataProvider;

Essa classe é necessária para que possamos utilizá-la na obtenção dos dados.

Na próxima linha:

var meuDP:Array = new Array();

Criamos uma nova matriz chamada **meuDP** para armazenar a fonte de dados (itens) para o componente **DataGrid**.

Na linha seguinte:

```
meuDP=[{Nome: "Maria", Idade: "50"}, {Nome: "Joana", Idade: "17"}, {Nome: "Joyce", Idade: "25"}];
```

Definimos os dados a serem inseridos no **DataGrid**. Os campos **Nome** e **Idade** serão usados como os cabeçalhos das colunas e seus valores preencherão as células em cada linha, semelhante a uma tabela de um banco de dados qualquer.

Nas linhas a seguir:

```
planilha.dataProvider = new DataProvider(meuDP);
planilha.rowCount=planilha.length;
```

Criamos uma nova instância da classe **DataProvider** através de seu construtor para recuperarmos os dados fornecidos na matriz **meuDP**. Em seguida definimos o número de linhas da planilha para que seja igual ao seu comprimento, a fim de evitar linhas em branco desnecessárias.

7. Execute a aplicação e confira o resultado com o da **Figura 7.2** a seguir:

Nome	Idade
Maria	50
Joana	17
Joyce	25

Figura 7.2 – Componente DataGrid devidamente preenchido.

Se você clicar em um dos cabeçalhos das colunas, a lista de itens será classificada em ordem crescente. Se clicar novamente, a lista ficará na ordem inversa. Experimente.

*****

Exemplo 2

Nesse exemplo mostraremos como criar e preencher um componente **DataGrid** em tempo de execução utilizando o ActionScript. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **DataGrid** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. Clique no primeiro frame dessa camada e insira o seguinte código:

```
stop();
//
import fl.controls.DataGrid;
import fl.data.DataProvider;
//
var planilha:DataGrid=new DataGrid();
addChild(planilha);
//
planilha.columns=["Nome","Idade"];
planilha.setSize(220, 100);
planilha.move(120, 80);
//
var meuDP:Array = new Array();
```

```
meuDP=[{Nome:"Jorge",Idade:55},{Nome:"Maria",Idade:51},{Nome:"Emmanuelle",
Idade:29},{Nome:"Diego", Idade:27},{Nome:"Carol", Idade:22}];
planilha.dataProvider = new DataProvider(meuDP);
planilha.rowCount=planilha.length;
```

Vejamos os comentários desse código:

Nas linhas:

```
import fl.controls.DataGrid;
import fl.data.DataProvider;
```

Importamos as classes necessárias para podermos utilizar os recursos do componente adequadamente. A primeira linha refere-se ao componente e a segunda linha à fonte de dados.

Nas linhas a seguir:

```
var planilha:DataGrid=new DataGrid();
addChild(planilha);
```

Criamos uma nova instância do objeto **DataGrid** através de seu construtor. Em seguida, inserimos essa instância no palco através do método **addChild()**.

Nas linhas seguintes:

```
planilha.columns=["Nome","Idade"];
planilha.setSize(220, 100);
planilha.move(120, 80);
```

Definimos:

- Os cabeçalhos das colunas através da propriedade **columns**.
- O tamanho da planilha (**setSize**).
- A posição (**move**) da planilha no palco.

Nas linhas:

```
var meuDP:Array = new Array();
meuDP=[{Nome:"Jorge",
Idade:55},{Nome:"Maria",Idade:51},{Nome:"Emmanuelle",Idade:29},{Nome:"Diego",
Idade:27},{Nome:"Carol", Idade:22}];
```

Criamos uma nova matriz (**meuDP**) para definirmos a fonte de dados do **DataGrid**. Em seguida, preenchemos essa matriz com os dados necessários para nossa aplicação.

E finalmente, nas linhas:

```
planilha.dataProvider = new DataProvider(meuDP);
planilha.rowCount=planilha.length;
```

Criamos um novo objeto **dataProvider** vinculado à instância (**planilha**) do componente **DataGrid** para preenchê-lo com as informações da matriz **meuDP**. Em seguida definimos o número de linhas da planilha para que seja igual ao seu comprimento, a fim de evitar linhas em branco desnecessárias.

