JSF

JAVA SERVER FACES

JAVA SERVER FACES

- Arcabouço para o desenvolvimento de aplicações Web com foco na construção da interface de usuário.
- Combina Struts e Swing (conceitual)
- Como o Struts: Permite o controle/gerenciamento do Ciclo de Vida de uma aplicação Web através do uso de Servlets como controladores
- Como Swing: Fornece um conjunto de componentes ricos que permitem manipulação de eventos.

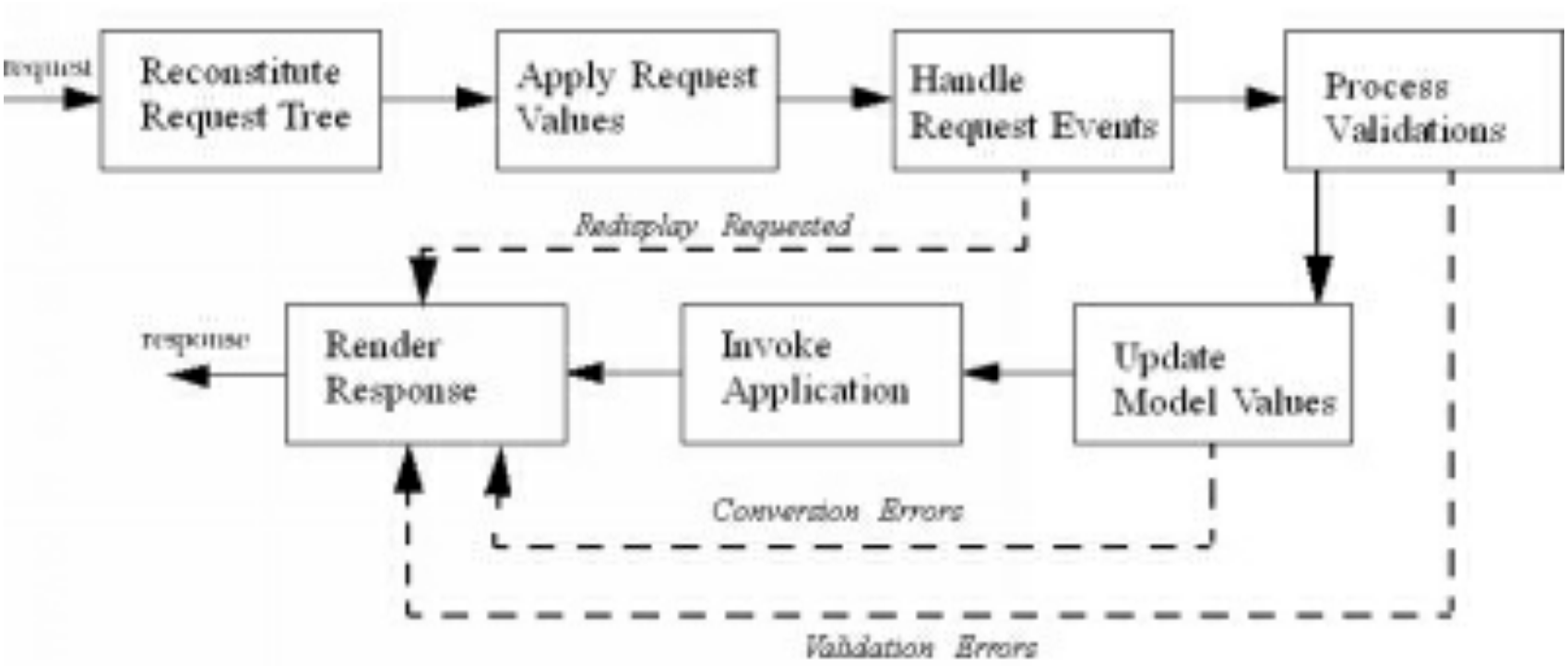
JAVA SERVER FACES

- Vantagens:
  - PERMITE CRIAR Uis A PARTIR DE COMPONETES (SERVER-SIDE) PADRÕES
  - FORNECE UM CONJUNTO DE TAGS JSP PARA ACESSAR COMPONENTES
  - PERMITE SALVAR O ESTADO DE INFORMAÇÕES DE FORMA TRANSPARENTE E REPOPULAR COMPONENTES NO MOMENTO DE SUA (RE)EXIBÇÃO
  - PERMITE CRIAR COMPONENTES CUSTOMIZADOS (EX.: PRIMEFACES)
  - ENCAPSULA A MANIPULAÇÃO DE EVENTOS E A RENDERIZAÇÃO DE COMPONENTES DE FORMA QUE É POSSÍVEL USAR COMPONENTES PADRÕES OU CUSTOMIZADOS DO JSF COM OUTRAS LINGUAGENS DE MARCAÇÃO DIFERENTES DO HTML

JAVA SERVER FACES

- É UM COMPETIDOR DIRETO DA MICROSOFT (WEBFORMS)
- SÃO (WEBFOMS E JSF) BEM SIMILARES TANTO NO CONCEITO QUNTO NA IMPLEMENTAÇÃO

JSF: CICLO DE VIDA



JSF: FASES DO CLICO DE VIDA

1. Reconstitute Request Tree: CRIA-SE UMA ARVORE DE COMPONENTES PARA A PÁGINA SOLICITADA. SE ESTA PÁGINA ESTIVESSE SENDO EXIBIDA PREVIAMENTE O ESTADO DA INFORMAÇÃO (JÁ SALVO) SERÁ ADICIONADO AO REQUEST.
2. Apply Request Values: NESTA FASE O JSF PERCORRE A ÁRVORE DE COMPONETES E EXECUTA O MÉTODO decode() DE CADA UM DELES. ESTE MÉTODO EXTRAI OS VALORES DO REQUEST E "ARMAZENA" NOS COMPONTES.
3. Handle Request Events: ESTA FASE PERMITE MANIPULAR EVENTOS DE REQUEST GERADOS NA FASE ANTEIOR (EX.: MUDANÇAO VISUAIS EM UM COMPONENTE COMO CLICAR EM UM NÓ DE UMA ÁRVORE E PARA EXPANDIR UM GRAFO)
    1. O JSF CHAMA O MÉTOD proccessEvents() PARA CADA COMPONETE QUE TENHA UM OU MAIS EVENTOS ASSOCIADOS.
    2. ESTE MÓTODO RETORNA TRUE OR FALSE. NO PRIMEIRO CASO O CILCO AVANÇA PARA FASE DE Proccess Validation E NO SEGUNDO PARA Render Response.

