

# Conhecendo o GWT-Ext

## Como criar uma interface web rica

O desenvolvimento de aplicações web sempre teve como vantagem principal a independência do cliente para a interação com o sistema. Máquinas clientes com os mais diversos tipos de configurações são beneficiadas pelo processamento no servidor, servindo apenas como ponto de interação com o usuário final das aplicações. À medida que esse tipo de desenvolvimento foi se difundindo e novas tecnologias foram surgindo, usuários e desenvolvedores passaram a ficar mais exigentes e seletivos no desejo de criar interfaces mais ricas. Porém, não era possível construir tais interfaces utilizando apenas a linguagem de programação empregada no sistema, ou seja, era necessário uma linguagem para desenvolvimento e outras para criação de layout, como JavaScript, HTML, DHTML, Flash, etc. O surgimento do Ajax (Asynchronous JavaScript And XML) permitiu a criação de páginas mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações e iniciando assim uma nova geração de aplicações web.

A partir daí novas bibliotecas surgiram para melhorar a interatividade do usuário utilizando Ajax. Uma delas é o ExtJS, uma biblioteca JavaScript *open source* para construção de aplicações web ricas com interfaces gráficas sofisticadas e com alto grau de interatividade com o usuário. O ExtJS é detalhado no artigo: Introdução ao Framework ExtJS (Edição 60).

### Resumo DevMan

#### De que se trata o artigo:

O artigo apresenta a biblioteca GWT-Ext e demonstra o desenvolvimento de uma aplicação de exemplo, apresentando através de um CRUD o potencial da biblioteca.

#### Para que serve:

Este artigo serve para mostrar a utilização da biblioteca GWT-Ext de maneira simples e produtiva, facilitando a vida do desenvolvedor na criação de aplicações web com interface rica utilizando apenas código Java.

#### Em que situação o tema é útil:

Para o desenvolvimento de aplicações web com interfaces ricas e interativas é necessária a utilização de uma diversidade de tecnologias, como JavaScript, DHTML Ajax, etc. dificultando assim a vida do desenvolvedor, que precisa conhecer várias linguagens, trabalhar com as diferentes sintaxes e limitações de cada uma delas. O GWT-Ext permite a criação de interfaces ricas e interativas utilizando apenas a linguagem Java, propondo uma maior produtividade e oferecendo um conjunto amplo de componentes sofisticados advindos do ExtJS e a facilidade da programação para a geração desses componentes advindos do GWT.

#### Conhecendo o GWT-Ext:

A biblioteca GWT-Ext consiste na integração das tecnologias GWT e ExtJS, proporcionando uma poderosa biblioteca de *widjets* que provê ricos componentes de interface, tais como: *grids* com ordenação, paginação e filtro; *árvores* com suporte a *drag and drop*; *comboboxes* altamente customizáveis; *tab panels*; *menus*; *tollbars*; *forms*; entre outros. O GWT consiste em um conjunto de ferramentas *open source* que permitem aos desenvolvedores criarem aplicações Ajax na linguagem Java. O ExtJS, por sua vez, é uma biblioteca de JavaScripts para a criação de componentes de interface web com alta interatividade. GWT apenas, não oferece um conjunto de componentes tão amigáveis quanto os oferecidos pelo ExtJS, enquanto este necessita que o desenvolvedor conheça outra linguagem além do Java para a construção de aplicações. O GWT-Ext surge como um meio de se criar componentes tão ricos como os apresentados pelo ExtJS usando apenas linguagem Java como proposto pelo GWT.



### Nota do Devman

Veja o que você vai aprender adicionalmente neste artigo:

- O conceito de Portlets;
- A definição de Binding.

Outra solução foi o GWT (*Google Web Toolkit*), um conjunto de ferramentas *open source* que permitem aos desenvolvedores criarem aplicações Ajax na linguagem Java. O compilador GWT traduz código Java para um JavaScript equivalente, que manipula programaticamente o HTML no

Utilize o poder dessa biblioteca de componentes para gerar interfaces web ricas escrevendo apenas código Java

IZALMO PRIMO E SAMUEL SANTOS



## Nota do Devman

**Portlets:** Um Portlet é um componente visual independente que pode ser utilizado para disponibilizar informações dentro de uma página Web. Um Portlet pode ser utilizado em qualquer portal, promovendo-se assim a reutilização. Esse fator fez com que este componente ganhasse grande popularidade junto as equipes de desenvolvimento de portais para Web.

A especificação JSR 168 surgiu com o intuito de criar um padrão para o desenvolvimento de "portlets" para portais baseados na plataforma Java. Na JSR 168, um Portlet é um componente para Web gerado por um container específico (*portlet container*) que processa um pedido e gera dinamicamente o conteúdo que será mostrado no cliente. São utilizados no contexto de um portal na camada de apresentação de um sistema de informação.

web browser usando técnicas de DHTML. O GWT trouxe uma grande facilidade para os desenvolvedores Java que normalmente brigavam com o JavaScript para desenvolver interfaces ricas no lado cliente. Outras vantagens de usar a linguagem Java pelo GWT é que ele proporciona um código melhor estruturado, além de facilitar o desenvolvimento RPC cliente-servidor para Java, e ainda elimina a necessidade de bi-

bliotecas como DWR (*Direct Web Remoting*) para construir pontes entre o serviço Java e os clientes JavaScript. Como desvantagem, o GWT não possui um conjunto de componentes sofisticados como o ExtJS. Para saber mais sobre GWT leia o artigo: Ajax avançado com GWT (Edição 39).

