

# Desenvolvimento Web I

Residência de TI Aplicada  
à Área Jurídica - JF e TCE

Professor: Uirá Kulesza

Outubro 2018

# [ Aula 3: Java Server Faces ]

O que vimos até agora?

Que arquitetura vocês  
implementaram no nosso mini-  
framework web?

# Motivação

- Tecnologias de *Servlet* e JSP são úteis para construção de funcionalidades de interface com o usuário
- Mas elas não oferecem um framework/ modelo para manipulação de entradas/ eventos de UI padronizado
- Isso impede que ferramentas de desenvolvimento ofereçam facilidades para a rápida construção de código de UI

# Motivação

- Plataforma JEE ao longo de vários anos não definiu um modelo de UI padronizado, tais como o Swing/AWT para JSE
- Outras linguagens oferecem ambientes para a construção de UI sejam elas para *web* (Visual Studio) ou *desktop* (Delphi, Visual Basic)
- Isso traz um grande aumento de produtividade
- Java Server Faces veio preencher essa lacuna !!!

# Java Server Faces

- Framework MVC para desenvolvimento de interfaces com o usuário de aplicações web na plataforma JEE
- Oferece grande variedade de componentes de interface do usuário, acessíveis através de bibliotecas de tags JSP
- Define estratégias no lado servidor para processamento de eventos e validação da entrada do usuário

# Características do JSF

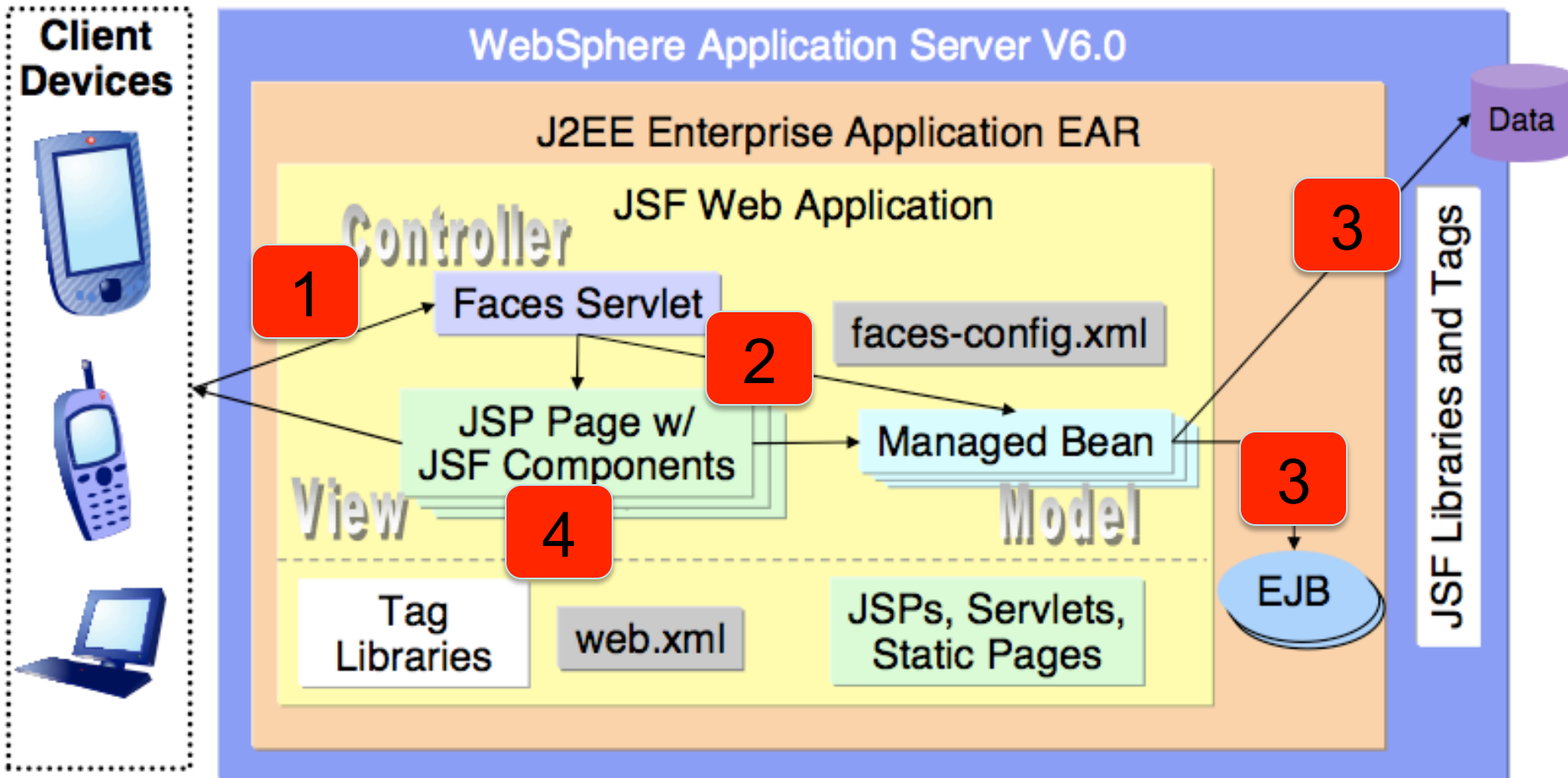
- Modelo de Componentes para UI
  - Oferece um framework para a criação de componentes customizados, extensível e que promove o reuso de componentes
- Modelo de Programação Orientada a Eventos
  - Tratamento de eventos do usuário
- Modelo de Validação
  - Oferece framework para validação de entrada do usuário

# Características do JSF (cont.)

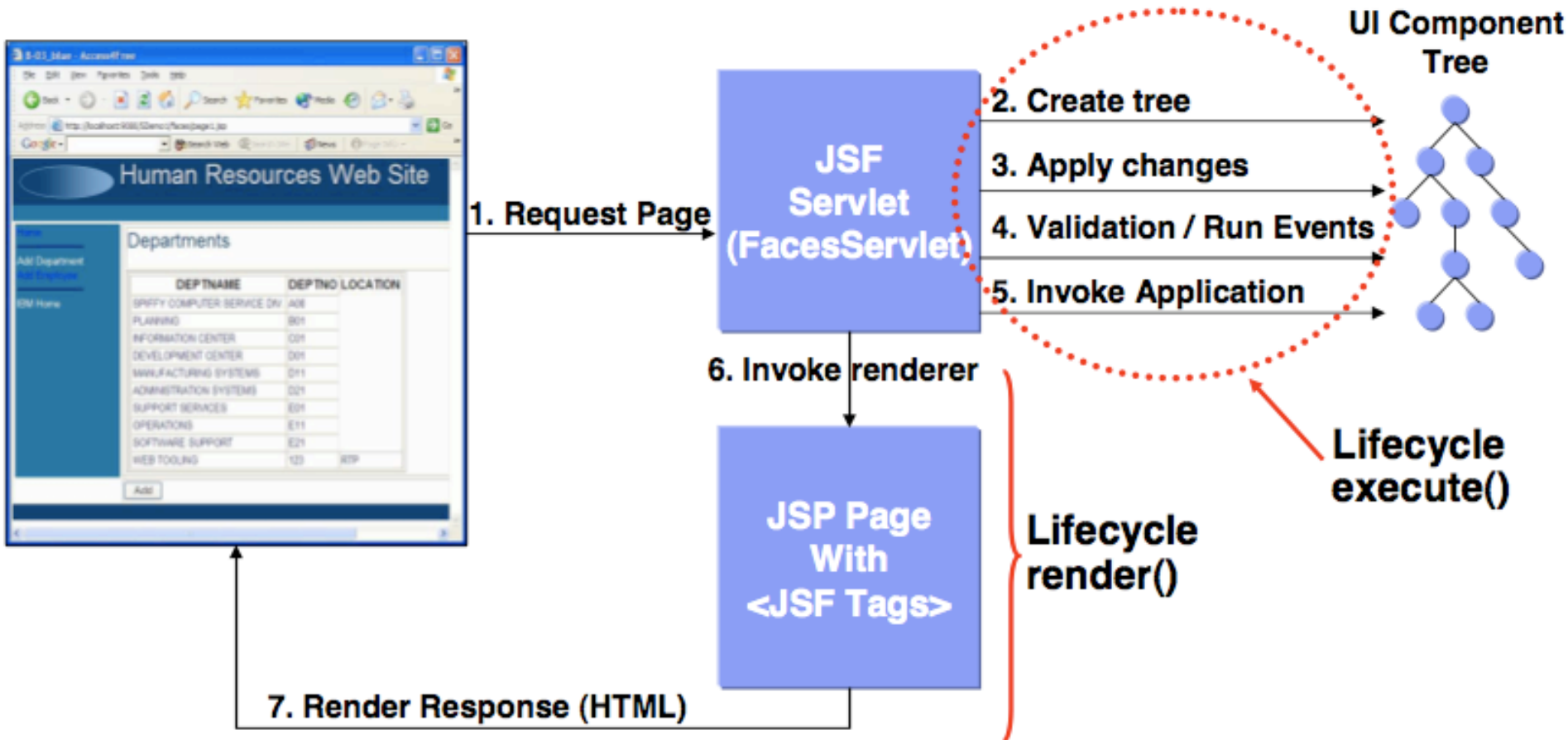
- Modelo de Navegação
  - Oferece mecanismos para declaração de rotas de navegação entre páginas sem codificação
- Modelo de Conversão
  - Permitir conversão automática de dados da UI para objetos do modelo
- Gerenciamento do Estado de componentes UI



