

Integrando Flash Builder 4 com o Eclipse

Quando comecei estudar WebServices sempre perguntei se conseguiria desenvolver utilizando toda a interoperabilidade que os WebServices proporcionam com toda a riqueza interativa do desktop sem a necessidade de distribuir executáveis ou DLLs. Na medida em que as aplicações desktop foram sendo substituídas por versões Web, foram surgindo novas tecnologias e soluções que permitem uma interatividade com o browser semelhante a que se tem nas aplicações desktop. Proponho o desenvolvimento de uma aplicação bem simples que demonstra alguns recursos do Flash Builder 4.

A intenção é possibilitar o entendimento da integração do Flash Builder 4 com o Java utilizando o BlazeDS, o Adobe Flash Builder antes chamado Adobe Flex Builder permite desenvolver Aplicações Ricas para Internet (RIA).

O Adobe Flash Builder 4 Eclipse Plug-in pode ser baixado no seguinte site:

https://www.adobe.com/cfusion/tdrc/index.cfm?product=flash_builder

Partindo do princípio que o Eclipse já esteja instalado, instale o Flash Builder 4 Eclipse Plug-in, a instalação é bem simples e intuitiva. Utilizei o Eclipse-jee-galileo-SR2-win32.

Utilizei a versão blazeds-bin-4.0.0.14931 do BlazeDS, a versão pode ser encontrada no site:

<http://opensource.adobe.com/wiki/display/blazeds/download+blazeds+4>

O BlazeDS permite a comunicação entre o cliente Flex e o servidor Java, usa o protocolo Action Message Format (AMF), que possibilita a troca de mensagens binárias entre cliente e servidor utilizando o protocolo HTTP. A transferência dos dados chega a ser 10 vezes mais rápido do que utilizando formato de texto como o XML SOAP.

Criando o projeto

Selecione File->New->Dynamic Web Project e configure de acordo com a Imagem 1.

Selecionando Next, na tela seguinte selecione Edit... e mude o build path de src para src_java, esta alteração é necessário para separar os arquivos Java dos arquivos Flex.

Finalize a criação do projeto selecionando Finish.

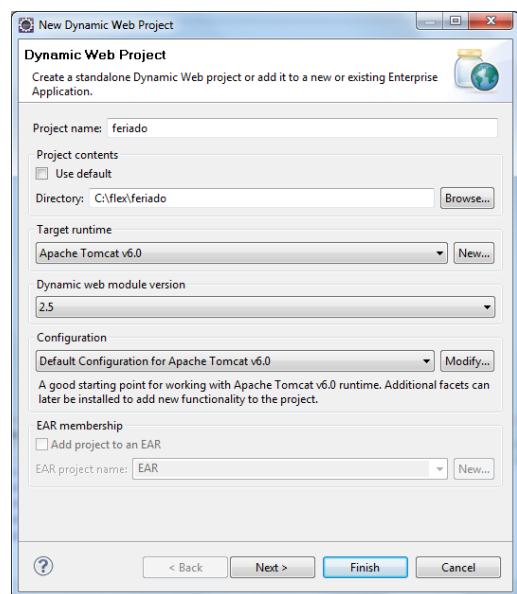


Imagem 1

Adicionando as dependências do Flex

O arquivo BlazeDS baixado contém um blazeds.war, descompacte-o, vá para o diretório WEB-INF do arquivo descompactado, copie os diretórios flex, lib e o arquivo web.xml para o diretório WEB-INF do projeto feriado substituindo os arquivos já existentes.

A estrutura do projeto deve ficar como mostrado na Imagem 2.