Em junho de 2007, Sanjiv Jivan propôs no seu blog a integração do GWT e ExtJS. Seu objetivo é permitir que desenvolvedores Java possam criar interfaces web ricas como as do ExtJS usando apenas a linguagem Java, como proposto pelo GWT. Dessa forma, foi criada a biblioteca GWT-Ext, uma poderosa biblioteca de *widgets* que provê ricos componentes de interface, tais como *grids* com ordenação, paginação e filtro, *árvores* com suporte a *drag and drop*, *comboboxes* altamente customizáveis, *tab panels*, *menus*, *tollbars*, *forms* e muitos outros componentes poderosos com uma API muito fácil de usar. Além disso, é possível construir *portlets* facilmente, fazer integração com mapas (utilizando o *google maps*) e ainda

a integração com gráficos (utilizando o *google charts*).

A versão mais atual do GWT-Ext é a 2.0.5, lançada em setembro de 2008. Ela utiliza o ExtJS na versão 2.0.2 e o GWT na versão 1.5.2. A **Figura 1** ilustra a diversidade de componentes oferecidos pela biblioteca apresentada no ShowCase Demo da biblioteca.

## Desenvolvendo uma aplicação

Para ilustrar o uso da biblioteca GWT-Ext apresentaremos um exemplo que consiste em um sistema de controle de pessoal para uma empresa multinacional. Iremos ilus-

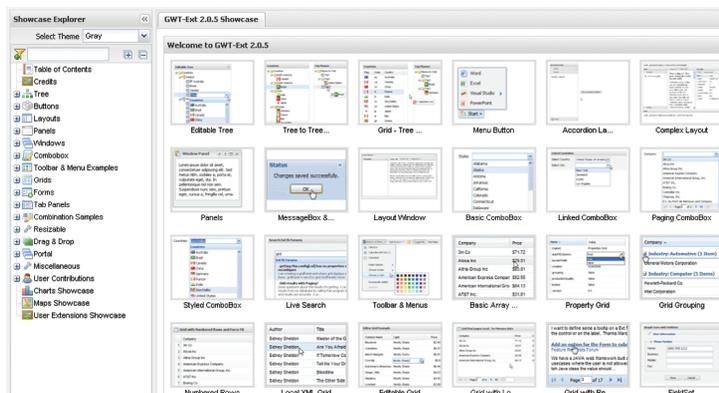


Figura 1. Vários tipos de widgets disponíveis para o GWT-Ext.

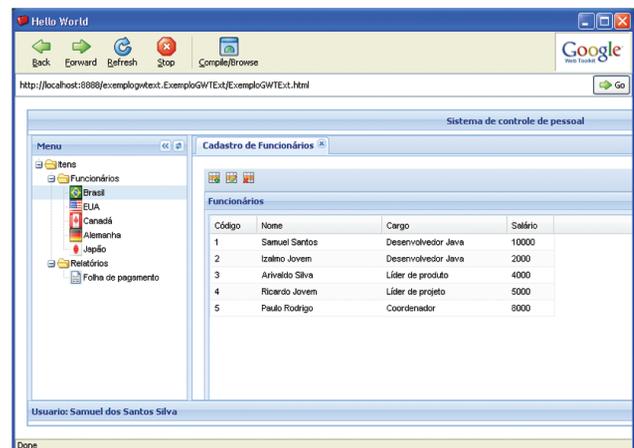


Figura 2. Tela da aplicação exemplo.

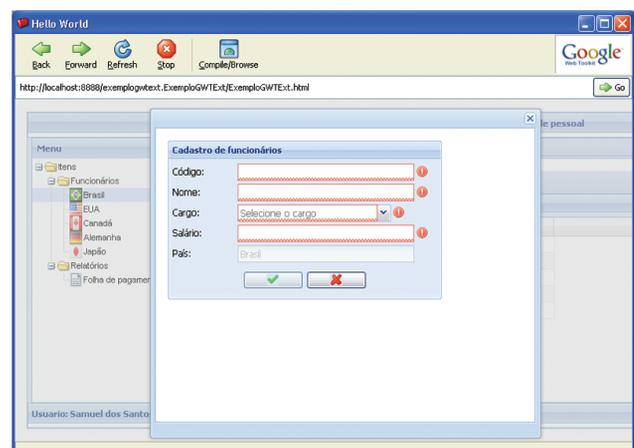


Figura 3. Formulário para criação e atualização de funcionário.

trar o módulo de cadastro de funcionários através das operações CRUD. A idéia é mostrar o potencial da biblioteca através da utilização dos seguintes componentes:

- **BorderLayout:** Utilizado na organização do layout da aplicação, disponibilizando o painel de título no norte, a barra de status no sul, o painel de menu na esquerda e o painel de informação no centro;
- **TreePanel:** Presente na construção da árvore com os menus da aplicação;
- **GridPanel:** Grid contendo a lista de funcionários de acordo com o país selecionado no menu. Esse grid já contém implicitamente um mecanismo de ordenação;
- **Window:** Janela que contém o formulário de criação e atualização de funcionário;
- **FormPanel:** Formulário com os campos para inserir e atualizar funcionários na aplicação.