# Visão Geral do JSF 1.2



# Visão Geral do JSF



# Aplicação Java Server Faces

- É composta de:
  - (1) Páginas JSF com tags apropriadas (componentes gráficos + componentes comando)
  - (2) Managed Beans – classes Java que possuem atributos associado as informações passadas para as páginas JSF
  - (3) Arquivo de configuração – web.xml, faces-config.xml
  - (4) Regras de navegação entre componentes e páginas JSF

# Arquivo de Configuração: faces-config.xml

- Define a configuração do framework JSF para uma dada aplicação
- Permite definir a configuração de:
  - Managed Beans
  - Validators
  - Regras de navegação
  - Componentes customizados
  - Conversores

# Bibliotecas de Tag JSF

- JSF Core

- Oferece tags que são usadas principalmente para cadastro de validadores e listeners de eventos

- Declaração:

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

- HTML Basic

- Oferece tags que suportam componentes de renderização de HTML padrão

- Declaração:

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

# Exemplo de Aplicação JSF

# Aplicação Exemplo

- Aplicação de conversão de temperatura Celsius para Farenheit



A screenshot of a web browser window displaying a temperature conversion application. The browser's address bar shows the URL `http://localhost:8080/testeJSF/faces/Convertor.jsp`. The application interface includes a label "Celsius" next to a text input field containing the value "30.0". Below the input field are two buttons: "Calcular" and "Limpar". Underneath the buttons, the word "Resultado" is displayed in a bold font. At the bottom of the interface, the text "Fahrenheit 86.0" is shown, indicating the converted temperature.

# Passo 1: Configurar o web.xml

- Necessidade de configurar o arquivo web.xml para que chamadas a aplicação sejam redirecionadas para o servlet Faces do JSF
- Esse arquivo é gerado automaticamente quando criando um projeto JSF no Eclipse WTP



# Exemplo de arquivo: web.xml

```
...  
<servlet>  
  <servlet-name>Faces Servlet</servlet-name>  
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>  
  <load-on-startup>1</load-on-startup>  
</servlet>  
  
<servlet-mapping>  
  <servlet-name>Faces Servlet</servlet-name>  
  <url-pattern>/faces/*</url-pattern>  
</servlet-mapping>  
...
```

## Passo 2: Definir o Managed Bean

- Os Managed Beans são classes Java simples (POJOs) que são mapeadas para tratar o processamento de páginas JSF
- Eles definem atributos e métodos que são associados com os componentes UI apresentados nas páginas
- Eles funcionam como controladores na arquitetura MVC

# Managed Bean >> TemperatureConvertor.java

```
public class TemperatureConvertor {  
    private double celsius;  
    private double fahrenheit;  
    private boolean initial= true;  
    public double getCelsius() {  
        return celsius;  
    }  
    public void setCelsius(double celsius) {  
        this.celsius = celsius;  
    }  
    public double getFahrenheit() {  
        return fahrenheit;  
    }  
    ....  
}
```

# Managed Bean >> TemperatureConvertor.java

```
public class TemperatureConvertor {  
    public boolean getInitial(){  
        return initial;  
    }  
    public String reset (){  
        initial = true;  
        fahrenheit =0;  
        celsius = 0;  
        return "reset";  
    }  
    public String celsiusToFahrenheit(){  
        initial = false;  
        fahrenheit = (celsius *9 / 5) +32;  
        return "calculated";  
    }  
}
```

# Passo 3: Definir as páginas JSF

- Cada página JSF utiliza componentes UI das 2 bibliotecas de tags padrões do JSF para definir:
  - Componentes gráficos que representam informações na tela
    - Exemplo: formulários, tabelas, botões
  - Componentes de comando que indicam o que deve ser feito para diferentes eventos da aplicação:
    - Exemplos: clique no botão, validação

# Página JSF – Conversor.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loos
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Celsius to Fahrenheit Convertor</title>
</head>
<body>
```

# Página JSF – Conversor.jsp

```
<body>
<f:view>
  <h:form>
    <h:panelGrid columns="2">
      <h:outputLabel value="Celsius"></h:outputLabel>
      <h:inputText value="#{temparaturConvertor.celsius}"></h:inputText>
    </h:panelGrid>
    <h:commandButton action="#{temparaturConvertor.celsiusToFahrenheit}" value="Calcular">
    </h:commandButton>
    <h:commandButton action="#{temparaturConvertor.reset}" value="Limpar"></h:commandButton>
    <h:messages layout="table"></h:messages>
  </h:form>

