Globalcode®
The Developers Company

Open-source Education

JavaServer Faces 1.1

Iniciativa Globalcode

Introdução

Motivação

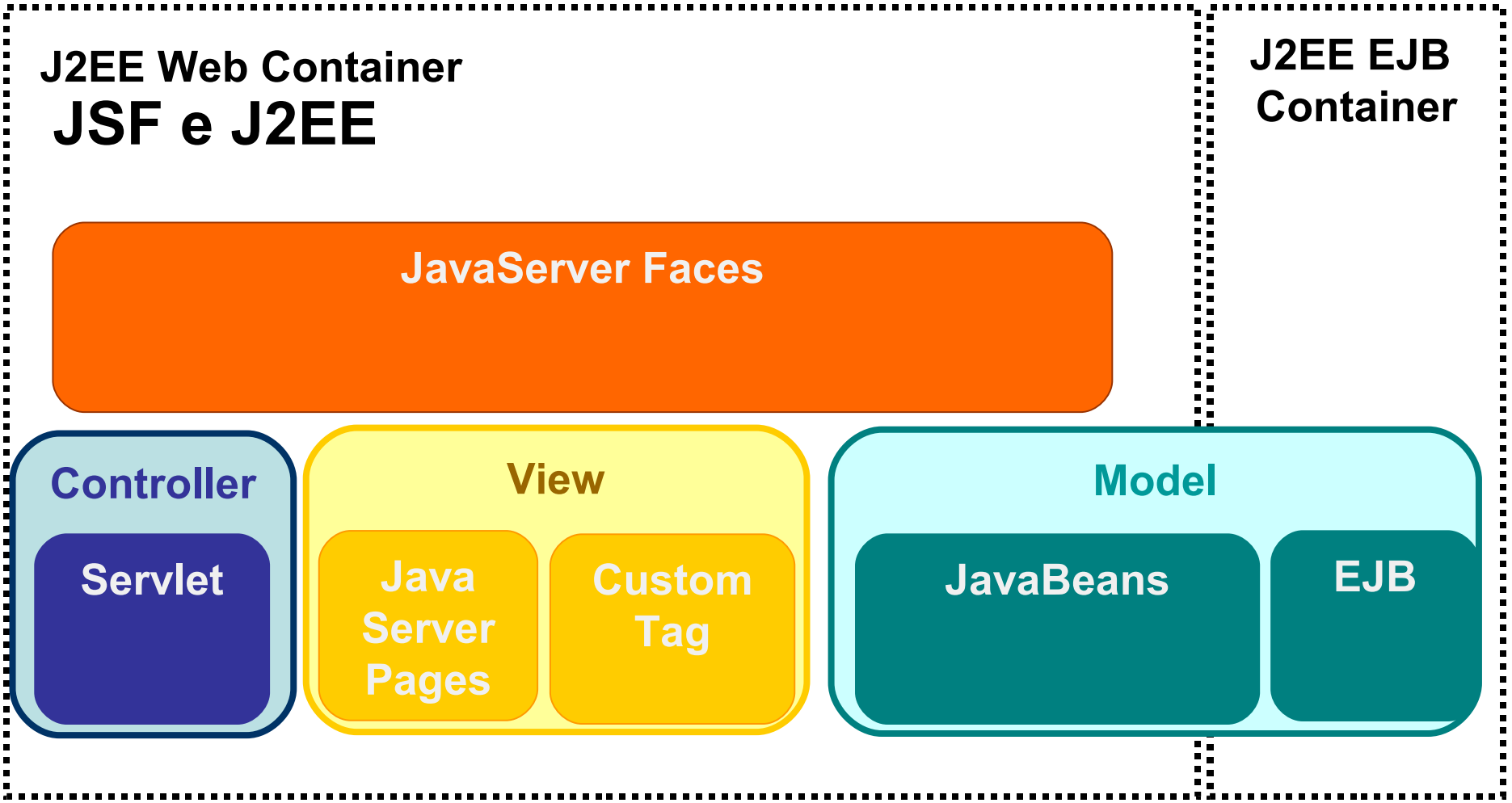
- MVC tornou-se um padrão de mercado;
- As interfaces gráficas exigidas são muito complexas para serem desenvolvidas somente com HTML exigindo muito JavaScript;
- Muitos componentes de UI sendo desenvolvidos com Custom Tags ou JavaScript sem padronização;
- Baixa produtividade no desenvolvimento de aplicações Web;

Introdução

O que é JavaServer Faces (JSF)?

- Paradigma de programação visual de User-interfaces aplicado à web
- É um framework que permite a criação de aplicações Web com semântica de Swing implementando MVC;
- “Toolability = Ferramentabilidade” ;
- É Uma especificação J2EE – JSR 127;

Introdução


J2EE Web Container
JSF e J2EE
J2EE EJB Container
JavaServer Faces
Controller
Servlet
View
Java Server Pages
Custom Tag
Model
JavaBeans
EJB

Introdução

Quais os próximos objetivos?

- No futuro todo o Web Container compatível com a especificação Java EE 5.0 terá que implementar JSF;
- Criação de componentes de UI compatíveis com JSF por terceiros;
- Suporte a JSF na maioria dos IDEs
- Aumento da produtividade de aplicação J2EE

Introdução

OC4J - Oracle

Introdução

Sun Application Server

J2EE Web Container

J2EE EJB Container

Implementação Sun de J2EE 1.5

- Servlets
- JSP
- Custom Tags
- JSF

EJB

Arquitetura JSF

- Arquitetura Client-Server baseada em HTTP;
- Dificuldade em prover o mesmo dinamismo de uma aplicação desktop;
- Todo o processamento Java acontece no servidor;