JSF: FASES DO CLICO DE VIDA

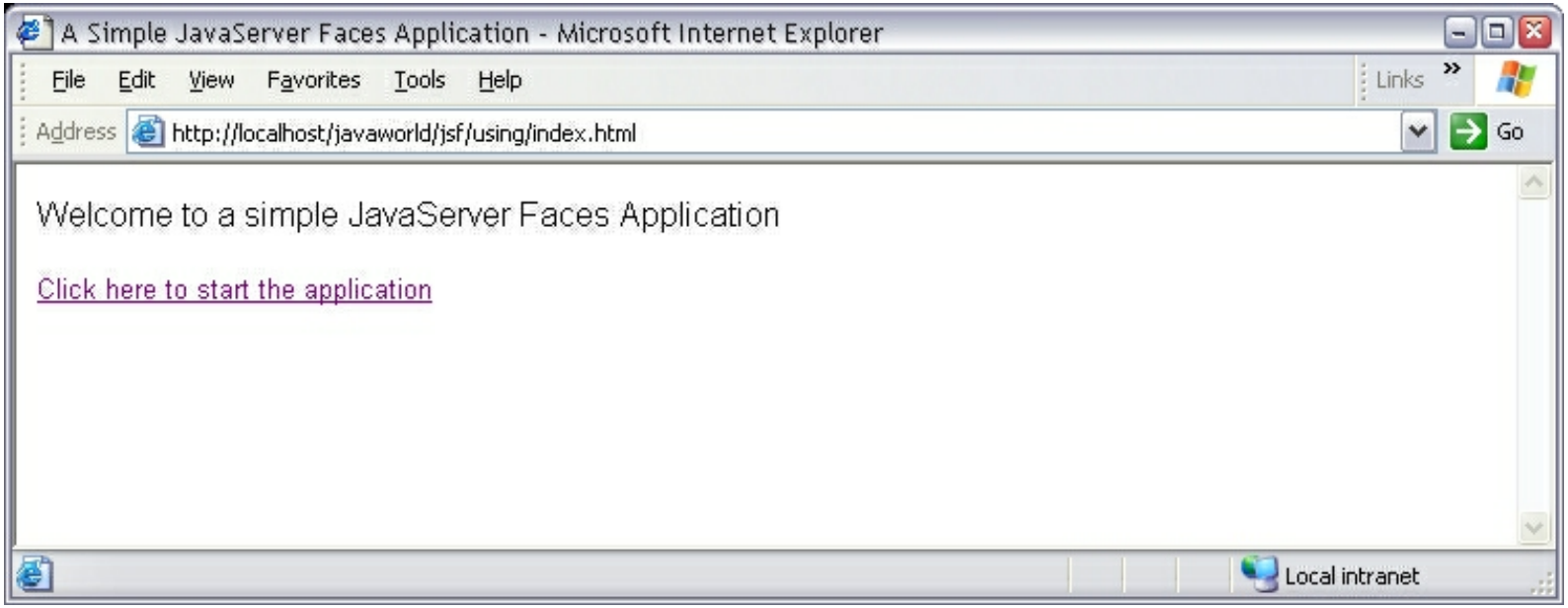
4 . Process Validations: NESTA FASE O JSF INVOCA O MÉTODO validate() PARA CADA OBJETO QUE PRECISE DE VALIDAÇÃO. ESTE MÉTODO RETORNA TRUE OU FALSE. NO PRIMEIRO CASO O CICLO PROCEDE NORMALMENTE E NO SEGUNDO ELE PULA PARA FASE Render Response.

5. Update Model: NESTA FASE, CADA CAMPO DE FORMULÁRIOS JSF PODE SER ASSOCIADO A UM ATRIBUTO DE UM COMPONETE EM JAVA (MODEL OBJECT). PARA ISSO O JSF CHAMA O MÉTODO updateModel() para cada COMPONETE. COMO OS VALORES DE UM REQUEST SÃO STRINGS E OS ATRIBUTOS DE UM MODEL SÃO TRATADOS COMO JAVA OBJECTS AS CONVERSÕES NECESSÁRIAS SÃO FEITAS NESSA FASE. CASO UM ERRO DE CONVESÃO OCORRA O CICLO PULA PARA A FASE Reder Response.

6. Invoke Applicattion: EM JSF, QUANDO SE SUBMETE UM FORM OU CLICA EM UM BOTÃO OU LINK A IMPLEMENTAÇÃO JSF CRIA UM EVENTO OU EVENTO DE COMANDO RESPECTIVAMENTE. ESTES EVENTOS SÃO TRATADOS NESTA FASE.

7. Render Response: POR FIM, NESTA FASE CRIA-SE UMA ÁRVORE DE COMPONETES DE RESPOSTA E A ENCAMINA AO CLIENTE. E OCLICLO COMEÇA NOVAMETE...

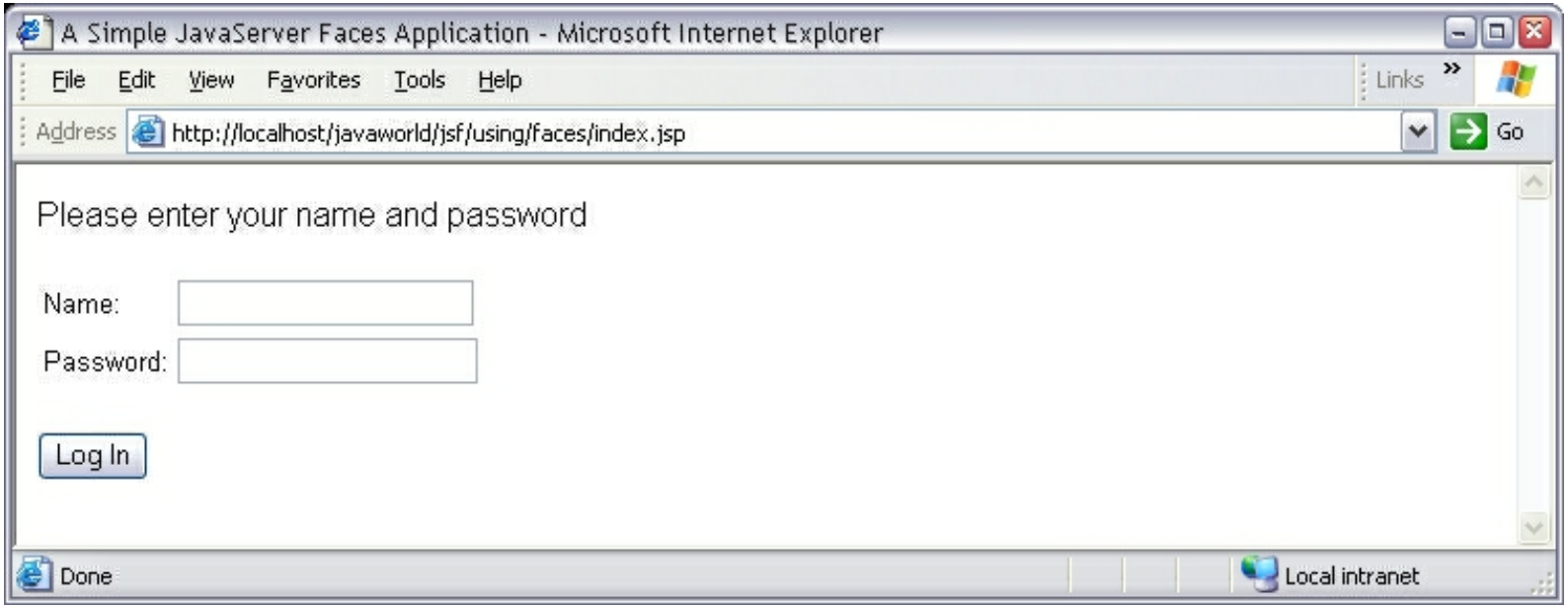
EXEMPLO SIMPLES



EXEMPLO SIMPLES

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>A Simple JavaServer Faces Application</title>
  </head>
  <body>
    <font size='4'>Welcome to a simple JavaServer Faces Application</font>
    <p>
    <a href='faces/index.jsp'>Click here to start the application</a>
  <body>
</html>
```

EXEMPLO SIMPLES



EXEMPLO SIMPLES

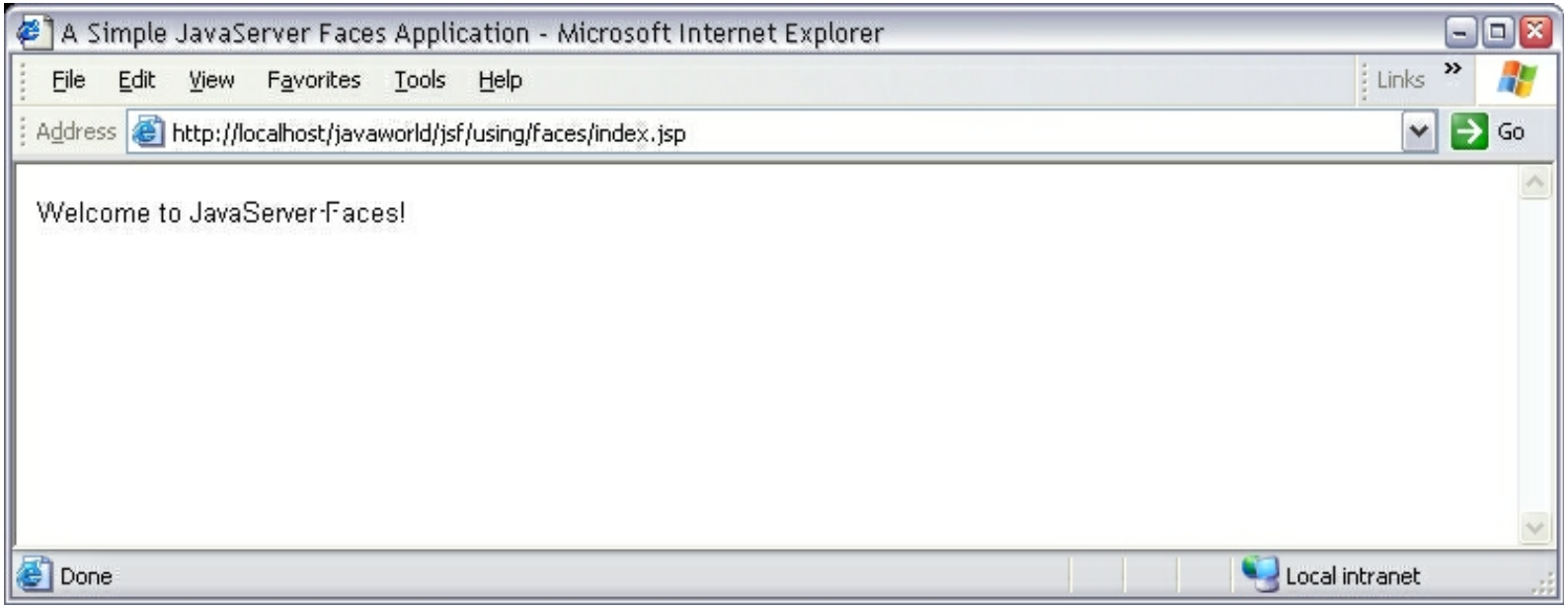
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>A Simple JavaServer Faces Application</title>
  </head>
  <body>
    <%@ taglib uri="http://java.sun.com/j2ee/html_basic/" prefix="faces" %>
    <font size="4">Please enter your name and password</font>

    <faces:usefaces>
      <faces:form id="simpleForm" formName="simpleForm">
        <table>
          <tr>
            <td>Name:</td>
```