Além da utilização dos componentes visuais propriamente ditos, apresentaremos também alguns recursos adicionais da biblioteca, como: preenchimento automático de formulário a partir de um registro selecionado no grid e validação de campos do formulário no cliente.

Nossa proposta principal é demonstrar os componentes visuais da biblioteca. Com isso, toda implementação de acesso ao serviço remoto e persistência de dados está fora do escopo de nosso exemplo.

O exemplo ilustrado na **Figura 2** apresenta um menu lateral de funcionários da multinacional por país. Ao clicar em um país, o grid apresenta os funcionários a este associado. Logo acima do grid foi adicionado um *toolbar* contendo os botões de criação, atualização e exclusão de funcionários. Para atualização e exclusão é necessária a seleção de uma linha no grid. A **Figura 3** ilustra o painel de formulário para a criação e atualização de um funcionário. Note que o visual da aplicação se aproxima bastante de uma aplicação desktop.

## Criando o projeto GWT

Vamos agora apresentar um passo-a-passo de como criar um projeto GWT no eclipse. Após a criação do projeto o desenvolvedor estará apto a implementar a solução e executá-la utilizando o ambiente do eclipse.

1. Baixar do site oficial do projeto, a distribuição binária mais recente do GWT e descompactá-la em qualquer diretório, que

será referenciado daqui para frente como *GWT\_HOME* (ver o endereço do GWT em **Links**);

2. Criar a pasta *ExemploGWText*. Ela será utilizada posteriormente para armazenar todo o conteúdo do projeto GWT;

3. Abrir uma janela de interpretador de linha de comandos a partir do diretório *ExemploGWText* e criar a estrutura do projeto eclipse utilizando o *toolkit* através do seguinte comando:

```
GWT_HOME/projectCreator -eclipse ExemploGWText
```

Nesse ponto, são criados os arquivos de configuração de projeto eclipse, *.classpath*, *.project*, e os diretórios *src* e *test*;

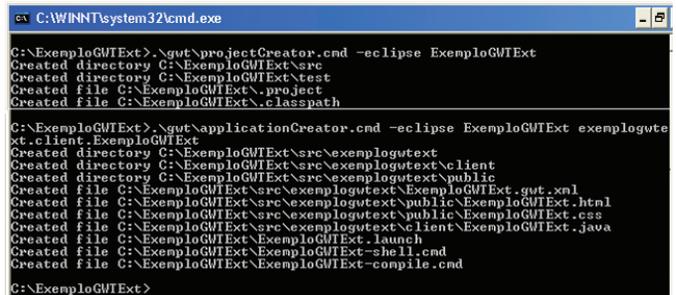
4. No mesmo diretório *ExemploGWText*, gerar os arquivos iniciais necessários para uma aplicação GWT através do seguinte comando:

```
GWT_HOME/applicationCreator -eclipse
ExemploGWText exemplotext.client.
ExemploGWText
```

O primeiro parâmetro consiste no nome do projeto e o segundo corresponde ao nome completo da classe principal da aplicação. Nesse momento são criados os seguintes pacotes e arquivos:

- *src/exemplotext/client*: Esse pacote deve conter todas as classes que irão rodar na parte cliente da aplicação;
- *src/exemplotext/public*: Esse pacote deve conter todos os arquivos HTML, JavaScripts, css, etc.;
- *src/exemplotext/ExemploGWText.xml*: Arquivo contendo a configuração dos módulos da aplicação GWT;
- *src/exemplotext/public/ExemploGWText.html*: Arquivo contendo a página HTML inicial da aplicação;
- *src/exemplotext/client/ExemploGWText.java*: Arquivo contendo a implementação da construção da tela inicial da aplicação;
- *.ExemploGWText.launch*: Arquivo contendo a configuração de execução da aplicação para abrir em uma janela de *browser* GWT.

5. Importar o projeto no eclipse a partir do diretório *ExemploGWText*.



**Figura 4.** Tela com a interpretação de comandos para a criação do projeto GWT.

A **Figura 4** ilustra a seqüência de execução dos comandos utilizando um interpretador de comandos do Windows, e a **Figura 5** ilustra a estrutura do projeto criado no eclipse.