  <h:panelGroup rendered="#{temparaturConvertor.initial!=true}">
    <h3> Result </h3>
    <h:outputLabel value="Fahrenheit "></h:outputLabel>
    <h:outputLabel value="#{temparaturConvertor.fahrenheit}"></h:outputLabel>
  </h:panelGroup>
</f:view>
</body>
</html>
```

# Formulário JSF Final



A screenshot of a web browser window displaying a temperature converter application. The browser's address bar shows the URL `http://localhost:8080/testeJSF/faces/Convertor.jsp`. The page content includes a label "Celsius" followed by a text input field containing the value "30.0". Below the input field are two buttons: "Calcular" and "Limpar". Further down, the heading "Resultado" is displayed in bold. Underneath the heading, the text "Fahrenheit 86.0" is shown, indicating the result of the conversion.

← → [Red Square] [Yellow Dollar Sign]

`http://localhost:8080/testeJSF/faces/Convertor.jsp`

Celsius 30.0

Calcular Limpar

**Resultado**

Fahrenheit 86.0



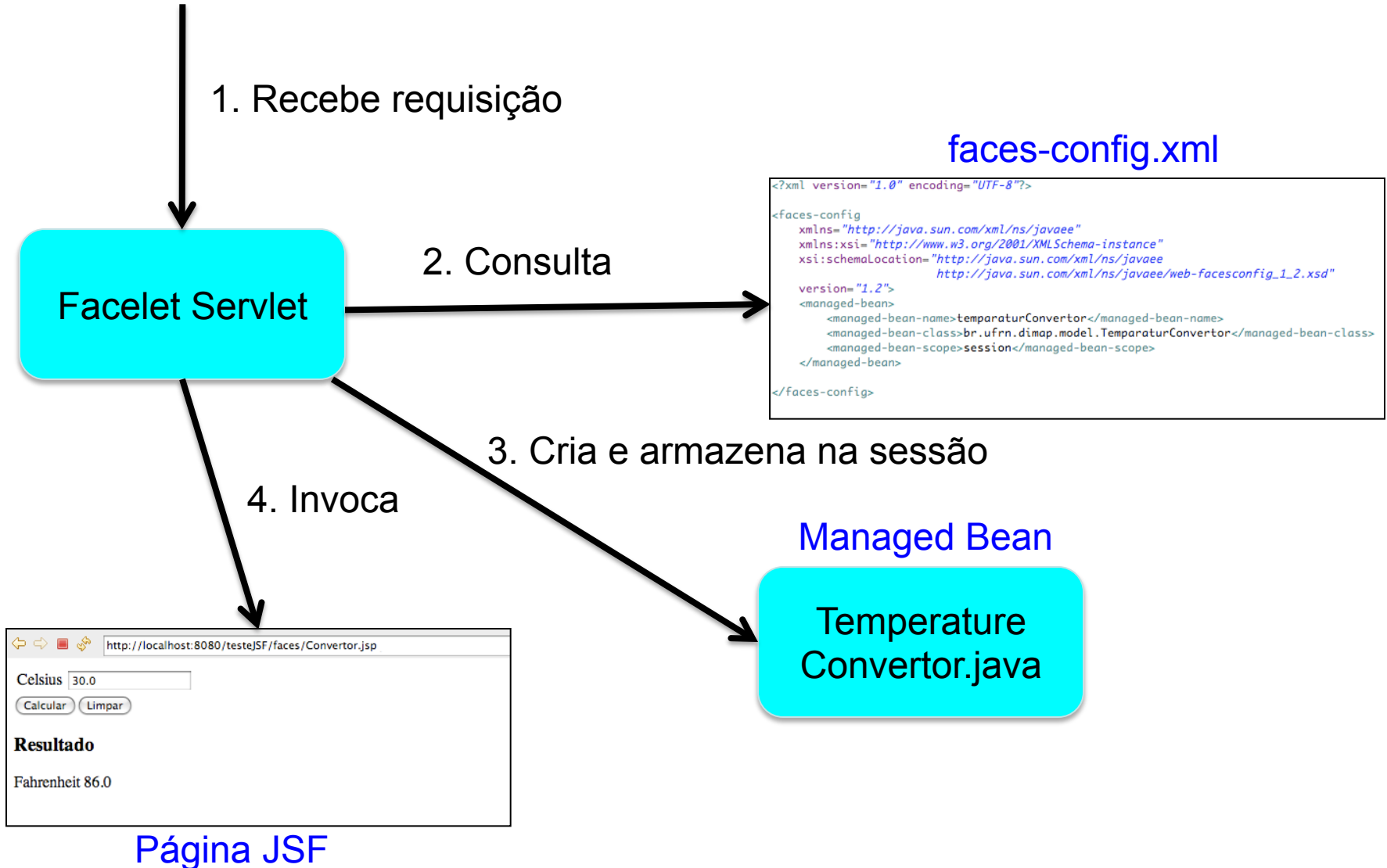
## Passo 4: Definir configurações de navegação

- O arquivo de configuração do JSF deve ser configurado para:
  - definir mecanismos de navegação entre páginas
  - declarar os *managed beans* que serão usados na aplicação
- No caso dessa aplicação de conversão, apenas um managed bean será declarado, pois não existe outra página JSF no processo

# Arquivo faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
  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/web-facesconfig_1_2.xsd"
  version="1.2">
  <managed-bean>
    <managed-bean-name>temperaturConvertor</managed-bean-name>
    <managed-bean-class>br.ufrn.dimap.model.TemperaturConvertor</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
</faces-config>
```

# Resumindo



# Configuração da Aplicação JSF no Eclipse

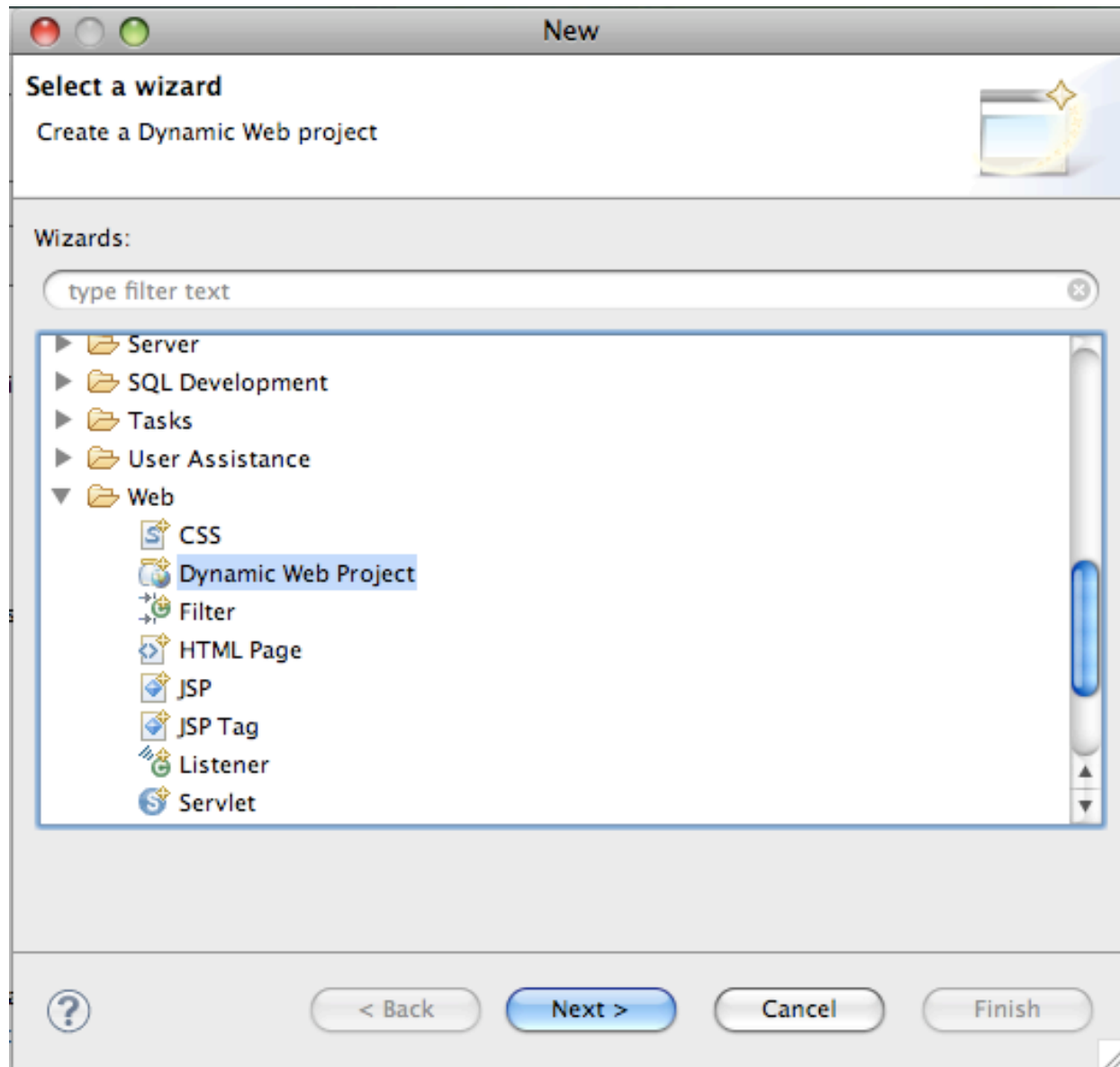
# Eclipse WTP

- O plugin WTP (*Web Tooling Project*) do Eclipse oferece uma série de facilidades para o desenvolvimento de aplicações JSF
- Recursos oferecidos:
  - Criação e configuração automática de projeto JSF
  - Editores gráficos para configuração da aplicação
  - Editores gráficos para edição de páginas JSF
- Os slides a seguir, mostram o passo-a-passo de configuração de uma aplicação no Eclipse

