Introdução ao JSF

Daniel Bruno
dbconrado@gmail.com

II Semana de Tecnologia da Informação
UNIVEM

O que é?

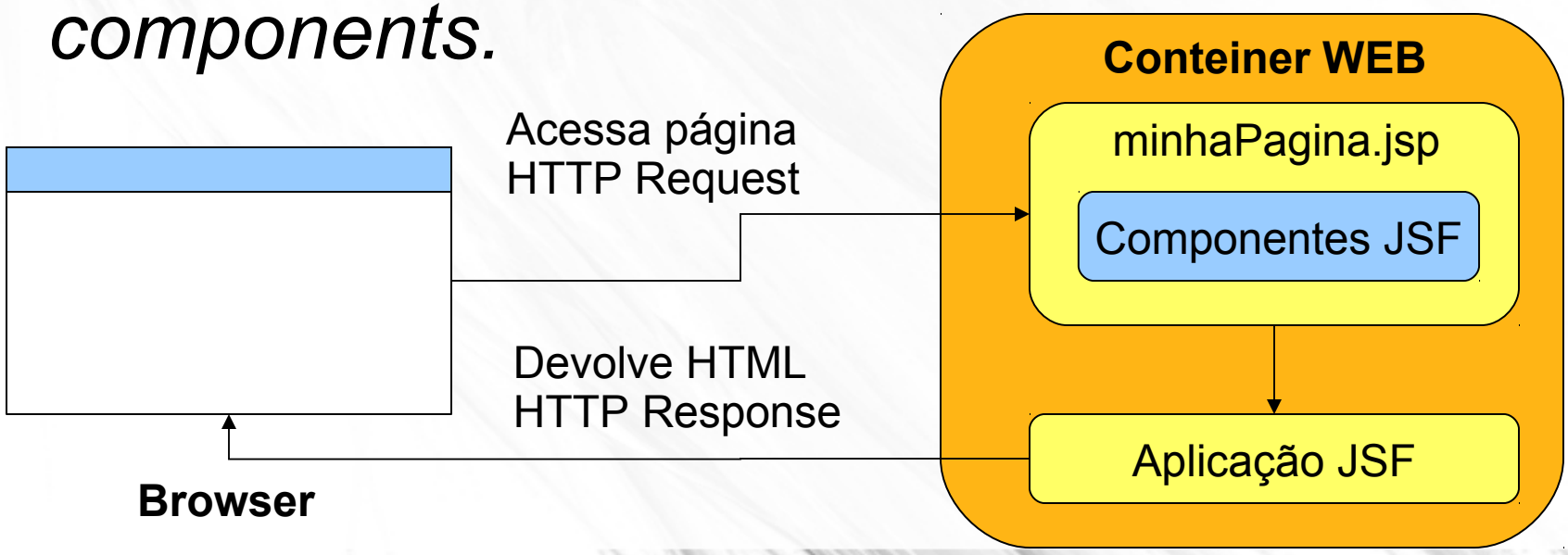
- Framework de interface com o usuário;
- Orientado a componentes;
- Altamente extensível/personalizável;
- Especificado pelo JCP 252 (participação de várias empresas);

O que fornece?

- Gerência de estados dos componentes;
- Tratamento de eventos;
- Validação;
- Conversão de dados;
- Mapeamento das navegações entre páginas;
- Internacionalização, entre outros.

Como funciona

- Os componentes, seus estados, seus eventos, validações, etc., acontecem **no servidor**. Isto se chama *server-side components.*



Como funciona

- minhaPagina.jsp é uma página JSP normal que contém tags de componentes JSF;
- Aplicação JSF é a que você criou, contendo:
  - Tratadores de eventos;
  - Validadores e conversores;
  - **JavaBeans** que encapsulam dados e comportamentos específicos para os componentes inseridos na página;
  - Entre outros.

Benefícios

- Separação limpa entre apresentação e comportamento;
- Separação de papéis dentro de uma equipe (i.e., programador de páginas e programador de aplicação);
- Não se limita à JSP. Outras tecnologias de apresenção podem ser utilizadas (i.e., Facelets).

Uma aplicação web JSF

- É uma aplicação web em Java comum, com algumas coisas a mais:
  - Conjunto de páginas JSP (se for a tecnologia de apresentação usada);
  - Conjunto de **backing beans** (componentes JavaBeans que definem propriedades e funções para componentes UI em uma página);

Uma aplicação JSF

- Continuando:
  - Arquivo de configuração de recursos, que define navegação de páginas, configurações dos backing beans, entre outros;
  - Objetos validadores, conversores, ou tratadores de eventos;
  - Conjunto de tags customizadas, para representar objetos em uma página.