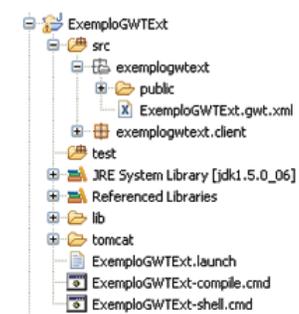
## Configurando o projeto para utilização do GWT-Ext

Até esse ponto criamos apenas um projeto GWT. Vamos agora configurá-lo para utilizarmos a biblioteca GWT-Ext. Para isso precisamos da biblioteca *gwttext.jar* e de alguns arquivos de JavaScript e estilos da biblioteca ExtJS. Seguem os passos a serem seguidos:

1. Baixar do site oficial do projeto a versão mais recente da biblioteca GWT-Ext. Adicionar o arquivo *gwttext.jar* ao projeto (veja o endereço na seção **Links**);

2. Baixar do site oficial do projeto a mais recente versão da biblioteca ExtJS e descompactá-la em qualquer diretório (que será referenciado daqui para frente como *EXT\_HOME*). Adicionar os seguintes arquivos ao projeto:

- *EXT\_HOME/ext-all.js* e *EXT\_HOME/ext-base.js* para o diretório *ExemploGWText\src\exemplotext\public\js*;
- *EXT\_HOME/resources/css/ext-all.css* para o diretório *ExemploGWText\src\exemplotext\public\css*;
- Copiar o diretório *EXT\_HOME/resources/images* para *ExemploGWText\src\exemplotext\public\images*;



**Figura 5.** Estrutura do projeto GWT criado.

Listagem 1. Classe principal *ExemploGWText.java*.

```
1 package exemplogwtext.client;
2 ...
3 public class ExemploGWText implements EntryPoint {
4 ...
5 public void onModuleLoad() {
6     Panel panel = new Panel();
7     panel.setBorder(false);
8     panel.setPadding(15);
9     panel.setLayout(new FitLayout());
10
11     Panel borderPanel = new Panel();
12     borderPanel.setLayout(new BorderLayout());
13
14     ...
15
16     //add north panel
17     Panel northPanel = new Panel();
18     northPanel.setHeight(32);
19     northPanel.setTitle("<center>Sistema de controle de pessoal</center>");
20     borderPanel.add(northPanel, new BorderLayoutData(RegionPosition.NORTH));
21
22     //add south panel
23     Panel southPanel = new Panel();
24     southPanel.setHeight(32);
25     southPanel.setTitle("Usuario: Samuel dos Santos Silva");
26     borderPanel.add(southPanel, new BorderLayoutData(RegionPosition.SOUTH));
27
28     //add west panel
29     Panel westPanel = new Panel();
30     westPanel.setTitle("Menu");
31     westPanel.setCollapsible(true);
32     westPanel.setWidth(200);
33     westPanel.add(criaPanelTree());
34     BorderLayoutData westData = new BorderLayoutData(RegionPosition.WEST);
35     westData.setSplit(true);
36     westData.setMinSize(175);
37     westData.setMaxSize(400);
38     westData.setMargins(new Margins(0, 5, 0, 0));
39
40     borderPanel.add(westPanel, westData);
41
42     borderPanel.add(centerPanel, new BorderLayoutData(RegionPosition.CENTER));
43
44     panel.add(borderPanel);
45
46     new Viewport(panel);
47 }
```

3. Configurar o módulo do projeto GWT para utilizar o GWT-Ext. Adicionar as linhas 4, 8, 9 e 10 da **Listagem 7** no arquivo *ExemploGWText.gwt.xml*. Lembre-se que o arquivo de configuração de módulos de um projeto GWT tem sempre o formato *<nome-projeto>*.

*gwt.xml*. Esse arquivo deve ser alterado para a inclusão do módulo GWT-Ext.

### Desenvolvendo a aplicação

Com o projeto configurado, vamos iniciar o desenvolvimento da aplicação. Basicamente precisamos de um arquivo HTML

referente à página principal e as classes **ExemploGWText**, que trata da montagem da tela principal da aplicação; **Funcionario** que contém as informações de um funcionário; **FuncionariosPanel** que representa o painel contendo o grid de funcionários e uma toolbar com as operações de criação, edição e exclusão; e finalmente a classe **FuncionariosForm**, que contém o formulário utilizado na criação e edição de funcionário. Essa aplicação está bem focada na camada de visualização, que é o objetivo principal do artigo, dessa forma o modelo está bastante simplificado, onde não estamos trabalhando com serviços remotos nem persistência dos dados. Iremos detalhar a utilização dessas classes na montagem da aplicação nos próximos tópicos.

### Exibindo a tela principal

O GWT sempre invoca o método **onModuleLoad()** na execução da classe principal do projeto, que é o ponto de entrada da aplicação. A definição da classe principal é configurada no arquivo de módulos do projeto, através do parâmetro *entry-point*. No nosso exemplo, a classe principal é a **ExemploGWText** (**Listagem 7**). O método **onModuleLoad()** irá montar o layout da aplicação utilizando o **BorderLayout** (**Listagem 1**). Primeiramente é criado o painel de título, e neste são setados os valores de título (com **setTitle()**) e altura (com **setHeight()**). Da mesma forma é criado o painel do sul, configurando o nome do usuário logado e a altura. Para o painel de menus que é adicionado à esquerda no layout são configurados os valores de título, **collapsible** (indicando se o menu pode ou não ser ocultado, caso não seja setado o menu permanece fixo) e largura. Além disso, foram configuradas no próprio layout que define a posição desse painel as propriedades **split** (permitindo que o painel de menus possa ser redimensionado e os