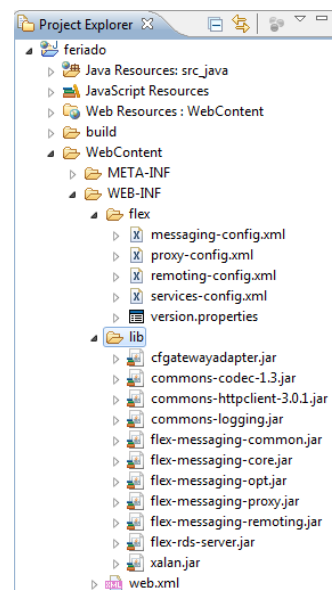
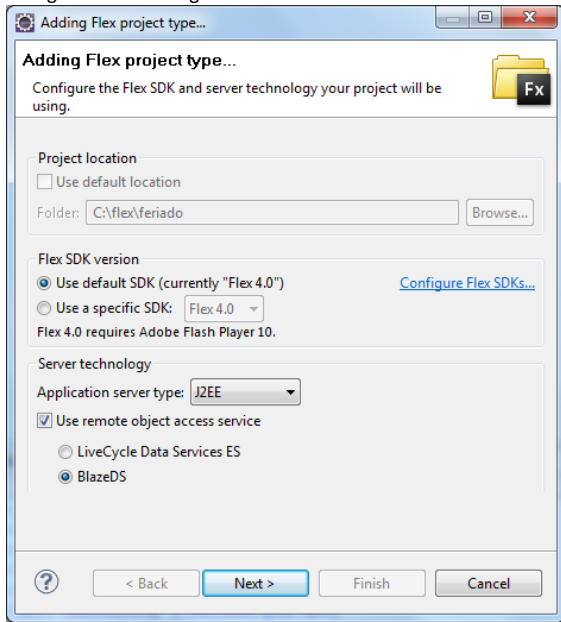


Imagem 2

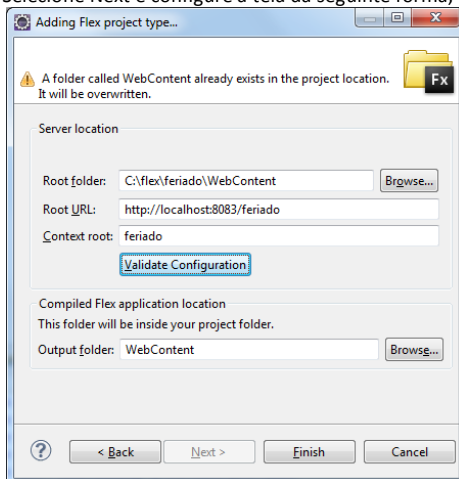
Adicionando o projeto tipo Flex

Selecione o projeto Feriados com o botão direito do mouse -> Add/Change Project Type-> Add Flex Project Type...

Configure a tela da seguinte forma:



Selecione Next e configure a tela da seguinte forma, foi usada a porta 8085 do servidor web, coloque a porta de acordo com o seu ambiente:



Selecione Finish, será feita uma pergunta se deseja mudar para a perspectiva Flex, selecione Yes.

O Eclipse mostra um erro informando que não foi possível criar a página HTML que carrega o arquivo feriado.swf pedindo para você clicar com o botão direito do mouse e recriar o arquivo, faça isso.

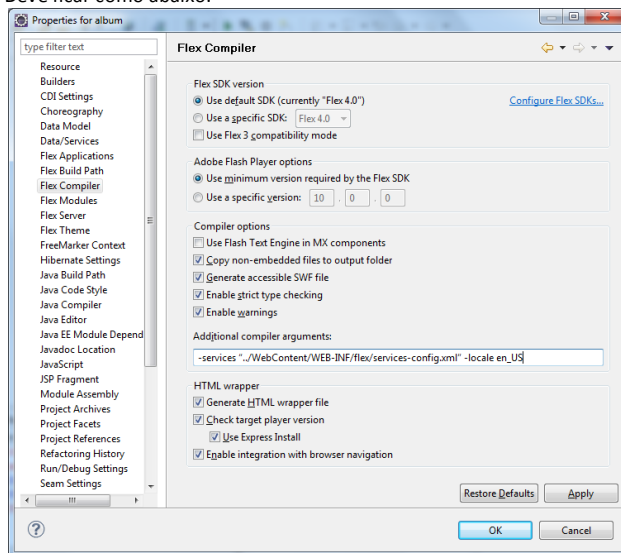
É necessário fazer uma alteração nas configurações do Flex, clique com o botão direito do mouse no projeto, selecione Properties e Flex Compiler, no campo Additional compiler arguments mude de

```
-services "C:\Projeto\album\WebContent\WEB-INF\flex\services-config.xml" -locale pt_BR
```