Papéis em uma aplicação JSF

- Autor de páginas:
  - Cria as páginas utilizando as bibliotecas de tags do JSF;
- Desenvolvedor de aplicação:
  - Criador de conversores, validadores, tratadores de eventos e backing beans;
- Autor de componentes:
  - Cria componentes e renderizadores;

Papéis em uma aplicação JSF

- Arquiteto da aplicação:
  - Configura toda a aplicação;
  - Define regras de navegação entre páginas;
  - Configura objetos (conversores, validadores, etc);
  - Entre outros;

Hello World em JSF

- Vamos criar uma aplicação com duas páginas, sendo:
  - helloForm.jsp: a página que receberá o nome do usuário;
  - helloMessage.jsp: a página que exibirá uma mensagem customizada com o nome do usuário.

Hello World em JSF

- Abra o NetBeans 6.5.1;
- Vá em File → New Project...;
- Em Categories, selecione *Java Web;*
- Em Projects, selecione *Web Application*;
- Clique em Next;
- Em Project Name, informe *HelloJSF*;

Hello World em JSF

- Opcionalmente, você pode definir um diretório alternativo onde será gravado o projeto em *Project Location*;
- Clique em Next;
- Em Server, selecione Tomcat 6.0;
- Em Java EE Version, selecione Java EE 5;
- Context path provavelmente será /HelloJSF;

Hello World em JSF

- O context path é como sua aplicação será chamada para execução, dentro do servidor. Neste caso, a nossa aplicação será chamada assim:
- http://localhost:8080/HelloJSF;
- Onde:
  - Localhost é o servidor (neste caso, a máquina local);

Hello World em JSF

- Continuando...:
  - 8080 é a porta onde o servidor Tomcat está executando;
  - /HelloJSF é o context path;
- Clique em Next;
- Em Frameworks, selecione **apenas** *JavaServer Faces* (o nosso JSF);

Hello World em JSF

- Na aba Configuration temos JSF Servlet Name como Faces Servlet. Esta servlet é responsável por toda execução do JSF. Toda página que contiver componentes JSF deve ser executada por ela. Por isso, temos abaixo o Servlet URL Pattern. Este indica quais páginas serão executadas pela Servlet;

Hello World em JSF

- Neste caso, informamos /faces/*;
- Isto significa que todas as páginas chamadas com o prefixo /faces/ serão executadas pela Faces Servlet. Por Exemplo:
- http://localhost:8080/HelloJSF/faces/helloForm.jsp;
- http://localhost:8080/HelloJSF/faces/helloMessage.jsp;

Hello World em JSF

- Se chamarmos a página helloForm.jsp desta forma: http://localhost:8080/HelloJSF/helloForm.jsp, ela não será executada pela Faces Servlet. Como consequência, os componentes JSF desta página não serão executados.

Hello World em JSF

- É comum os desenvolvedores mudarem este URL pattern para *.jsf. Desta forma, as páginas seriam acessadas assim:
- http://localhost:8080/HelloJSF/helloForm.jsf;
- http://localhost:8080/HelloJSF/helloMessage.jsf;
- Vamos deixar a configuração padrão;