Listagem 2. Arquivo contendo a configuração do menu da aplicação: *menu.xml*.

```
<menu title="Empresa">
  <menuitem title="Funcionários">
    <item id="1" title="Brasil" icon="images/br.gif" qtip="Atualizar funcionários do Brasil"/>
    <item id="2" title="EUA" icon="images/us.gif" qtip="Atualizar funcionários dos EUA"/>
    <item id="3" title="Canadá" icon="images/ca.gif" qtip="Atualizar funcionários do Canadá"/>
    <item id="4" title="Alemanha" icon="images/de.gif" qtip="Atualizar funcionários Alemanha"/>
    <item id="5" title="Japão" icon="images/jp.gif" qtip="Atualizar funcionários do Japão"/>
  </menuitem>
  <menuitem title="Relatórios">
    <item id="3" title="Folha de pagamento" icon="images/document_text.png" qtip="Relatório de folha de pagamento de funcionários"/>
  </menuitem>
</menu>
```

**Listagem 3.** Método que monta a árvore de menus: *ExemploGWText.java*.

```

1 private Panel criaPanelTree() {
2     treePanel.setWidth(200);
3     treePanel.setHeight(400);
4     treePanel.setAnimate(true);
5     treePanel.setEnabledDD(true);
6     treePanel.setContainerScroll(true);
7     treePanel.setRootVisible(true);
8     treePanel.setBorder(false);
9
10    treePanel.addListener(new TreePanelListenerAdapter(){
11
12        public void onDbClick(TreeNode node,EventObject e)
13        {
14            int nodeId = Integer.parseInt(node.getId());
15            switch (nodeId) {
16                case 1:
17                    Panel panelTab = new FuncionariosPanel();
18                    panelTab.setTitle("Cadastro de Funcionários");
19                    panelTab.setClosable(true);
20                    centerPanel.add(panelTab);
21                    centerPanel.setActiveItem(0);
22                    break;
23                default:
24                    break;
25            }
26        }
27
28    });
29
30    final XMLTreeLoader loader = new XMLTreeLoader() {
31        {
32            setDataUrl("menu.xml");
33            setRootTag("menu");
34            setFolderIdMapping("@id");
35            setLeafIdMapping("@id");
36            setFolderTitleMapping("@title");
37            setFolderTag("menuitem");
38            setLeafTitleMapping("@title");
39            setLeafTag("item");
40            setQtipMapping("@qtip");
41            setIconMapping("@icon");
42        }
43    };
44    root = new AsyncTreeNode("Itens", loader);
45    treePanel.setRootNode(root);
46
47    root.expand();
48    treePanel.expandAll();
49
50    return treePanel;
51 }

```

**Listagem 4.** Painel contendo o grid de funcionários: *FuncionariosPanel.java*.

```

1 package exemplogwtext.client;
2
3 ...
4 public class FuncionariosPanel extends Panel {
5     ..
6     public FuncionariosPanel() {
7         this.init();
8     }
9
10    private void init() {
11        this.panelForm = new FuncionariosForm(this);
12        this.setBorder(false);
13        this.setPadding(15);
14
15        recordDef = new RecordDef(new FieldDef[] {
16            new StringFieldDef("codigo"),
17            new StringFieldDef("nome"),
18            new StringFieldDef("cargo"),
19            new FloatFieldDef("salario"),
20            new StringFieldDef("pais"),
21        });
22
23        store = new Store(recordDef);
24        grid.setStore(store);
25
26        ColumnConfig[] columns = new ColumnConfig[] {
27            new ColumnConfig("Código", "codigo", 60, true),
28            new ColumnConfig("Nome", "nome", 160, true),
29            new ColumnConfig("Cargo", "cargo", 160, true),
30            new ColumnConfig("Salário", "salario", 60, true)
31        };
32
33        ColumnModel columnModel = new ColumnModel(columns);
34        grid.setColumnModel(columnModel);
35
36        grid.setFrame(true);
37        grid.setStripeRows(true);
38
39        grid.setHeight(350);
40        grid.setWidth(600);
41        grid.setTitle("Funcionários");
42
43        Toolbar toolbar = getToolbar();
44        toolbar.setSize(grid.getWidth(), 30);
45        this.add(toolbar);
46        this.add(grid);
47    }
48
49    ...
50    private void removeFuncionario() {
51        System.out.println("Removendo funcionário");
52        store.remove
53            (grid.getSelectionModel().getSelected());
54    }
55
56    public void addRecord(Funcionario func) {
57        Object[] rowData = new Object[]{func.getCodigo(),
58            func.getNome(), func.getCargo(),
59            func.getSalario(), func.getPais()};
60        Record record = recordDef.createRecord
61            (rowData[0].toString(), rowData);
62        store.add(record);
63    }
64 }

```

limites desse redimensionamento: **minSize** e **maxSize**). Por fim é criado o painel central, um **TabPanel**, que irá gerenciar as abas das janelas da aplicação.