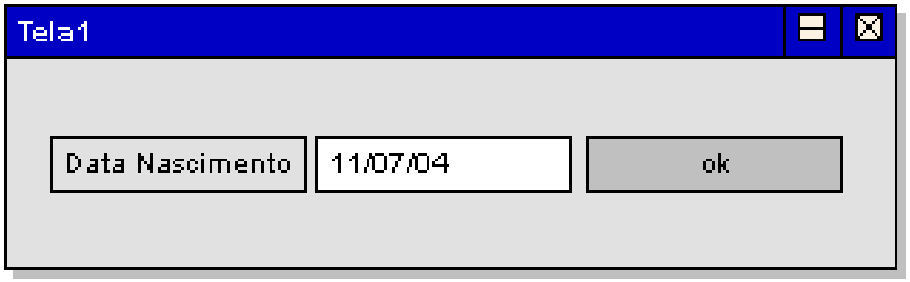
Arquitetura Swing

GUI e Listener são processados pela mesma máquina

Listener

- **Validação**
- **Conversão de dados**
- **Integração com a camada Model**
- **Lógica de negócios, etc…**

Java
Chamada a métodos

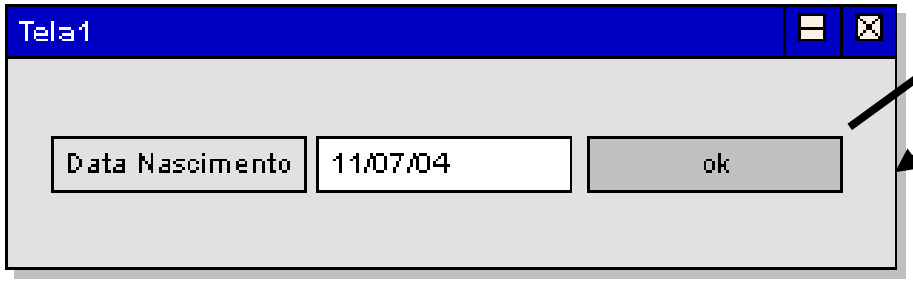


Arquitetura JSF

J2EE Web Container

Java Servlet

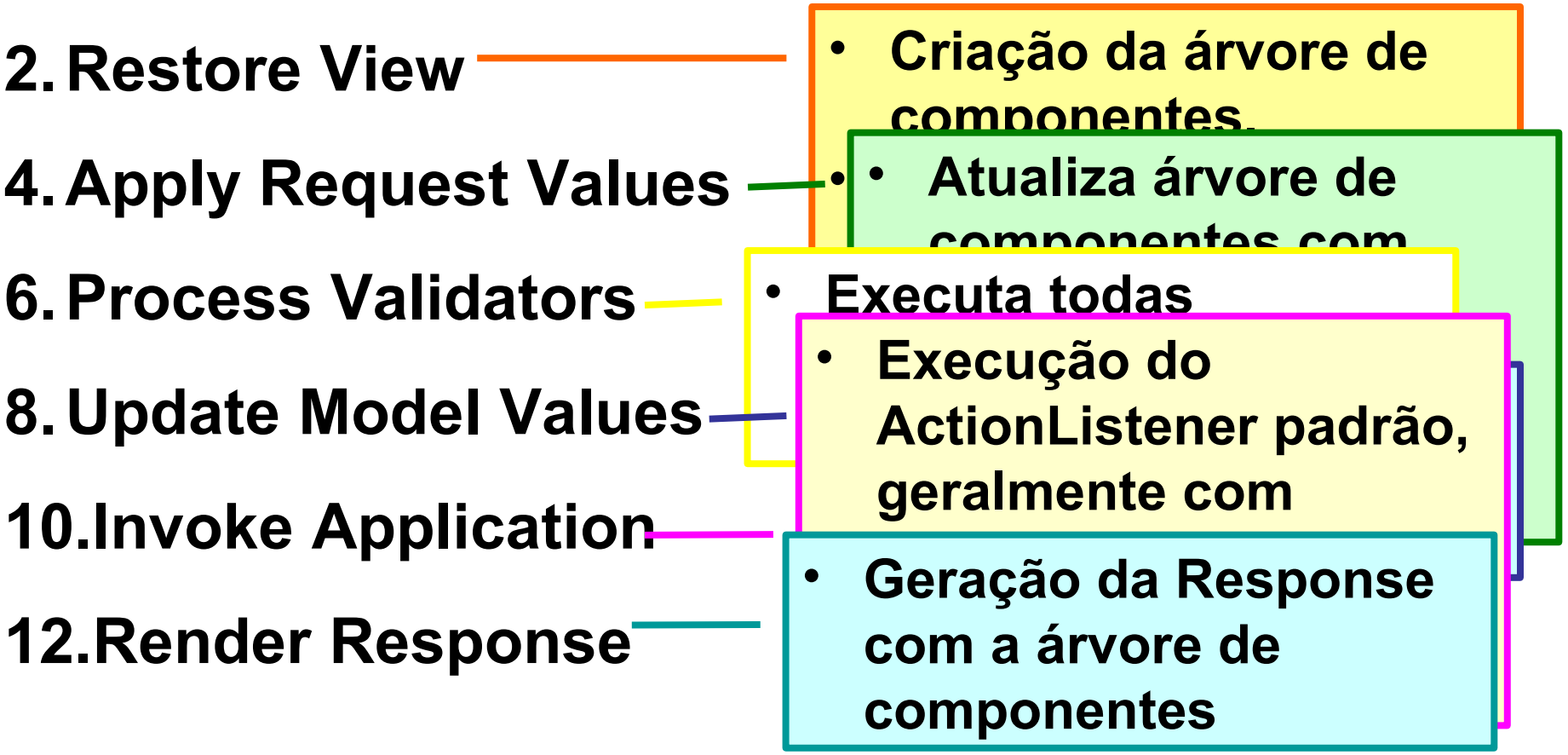
- Validação
- Conversão de dados
- Integração com a camada Model
- Lógica de negócios, etc…