5. Execute a aplicação e confira com o resultado mostrado na **Figura 7.3** abaixo:

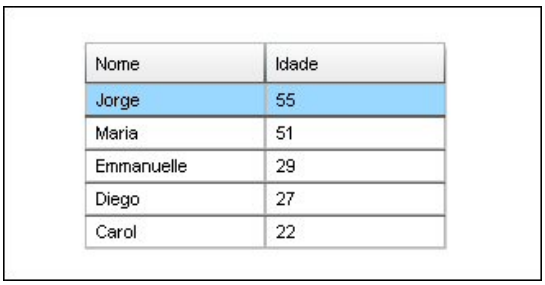


Figura 7.3 – DataGrid criado e preenchido Em tempo de execução.

6. Experimente clicar nos cabeçalhos das colunas para classificá-las em ordem crescente e decrescente.

*****

Exemplo 3

Esse exemplo mostrará como criar um componente **DataGrid** com três colunas de dados preenchidas com números aleatórios entre 1 e 100 em tempo de execução utilizando o ActionScript. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **DataGrid** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. Clique no primeiro frame da camada atual e insira o seguinte código:

```
stop();
//
import fl.controls.DataGrid;
import fl.data.DataProvider;
//
var totalLinhas = 16;
var dp:DataProvider = new DataProvider();
for (var i = 0; i < totalLinhas; i++) {
    dp.addItem({col1: calculaNumero (),col2: calculaNumero (),col3: calculaNumero
()});
}
//
var dg:DataGrid = new DataGrid();
dg.setSize(200, 320);
dg.columns = ["col1", "col2", "col3"];
dg.dataProvider = dp;
dg.move(180,50);
addChild(dg);
//
function calculaNumero ():uint {
    return Math.floor(Math.random() * 100 + 1);
}
```

Vejamos os comentários desse código:

Nas linhas:

```
import fl.controls.DataGrid;
import fl.data.DataProvider;
```

Importamos as classes necessárias para podermos utilizar os recursos do componente adequadamente. A primeira linha refere-se ao componente e a segunda linha à fonte de dados.

Nas linhas a seguir:

```
var totalLinhas:uint = 16;
var dp:DataProvider = new DataProvider();
```

Criamos uma variável (**totalLinhas**) para armazenar o número de linhas visíveis do **DataGrid**. Em seguida, criamos uma nova instância do objeto **DataProvider** através de seu construtor, para armazenar as informações.

No código seguinte:

```
for (var i = 0; i < totalLinhas; i++) {
    dp.addItem({col1: calculaNumero (),col2: calculaNumero
(),col3:calculaNumero()});
}
```

Criamos um laço com o número de linhas especificado para preencher o **DataGrid** com os números aleatórios fazendo uma chamada à função **calculaNumero**, utilizando-se para isso o método **addItem()**.

Na linha:

```
var dg:DataGrid = new DataGrid();
```

Criamos uma nova instância (**dg**) do componente **DataGrid** através de seu construtor.

Nas linhas seguintes:

```
dg.setSize(200, 320);
dg.columns = ["col1", "col2", "col3"];
dg.dataProvider = dp;
dg.move(180,50);
addChild(dg);
```

Definimos o tamanho do **DataGrid** (**setSize**), os labels das colunas, a fonte de dados (**dp**), a sua localização no palco (**move**), e logo em seguida colocamos o respectivo componente no palco através do método **addChild()**.

E finalmente, nas linhas a seguir:

```
function calculaNumero():uint {
    return Math.floor(Math.random() * 100 + 1);
}
```

Criamos a função **calculaNumero** para calcular os números aleatórios que deverão preencher as colunas do **DataGrid**, utilizando-se para isso as funções **Math.random** para gerar o número e **Math.floor** para arredondar esse número para baixo.