# Passo 1: Criar Projeto JSF

- Criação de projeto JSF, usando a seguinte opção “New / Project / Web / Dynamic Web Project”
- Configuração de bibliotecas de tags JSF a serem usadas no projeto
- Bibliotecas do JSF propriamente ditas são automaticamente inseridas pela wizard de criação do projeto


# Criar Projeto JSF



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.



d

Project name:

Project contents

☒ Use default

Directory:

Target runtime

Dynamic web module version

Configuration

Configures a Dynamic Web application to use JSF v1.2

EAR membership


☐ Add project to an EAR

EAR project name:

Working sets

☐ Add project to working sets

Working sets:






New Dynamic Web Project

JSF Capabilities


Add JSF capabilities to this Web Project





JSF Implementation Library

Type: 

User Library

☒  JSF 1.2 (Apache Myfaces JSF Core-1.2 API 1.2.8)

☒  JSTL



☒ Include libraries with this application

JSF Configuration File: 

/WEB-INF/faces-config.xml

JSF Servlet Name: 

Faces Servlet

JSF Servlet Classname: 


javax.faces.webapp.FacesServlet

URL Mapping Patterns: 

/faces/\*

Add...

Remove



< Back

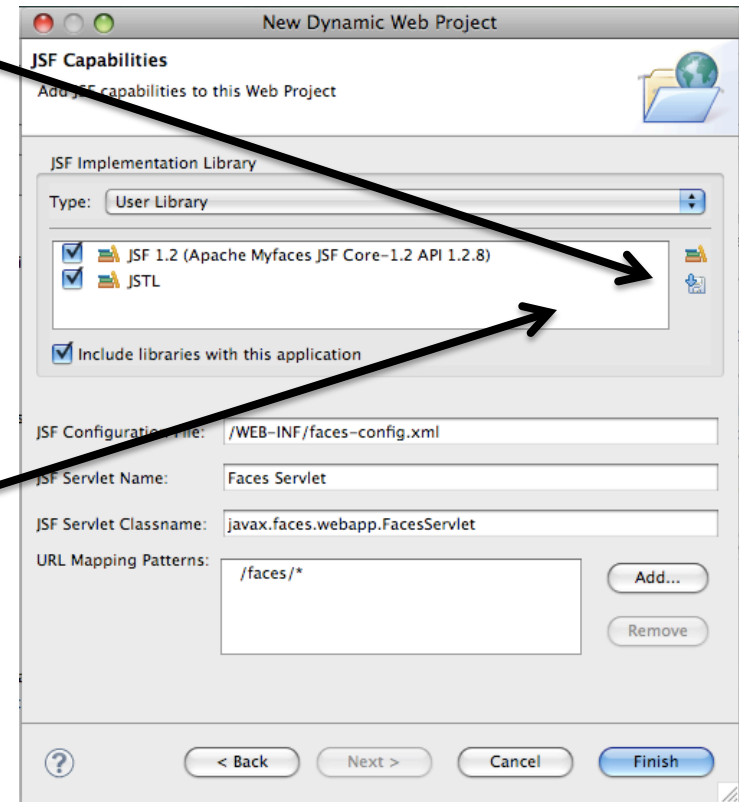
Next >

Cancel

Finish

# Configuração de Bibliotecas do JSF

- Apache MyFaces JSF:
  - Clicar no botão para fazer download
- Baixar bibliotecas JSTL
  - jstl-api-1.2.jar
  - jstl-impl-1.2.jar
  - E criar uma library local
    - (ver detalhes no link do tutorial)



## Passo 2: Configurar Managed Bean

- Após a criação de uma classe *Managed Bean*, esta pode ser automaticamente configurada usando o editor do faces-config.xml do JSF oferecido pelo Eclipse WTP
- Clicar no arquivo faces-config.xml para abrir Editor

# Editor de configurações do JSF

register.jsp

NovoDepartamento.jsp

faces-config.xml


Convertor.jsp

Celsius to Fahrenheit

”21

## Faces Configuration Introduction

Introduction



### The Faces Configuration Editor

The faces configuration editor helps you to complete the JSF web application development process by allowing you to edit faces configuration. Use the pages in this editor to define and edit page navigations, managed beans, component, converter, validator, renderkit and other element configurations.

[Start](#)

Start working with the editor by selecting PageFlow and defining page navigations.

[Tutorial](#)

Launch the cheat sheet for guidance in working with the editor.

[Help](#)

Launch the help system and review topics about working with faces configuration descriptor.

☐ Don't show this page next time.

Introduction | Overview | Navigation Rule | ManagedBean | Component | Others | Source

# Editor de configurações do JSF

register.jsp

NovoDepartamento.jsp

faces-config.xml


Convertor.jsp

Celsius to Fahrenheit

”21

## Faces Configuration Introduction

Introduction



### The Faces Configuration Editor

The faces configuration editor helps you to complete the JSF web application development process by allowing you to edit faces configuration. Use the pages in this editor to define and edit page navigations, managed beans, component, converter, validator, renderkit and other element configurations.

[Start](#)

Start working with the editor by selecting PageFlow and defining page navigations.

[Tutorial](#)

Launch the cheat sheet for guidance in working with the editor.

[Help](#)

Launch the help system and review topics about working with faces configuration descriptor.

☐ Don't show this page next time.

Introduction | Overview | Navigation Rule | ManagedBean | Component | Others | Source

# Configuração de Managed Beans (1/6)

The screenshot shows a web IDE interface with several tabs at the top: `register.jsp`, `NovoDepartamento.jsp`, `faces-config.xml` (active), `Convertor.jsp`, and `Celsius to Fahrenheit`. The main window is titled **ManagedBean**. Inside, there is a section labeled **Managed Bean Elements** with the text "The following managed beans are defined". Below this, a list of managed beans is shown, each with a small icon and a name: 

- `session` (with a session icon)
- `temparaturConvertor` (with a temperature icon)
- `request` (with a request icon)
- `application` (with an application icon)
- `none` (with a none icon)

 To the right of the list, there are two buttons: **Add** and **Remove**. At the bottom of the window, there is a navigation bar with tabs: `Introduction`, `Overview`, `Navigation Rule`, `ManagedBean` (active), `Component`, `Others`, and `Source`.

# Configuração de Managed Beans (2/6)

New Managed Bean Wizard

## Java Class Selection

Search for an existing class or generate a new one.