EXEMPLO SIMPLES

```
<td><faces:textentry_input id="name"/></td>
</tr>
            <tr>
              <td>Password:</td>
              <td><faces:textentry_secret id="password"/></td>
            </tr>
          </table>
          <p><faces:command_button id="submit" commandName="Log In"/>
        </faces:form>
      </faces:usefaces>
    </body>
</html>
```

EXEMPLO SIMPLES



BEAN GERENCIADO

- Uma classe Java usada essencialmente para armazenar dados de usuário e que é controlada por uma página JSF.
- Para criar um bean gerenciado basta anotar com @ManagedBean (no JSF 2.0) e implementar a interface *Serializable*.
- Ex:

```
public class HelloBean implements Serializable {
        private String name;

        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
}
```

PÁGINAS JSF

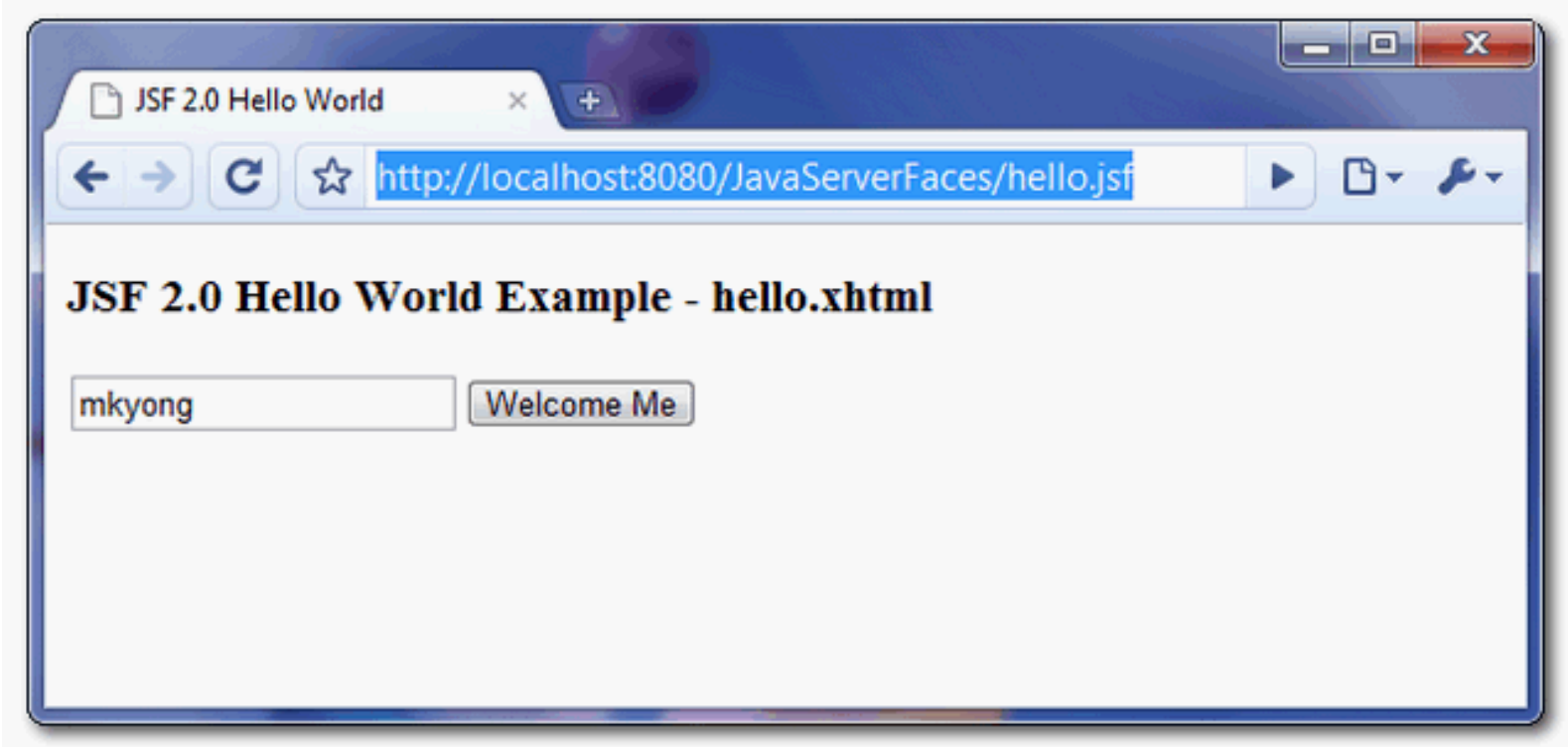
- Têm a extensão “.xhtml”
- A biblioteca de tags é encontrada através dos namespaces:
  - <html xmlns="http://www.w3.org/1999/xhtml"
  - xmlns:f="http://java.sun.com/jsf/core"
  - xmlns:h="http://java.sun.com/jsf/html">
- O exemplo a seguir mostra o uso do HelloBean (visto) a partir de uma página JSF.

```
<h:head>
    <title>JSF 2.0 Hello World</title>
  </h:head>
  <h:body>
      <h3>JSF 2.0 Hello World Example - hello.xhtml</h3>
      <h:form>
        <h:inputText value="#{helloBean.name}"></h:inputText>
        <h:commandButton value="Welcome Me" action="welcome"></h:commandButton>
      </h:form>
  </h:body>
</html>
```