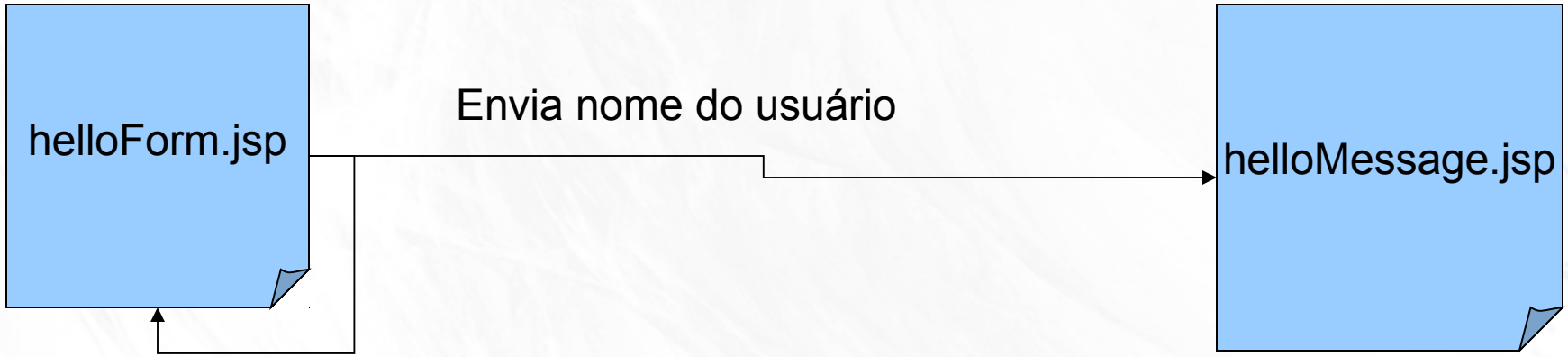
Hello World em JSF

- O NetBeans constrói o projeto e nos mostra várias pastas lógicas. Por enquanto, vamos nos concentrar em apenas três, a saber:
  - Web Pages – aí é onde o NetBeans organizará as páginas JSP, bem como arquivos CSS, javascript's, imagens, etc.;
  - Configuration Files – arquivos de configuração da aplicação;
  - Source Packages – as classes Java de nossa aplicação.

Hello World em JSF

- O NetBeans, por padrão, cria uma página de Hello World chamada welcomeJSF.jsp. Exclua-a. Não vamos precisar dela. Para isso, clique nela, na parte Web Pages, e aperte Delete.

Estrutura da aplicação Hello JSF



Mostra mensagem de erro,
Caso nome esteja vazio

Backing Bean NomeBean

- Para enviarmos o nome de uma página a outra, precisamos de um backing bean;
- É uma classe Java normal que segue o padrão JavaBeans;
- Simplesmente conterá um atributo String nome e seus métodos acessores get/set;

Backing Bean NomeBean

- Clique com o botão direito em Source Packages e, no menu que aparecer, selecione New → Java Class;
- Em Class Name, informe NomeBean;
- Em Package, informe bean;
- Clique em Finish;
- Insira um atributo String nome e os métodos get/set. A classe ficará conforme próx. slide;

Backing Bean NomeBean

```
public class NomeBean {
    private String nome;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Configurando NomeBean

- Vamos configurar o bean para que ele possa ser utilizado em nossas páginas. Para isso, vá em Configuration Files e dê duplo clique em faces-config.xml;
- Clique na parte XML e, dentro da tag faces-config, coloque:

```
<managed-bean>
   <managed-bean-name>nomeBean</managed-bean-name>
   <managed-bean-class>bean.NomeBean</managed-bean-class>
   <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

Configuranto NomeBean

```
<faces-config version="1.2"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/

    <managed-bean>
        <managed-bean-name>nomeBean</managed-bean-name>
        <managed-bean-class>bean.NomeBean</managed-bean-class>
        <managed-bean-scope>request</managed-bean-scope>
    </managed-bean>

</faces-config>
```

Explicando a configuração

- Managed-bean-name: é o nome pelo qual as páginas JSP acessarão o bean;
- Managed-bean-class: a classe java do bean. Neste caso, dizemos que é a classe NomeBean que está dentro do pacote bean;
- Managed-bean-scope: o escopo do bean. Isto será explicado posteriormente.

Criando a primeira página

- Agora que já temos o bean configurado, vamos criar nossa primeira página JSP com componentes JSF.

Página helloForm.jsp

- Clique com o botão direito em Web Pages. No menu que aparecer, selecione New → JSP;
- Em JSP File Name, informe *helloForm*;
- Clique em Finish;

Página helloForm.jsp

- Abaixo da diretiva page, inclua as taglib's do JSF:

```
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html" %>
```

Página helloForm.jsp

- Dentro da tag <body> insira:

```
<f:view>
    <h:form>
        <h:outputLabel for="itNome" value="Informe seu Nome"/>
        <h:inputText id="itNome" value="#{nomeBean.nome}"/>
        <br/>
        <h:commandButton value="OK"/>
    </h:form>
</f:view>
```

Explicando a tag <f:view>

- A tag <f:view> cria uma **view JSF**. Ela é um container que tem todos os componentes que fazem parte da view;
- Cada página só pode ter uma única view;
- Todos os demais componentes JSF devem estar dentro dela;
- Qualquer componente que fique fora pode gerar um erro.