☒ Using an existing Java class

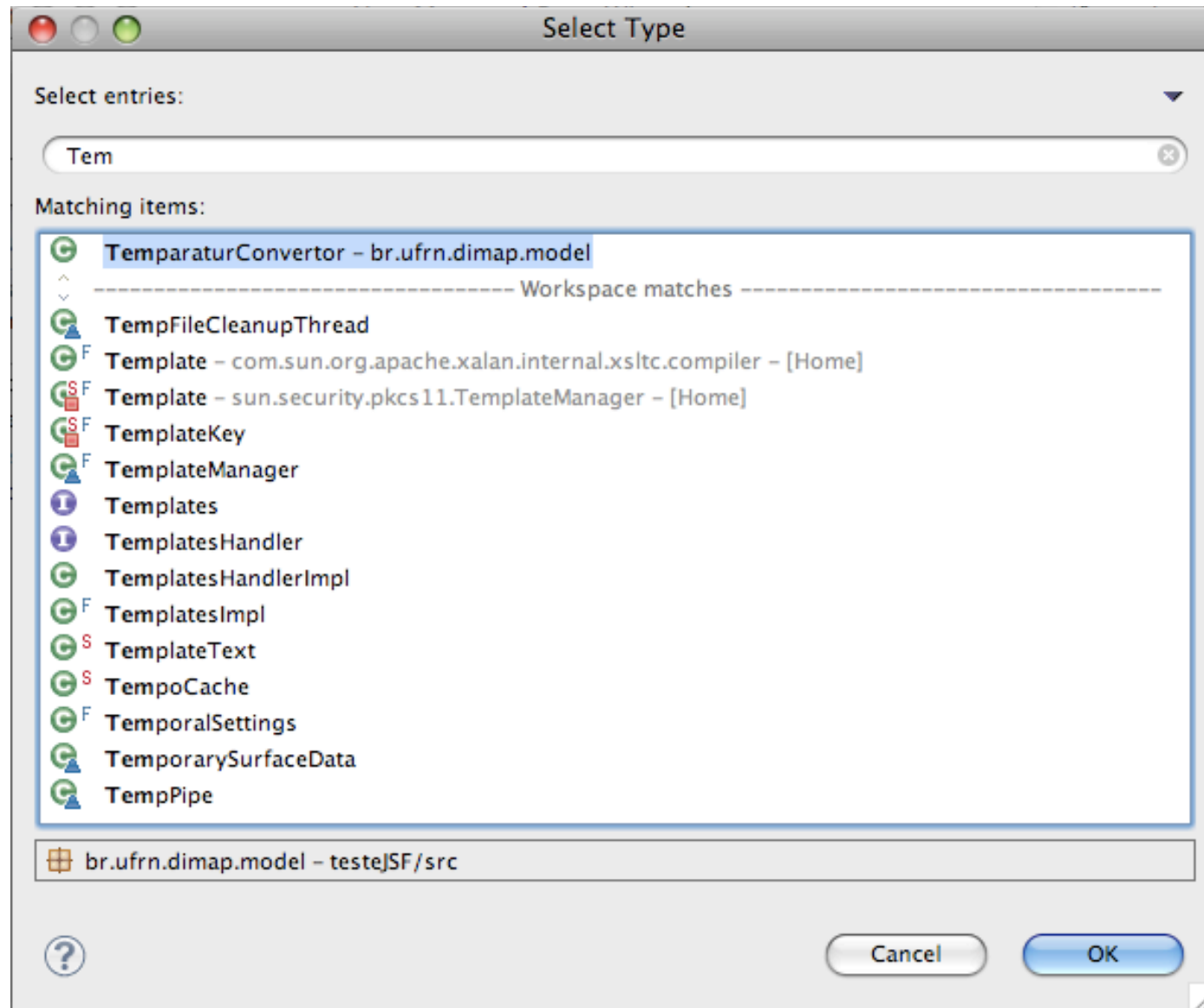
Qualified class name:

(This option will use an existing java class as managed bean's type.)

☐ Create a new Java class

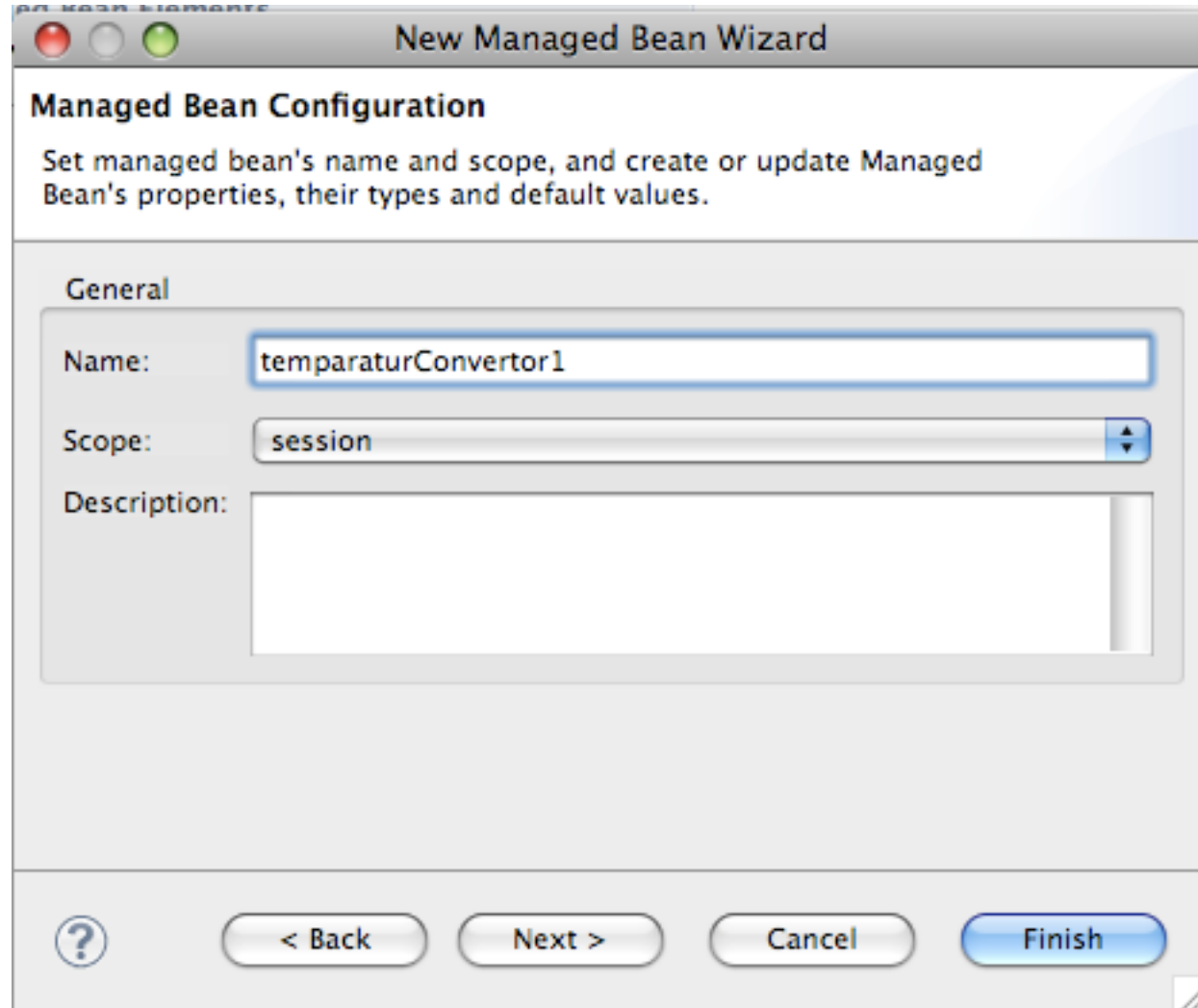
(This option will create a new java class in the next wizard page.)