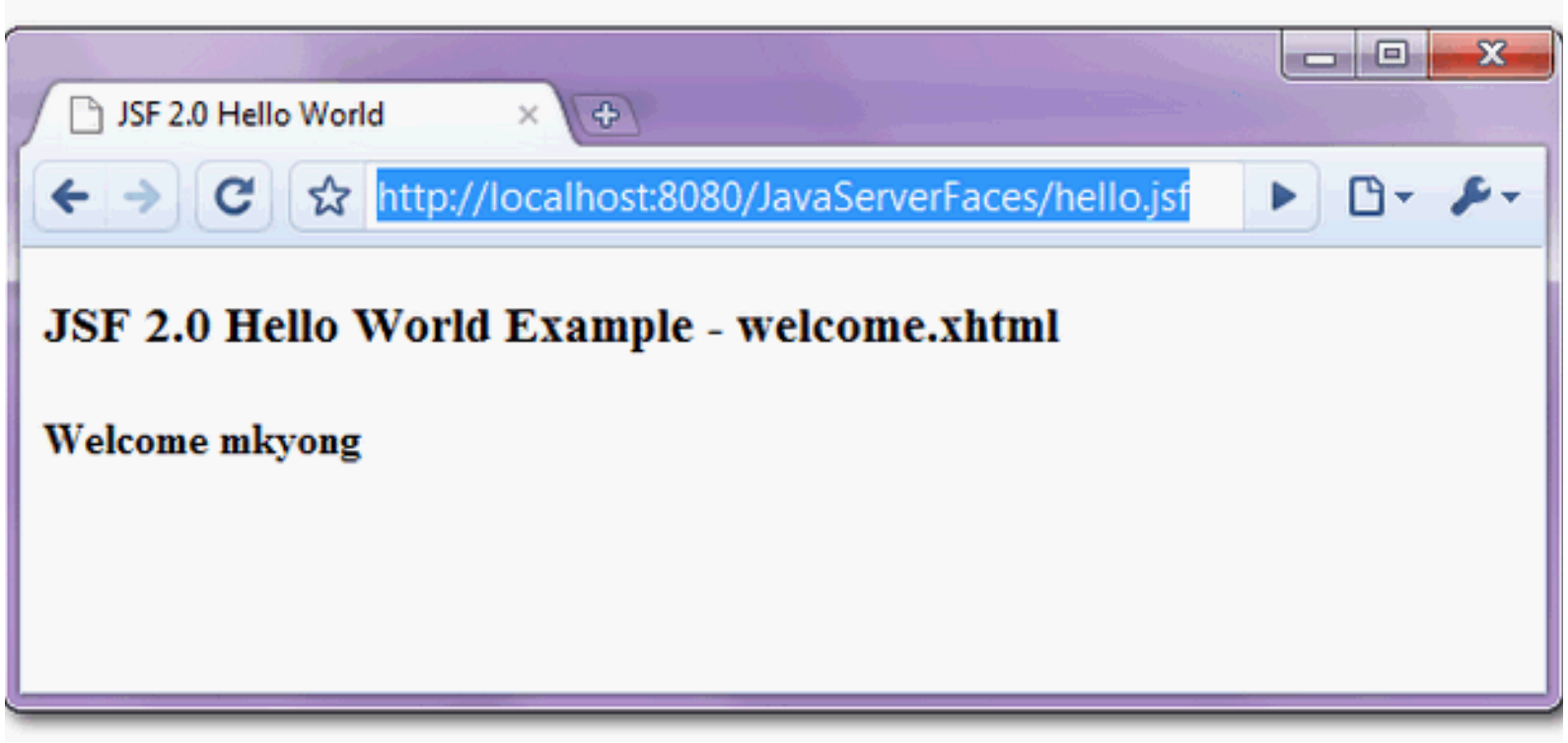
A NEVEGAÇÃO

- <h:commandButton value="Welcome Me" action="welcome"></h:commandButton>
- O atributo action permite que uma página JSF (welcome.xhtml) seja chamada:
  - <?xml version="1.0" encoding="UTF-8"?>
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  - "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  - <html xmlns="http://www.w3.org/1999/xhtml"
  - xmlns:h="http://java.sun.com/jsf/html">
  - <h:head>
  - <title>JSF 2.0 Hello World</title>
  - </h:head>
  - <h:body bgcolor="white">
  - <h3>JSF 2.0 Hello World Example - welcome.xhtml</h3>
  - <h4>Welcome #{helloBean.name}</h4>
  - </h:body>
  - </html>

RODANDO O EXEMPLO



RODANDO O EXEMPLO



CODIFICANDO AJAX EM PÁGINAS JSF

- Em JSF é possível codificar AJAX como através apenas de código de marcação ex:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:body>
         <h3>JSF 2.0 + Ajax Hello World Example</h3>
         <h:form>
           <h:inputText id="name" value="#{helloBean.name}"></h:inputText>
           <h:commandButton value="Welcome Me">
                        <f:ajax execute="name" render="output" />
           </h:commandButton>
           <h2><h:outputText id="output" value="#{helloBean.sayWelcome}" /></h2>
         </h:form>
   </h:body>
</html>
```

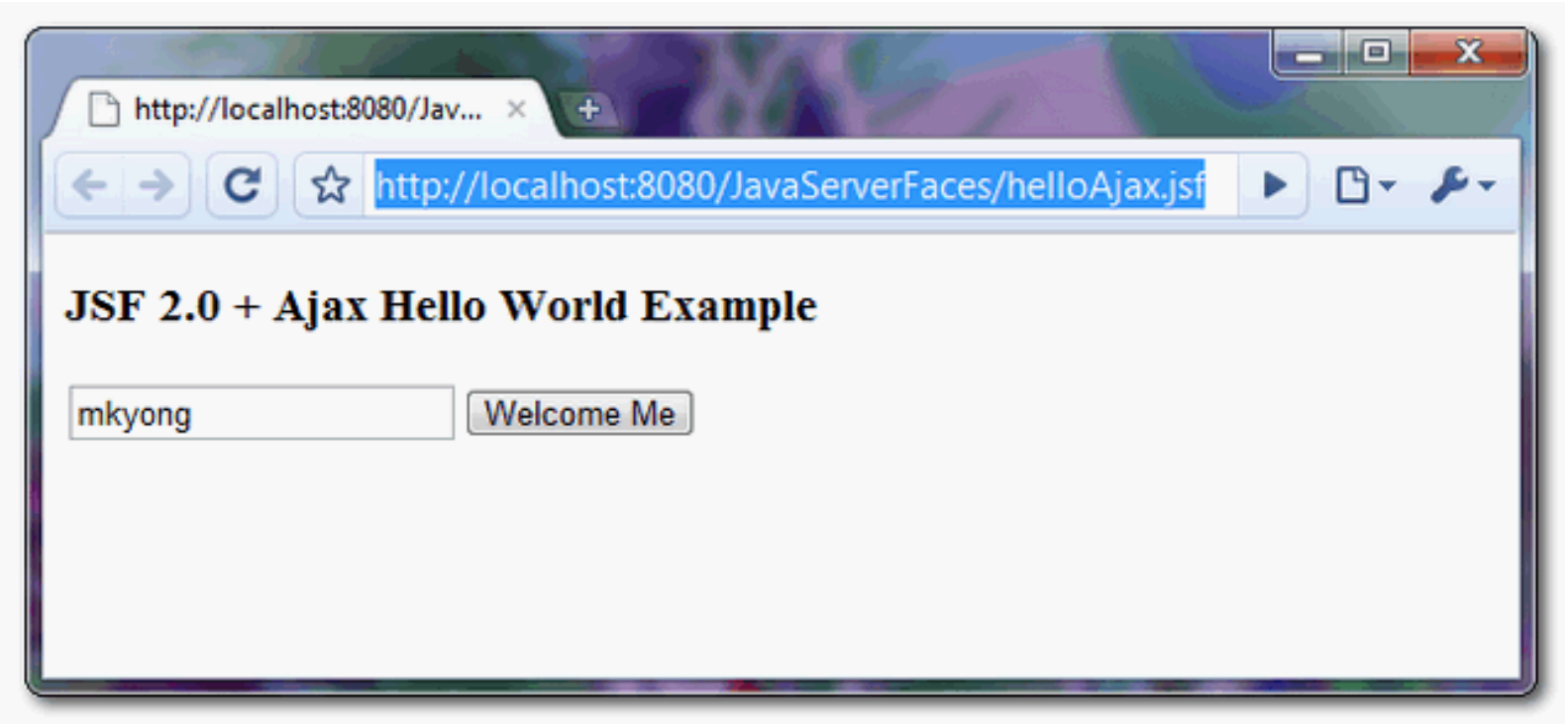
CODIFICANDO AJAX EM PÁGINAS JSF

- No exemplo anterior, quando o botão é acionado, um requisição de execução para o componente helloBean (método getName()).
- Isso é feito sem que todo o formulário seja submetido e atualizado.
- Na TAG <f:ajax>:
  - O atributo **execute** define que o objeto com id="name" (inputtext) deve ser executado no servidor (lado) e o resultado dessa execução deve ser "desenhado" (atributo render) no objeto de id="output" (outputtext).
  - Se mais de um método de um ou mais objetos precisar ser executado, basta separar o valor de id (Ex.: id="id1 id2 id3....")

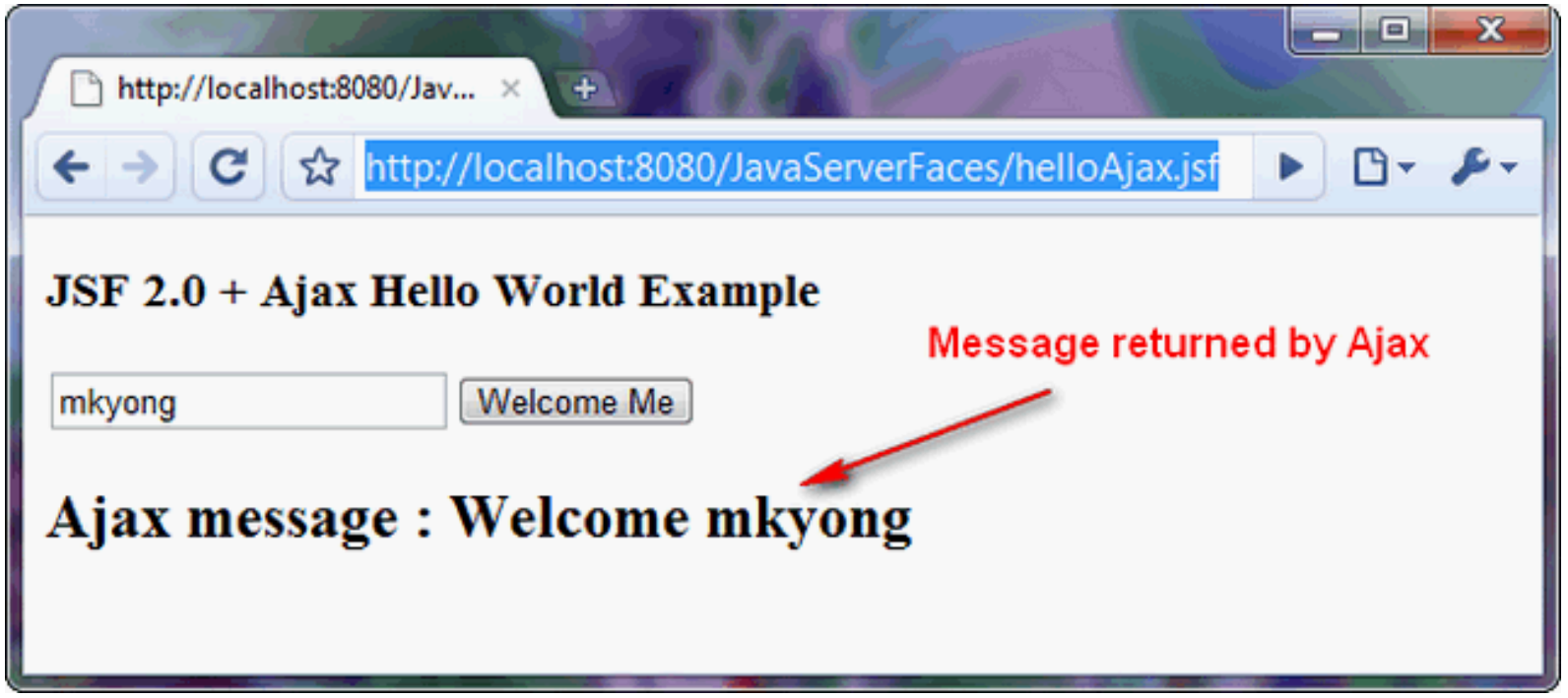
RODANDO JSF/AJAX