Explicando a tag <h:form>

- Representa a tag form do HTML;
- Dentro do form, devem estar os componentes para entrada de dados, que serão posteriormente enviados;

Explicando a <h:outputLabel>

- Esta tag representa uma tag HTML <label>;
- Atributo **for** indica a qual componente este label está associado;
- Atributo value é o texto que será exibido;

Explicando a <h:inputText>

- Representa uma caixa de texto;
- Atributo **id** é o nome do componente dentro da página;
- Value é o valor da caixa de texto. Este atributo também é utilizado para gravar o valor digitado pelo usuário.

EL Expressions

- Colocamos no atributo value do inputText o seguinte: #{nomeBean.nome};
- Isto se chama expressão EL ou EL expression;
- Utilizada para acessar os backing beans;

EL Expressions

- Decompondo esta expressão, temos:
  - nomeBean – o nome do bean que configuramos no faces-config;
  - nome – a propriedade **nome** do bean nomeBean.
- O que define uma propriedade são os métodos **set/get**;
- Como criamos os métodos getNome() e setNome(), **nome** virou uma propriedade;

EL Expressions

- Quando a página for renderizada, o componente inputText chamará o método getNome() para pegar o valor atual da propriedade;
- Quando a página for submetida, o inputText chamará o método setNome(), passando por parâmetro o texto que foi digitado pelo usuário;

Explicando

- Uma simples quebra de linha;

Explicando <h:commandButton>

- Representa um botão;
- Atributo value é o texto do botão;
- Alteraremos este componente mais pra frente;

Criando a helloMessage.jsp

- Clique com o botão direito em Web Pages e, no menu que aparecer, clique em New → JSP;
- Dê o nome de helloMessage e clique em Finish;
- Adicione as taglib's do JSF, conforme página anterior;

Criando a helloMessage.jsp

- Dentro da tag <body> adicione:

```
<f:view>
    <h1><h:outputText value="Olá, #{nomeBean.nome}"/></h1>
</f:view>
```

Configurando navegação

- Vamos configurar a navegação entre as páginas;
- A navegação se dá através de string's chamadas de **outcome**'s;
- Configuramos a navegação através de regras. Uma regra de navegação contém a origem e uma série de pares outcome/destino;

Exemplo de regra de navegação

- Origem: menu.jsp;
  - Outcome: clientes, Destino: clientes.jsp;
  - Outcome: produtos, Destino: produtos.jsp;

Configurando Navegação

- Vá em Configuration Files e abra o faces-config.xml;
- Na parte PageFlow, o NetBeans colocou automaticamente todas as nossas páginas;
- Clique e segure no pequeno quadrado ao lado do sinal de mais (+) da página helloForm.jsp e arraste até o mesmo quadrado da página helloMessage.jsp;

Configurando Navegação

- O NetBeans criará uma linha ligando as duas páginas com o nome case1;
- Selecione a linha e vá na janela Properties;
- Em Outcome, digite **mensagem**;

Regra criada

- Origem: helloForm.jsp;
  - Outcome: mensagem, Destino: helloMessage.jsp

Voltando a helloForm.jsp

- Volte à pagina helloForm.jsp e altere o componente commandButton conforme abaixo:

```
<h:commandButton value="OK" action="mensagem"/>
```

Explicando...

- O atributo action do componente commandButton pode ser o outcome da regra de navegação que criamos;
- Desta forma, quando o usuário clicar no botão, será direcionado para a página helloMessage.jsp. O valor digitado no campo de texto será gravado no NomeBean.

Antes de rodar...

- Antes de rodarmos nossa aplicação, precisamos de uma página jsp que, ao ser acessada, direcione para a página helloForm.jsp;
- Por padrão, o servidor Web executa a página index.jsp da aplicação;
- Então, criamos a index.jsp que direciona para a página helloForm.jsp;

index.jsp

- Clique com o botão direito em Web Pages e, no menu, clique em New → JSP;
- Dê o nome de index e clique em Finish;
- Coloque dentro da tag body o seguinte:

```
<jsp:forward page="/faces/helloForm.jsp"/>
```

Configurando a index.jsp

- Vá em Configuration Files e abra o web.xml;
- Vá na aba Pages;
- Em Welcome Files, digite: index.jsp;
- Salve e execute o projeto.