para

```
-services "../WebContent/WEB-INF/flex/services-config.xml" -locale en_US
```

Deve ficar como abaixo:



Mude a perspectiva para JavaEE e na aba Servers adicione um novo servidor, estou usando o Tomcat. Após configurar um servidor web, associar o projeto e iniciá-lo. Digite no browser a seguinte URL: <http://localhost:8083/feriado/feriado.html> deve aparecer uma tela em branco no browser.

Para que o servidor saiba em qual classe estão os métodos invocados pelo cliente é necessário vinculá-las a um identificador no arquivo `../WebContent/WEB-INF/flex/remoting-config.xml`

```
<destination id="feriadoControl">
    <properties>
        <source>br.com.estudos.controlador.FeriadoControlador</source>
    </properties>
</destination>
```

Criando as classes Java

Classe Entidade Feriado:

package br.com.estudos.entidade;

import java.util.Date;

public class Feriado {

private String descricao;

private Date data;

 /*

 * O BlazeDS exige ter um construtor padrao na classe para poder

 * serializar a classe Java para ActionScript

 * ou de ActionScript para Java

 */

public Feriado() {}

public Feriado(String descricao, Date data) {

this.descricao = descricao;

this.data = data;

 }

public String getDescricao() {

return descricao;

 }

public void setDescricao(String descricao) {

this.descricao = descricao;

 }

public Date getData() {

return data;

 }

public void setData(Date data) {

this.data = data;

```

}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((data == null) ? 0 : data.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Feriado other = (Feriado) obj;
    if (data == null) {
        if (other.data != null)
            return false;
    } else if (!data.equals(other.data))
        return false;
    return true;
}
}

```

Classe Controladora:

```

package br.com.estudos.controlador;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import br.com.estudos.entidade.Feriado;

public class FeriadoControlador {

    private static List<Feriado> feriados = new ArrayList<Feriado>();

    /*
     * Metodo que adiciona alguns feriados na lista de feriados
     */
    static {
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        try {
            feriados.add(new Feriado("Carnaval", sdf.parse("15/02/2010")));
            feriados.add(new Feriado("Paixão de Cristo", sdf
                .parse("02/04/2010")));
            feriados.add(new Feriado("Tiradentes", sdf.parse("21/04/2010")));
            feriados.add(new Feriado("Dia do Trabalho", df.parse("01/05/2010")));
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }

    public String adiciona(Feriado feriado) {
        if (this.getFeriados().contains(feriado)) {
            this.getFeriados().get(this.getFeriados().indexOf(feriado))
                .setDescricao(feriado.getDescricao());
        } else
            this.feriados.add(feriado);
        return "ok";
    }

    public boolean remover(Feriado feriado) {
        boolean excluido = false;
        for (Feriado f : this.getFeriados()) {
            if (f.equals(feriado)) {
                this.getFeriados().remove(f);
                excluido = true;
            }
        }
    }
}

```


Código do MXML Application:

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" minHeight="600">

  <fx:Script>
    <![CDATA[
      import br.com.estudos.entidade.Feriado;
      import br.com.estudos.utils.DateUtil;

      import mx.collections.ArrayCollection;
      import mx.controls.Alert;
      import mx.events.ValidationResultEvent;
      import mx.rpc.events.FaultEvent;
      import mx.rpc.events.ResultEvent;

      [Bindable]
      private var feriados:ArrayCollection = new ArrayCollection();