```
public class HelloBean implements Serializable {
        private String name;
        public String getName() {
          return name;
        }
        public void setName(String name) {
          this.name = name;
        }
        public String getSayWelcome(){
          if("".equals(name) || name ==null){
                    return "";
          }else{
                    return "Ajax message : Welcome " + name;
          }
        }
}
```

RODANDO JSF/AJAX



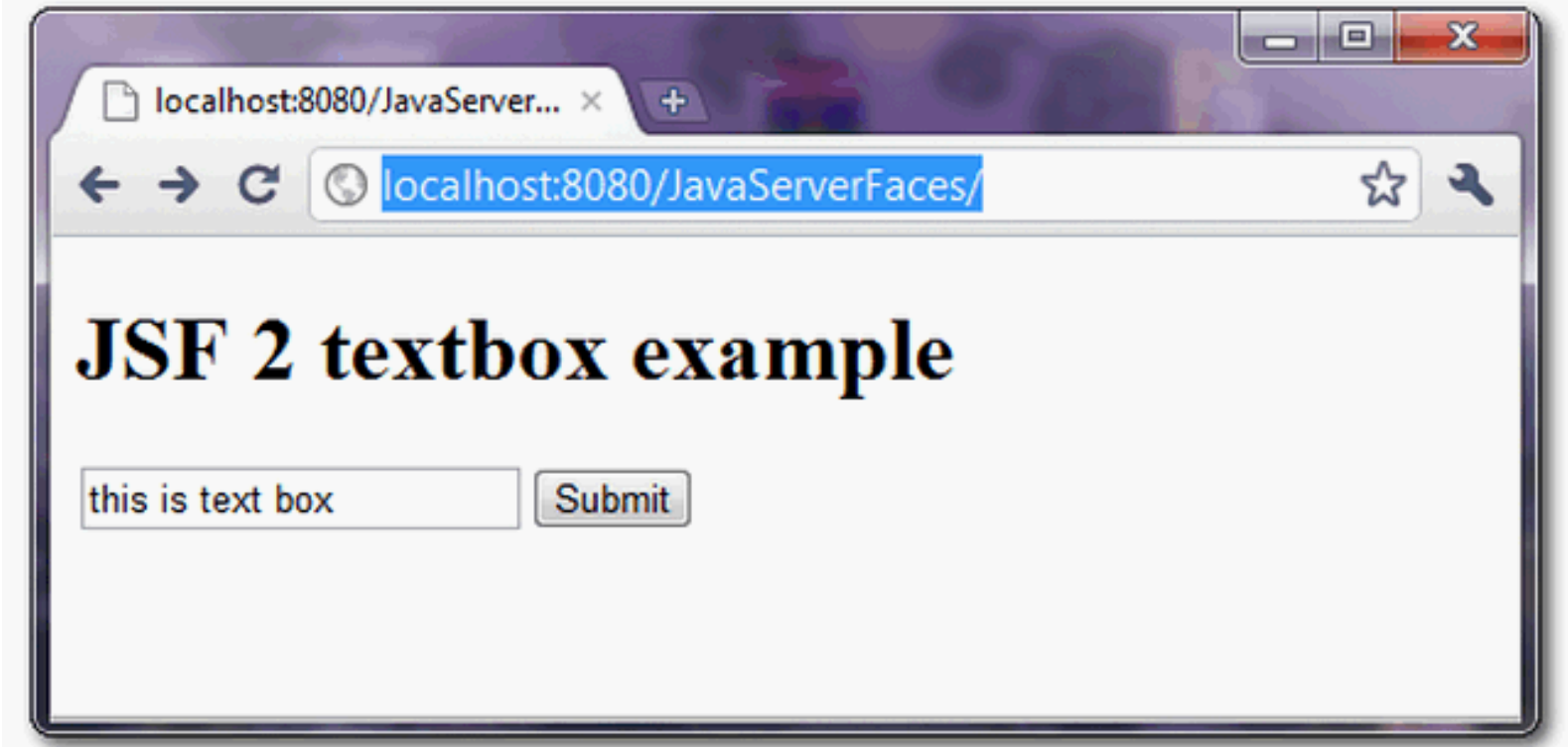
RODANDO JSF/AJAX



EXEMPLO CAIXA DE TEXTO (INPUTTEXT)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:body>
      <h1>JSF 2 textbox example</h1>
       <h:form>
              <h:inputText value="#{userBean.userName}" />
              <h:commandButton value="Submit" action="user" />
       </h:form>
   </h:body>
</html>
```