Adicionando validação

- Vamos validar a entrada do nome do usuário;
- Ele não pode ser vazio;
- Para isso, vamos alterar o componente inputText;
- Abra a página helloForm.jsp;

Adicionando Validação

- No componente inputText, adicione mais dois atributos conforme abaixo:

```
<h:inputText id="itNome" value="#{nomeBean.nome}"
required="true" requiredMessage="Nome não pode ser vazio."/>
```

Adicionando Validação

- Além disso, adicione um componente <h:message> logo após o inputText, conforme abaixo:

```
<h:message for="itNome"/>
```

Explicando <h:message>

- É um componente que exibirá mensagens associadas ao componente descrito pelo atributo **for**;
- Ou seja, qualquer mensagem que o inputText lançar, será exibida por este componente;

Rodando

- Rode novamente a aplicação e clique em OK sem digitar nada no campo de texto;

Internacionalização

- Um recurso muito poderoso do JSF é a internacionalização;
- A aplicação pode ser facilmente traduzida para praticamente qualquer idioma;
- As mensagens ficam gravadas em arquivo de recursos chamado de Resource Bundle;
- Pode ser uma classe Java ou um arquivo texto.

Internacionalização

- Para internacionalizar nossa aplicação, vamos criar uns arquivos texto, contendo as mensagens de nossas páginas;
- Serão dois, um em português e outro em inglês;

Internacionalização

- Vamos criar um pacote para guardar nossos arquivos texto;
- Clique com o botão direito no projeto, selecione new → Package;
- No nome do pacote, informe *mensagem*;

Internacionalização

- Clique com o botão direito em cima do pacote mensagem, selecione new → Other;
- Em Categories, selecione Other;
- Em File Types, selecione Properties File;
- Dê o nome de Mensagens_pt_BR;

Entendendo o nome do arquivo

- Mensagens – será o nome base de todos os idiomas;
- pt_BR – indica a língua (pt – português) e o país (BR – Brasil);
- A extensão de um arquivo de recurso texto é .properties;

Mensagens_pt_BR.properties

- Digite dentro deste arquivo o seguinte:

```
informeNome=Informe seu Nome
nomeNaoVazio=Nome não pode estar vazio.
ok=OK
ola=Olá
ingles=English
portugues=Português
```

Arquivo em Inglês

- Agora, vamos criar as mensagens em inglês;
- Crie um novo arquivo properties e dê o nome de Mensagens_en_US;

Mensagens_en_US.properties

- Neste arquivo, digite:

```
informeNome=Enter your name
nomeNaoVazio=Name cannot be null.
ok=OK
ola=Hello
ingles=English
portugues=Português
```

Configurando o ResourceBundle

- Abra o faces-config.xml e, dentro da tag faces-config, digite:

```
<application>
    <resource-bundle>
        <base-name>mensagem.Mensagens</base-name>
        <var>mensagens</var>
    </resource-bundle>
    <locale-config>
      <default-locale>pt_BR</default-locale>
      <supported-locale>en_US</supported-locale>
      <supported-locale>pt_BR</supported-locale>
    </locale-config>
</application>
```

Explicando...

- Tag application: define configurações gerais da aplicação toda;
- Tag resource-bundle: serve para configurar um resource bundle;
- Tag base-name: indica o nome base dos arquivos de recurso. Neste caso, indicamos que está no pacote mensagem, e os arquivos começam com Mensagens.

Explicando...

- Assim, o JSF sabe que, para a língua portuguesa, país Brasil, ele deve procurar por: mensagem.Mensagens_pt_BR;
- Para a língua inglesa, país Estados Unidos, por: Mensagens_en_US;
- E assim por diante;

Explicando...

- Tag var: indica o nome da variável que será visível nas páginas JSP. Esta variável será utilizada para colocar as mensagens nas páginas. O nome que demos foi mensagens. Veremos mais adiante;
- Tag locale-config: configura os idiomas que podem ser usados na aplicação;
- Tag default-locale: língua padrão da aplicação;

Explicando...

- Tag supported-locale: indica os idiomas suportados pela aplicação. Neste caso, temos dois: pt_BR e en_US;
- Definimos, através da tag default-locale, que a aplicação será em português, caso o JSF não consiga identificar a língua do browser;

Alterando as páginas

- Agora, vamos substituir as mensagens estáticas de nossas páginas pelas mensagens dos arquivos de recurso;
- Abra a página helloForm.jsp;
- Substitua a frase “Informe seu nome” por “#{mensagens.informeNome}”;
- Veja que utilizamos a variável *mensagens*, previamente definida no faces-config;

Alterando helloForm.jsp

```
<h:outputLabel for="itNome" value="#{mensagens.informeNome}"/>
<h:inputText id="itNome" value="#{nomeBean.nome}"
required="true" requiredMessage="#{mensagens.nomeNaoVazio}"/>
<h:message for="itNome"/>
<br/>
<h:commandButton value="#{mensagens.ok}" action="mensagem"/>
```