      [Bindable]
      private var _feriado:Feriado = new Feriado();

      protected function remotferiadoControl_faultHandler(event:FaultEvent):void
      {
          Alert.show(event.fault.faultString);
      }

      private function resultPesquisaFeriados(event:ResultEvent):void {
          this.feriados = event.result as ArrayCollection;
      }

      private function resultAdiciona(event:ResultEvent):void {
          remotferiadoControl.getFeriados();
      }

      private function resultExcluir(event:ResultEvent):void {
          remotferiadoControl.getFeriados();
      }

      protected function salvar_clickHandler(event:MouseEvent):void
      {
          var evTituloValid:ValidationResultEvent = validaFeriado.validate();

          if (evTituloValid.type== ValidationResultEvent.INVALID) {
              Alert.show(validaFeriado.requiredFieldError);
          }
          else
              remotferiadoControl.adiciona(this._feriado);
      }
    ]]>
  </fx:Script>

  <fx:Declarations>
    <s:RemoteObject id="remotferiadoControl" destination="feriadoControl"
      fault="remotferiadoControl_faultHandler(event)">
      <s:method name="getFeriados" result="resultPesquisaFeriados(event)" />
      <s:method name="adiciona" result="resultAdiciona(event)" />
      <s:method name="remover" result="resultExcluir(event)" />
    </s:RemoteObject>
    <mx:StringValidator id="validaFeriado" source="{txtDescricao}" property="text" minLength="4"
      maxLength="50"
      requiredFieldError="Obrigatório o preenchimento do campo Feriado." required="true"/>
  </fx:Declarations>

  <fx:Binding source="dtgrdFeriados.selectedItem as Feriado" destination="_feriado"/>
  <fx:Binding source="txtDescricao.text" destination="_feriado.descricao" twoWay="true"/>
  <fx:Binding source="_feriado.data" destination="dtfldData.selectedDate" twoWay="true"/>
</s:Application>
```

```

<mx:VBox>
  <mx:Form x="15" y="2">
    <mx:FormHeading label="Cadastra feriados." />

    <mx:FormItem label="Feriado:" required="true">
      <s:TextInput id="txtDescricao" width="230" />
    </mx:FormItem>

    <mx:FormItem label="Data:" required="true">
      <mx:DateField id="dtfldData" selectedDate="{_feriado.data}" width="230" />
    </mx:FormItem>

    <mx:FormItem label="Dia:" enabled="false">
      <s:TextInput id="txtDia" text="{DateUtil.diaSemana(_feriado.data.day)}" width="230" />
    </mx:FormItem>

    <mx:FormItem>
      <mx:HBox>
        <s:Button id="btnSalvar" label="Salvar" click="salvar_clickHandler(event)" />
        <s:Button id="btnExcluir" label="Excluir" click="remotferiadoControl.remover(this._feriado)"
          enabled="{(dtgrdFeriados.selectedIndex > -1)}" />
      </mx:HBox>
    </mx:FormItem>
  </mx:Form>

  <mx:DataGrid id="dtgrdFeriados" dataProvider="{this.feriados}"
    creationComplete="{remotferiadoControl.getFeriados()}" width="461">
    <mx:columns>
      <mx:DataGridColumn headerText="Feriado" dataField="descricao" />
      <mx:DataGridColumn headerText="Data"
        dataField="data" labelFunction="{DateUtil.formataData}" />
      <mx:DataGridColumn headerText="Dia"
        dataField="data" labelFunction="{DateUtil.retornaDiaSemana}" />
    </mx:columns>
  </mx:DataGrid>
</mx:VBox>
</s:Application>

```

Dentro da tag <fx:Script> é adicionado os métodos, variáveis e referencias que serão utilizados:

`feriados` – Armazena uma coleção de instancias da classe `Feriado`

`_feriado` – O Flex não permite ter nome de variável igual ao nome do arquivo MXML, por este motivo foi colocado o caractere underscore no início, esta variável armazena os valores do objeto visualizado no formulário.

Binding e Bindable

Binding é uma forma simples e elegante de atribuir valores dos campos dos formulários às propriedades das classes ou variáveis, o Bindable trabalha com o padrão Observer, ou seja: Quando definimos Bindable a uma variável, qualquer alteração nesta variável será refletida para as dependências, no padrão observer é define um evento que atualiza as dependências, no Flex não é diferente, porem pode-se deixar a criação de tais eventos para o próprio Flex criá-los na hora da compilação.