HTTP
REQUEST / RESPONSE

Cliente – Browser: HTML + JavaScript

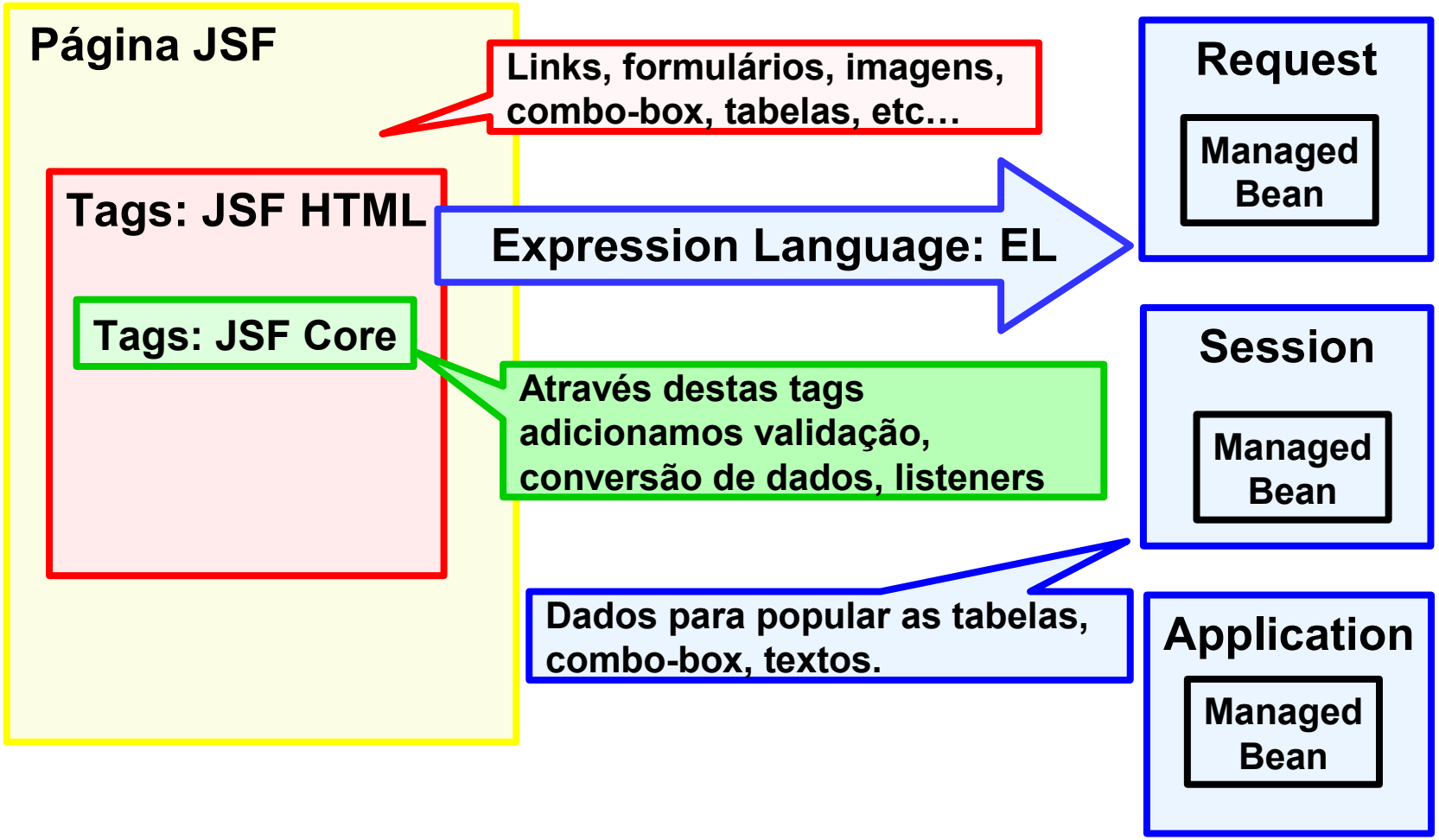
Ciclo de vida do JSF



Criação de páginas JSF

- **Páginas JSF geralmente utilizam duas bibliotecas de Tags:**
  - **JSF Html: renderização de componentes HTML**
  - **JSF Core: integração dos componentes de UI com validadores, conversores**

Criação de páginas JSF



Criação de páginas JSF

A navegação em páginas HTML pode acontecer das seguintes formas:

- Um link : <A HREF="catalogo.jsp"> texto do link </A>
- Um formulário <FORM action="catalogo.jsp>

Podemos utilizar os seguintes componentes JSF para gerar hyper links:

- OutputLink
- CommandButton
- CommandLink

Criação de páginas JSF

1. **OutputLink:** Utilizamos OutputLink quando nenhuma ação deve ser realizada quando o link for clicado, ou seja, somente o redirecionamento nos interessa.

3. **CommandLink e CommandButton:** são utilizados para gerar links e submeter formulários, e podem ser configurados para executar uma determinada ação antes de enviar a Request para a página configurada.