7. Execute a aplicação (**Ctrl + Enter**) e confira o resultado com o mostrado na **Figura 7.4** a seguir:

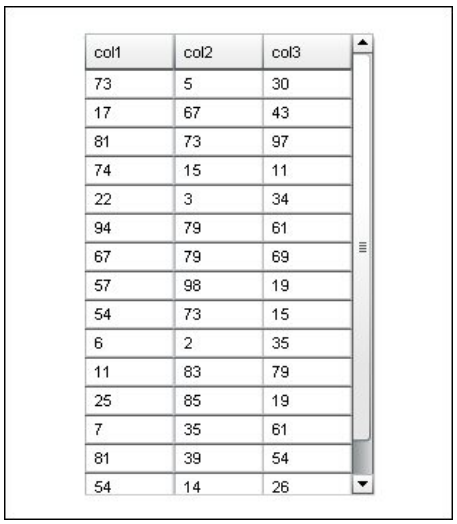


Figura 7.4 – DataGrid com colunas preenchidas com números randômicos.

*****

Exemplo 4

Esse exemplo mostra como criar e preencher um componente **DataGrid** com apenas duas colunas de dados provenientes de um arquivo XML externo em tempo de execução utilizando o ActionScript. O arquivo XML se encontra no DVD na mesma pasta do exemplo. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **DataGrid** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. Clique no primeiro frame da camada atual e insira o seguinte código:

```
stop();
//
import fl.controls.DataGrid;
import fl.data.DataProvider;
//
var dp:DataProvider;
//
var meuDataGrid:DataGrid = new DataGrid();
meuDataGrid.addColumn("ColunaA");
meuDataGrid.addColumn("ColunaB");
meuDataGrid.width = 200;
meuDataGrid.move(20, 30);
addChild(meuDataGrid);
//
var arquivoXML:String = "itemsDP.xml";
var req:URLRequest = new URLRequest(arquivoXML);
var carregador:URLLoader = new URLLoader();
//
carregador.addEventListener(Event.COMPLETE, carregaArquivo);
carregador.load(req);
//
function carregaArquivo(event:Event):void {
    var ldr:URLLoader = event.currentTarget as URLLoader;
    var xmlDP:XML = new XML(ldr.data);
    dp = new DataProvider(xmlDP);
    meuDataGrid.dataProvider = dp;
}
```

Vejamos os comentários desse código:

Nas linhas:

```
import fl.controls.DataGrid;
import fl.data.DataProvider;
```

Importamos as classes necessárias para podermos utilizar os recursos do componente adequadamente. A primeira linha refere-se ao componente e a segunda linha à fonte de dados.

Na linha a seguir:

```
var dp:DataProvider;
```

Criamos uma nova instância (**dp**) da classe **DataProvider** através de seu construtor, para posteriormente obtermos as informações necessárias provenientes de um arquivo XML.

No bloco de código seguinte:

```
var meuDataGrid:DataGrid = new DataGrid();
meuDataGrid.addColumn("ColunaA");
meuDataGrid.addColumn("ColunaB");
meuDataGrid.width = 200;
meuDataGrid.move(20, 30);
addChild(meuDataGrid);
```

Criamos uma nova instância (**meuDataGrid**) do componente **DataGrid** através de seu construtor, e definimos:

- Os títulos das colunas do **DataGrid** utilizando o método **addColumn()**.
- A sua largura utilizando-se o método **width**.
- A sua posição no palco utilizando o método **move()**.

Em seguida, inserimos o componente no palco através do método **addChild()**.

Nas linhas seguintes:

```
var arquivoXML:String = "itemsDP.xml";
var req:URLRequest = new URLRequest(url);
var carregador:URLLoader = new URLLoader();
```

Criamos uma variável (**arquivoXML**) do tipo string para armazenar o arquivo XML com as informações que iremos utilizar. Em seguida criamos uma instância (**req**) da classe **URLRequest** para armazenar o conteúdo da variável **arquivoXML**, e finalmente, criamos uma outra instância (**carregador**) da classe **URLLoader**.

Nas linhas a seguir:

```
carregador.addEventListener(Event.COMPLETE, carregaArquivo);
carregador.load(req);
```

Utilizamos o evento **COMPLETE** associado à instância **carregador**, de forma que quando a carga do arquivo for completada, a função **carregaArquivo** seja executada. Em seguida, utilizamos o método **load** para carregar o arquivo armazenado na variável **req**.