DEMO


localhost:8080/JavaServer...
localhost:8080/JavaServerFaces/
JSF 2 textbox example
this is text box
Submit

EXEMPLO RÓTULO (OUTPUTTEXT)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:body>
      <h1>JSF 2 textbox example</h1>
      Submitted value : <h:outputText value="#{userBean.userName}" />
    </h:body>
</html>
```

DEMO


localhost:8080/JavaServer...
localhost:8080/JavaServerFaces/faces/demo.xhtml
JSF 2 textbox example
Submitted value : this is text box

EXEMPLO PASSWORD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:body>
      <h1>JSF 2 password example</h1>
       <h:form>
              Password : <h:inputSecret value="#{userBean.password}" />
              <h:commandButton value="Submit" action="user" />
       </h:form>
   </h:body>
</html>
```

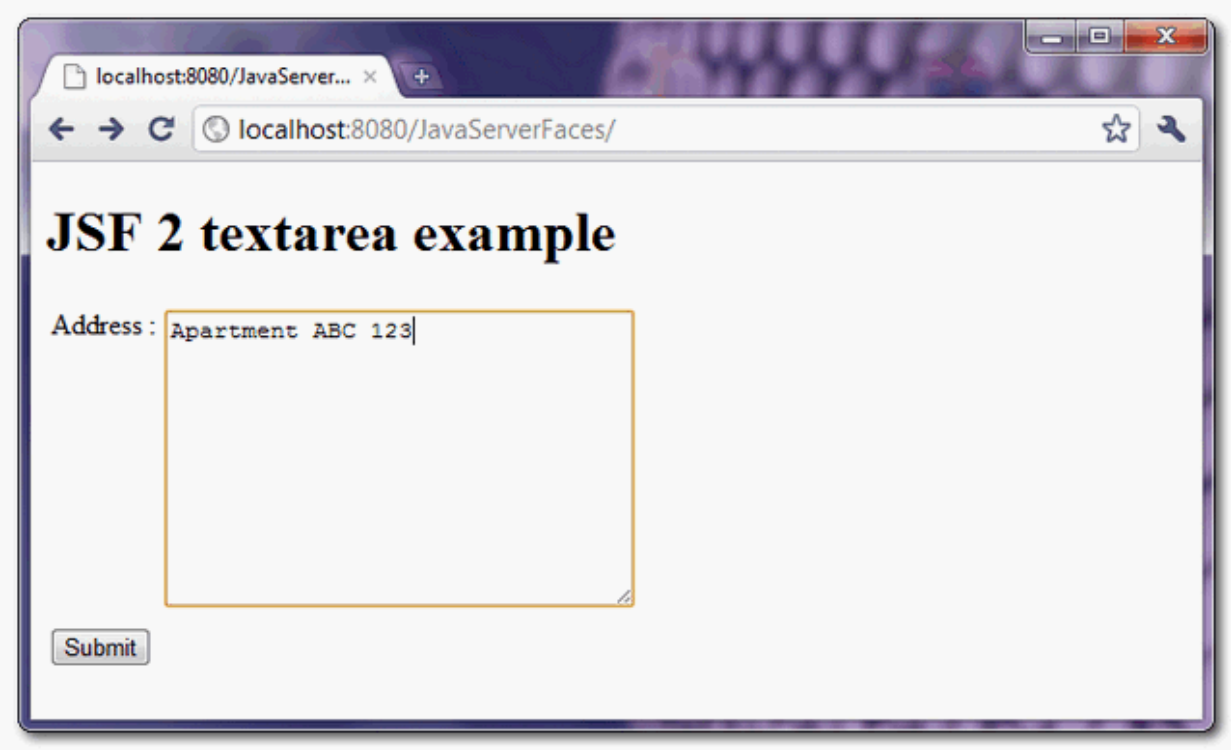
DEMO


localhost:8080/JavaServer...
localhost:8080/JavaServerFaces/
JSF 2 password example
Password :
Submit

EXEMPLO TEXTAREA

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:body>
        <h1>JSF 2 textarea example</h1>
        <h:form><table><tr>
                     <td valign="top">Address :</td>
                     <td><h:inputTextarea value="#{user.address}" cols="30" rows="10" /></td></tr>
                   </table>
                   <h:commandButton value="Submit" action="user" />
         </h:form>
   </h:body>
</html>
```

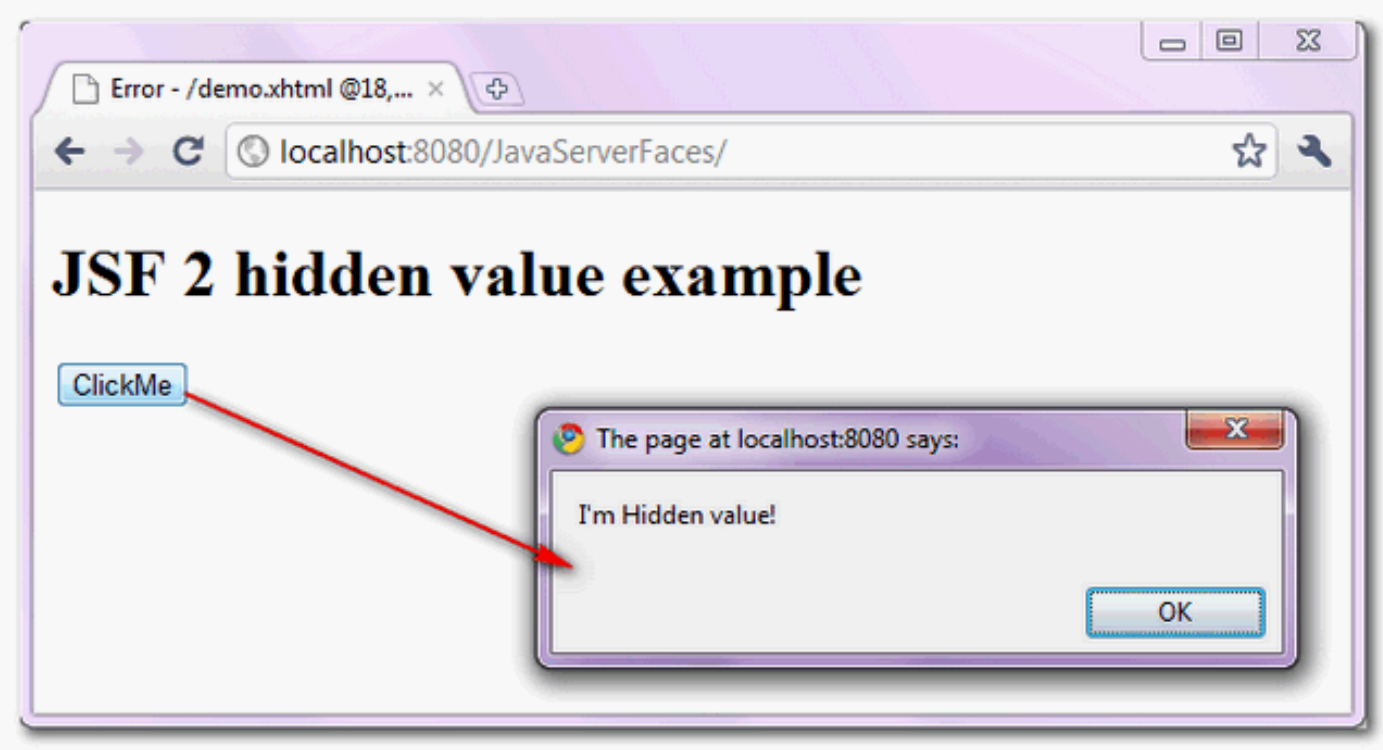
DEMO


localhost:8080/JavaServer...
localhost:8080/JavaServerFaces/
JSF 2 textarea example
Address :
Apartment ABC 123
Submit

EXEMPLO HIDDEN (INVISÍVEL)