Note que a construção do layout é similar a de um desenvolvimento desktop para criação de telas Swing. A árvore de menus, que é adicionada ao painel da esquerda, é construída através do método **criaPanelTree()** (Listagem 3). Para a criação da árvore utili-

zamos a classe **TreePanel**, e setamos diversos parâmetros de exibição: largura, altura, animação e barra de rolagem. Para a obtenção dos itens de menu configurados no arquivo **menu.xml** (Listagem 2) utilizamos a classe **XMLTreeLoader**, onde configuramos o arquivo XML e as tags de título, tooltip e ícone que serão utilizadas na montagem dos nós das árvores. Depois disso criamos um objeto da classe **AsyncTreeNode**, que corresponde ao nó

raiz da árvore e finalmente setamos o nó raiz da **TreePanel**. Após a execução do método **onModuleLoad()** a página inicial *ExemploGWText.html* (Listagem 8) é carregada.

### Exibindo a tela de funcionários

É adicionado um *listener* no painel da árvore para capturar o evento de duplo clique (Listagem 3). No tratamento do evento é criado o painel de funcionários (**Funciona-**

**Listagem 5.** Método de construção da toolbar de criação, atualização e exclusão de funcionário.

```
01 private Toolbar getToolbar() {
02 //toolbar
03 ToolbarButton btnNovo = new ToolbarButton();
04 btnNovo.setIcon("images/toolbar_add.png");
05 btnNovo.addListener(new ButtonListenerAdapter(){
06     public void onClick(Button button, EventObject e) {
07         panelForm.show(false);
08         Object[] rowData = new Object[]{"", "", "", "", "Brasil"};
09         Record record = recordDef.createRecord(rowData[0].toString(),
10             rowData);
11         panelForm.loadRecord(record);
12     }
13 });
14 ToolbarButton btnEdit = new ToolbarButton();
15 btnEdit.setIcon("images/toolbar_edit.png");
16 btnEdit.addListener(new ButtonListenerAdapter(){
17     public void onClick(Button button, EventObject e) {
18         panelForm.show(true);
19         panelForm.loadRecord(grid.getSelectionModel().getSelected());
20     }
21 });
22 ToolbarButton btnRemove = new ToolbarButton();
23 btnRemove.setIcon("images/toolbar_delete.png");
24 btnRemove.addListener(new ButtonListenerAdapter(){
25     public void onClick(Button button, EventObject e) {
26         MessageBox.confirm("Confirmacao de exclusao",
27             "Deseja realmente excluir o registro?",
28             new MessageBox.ConfirmCallback() {
29                 public void execute(String btnID) {
30                     if (btnID.equals("yes")) {
31                         removeFuncionario();
32                     }
33                 }
34             });
35     }
36 });
37 Toolbar toolbar = new Toolbar();
38 toolbar.addButton(btnNovo);
39 toolbar.addButton(btnEdit);
40 toolbar.addButton(btnRemove);
41
42 return toolbar;
43 }
```

riosPanel), que contém o grid onde serão exibidos os funcionários e a *toolbar* contendo os botões de criação, edição e exclusão de funcionário. Para a construção do grid utilizam-se as classes **RecordDef** e **ColumnModel** (Listagem 4). A primeira configura os tipos de dados do registro, por exemplo, **StringFieldDef** para tipo **String** e **FloatFieldDef** para o tipo **Float**. A classe **ColumnModel** configura o nome de apresentação, *alias* para o campo, tamanho das colunas e se estas podem ser ordenadas. **RecordDef** e **ColumnModel** utilizam um *alias* para cada coluna que podem ser posteriormente utilizados para fazer **binding** com os campos do formulário. Além da configuração das colunas, ainda setamos as propriedades de título, **stripeRows** (se verdadeiro, permite que as linhas sejam exibidas em duas cores, as linhas ímpares em uma cor e as pares em outra, para facilitar a visualização), largura e altura. Para armazenamento dos dados propriamente

ditos utiliza-se a classe **Store** e a partir dessa pode ser feita adição e remoção de dados. Note que é bastante simples a criação de um grid com recursos de ordenação, cores de linhas alternadas, coloração diferente na seleção de linha e alinhamento das colunas de acordo com o tipo.

#### Realizando as operações CRUD

Agora vamos entender como são realizadas as operações de criação, edição e exclusão de funcionários. Os botões de criação e atualização chamam a mesma tela de formulário, a diferença é que na criação passamos um registro com os campos vazios e na atualização recuperamos o registro do grid e o passamos para o formulário (Listagem 5). Para a realização do **binding** os campos criados no formulário devem ter os mesmos *alias* utilizados em **RecordDef**. Dessa forma, a criação dos campos de texto (**TextField**), campos numéricos (**NumberField**) e



## Nota do Devman

**Binding:** Um *Binding* (ligação) é uma ligação de uma referencia simples para um pedaço de código que é muito grande, muito complicado ou constantemente utilizado. Assim, o *binding* permite substituir a repetição do código por uma simples referencia para ele. Em linguagem de programação, o termo *binding* significa que existe uma referencia entre um identificador e um valor. Neste contexto, o *binding* é o ator que associa um nome ou um símbolo com o endereço de máquina. Esta associação ocorre em tempo de compilação, denominada ligação estática ou pode ocorrer em tempo de execução sendo chamado de ligação dinâmica.

lista de opções (**ComboBox**) recebem o label e o alias do campo.