# Configuração de Managed Beans (3/6)





# Configuração de Managed Beans (4/6)



The image shows a 'New Managed Bean Wizard' dialog box. The title bar reads 'New Managed Bean Wizard'. The main heading is 'Managed Bean Configuration'. Below this, a descriptive text says: 'Set managed bean's name and scope, and create or update Managed Bean's properties, their types and default values.' The 'General' tab is selected, showing fields for 'Name', 'Scope', and 'Description'. The 'Name' field contains 'temparaturConvector1'. The 'Scope' dropdown menu is set to 'session'. The 'Description' field is empty. At the bottom, there are four buttons: a help button (question mark icon), '< Back', 'Next >', and 'Finish'.

**New Managed Bean Wizard**

**Managed Bean Configuration**


Set managed bean's name and scope, and create or update Managed Bean's properties, their types and default values.

**General**

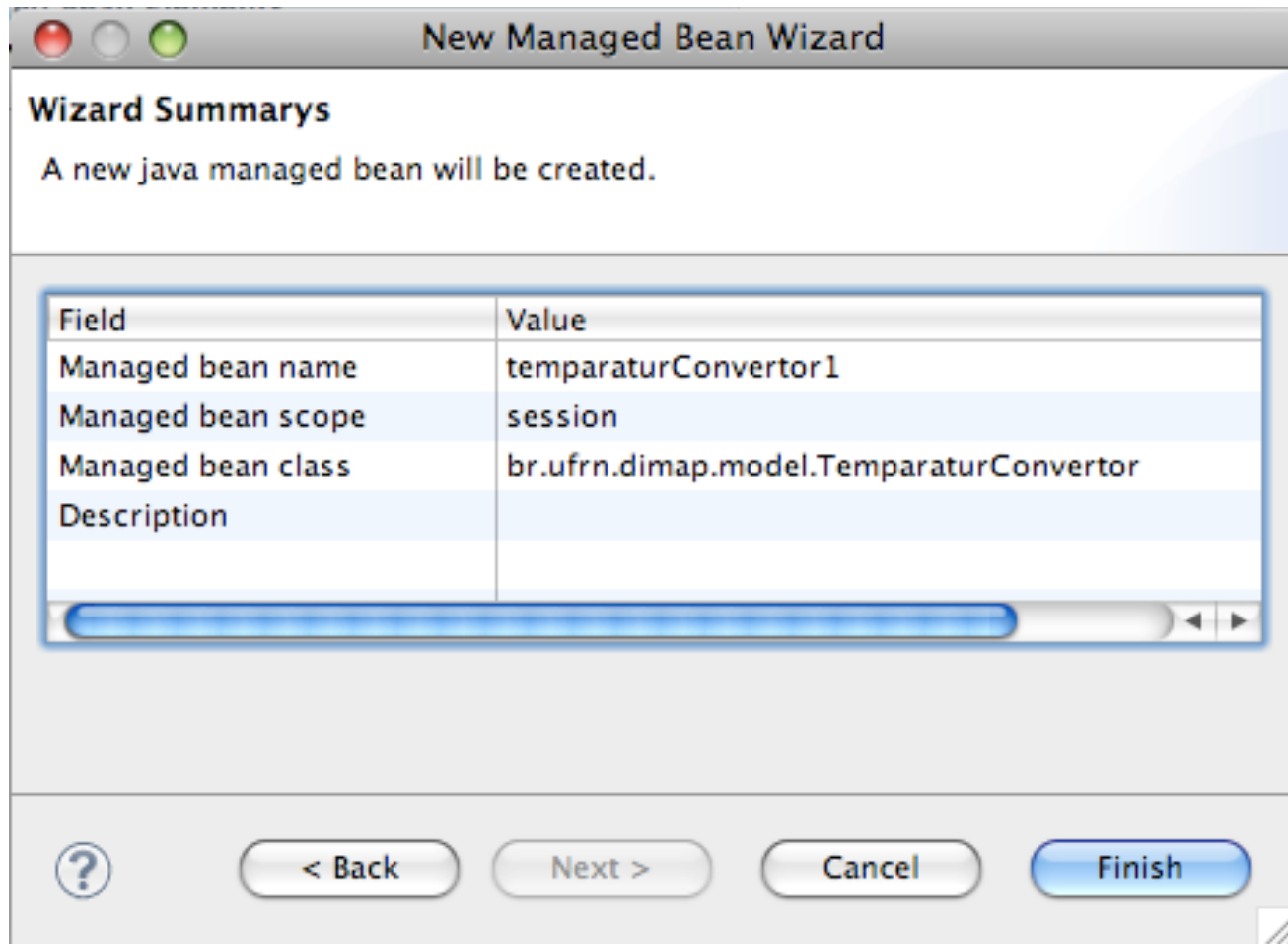
Name:

Scope:

Description:



# Configuração de Managed Beans (5/6)



# Configuração de Managed Beans (6/6)

register.jsp NovoDepartamento.jsp faces-config.xml Converter.jsp Celsius to Fahrenheit »21

## ManagedBean

Managed Bean Elements

The following managed beans are defined

session

temparaturConvertor

request

application

none

AddRemove

Managed Bean

This section describes general configuration of this managed bean

Managed Bean name\*: temparaturConvertor

Managed Bean class\*: br.ufrrn.dimap.model.TemparaturConvertor

Browse...

Managed Bean scope\*: session

Initialization

You can initialize the managed bean's properties or itself if it is a subclass of java.util.Map or java.util.List