Finalmente, no bloco de código seguinte:

```
function carregaArquivo(event:Event):void {
    var ldr:URLLoader = event.currentTarget as URLLoader;
    var xmlDP:XML = new XML(ldr.data);
    dp = new DataProvider(xmlDP);
    meuDataGrid.dataProvider = dp;
}
```

Criamos a função chamada **carregaArquivo** que deverá ser executada automaticamente para carregar o arquivo XML. Para isso:

- Criamos uma nova instância (**ldr**) da classe **URLLoader** para vincular o evento.
- Criamos uma nova instância (**xmlDP**) da classe XML para carregar os dados da variável (**ldr**) criada.
- Criamos uma nova instância (**dp**) da classe **DataProvider** para armazenar os dados do **DataGrid**.
- Finalmente, inserimos as informações no **DataGrid** (**meuDataGrid**) provenientes do arquivo XML.

8. Execute a aplicação e confira com o resultado mostrado na **Figura 7.5** abaixo:

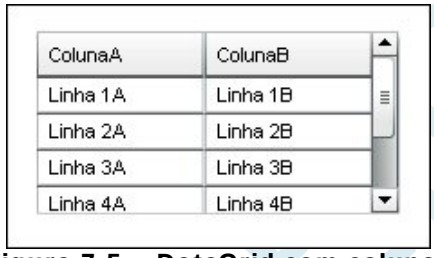


Figura 7.5 – DataGrid com colunas preenchidas utilizando-se um arquivo externo XML.

*****

Exemplo 5

Nesse exemplo serão utilizados um componente **DataGrid** com duas colunas, preenchido através de um bloco de dados em XML, um componente **Button** e um componente **Slider**, de forma que quando o usuário alterar a largura das colunas, o componente **Button** restaurará as mesmas para uma largura igual para ambas, e se o usuário alterar o valor do **Slider** e clicar em uma das setas da barra de rolagem, o número de pixels rolados será igual ao estabelecido no **Slider**. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **DataGrid**, um componente **Slider** e um componente **Button** para o palco;
3. Nomeie a camada atual para: **componentes**;
4. Abra o painel **Properties**, e crie as seguintes instâncias para os componentes:

   DataGrid: meuDG
   Slider: meuSlider
   Button: meuBotao

5. Crie uma nova camada e chame-a de: **ações**;
6. Clique no primeiro frame da camada atual e insira o seguinte código:

```
stop();
//
import fl.data.DataProvider;
//
```

```
var data:XML = <dataProvider>
      <data Coluna1="Carol" Coluna2="23" />
      <data Coluna1="Diego" Coluna2="27" />
      <data Coluna1="Emmanuelle" Coluna2="29" />
      <data Coluna1="Mariana" Coluna2="22" />
      <data Coluna1="Thiago" Coluna2="26" />
      <data Coluna1="Thiara" Coluna2="20" />
      <data Coluna1="Thieta" Coluna2="22" />
      <data Coluna1="Laura" Coluna2="56" />
      <data Coluna1="Maria" Coluna2="52" />
      <data Coluna1="Jorge" Coluna2="56" />
   </dataProvider>;
//
meuDG.setSize(160,120);
var dp:DataProvider = new DataProvider(data);
//
meuDG.dataProvider = new DataProvider(dp);
//
meuDG.addEventListener(MouseEvent.CLICK, alteraRolagem);
//
function alteraRolagem(event:Event):void {
    meuDG.verticalLineScrollSize = meuSlider.value;
}
//
meuBotao.addEventListener(MouseEvent.CLICK,restabeleceColuna);
//
function restabeleceColuna(event:Event):void {
    meuDG.spaceColumnsEqually();
}
```

Vejamos os comentários desse código:

Na linha:

```
import fl.data.DataProvider;
```

Importamos a classe necessária para que possamos criar nossa fonte de dados.

No bloco de código a seguir:

```
var data:XML = <dataProvider>
      <data Coluna1="Carol" Coluna2="23" />
      <data Coluna1="Diego" Coluna2="27" />
      <data Coluna1="Emmanuelle" Coluna2="29" />
      <data Coluna1="Mariana" Coluna2="22" />
      <data Coluna1="Thiago" Coluna2="26" />
      <data Coluna1="Thiara" Coluna2="20" />
      <data Coluna1="Thieta" Coluna2="22" />
      <data Coluna1="Laura" Coluna2="56" />
      <data Coluna1="Maria" Coluna2="52" />
      <data Coluna1="Jorge" Coluna2="56" />
   </dataProvider>;
```

Criamos um bloco de dados em XML chamado **data** para preencher o nosso **DataGrid** com duas colunas.