A classe **FuncionarioForm** (apresentada na Listagem 6) estende da classe **Window** da biblioteca GWT-Ext e constrói o formulário de funcionário a partir do método **getPanelCentral()**. A classe **Window** possui alguns parâmetros que são setados na inicialização, como: o **closable** (se for verdadeiro, apresenta o ícone de fechar no canto superior direito da janela), o tamanho, se a janela é modal ou não e o layout, o qual é setado para **BorderLayout**, que conterà o **FormPanel** no centro, ou seja, em toda a janela.

Para a montagem do formulário usamos a classe **FormPanel**, que é um painel específico para a adição de componentes de formulário (**TextField**, **NumberField**, **ComboBox**, etc.). Dessa forma os componentes são criados recebendo o label no seu construtor, que será apresentado no formulário, não necessitando assim criar um componente label para a descrição de cada componente que será apresentado na tela.

Observe que todos os campos foram setados como obrigatório através do método **setAllowBlank()**, com isso a tela de formulário já aparece com a exclamação do lado direito dos campos, exigindo o seu preenchimento. No nosso exemplo, é informada uma mensagem de validação padrão do componente, entretanto esta pode ser customizada através do método **setBlankText()**. Além da mensagem de validação, também pode ser customizada uma tooltip para cada campo através do método **setTitle()**.

O botão de confirmação dos dados só será habilitado após todas as validações

**Listagem 6.** Janela de formulário de funcionário: FuncionarioForm.java.

```

1 package exemlogwtext.client;
2
3 ...
4 public class FuncionariosForm extends Window {
5
6 ...
7     public FuncionariosForm(FuncionariosPanel
8         funcionarioPanel) {
9         this.funcionarioPanel = funcionarioPanel;
10        this.init();
11    }
12    private void init() {
13        ...
14    }
15
16    private Panel getPanelCentral() {
17        Panel panel = new Panel();
18        panel.setBorder(false);
19        panel.setPadding(15);
20
21        formPanel.setFrame(true);
22        formPanel.setTitle("Cadastro de funcionários");
23
24        formPanel.setWidth(350);
25        formPanel.setLabelWidth(75);
26
27        txtCodigo = new TextField("Código", "codigo", 230);
28        txtCodigo.setAllowBlank(false);
29        formPanel.add(txtCodigo);
30
31        txtNome = new TextField("Nome", "nome", 230);
32        txtNome.setAllowBlank(false);
33        formPanel.add(txtNome);
34
35        cboCargo = getCboCargo();
36        formPanel.add(cboCargo);
37
38        txtSalario = new NumberField("Salário",
39            "salario", 230);
40        txtSalario.setAllowBlank(false);
41        formPanel.add(txtSalario);
42
43        txtPais = new TextField("País", "pais", 230);
44        txtPais.setEnabled(false);
45        formPanel.add(txtPais);
46
47        final Button save = new Button();
48        save.setIcon("images/check2.png");
49        save.addListener(new ButtonListenerAdapter(){
50            public void onClick(Button button, EventObject e) {
51                Funcionario func = getFuncionarioForm();
52                if (edit) {
53                    updateEntity(func);
54                } else {
55                    saveEntity(func);
56                }
57                hide();
58            }
59        });
60        formPanel.addButton(save);
61
62        Button cancel = new Button();
63        cancel.setIcon("images/delete2.png");
64        cancel.addListener(new ButtonListenerAdapter(){
65            public void onClick(Button button, EventObject e) {
66                hide();
67            }
68        });
69
70        //habilita a validação
71        formPanel.setMonitorValid(true);
72        formPanel.addListener(new FormPanelListenerAdapter() {
73            public void onClientValidation(FormPanel formPanel,
74                boolean valid) {
75                save.setEnabled(!valid);
76            }
77        });
78        panel.add(formPanel);
79        return panel;
80    }
81 }
82
83 ...
84 private Funcionario getFuncionarioForm() {
85     Funcionario func = new Funcionario();
86     func.setCodigo(new Long(txtCodigo.getText()));
87     func.setNome(txtNome.getText());
88     func.setSalario(new Double(txtSalario.getText()));
89     func.setCargo(cboCargo.getValue());
90     func.setPais(txtPais.getText());
91     return func;
92 }
93
94 public void show(boolean edit) {
95     this.edit = edit;
96     if (edit) {
97         initEdit();
98     } else {
99         initNew();
100    }
101    show();
102 }
103
104 public void initEdit() {
105     txtCodigo.setEnabled(true);
106     txtNome.focus(true);
107 }
108
109 public void initNew() {
110     txtCodigo.setEnabled(false);
111     txtCodigo.focus(true);
112 }
113
114 public void loadRecord(Record record) {
115     this.record = record;
116     formPanel.getForm().loadRecord(record);
117 }
118
119 private void saveEntity(Funcionario func) {
120     System.out.println("Salvando func");
121     this.funcionarioPanel.addRecord(func);
122 }
123
124 private void updateEntity(Funcionario func) {
125     System.out.println("Atualizando func");
126     formPanel.getForm().updateRecord(record);
127 }
128
129 ...
130 }