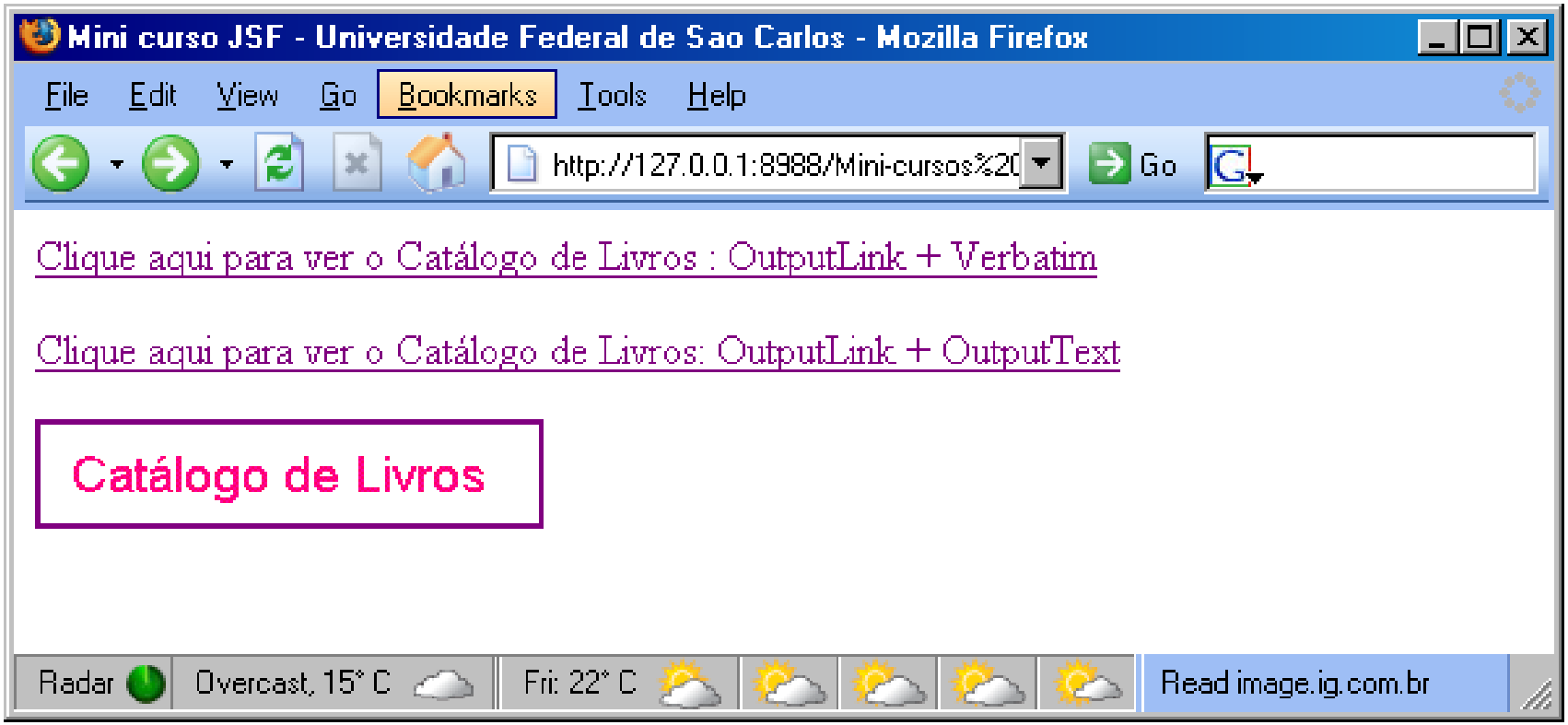
5. **CommandLink:** utiliza JavaScript para submissão da Request.

Criação de páginas JSF

Demo:

Criação de uma página index.jsp, com um link para a página catalogoLivros.jsp

Criação de páginas JSF



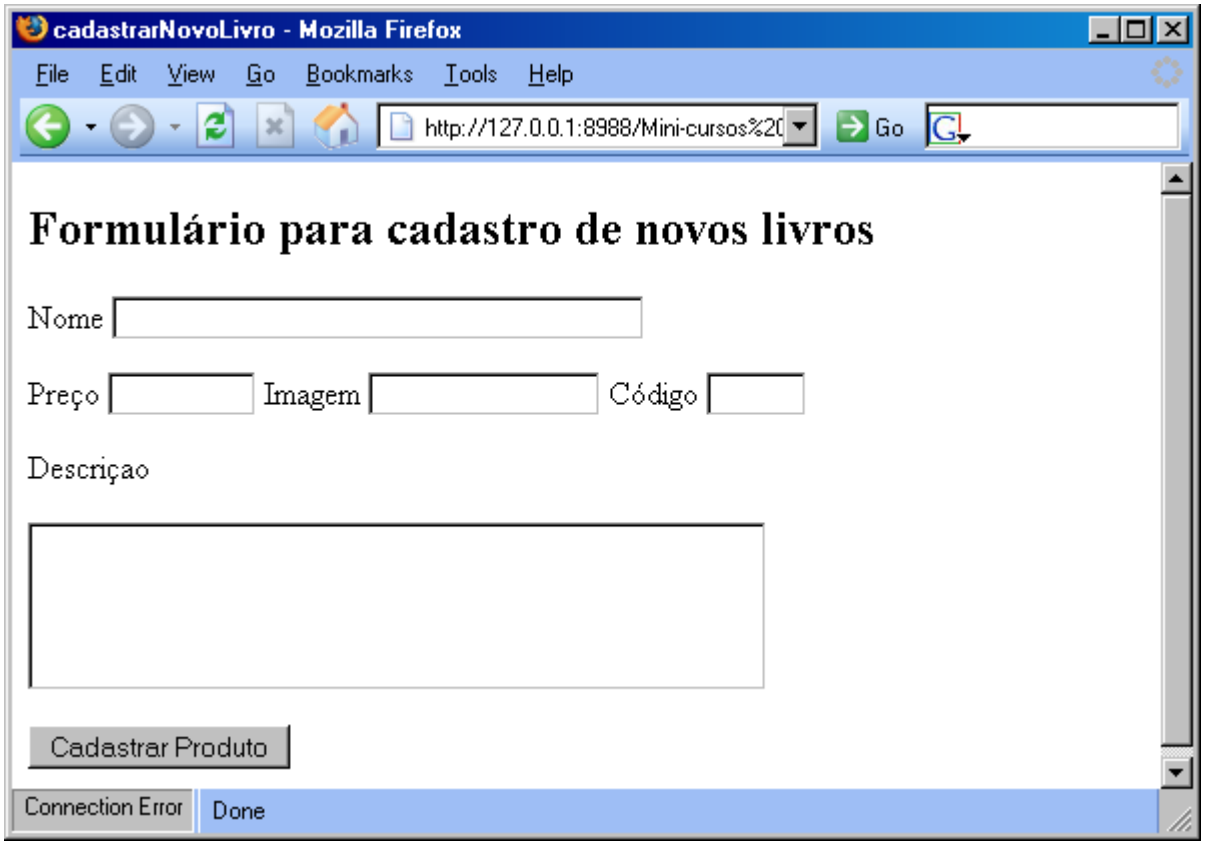
Veja o código-fonte desta página

Criação de páginas JSF

Demo:

Criação de uma página com formulário para cadastro de um novo Livro, utilizando CommandButton ou CommandLink para submeter o formulário.