Managed Bean class type: ☒ General class ☐ Map ☐ List

Name	Class	Value

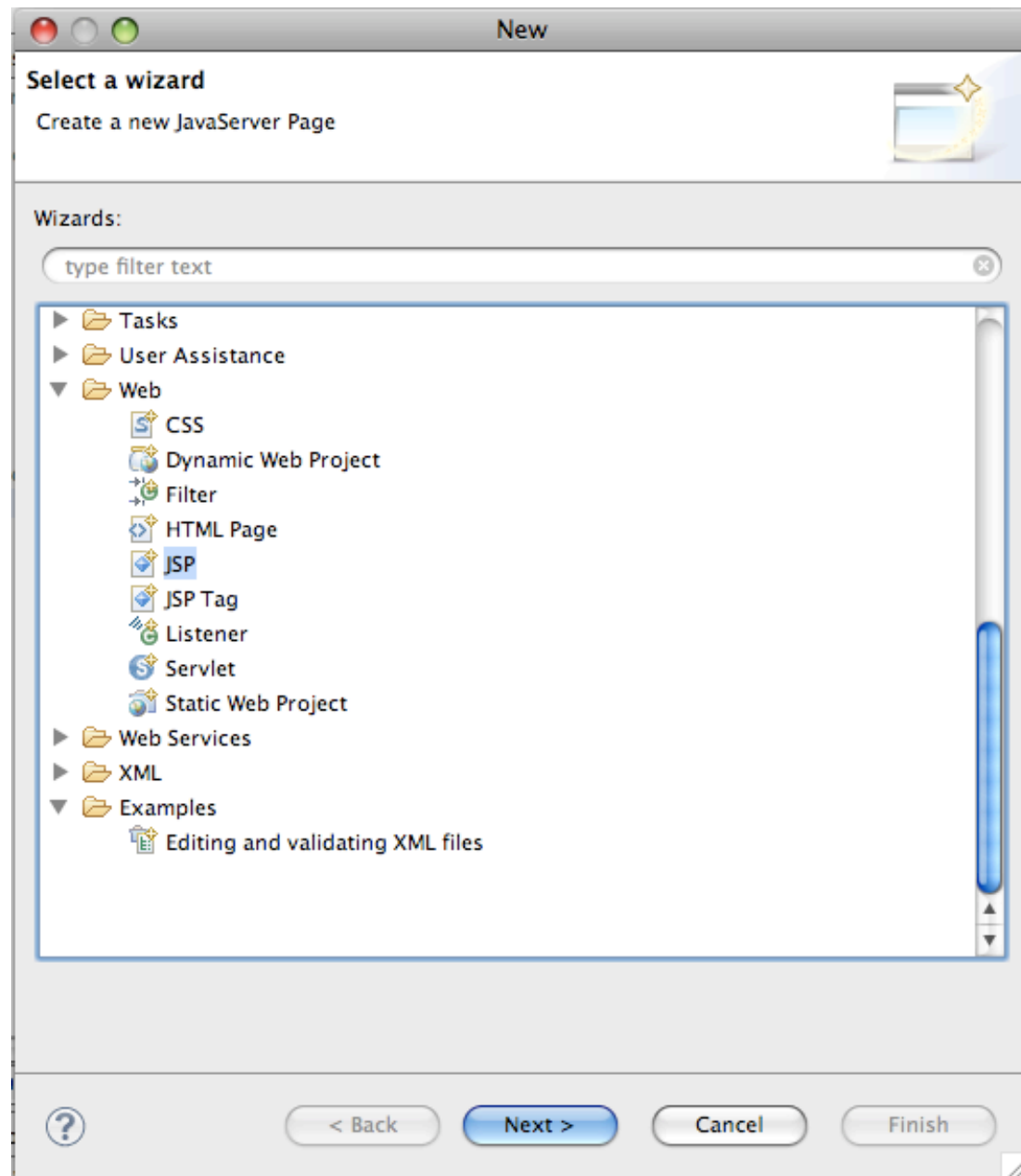
Add...Edit...Remove

IntroductionOverviewNavigation RuleManagedBeanComponentOthersSource

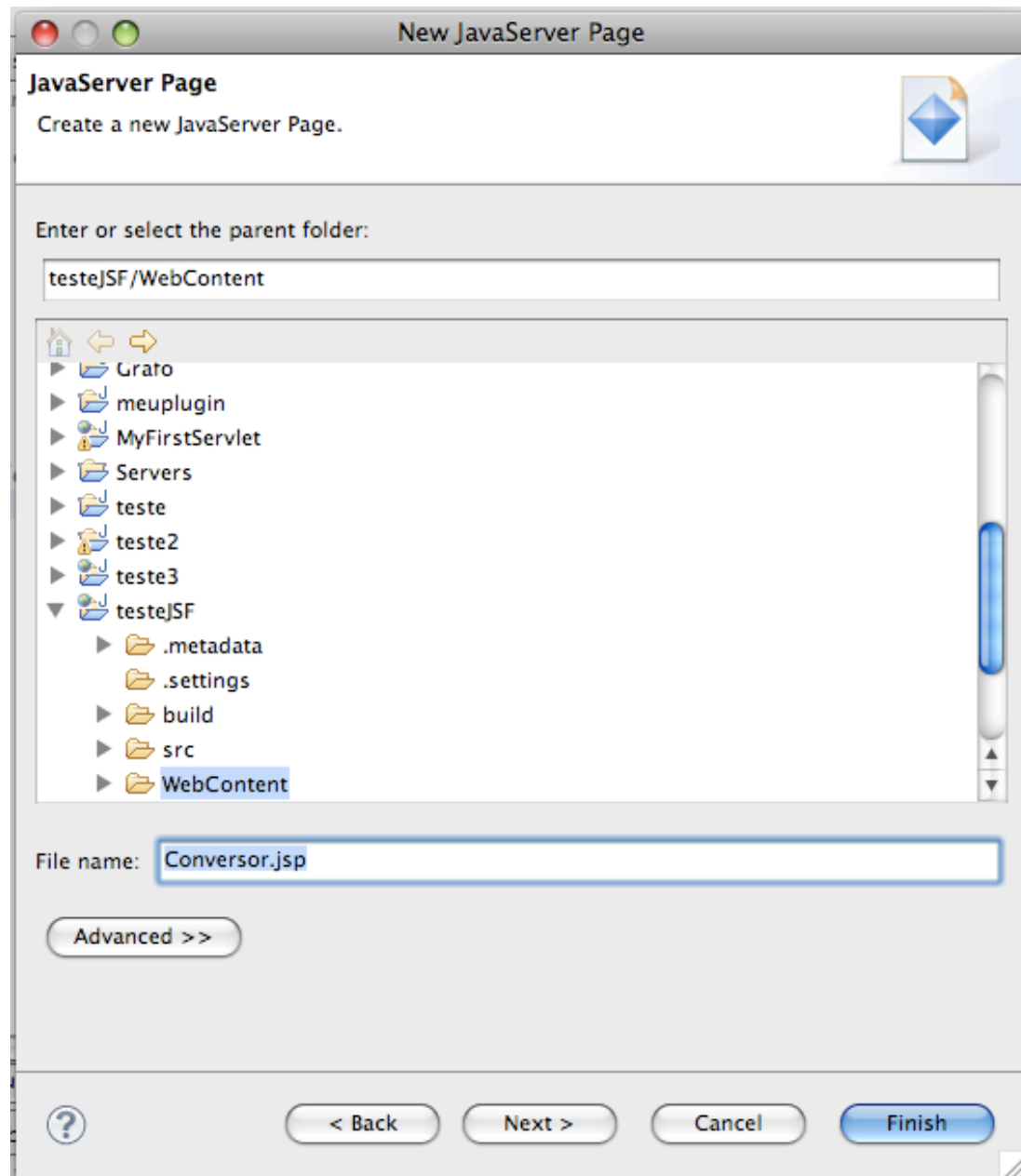
## Passo 3: Criação e Edição de páginas JSF

- O Eclipse WTP também oferece facilidades para a criação de páginas JSF
- Diversas wizards estão disponíveis para a criação de páginas JSF padrões
- Além disso, também um editor gráfico do WTP permite a edição e visualização de páginas JSF