Alterando helloMessage.jsp

```
<h:outputText value="#{mensagens.ola}, #{nomeBean.nome}"/>
```

Rodando a Aplicação

- Ao rodar a aplicação, ela deverá aparecer com o idioma padrão do seu browser (ou de sua máquina);

Forçando um idioma

- Podemos forçar um idioma na nossa aplicação;
- Vamos colocar dois links na página helloForm.jsp, Inglês e Português;
- Vamos também criar um backing bean que mude o idioma de nossa aplicação;

IdiomaBean

- Clique com o botão direito no projeto, selecione New → Java Class;
- Dê o nome de *IdiomaBean* e coloque no pacote *bean*;
- Crie um método chamado mudarIdioma() que receba, como parâmetro, um objeto Locale. Deverá ficar conforme próximo slide;

IdiomaBean

```
public void mudarIdioma(Locale locale) {
   FacesContext fc = FacesContext.getCurrentInstance();
   fc.getViewRoot().setLocale(locale);
}
```

Explicando...

- Um objeto Locale representa um idioma e, opcionalmente, um país;
- FacesContext representa o contexto da aplicação JSF. Contém todas as configurações;
- getViewRoot() retorna a view atual (a página que está sendo exibida/processada);

Explicando...

- O método setLocale() muda o idioma da aplicação para o idioma especificado pelo objeto Locale;

IdiomaBean - continuando

- Agora, vamos adicionar mais dois métodos especiais;
- Estes métodos são chamados por dois links que colocaremos na página helloForm.jsp;
- Eles devem obedecer à algumas regras, que são: a) devem retornar uma String (um outcome) e; b) não podem receber parâmetros;

IdiomaBean - continuando

```
public String ingles() {
   Locale locale = new Locale("en", "US");
   mudarIdioma(locale);
   return null;
}

public String portugues() {
   Locale locale = new Locale("pt", "BR");
   mudarIdioma(locale);
   return null;
}
```

Explicando...

- O método ingles() é responsável por mudar o idioma da aplicação para inglês;
- Ele cria um objeto Locale com o idioma inglês (en) e o país Estados Unidos (US);
- Utiliza o método mudarIdioma e retorna null;
- O método portugues() funciona de forma semelhante ao método ingles() exceto por mudar o idioma para português brasileiro;

Explicando...

- Por que retornar null?
- Quando utilizamos um link, geralmente é para navegar para uma outra página. No JSF, a navegação se dá pelos *outcome*'s;
- Neste caso, queremos ficar na mesma página, por isso, retornamos null;

Configurando o IdiomaBean

- Abra o faces-config.xml e, embaixo da configuração do NomeBean, digite:

```
<managed-bean>
  <managed-bean-name>idiomaBean</managed-bean-name>
  <managed-bean-class>bean.IdiomaBean</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

Colocando os links

- Abra a helloForm.jsp e, depois do commandButton mas antes do </h:form>, coloque:

```
<h:commandLink action="#{idiomaBean.ingles}"
value="#{mensagens.ingles}"
immediate=”true”/>
<br/>
<h:commandLink action="#{idiomaBean.portugues}"
value="#{mensagens.portugues}"
immediate=”true”/>
```

Rodando...

- Rode a aplicação e você verá que é possível mudar o idioma através dos links!

Explicando...

- commandLink representa um link em uma página;
- O atributo action não só pode receber uma string como também um método, ou a expressão do método (MethodExpression);
- Mas este método deve ser especial, conforme foi citado anteriormente (vide IdiomaBean);

Explicando...

- Atributo immediate, quando true, serve para pular antecipar algumas etapas;
- Se este atributo não estivesse aí, o link não funcionaria se o campo de texto estivesse vazio, pois a validação barraria.
  Experimente fazer isso;
- Quando true, o link será executado antes da validação;

Frameworks sobre frameworks

- Existem muitos frameworks que auxiliam o desenvolvimento com JSF;
- Frameworks de RIA (Rich Internet Applications);
- Frameworks de extensão do *core* do JSF;

Frameworks

- MyFaces Tomahawk;
- MyFaces Trinidad;
- MyFaces Tobago;
- MyFaces Orchestra;
- http://myfaces.apache.org/

Mais usados

- Richfaces (www.jboss.org/richfaces);
- ICEFaces (www.icefaces.org);
- ADF faces (http://jdevadf.oracle.com/adf-richclient-demo);

Emergente

- PrimeFaces (primefaces.prime.com.tr);

Muito obrigado.