```
        <h:head>
          <script type="text/javascript">
           function printHiddenValue(){
             alert(document.getElementById('myform:hiddenId').value);
           }
          </script>
        </h:head>
    <h:body>
        <h1>JSF 2 hidden value example</h1>
          <h:form id="myform">
                    <h:inputHidden value="#{user.answer}" id="hiddenId" />
                    <h:commandButton type="button" value="ClickMe" onclick="printHiddenValue()" />
          </h:form>
    </h:body>
</html>
```

DEMO


Error - /demo.xhtml @18,...
localhost:8080/JavaServerFaces/
JSF 2 hidden value example
ClickMe
The page at localhost:8080 says:
I'm Hidden value!
OK

EXEMPLO CHECKBOXES (SINGLE)

```
<h:selectBooleanCheckbox value="#{user.rememberMe}" />
```

- Retorna um boolean (true ou false)

EXEMPLO CHECKBOXES (MULTIPLO)

```
<h:selectManyCheckbox value="#{user.favNumber1}">
                        <f:selectItem itemValue="1" itemLabel="Number1 - 1" />
                        <f:selectItem itemValue="2" itemLabel="Number1 - 2" />
                        <f:selectItem itemValue="3" itemLabel="Number1 - 3" />
                        <f:selectItem itemValue="4" itemLabel="Number1 - 4" />
                        <f:selectItem itemValue="5" itemLabel="Number1 - 5" />
                </h:selectManyCheckbox>
```

- user.faNumber1 é um array de String com tamanho 5 que é preenchido em cada tag <f:selectItem

EXEMPLO CHECKBOXES (MULTIPLO)

```
<h:selectManyCheckbox value="#{user.favNumber2}">
                        <f:selectItems value="#{user.favNumber2Value}" />
            </h:selectManyCheckbox>
```

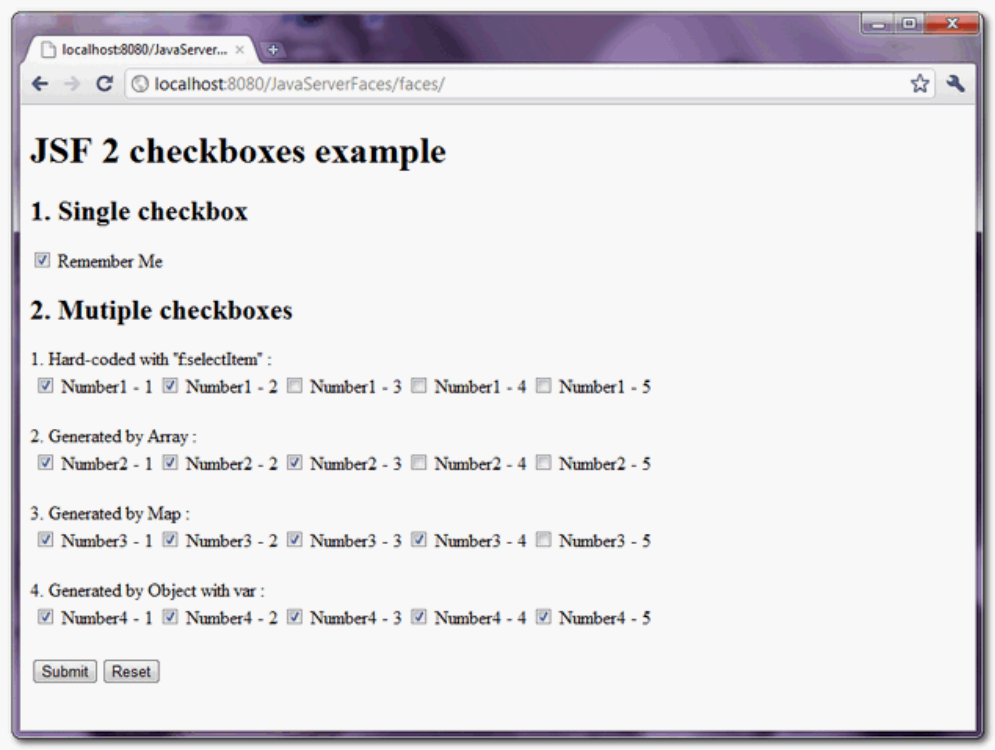
- user.faNumber2 é um array de String com tamanho 5 que é preenchido na tag <f:selectItems com os valores do array user.favNumber2Value.
- Ele (user.faNumber2) pode ser também um Map

EXEMPLO CHECKBOXES (MULTIPLO)

```
<h:selectManyCheckbox value="#{user.favNumber4}">
    <f:selectItems value="#{user.favNumber4Value}" var="n"
    itemLabel="#{n.numberLabel}" itemValue="#{n.numberValue}" />
    </h:selectManyCheckbox>
```

user.faNumber4 é um array de Objetos com propriedades getNuberValue e getNumberLabel.

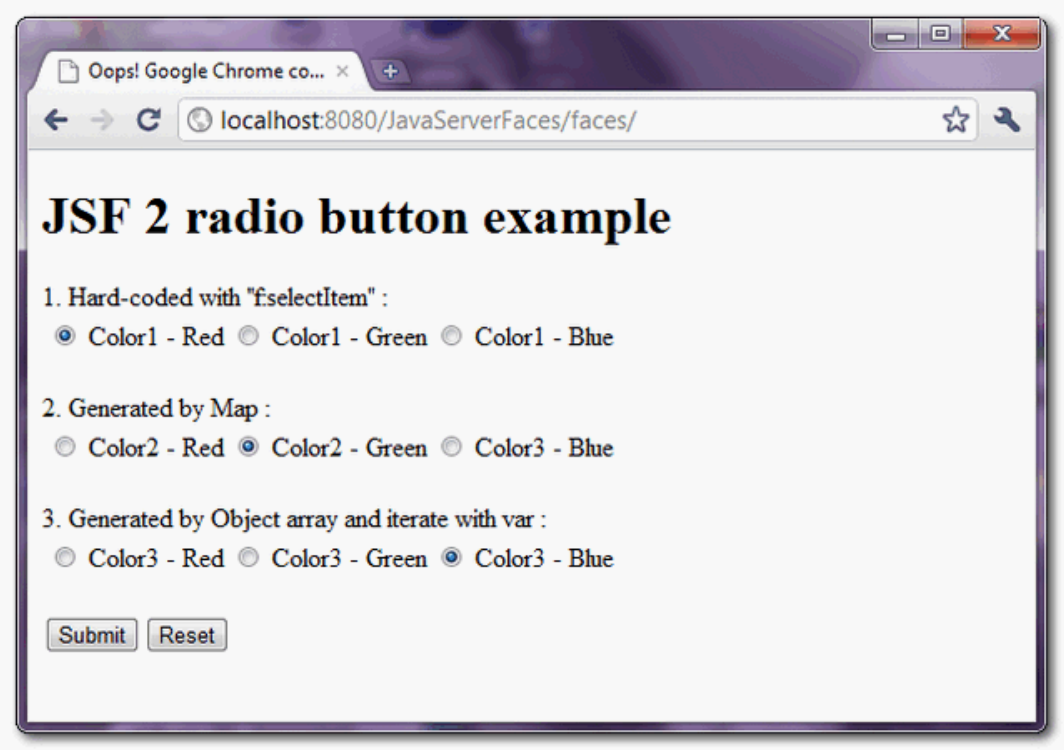
DEMO


localhost:8080/JavaServer...
localhost:8080/JavaServerFaces/faces/
JSF 2 checkboxes example
1. Single checkbox
Remember Me
2. Mutiple checkboxes
1. Hard-coded with "f:selectItem" :
Number1 - 1 Number1 - 2 Number1 - 3 Number1 - 4 Number1 - 5
2. Generated by Array :
Number2 - 1 Number2 - 2 Number2 - 3 Number2 - 4 Number2 - 5
3. Generated by Map :
Number3 - 1 Number3 - 2 Number3 - 3 Number3 - 4 Number3 - 5
4. Generated by Object with var :
Number4 - 1 Number4 - 2 Number4 - 3 Number4 - 4 Number4 - 5
Submit
Reset

EXEMPLO RADIO BUTTONS

- O mesmo que vale para checkbox vale para radio buutons
- A tag é :
- <h:selectOneRadio value="#{user.favColor1}">
- <f:selectItem itemValue="Red" itemLabel="Color1 - Red" />
- <f:selectItem itemValue="Green" itemLabel="Color1 - Green" />
- <f:selectItem itemValue="Blue" itemLabel="Color1 - Blue" />
- </h:selectOneRadio>

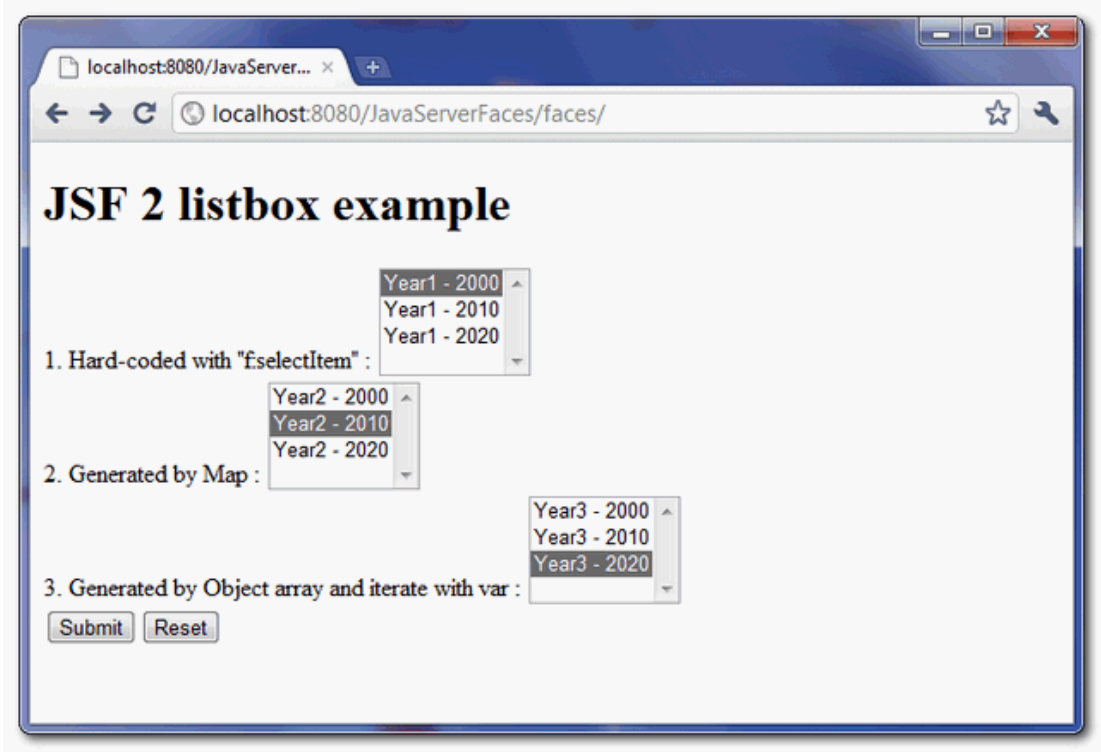
DEMO


Oops! Google Chrome co...
localhost:8080/JavaServerFaces/faces/
JSF 2 radio button example
1. Hard-coded with "f:selectItem" :
Color1 - Red Color1 - Green Color1 - Blue
2. Generated by Map :
Color2 - Red Color2 - Green Color3 - Blue
3. Generated by Object array and iterate with var :
Color3 - Red Color3 - Green Color3 - Blue
Submit
Reset

EXEMPLO LIST BOX