Criação de páginas JSF



Veja o código-fonte desta página

Managed Beans

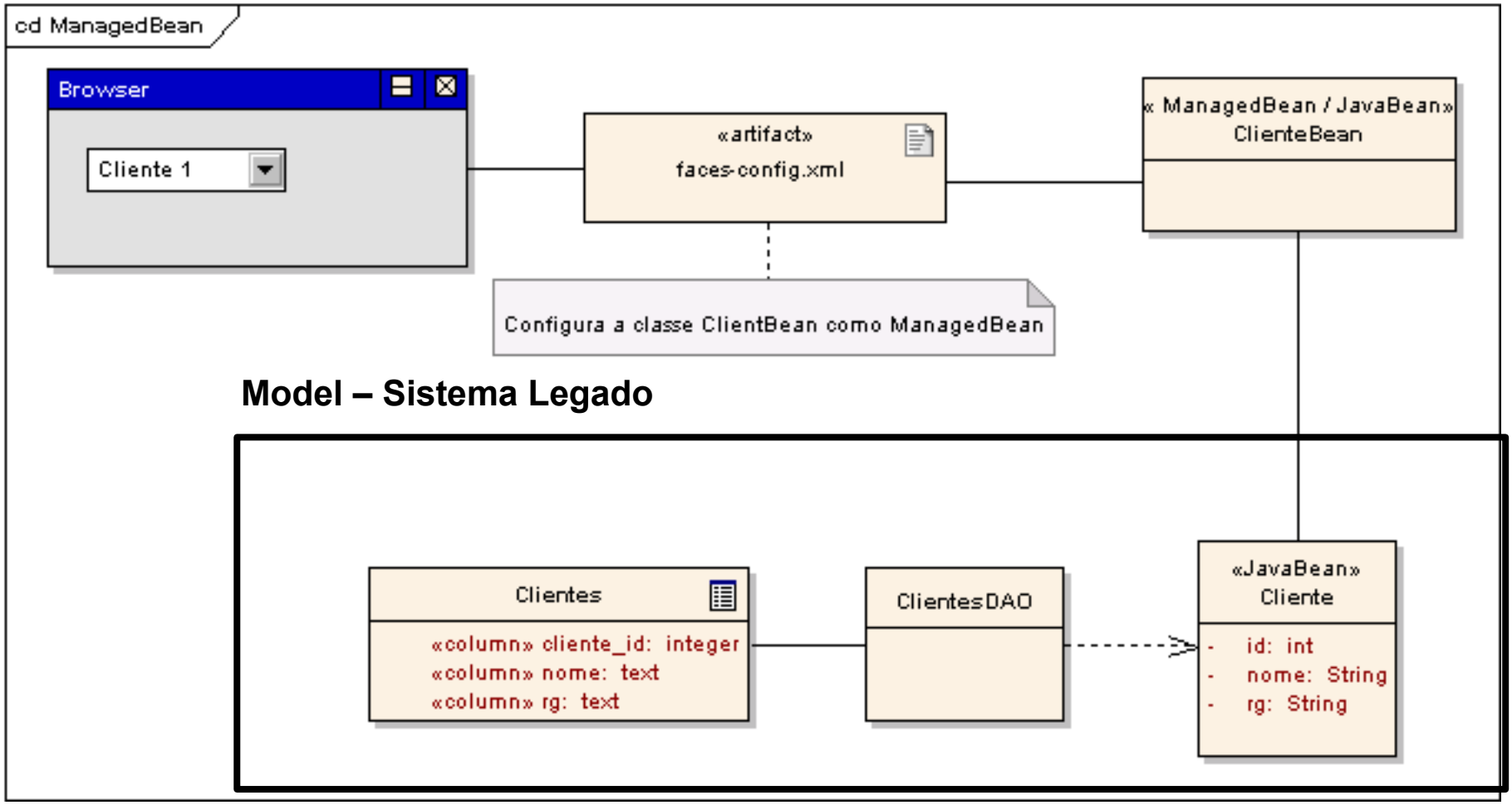
Um Managed Bean é um JavaBean gerenciado pelo framework JSF, ou seja, ele é instanciado, e colocado no escopo de acordo com as configurações encontradas no faces-config.xml

Um ManagedBean também é chamado de backing bean, pois contém os dados e os métodos que serão executados quando algum dos componentes da página JSF tiver que executar uma ação.

Managed Beans

Chamamos de *binding* o vinculo entre um componente da página JSF e o seu backing model / managed bean.

Managed Beans



Managed Beans

- Utilizamos Taglibs e EL (Expression Language) para associar (fazer o binding) de um componente de UI com um ManagedBean;

```
<h:outputText value="#{clienteBean.nome}"/>
```

A String clienteBean está associada a classe ClienteBean no faces-config.xml.