```
<fx:Binding source="dtgrdFeriados.selectedItem as Feriado" destination="_feriado" />
```

A linha de código acima pega o objeto retornado pelo método `selectedItem` faz um CAST para o tipo `Feriado` e atribui uma referencia para o objeto `_feriado`.

```
<fx:Binding source="txtDescricao.text" destination="_feriado.descricao" twoWay="true" />
```

O código acima cria um vinculo entre a propriedade `text` do controle `txtDescricao` e a propriedade `descricao` do objeto `_feriado`, o parâmetro `twoWay` faz a atribuição nos dois sentidos, ou seja, uma alteração no valor da propriedade `descricao` do objeto `_feriado` será refletida na propriedade `text` do controle `txtDescricao` e se a alteração for na propriedade `text` do controle `txtDescricao` vai ser refletida na propriedade `descricao` do objeto `_feriado`. Para ter a alteração apenas do `source` para o `destination` basta colocar `twoWay` com o valor `false`.

Dentro da tag <fx:Declarations> deve ser informado onde os métodos estão no servidor

`id="remotferiadoControl"`: Identificador do RemoteObject.

`destination="feriadoControl"`: Este parâmetro recebe o nome que foi dado para o destination no arquivo `remoting-config.xml` no servidor que esta vinculado à classe que implementa os métodos invocados neste `RemoteObject`.

`fault="remotferiadoControl_faultHandler(event)"`: Método padrão que será invocado caso ocorra algum erro na chamada dos métodos pertencentes a este `RemoteObject`.

`s:method name="getFeriados" result="resultPesquisaFeriados(event) "`: Identifica o nome do método invocado, não é informado os parâmetros caso exista, o método informado no parâmetro `result` será invocado quando o servidor retornar uma resposta, `event` armazena o valor retornado pelo método do servidor.

```
<mx:StringValidator id="validaFeriado" source="{txtDescricao}" property="text" minLength="4" maxLength="50"
    requiredFieldError="Obrigatório o preenchimento do campo Feriado." required="true"/>
```

O Flex inclui vários validadores como `DateValidator`, `EmailValidator`, `NumberValidator`, no entanto podemos criar nossos próprios validadores em ActionScript a partir da classe `Validator`, neste exemplo esta sendo utilizado um `StringValidator` para impedir o cadastro de um feriado com nome inferior a quatro caracteres ou com mais de cinquenta, se o usuário não preencher o campo será mostrado uma mensagem informando que o preenchimento do campo é obrigatório. Em uma aplicação real é necessário validar alguns campos tanto do lado cliente quanto do lado servidor, não esta sendo feito esta validação por ser apenas um exemplo.

Este foi apenas um exemplo simples utilizando poucos recursos da tecnologia, direcionado para quem esta iniciando os estudos da integração Java, Flex e BlazeDS.

Recomendo dar uma olhada nos seguintes Links:

<http://www.adobe.com/br/products/flashbuilder/>

http://livedocs.adobe.com/livecycle/es/sdkHelp/programmer/lcds/wwhelp/wwhimpl/common/html/wwhelp.htm?context=LiveDocs_Parts&file=serialize_data_2.html

Livro: Adobe FLEX 3 – Treinamento diretor da Fonte, Jeff Tapper, Michael Labriola e Matthew Boles com James Talbot

Criado por: Juraci Alves dos Santos
Goiânia – GO
juracisantos@gmail.com