Nas linhas seguintes:

```
meuDG.setSize(160,120);
var dp:DataProvider = new DataProvider(data);
```

Definimos o tamanho do **DataGrid** através da propriedade **setSize()** e logo em seguida criamos uma nova instância da classe **DataProvider** chamada **dp**, e armazenamos os dados do arquivo **XML** na mesma.

Na linha:

meuDG.dataProvider = new DataProvider(dp);

Criamos uma nova instância (**meuDG**) da classe **DataProvider** para preenchermos definitivamente o **DataGrid**.

Na linha seguinte:

meuDG.addEventListener(MouseEvent.CLICK, alteraRolagem);

Utilizamos um evento **CLICK** do mouse e vinculamos o mesmo à instância do **DataGrid** (**meuDG**), de forma que quando a barra de rolagem for clicada, a função **alteraRolagem** seja executada.

No código a seguir:

```
function alteraRolagem(event:Event):void {
    meuDG.verticalLineScrollSize = meuSlider.value;
}
```

Criamos a função **alteraRolagem** para associar o valor do **Slider** ao número de pixels que deverão ser rolados na barra de rolagem do **DataGrid** quando a mesma for executada.

Na linha:

meuBotao.addEventListener(MouseEvent.CLICK,restabeleceColuna);

Utilizamos um evento **CLICK** do mouse e vinculamos o mesmo à instância do botão (**meuBotao**), de forma que quando o mesmo for clicado a função **restabeleceColuna** seja executada.

E finalmente, no código:

```
function restabeleceColuna(event:Event):void {
    meuDG.spaceColumnsEqually();
}
```

Criamos a função **restabeleceColuna** para ajustar as colunas do **DataGrid** para o mesmo tamanho utilizando-se para isso o método **spaceColumnsEqually()**, quando a mesma for executada.

9. Execute a aplicação e confira com o resultado mostrado na **Figura 7.6** abaixo:

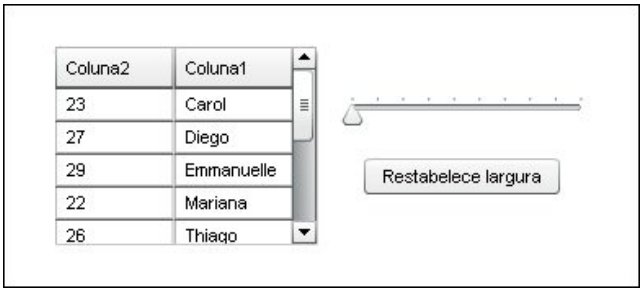


Figura 7.6 – DataGrid utilizando os componentes Slider e Button para manipular sua estrutura.

*****

Exercícios de Fixação

1. Um componente **DataGrid** exibe os dados em uma espécie de planilha em forma de:

   a) linhas
   b) itens
   c) linhas e colunas
   d) colunas
   e) campos e colunas

2. A lista de itens com a qual os dados do **DataGrid** serão preenchidos chama-se:

   a) DataGridColumn
   b) DataProvider
   c) DataGridRow
   d) DataGridView
   e) DataGridProvider

3. Um célula individual de um componente **DataGrid** é editável somente quando as propriedades **DataGrid.editable** e **DataGrid.Column.editable** forem configuradas para **true**.

   a) Verdadeiro
   b) Falso

4. Qual o valor padrão do parâmetro **editable** de um componente **DataGrid**?

   a) True
   b) False

5. Que propriedade utilizamos para permitir que o usuário altere a largura de uma coluna de um componente **DataGrid**?

   a) spaceColumns
   b) countColumns
   c) selectedColumns
   d) resizableColumns

6. Qual o valor padrão do parâmetro **rowHeight** de um componente **DataGrid**?

   a) 10
   b) 15
   c) 20
   d) 25
   e) 30

7. O parâmetro **rowHeight** de um componente **DataGrid** indica:

   a) A altura da linha selecionada
   b) A altura da planilha
   c) A largura da planilha
   d) A altura de cada linha
   e) A largura de uma coluna