Managed Beans

Declaração de um ManagedBean no faces-config.xml

```
<managed-bean>
  <managed-bean-name>clienteBean</managed-bean-name>
  <managed-bean-class>ClienteBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Criação de páginas JSF

Demo:

Binding dos dados do formulário com um Managed Bean

Principais componentes de JSF

- Managed Beans
- Componentes de User-interface (UI)
- Renderizadores
- Validadores
- Conversores
- Eventos
- Listeners
- FacesController

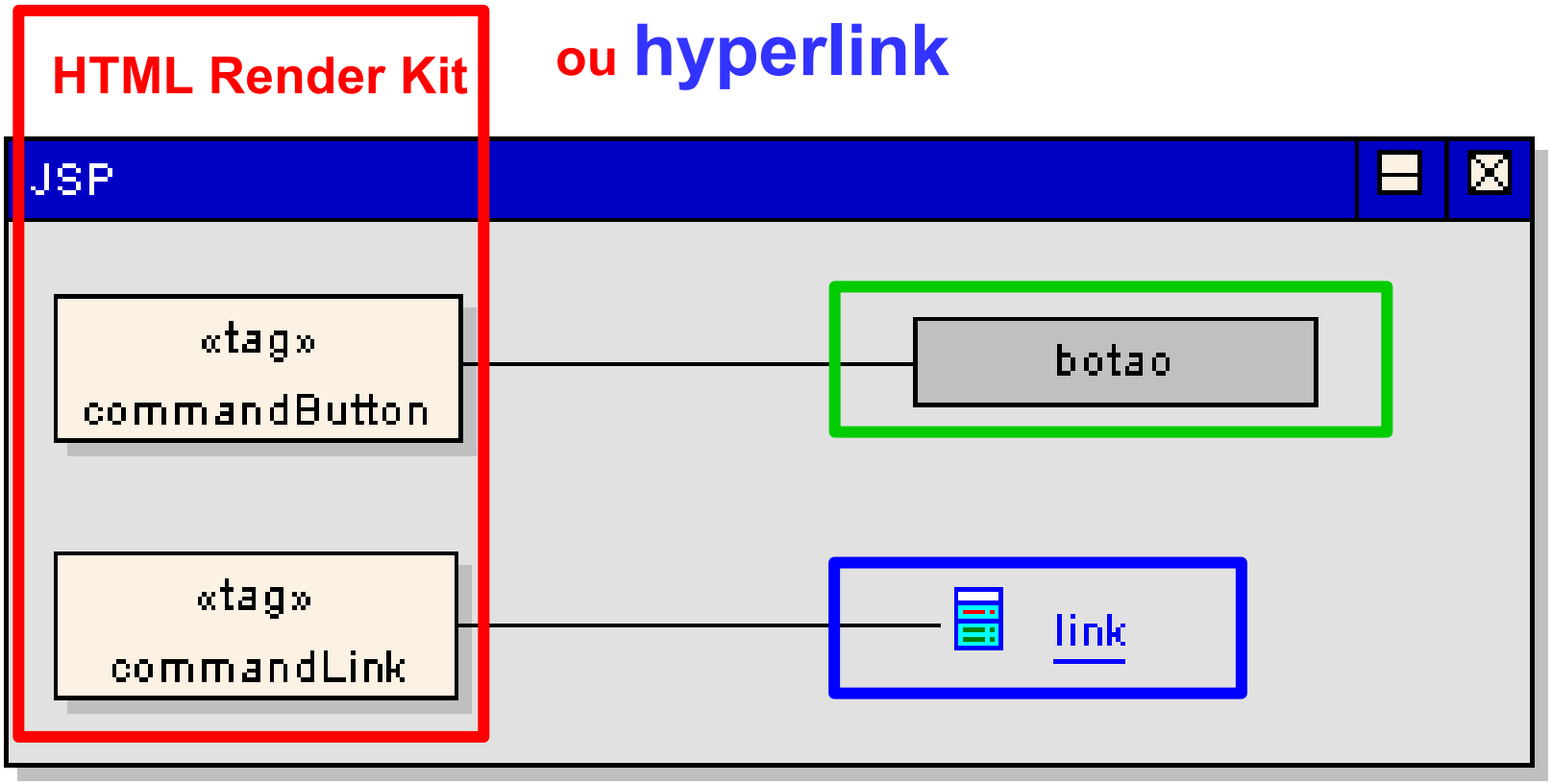
Componentes de User-interface (UI)

Técnicamente:

- Todos os componentes de UI implementam a interface UIComponent;
- A classe UIComponentBase já implementa a interface podendo ser estendida diretamente;

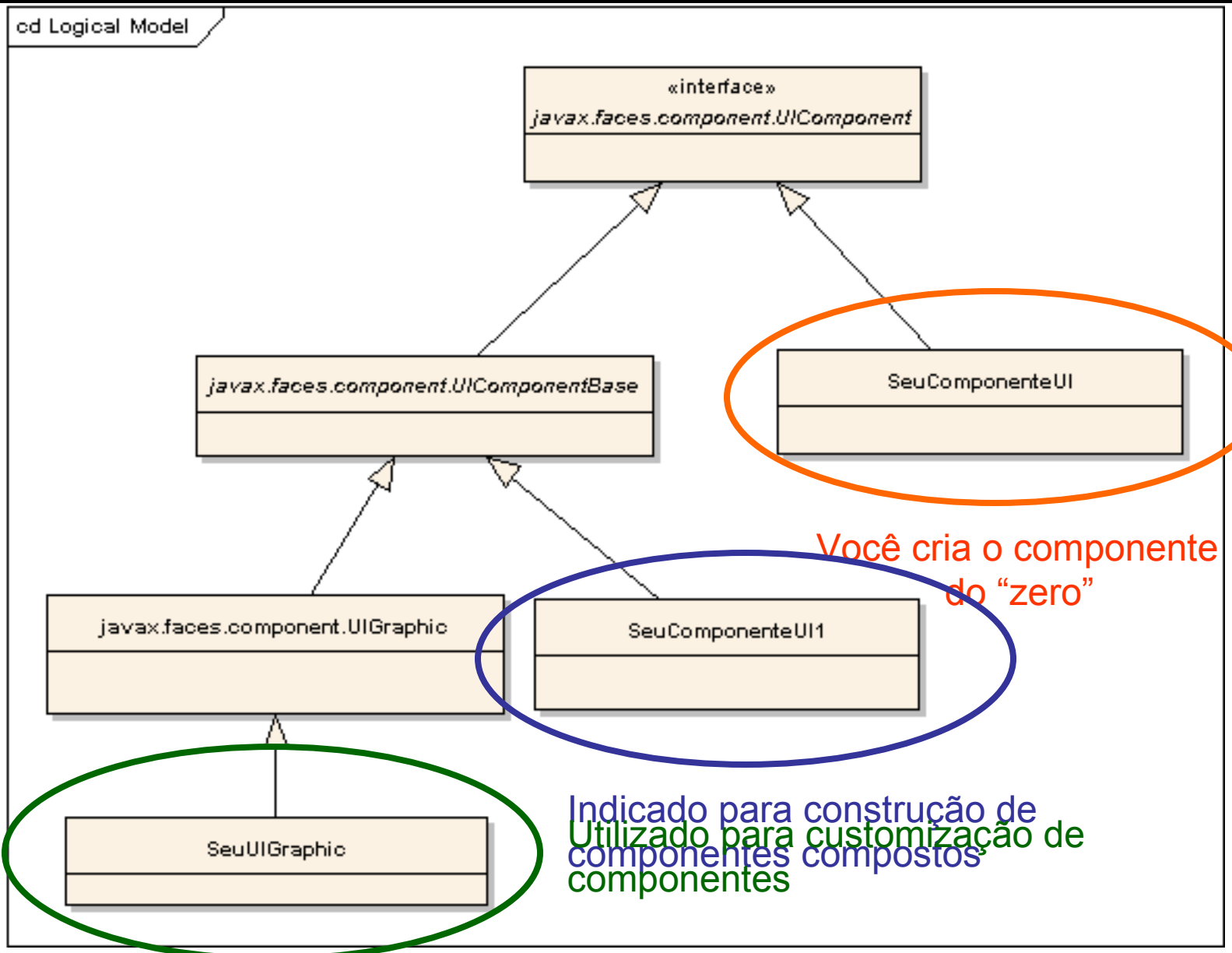
Componentes de User-interface (UI)

Custom Tag renderiza um **UICommand**
em forma de **botão**
ou **hyperlink**



Componentes de User-interface (UI)

Componente de UI não é responsável pela renderização;


cd Logical Model
«interface»
javax.faces.component.UIComponent
javax.faces.component.UIComponentBase
SeuComponenteUI
javax.faces.component.UIGraphic
SeuComponenteUI1
SeuUIGraphic
Você cria o componente do "zero"
Indicado para construção de componentes compostos
Utilizado para customização de componentes

Componentes de User-interface (UI)

Standard UI Components

- UICommand (botões, hyperlinks, menus)
- UIForm, UIInput, UIParameter,
- UIGraphic (imagens)
- UIMessage, UIMessages (msgs de erro)
- UIOutput, UIPanel

Componentes de User-interface (UI)

Standard UI Components

- UISelectBoolean (check box);
- UISelectItems, UISelectItem, UISelectMany UISelectOne (combo box, listas, conjunto de check boxes)
- UIData, UIColumn (tabelas, listas e árvores)

Componentes de User-interface (UI)

- Árvore de componentes
- Exemplo

Principais componentes do JSF

- Componentes de User-interface (UI)
- Validadores
- Conversores
- Eventos
- Listeners
- Renderizadores
- FacesController

Validadores

- **Todos os componentes de UI derivados de UIInput podem ser validados;**

Validadores

A validação pode ser feita das seguintes formas:

- Implementação do método validate(…) da interface UIComponent;
- Delegar a validação para um método de um JavaBean que estiver no escopo;
- Utilizar um Standard Validator
- Criar um Validator que implemente a interface javax.faces.validator.Validator

Validadores

Todas as implementações de JSF devem ter os seguintes Validators:

- DoubleRangeValidator
- LengthValidator
- LongRangeValidator

Principais componentes de JSF

- Managed Beans
- Componentes de User-interface (UI)
- Validadores
- Conversores
- Eventos
- Listeners
- Renderizadores
- FacesController

Conversores

Problemas comum:

- Dados digitados pelo usuário são Strings;
- Dados apresentados podem estar formatados ou ter representação gráfica, como um calendário;
- É necessário converter estas Strings para Date, long, char, int, Cliente ou qualquer outro tipo e vice-versa.

Conversores

- Todos os componentes de UI derivados de UIOutput podem ter conversores associados;
- Um conversor deve implementar a interface javax.faces.convert.Converter;

Conversores

Todas as implementações de JSF devem ter os seguintes Validators:

- DataTime
- Number (moeda, porcentagem, inteiros, etc…)

Principais componentes de JSF

- Managed Beans
- Componentes de User-interface (UI)
- Validadores
- Conversores
- Eventos
- Listeners
- Renderizadores
- FacesController

Eventos

- Modelo de eventos muito parecido com AWT e Swing
- Os Eventos são responsáveis pela propagação das ações sobre a interface com o usuário;