```

#### Listagem 7. Arquivo de configuração do GWT: ExemploGWText.gwt.xml.

```
1 <module>
2 <!-- Inherit the core Web Toolkit stuff. -->
3 <inherits name='com.google.gwt.user.User' />
4 <inherits name='com.gwt.ext.GwtExt' />
5
6 <!-- Specify the app entry point class. -->
7 <entry-point class='exemplogwtext.client.ExemploGWText' />
8 <stylesheet src='css/ext-all.css' />
9 <script src='js/ext-base.js' />
10 <script src='js/ext-all.js' />
11 </module>
```

#### Listagem 8. Arquivo HTML inicial da aplicação: ExemploGWText.html.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Author" content="Sankar Gorathi">
<title>Hello World</title>
</head>
<body>
<iframe id="__gwt_historyFrame" style="width:0;height:0;border:0">
</iframe>
<!-- The GWT js file generated at run time -->
<script type="text/javascript" src='exemplogwtext.ExemploGWText.
nocache.js'></script>
</body>
</html>
```

dos campos estiverem ok (linhas 71-76 da **Listagem 6**). Passando a etapa de validação, caso seja uma operação de inclusão, é invocado o método **saveEntity(Funcionario func)**, onde o funcionário é construído a partir dos campos do formulário. O método repassa a inclusão para **FuncionarioPanel (Listagem 4)**, que simplesmente cria o registro a partir dos dados do objeto funcionário e adiciona no **Store**. Para a operação de atualização basta a invocação de **formPanel.getForm().updateRecord(Record)**, que o registro é automaticamente atualizado no grid devido ao sincronismo criado entre o formulário e o grid (linhas 124-127 da **Listagem 6**). Por fim, a remoção de um registro do grid é realizada simplesmente através do comando **store.remove(grid.getSelectionModel().getSelected())**.

### Usando outros componentes

Além dos componentes apresentados na aplicação de exemplo, existem outros bastante sofisticados oferecidos pela biblioteca. Existem componentes de árvores editáveis, com opção de checkbox, ordenação, multi-seleção e *drag and drop*. Há painéis que podem ser ocultados, móveis e ainda apresentar conteúdos formatados

de arquivos Java e XML, por exemplo. Existem diversos tipos de janelas (**Window**), como diálogos de confirmação, alertas e progresso. **Comboboxes** linkados com outros e/ou paginados. **Grid** com agrupamentos entre linhas, com paginação local ou remota e/ou editáveis. As opções de layouts disponíveis também são bastante interessantes, além do **BorderLayout** temos **Horizontal**, **Vertical**, **Accordion**, **Anchor**, **Column**, **Table**, **CardLayout** e **RowLayout**, além da combinação destes. Existe ainda a possibilidade de se trabalhar com temas, assim como o *look and feel* do Swing em Java Desktop. Enfim, encontramos uma gama enorme de componentes que podem ser mais bem visualizados no demo exposto através do site do GWT-Ext (**Links**). E ainda é disponibilizado neste mesmo demo o código fonte dos componentes apresentados, tornando-o uma grande fonte de pesquisa e aprendizado.

### Conclusões

Apesar de ser uma tecnologia nova, o GWT-Ext já se mostra muito poderoso e tem grande chance de ser bem sucedida por unir duas bibliotecas bastante utilizadas: o GWT (que tem a vantagem de ser criada pela Google) e o ExtJS, que trouxe um grande avanço

para desenvolvimento de aplicações web com interfaces mais ricas e amigáveis.

Esse artigo apresentou, através de um exemplo simples, um passo-a-passo para quem deseja iniciar no desenvolvimento do GWT-Ext, além de como configurá-lo para utilização. A simplicidade e o fato de não ser necessário conhecer outra linguagem além do Java para o desenvolvimento são as grandes marcas dessa tecnologia que tem muito a crescer. Mãos à obra, e bom estudo!

<http://code.google.com/webtoolkit/>

Página principal do GWT.

<http://gwt-ext.com/>

Página principal do GWT-Ext.

<http://extjs.com/>

Página principal da biblioteca ExtJS.

<http://www.jroller.com/sjivan/>

Blog do criador do GWT-Ext.

<http://www.gwt-ext.com/demo/>

ShowCase Demo da versão 2 da biblioteca GWT-Ext.



#### Izalmo Primo da Silva

[izalmo@yahoo.com.br](mailto:izalmo@yahoo.com.br)

é Bacharel em Ciência da Computação pela Universidade Candido Mendes – Campos/RJ e Mestre em informática pela UFRJ. Desenvolve soluções em Java desde

2004 e atualmente trabalha na TI da Petrobras com aplicações voltadas para exploração e produção de petróleo. É também professor acadêmico da disciplina de Java aplicada em padrões de projetos.



#### Samuel dos Santos Silva

[samuca.santos@gmail.com](mailto:samuca.santos@gmail.com)

é Bacharel em Ciência da Computação pela Universidade Federal de Pernambuco. Possui a certificação SCJP. Desenvolve soluções em Java desde 2000 e atualmente trabalha na TI da Petrobras com aplicações voltadas para exploração e produção de petróleo.

### Dê seu feedback sobre esta edição!

A Java Magazine tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

[www.devmedia.com.br/javamagazine/feedback](http://www.devmedia.com.br/javamagazine/feedback)