```
<h:selectOneListbox value="#{user.favYear1}">
                        <f:selectItem itemValue="2000" itemLabel="Year1 - 2000" />
                        <f:selectItem itemValue="2010" itemLabel="Year1 - 2010" />
                        <f:selectItem itemValue="2020" itemLabel="Year1 - 2020" />
            </h:selectOneListbox>
            <br />
            <h:selectOneListbox value="#{user.favYear2}">
                        <f:selectItems value="#{user.favYear2Value}" />
            </h:selectOneListbox>
          <br />
            <h:selectOneListbox value="#{user.favYear3}">
                        <f:selectItems value="#{user.favYear3Value}" var="y"
                        itemLabel="#{y.yearLabel}" itemValue="#{y.yearValue}" />
            </h:selectOneListbox>
```

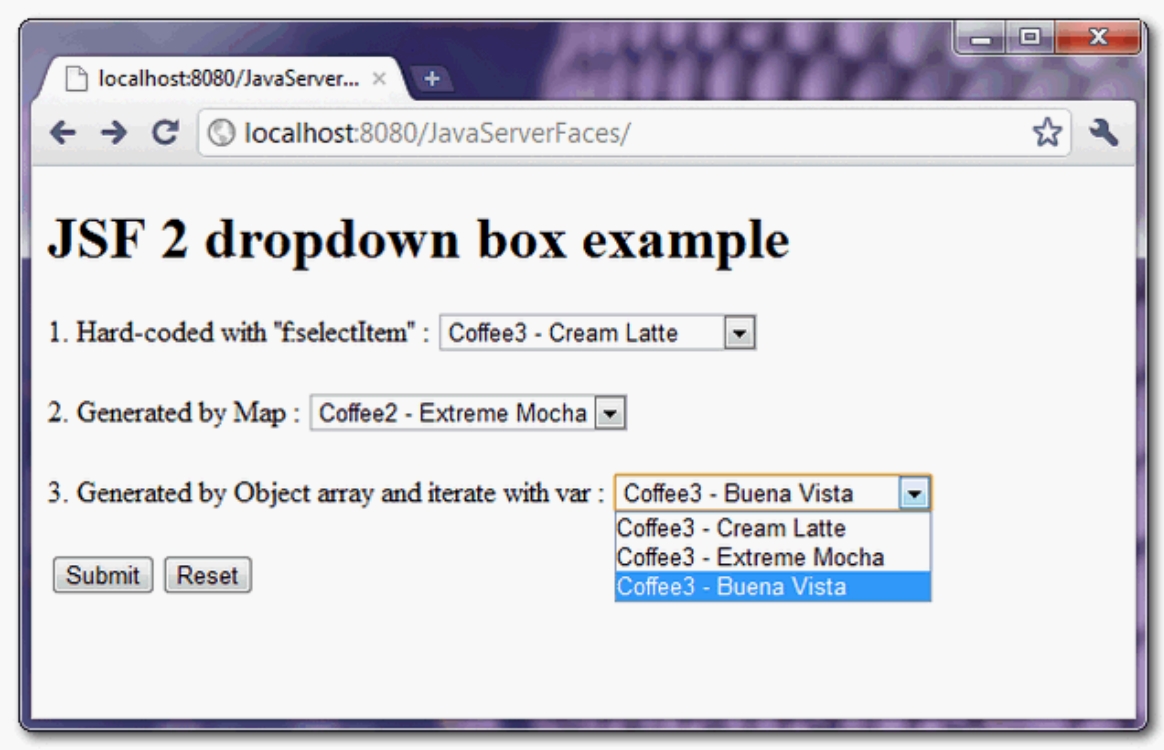
DEMO


localhost:8080/JavaServer...
localhost:8080/JavaServerFaces/faces/
JSF 2 listbox example
1. Hard-coded with "f:selectItem" :
Year1 - 2000
Year1 - 2010
Year1 - 2020
2. Generated by Map :
Year2 - 2000
Year2 - 2010
Year2 - 2020
3. Generated by Object array and iterate with var :
Year3 - 2000
Year3 - 2010
Year3 - 2020
Submit
Reset

EXEMPLO DROPDOWNLIST

```
<h:selectOneMenu value="#{user.favCoffee3}">
                        <f:selectItems value="#{user.favCoffee3Value}" var="c"
                        itemLabel="#{c.coffeeLabel}" itemValue="#{c.coffeeValue}" />
            </h:selectOneMenu>
```

DEMO


localhost:8080/JavaServer...
localhost:8080/JavaServerFaces/
JSF 2 dropdown box example
1. Hard-coded with "f:selectItem" : Coffee3 - Cream Latte
2. Generated by Map : Coffee2 - Extreme Mocha
3. Generated by Object array and iterate with var : Coffee3 - Buena Vista
Coffee3 - Cream Latte
Coffee3 - Extreme Mocha
Coffee3 - Buena Vista
Submit
Reset