- Cada componente de UI pode disparar quantos eventos forem necessários;
- Um evento deve implementar a interface javax.faces.event.FacesEvent

Eventos

- **A especificação do JSF padroniza duas interfaces de eventos:**
  - ActionEvent (geradas através da interação com UICommand)
  - ValueChangeEvent (geradas através da interação com UIInput)
- **A especificação JavaBeans recomenda que todas as classe que representam eventos sejam pós-fixadas com a palavra Event.**

Listeners

- Cada tipo de evento tem uma interface listener correspondente;
- Toda interface listener deve estender a interface javax.faces.event.FacesListener ;
- A especificação JavaBeans recomenda que toda a classe que representa um listener tenha seu nome baseado no evento que está associada e seja pós-fixada com Listener ;

Listeners

- **A especificação do JSF define duas interfaces de Listeners padrão:**
  - ActionListener (UICommand)
  - ValueChangeListener (UIInput)

Listeners

- Cada componente de UI pode ter quantos listeners forem necessários;
- Um Componente de UI também pode ser um listener;

Listeners

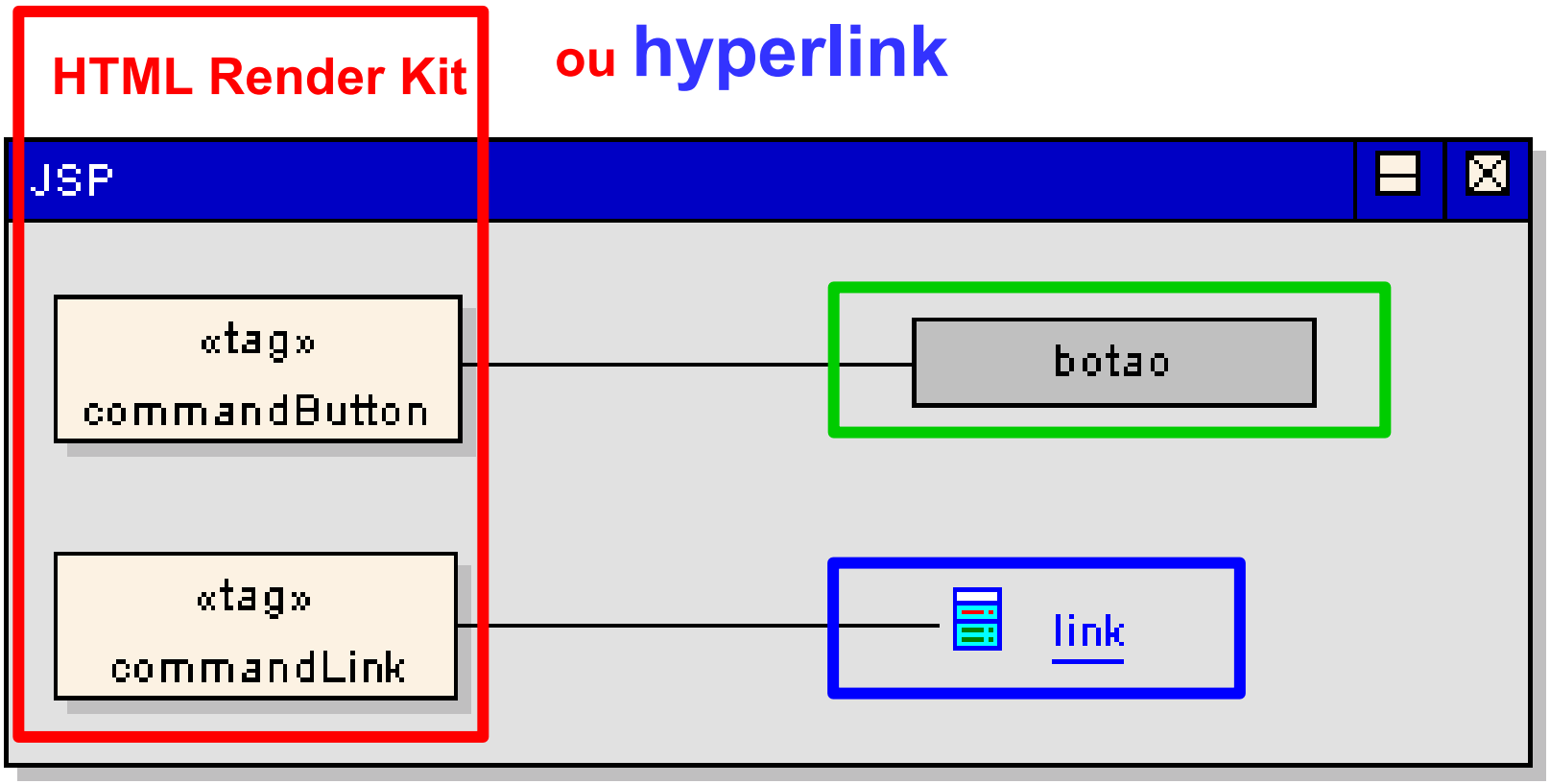
- **Os componentes de UI tem métodos para registrar e remover listeners. Exemplos:**
  - addActionListener
  - removeActionListener
- **O registro de listeners também pode ser feito através das custom tags encontradas na Standard JSF HTML tag library**

Principais componentes do JSF

- Managed Beans
- Componentes de User-interface (UI)
- Validadores
- Conversores
- Eventos
- Listeners
- Renderizadores
- FacesController

Renerizadores

Custom Tag renderiza um **UICommand**
em forma de **botão**
ou **hyperlink**



Renerizadores

- A renderização do componente pode ser feita pelo próprio componente ou delegada para um Renderizador.
- A renderização através de um Renderizador provê maior flexibilidade e reusabiliadde para aplicação.

Renerizadores

- A aparência fica separada da funcionalidade. O componente provê a funcionalidade e o renderizador a aparência;
- Utilizamos estes componentes de UI através de custom tags definidas no HTML RenderKit;
- O HTML RenderKit deve obrigatoriamente ser oferecido por todas as implementações de JSF;

Renerizadores

- Um RenderKit é uma coleção de renderizadores (javax.faces.render.Renderer)
- Um RenderKit deve implementar a interface javax.faces.render.RenderKit

Renerizadores

Standard JSF HTML Renderers:

- Button
- Web Link
- Table
- Form
- Image
- Hidden
- Secret
- Input Text
- TextArea
- Label
- Output Link
- Output Text
- Grid
- Group
- Checkbox
- Checkbox List
- Listbox
- Menu
- Radio

Principais componentes do JSF

- Managed Beans
- Componentes de User-interface (UI)
- Validadores
- Conversores
- Eventos
- Listeners
- Renderizadores
- FacesController

FacesController

- Um dos principais componentes de controle da aplicação;
- Implementação de FrontController conveniente para gerenciamento de segurança e acesso a aplicações Web;
- Responsável por grande parte do ciclo de vida de uma aplicação JSF, pois “inicia” vários processos;

Configuração

JARs necessários:

- jsf-api.jar
- jsf-ri.jar
- jstl.jar
- standard.jar
- commons-beansutils.jar
- commons-digester.jar
- commons-collections.jar
- commons-logging.jar

Configuração

Deployment Descriptor: web.xml

- Fazer o mapeamento do FacesController
- Configurar o tipo de persistência dos dados da tela (no cliente ou servidor)

Configuração

faces-config.xml

Neste arquivo são feitas todas as configurações da aplicação JSF, como por exemplo:

- Navegação
- Managed Beans
- Validators

Navegação

A navegação é configurada no faces-config.xml