8. Que propriedade de um componente **DataGrid** define o número de linhas que deverão ser parcialmente visíveis, desconsiderando a barra de rolagem?

   a) rowCount

b) selectedRow
c) rowNumber
d) rowSelected

9. Para adicionarmos uma nova coluna no final de uma matriz de colunas de um componente **DataGrid**, que método deveremos utilizar?

a) addDataGrid
b) includeColumn
c) selectColumn
d) addColumn

10. Qual o método utilizado para inserir uma instância de um componente **DataGrid** que se encontra na biblioteca, no palco?

a) AddChild()
b) StageDataGrid()
c) AddDataGrid()
d) IncludeDataGrid()

11. Qual a alternativa correta para criarmos uma nova instância (de nome **planilha**), de um componente **DataGrid**?

a) var planilha:DataGrid=newDataGrid()
b) var planilha.DataGrid=new DataGrid()
c) var planilha:DataGrid=new DataGrid()
d) var planilha:DataGrid=new.DataGrid()

12. Se quiséssemos saber o número de colunas de um componente **DataGrid**, que método utilizaríamos?

a) getColumnAt
b) getColumnCount
c) getColumnIndex
d) getColumnNumber

13. Que evento é executado quando o usuário expande uma coluna de um componente **DataGrid**?

a) columnRelease
b) columnStretch
c) columnChange
d) columnClick

14. Qual a propriedade que define a largura mínima de um componente **DataGrid**?

a) columnMin
b) columnMinWidth
c) minColumnWidth
d) minWidthColumn

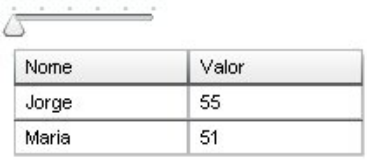
15. Que evento é executado quando o usuário clica no cabeçalho de uma coluna de um componente **DataGrid**?

a) headerRelease
b) headerClick
c) columnClick
d) columnHeader

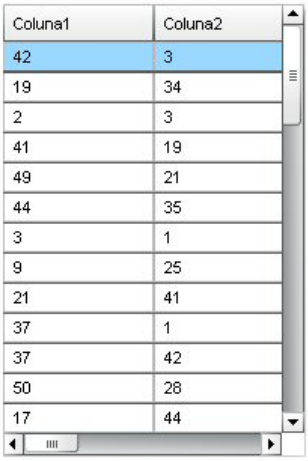
*****

Exercícios Propostos

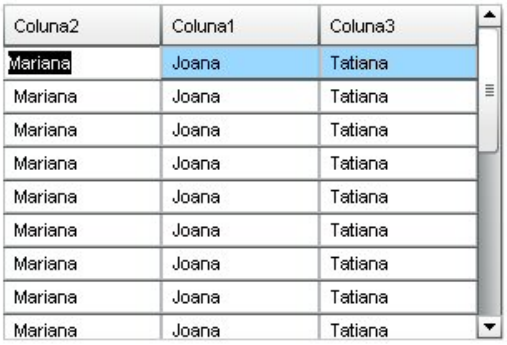
1. Crie uma aplicação utilizando um componente **DataGrid** com duas linhas e duas colunas, e um componente **Slider**, de forma que quando o valor do **Slider** for alterado, seja alterada a altura do cabeçalho. Veja uma sugestão de layout para esse exemplo:



2. Crie uma aplicação utilizando um componente **DataGrid** com cinqüenta linhas e oito colunas, de modo que sejam visíveis apenas duas colunas e quatorze linhas (incluindo o cabeçalho). O conteúdo da planilha deverá ser preenchido com números aleatórios entre 1 e 50. Para isso, utilize a função **Math.floor (Math.random())**. Veja uma sugestão de layout para esse exemplo:



3. Crie uma aplicação utilizando um componente **DataGrid** com vinte linhas e três colunas, de forma que todas as células sejam editáveis, e quando cada célula for alterada, o seu conteúdo seja mostrado no painel **Output**. O conteúdo da planilha poderá ser quaisquer informações. Veja uma sugestão de layout para esse exemplo:



*****