```
<navigation-rule>
  <from-view-id>/index.jsp</from-view-id>

  <navigation-case>
     <description>Descricao</description>
     <from-outcome>viewClienteData</from-outcome>
     <to-view-id>/viewClienteData.jsp</to-view-id>
  </navigation-case>

</navigation-rule>
```

Integração JSF e JSP

- JSP não é a única forma de construir interfaces para JSF;
- A integração é feita através de TagLibs;
- As TagLibs “ligam” os componentes server-side aos client-side (tipicamente HTML)

Integração JSF e JSP

- **Core Tag Library:** gerenciamento de listeners, configuração de componentes, validação, entre outros;
- **HTML Tag Library:** Definem o renderizador do componente de UI, utilizam EL para integração com os Managed Beans;
- Existe uma tag para cada combinação entre renderizador e componente;
  Por exemplo, um **UIInput** pode ser renderizado em forma de **inputText** ou de **inputSecret**;

Integração JSF e JSP

- Configurações necessárias:

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

Ferramentas

- Java Studio Creator
- JDeveloper
- JBuilder
- MyEclipseIDE
- Exadel Studio
- WSAD

Mais informações

- http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
- http://java.sun.com/j2ee/javaserverfaces
- Mastering JavaServer Faces – Editora Wiley
- http://www.jcp.org/en/jsr/detail?id=127