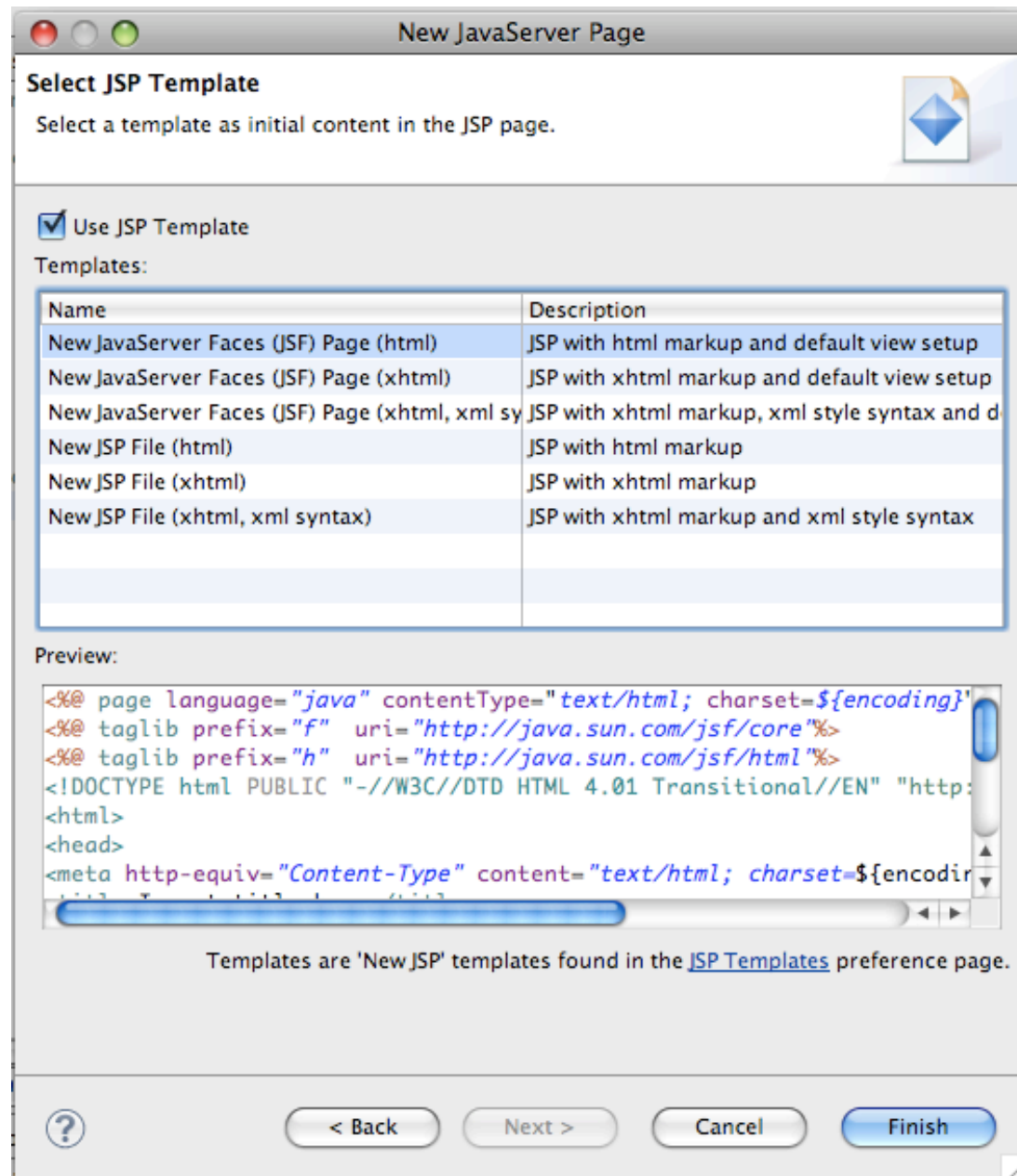
# Wizard de Criação de página JSF (1/3)



# Wizard de Criação de página JSF (2/3)



# Wizard de Criação de página JSF (3/3)



# Editor de páginas JSF (1/2)

register.jsp faces-config.xml Convertor.jsp Convertor.jsp Celsius to Fahrenheit 21

directive.page http://java.sun.com/jsf/core http://java.sun.com/jsf/html

< > head

Celsius

Calcular Limpar messages

**Resultado**

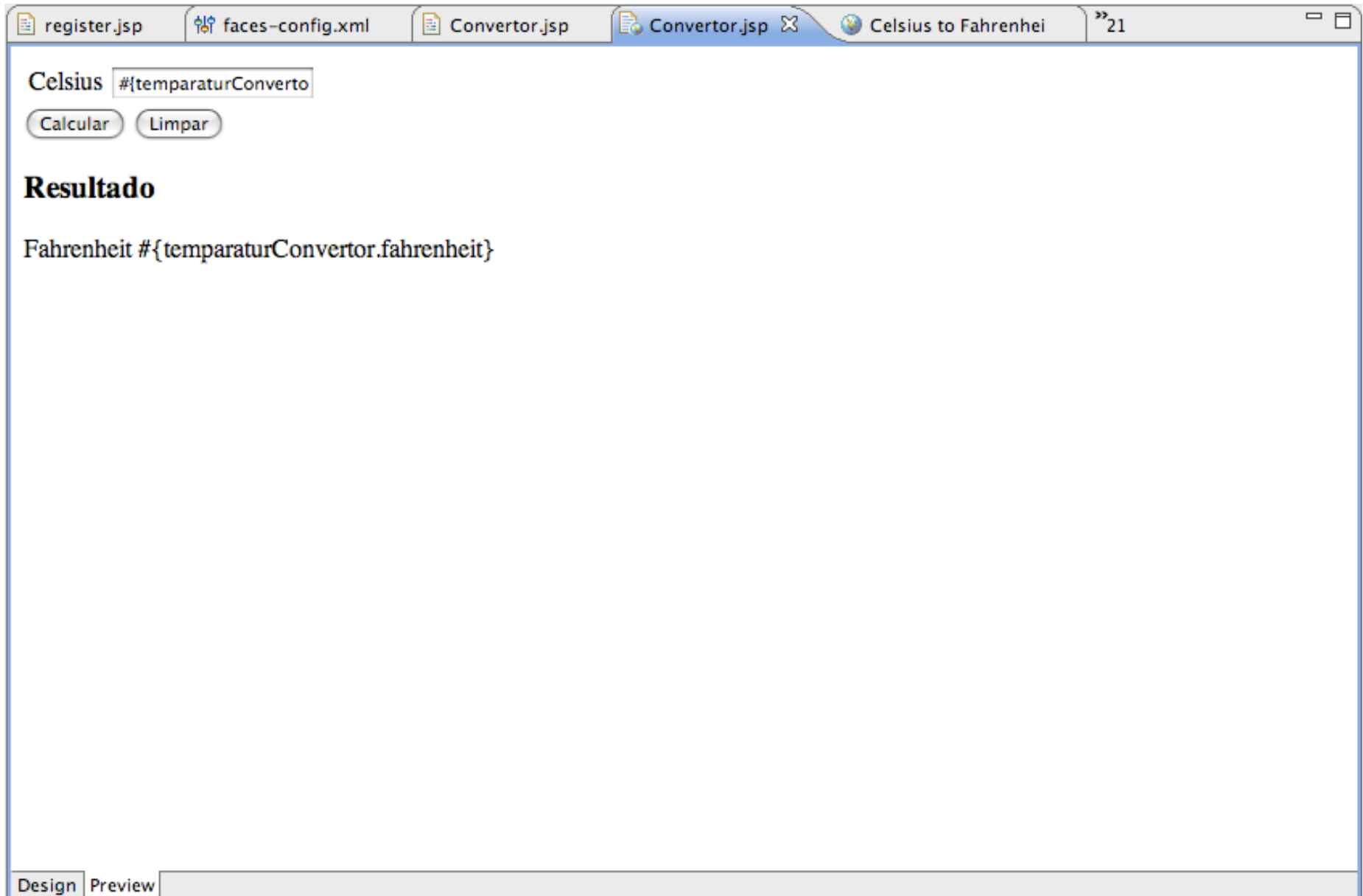
```
<body>
<f:view>
  <h:form>
    <h:panelGrid columns="2">
      <h:outputLabel value="Celsius"></h:outputLabel>
      <h:inputText value="#{temparaturConvertor.celsius}"></h:inputText>
    </h:panelGrid>
    <h:commandButton action="#{temparaturConvertor.celsiusToFahrenheit}" value="Calcular">
    </h:commandButton>
    <h:commandButton action="#{temparaturConvertor.reset}" value="Limpar"></h:commandButton>
    <h:messages layout="table"></h:messages>
  </h:form>

  <h:panelGroup rendered="#{temparaturConvertor.initial!=true}">
    <h3> Resultado </h3>
  </h:panelGroup>
</f:view>
</body>
```

Design Preview



# Editor de páginas JSF (2/2)



## Exemplo 2

Aplicação JSF com  
*Controller* e Validação

# Aplicação JSF com Controller e Validação

- Ilustração de uma aplicação JSF mais rica, contendo código de controle e validação
- Duas páginas JSF: login e formulário de multiplicação de números

The image displays two screenshots of a Java Server Faces (JSF) application running in a web browser.

The left screenshot shows a login page with the URL `http://localhost:8080/testejSF4/faces/Trainer.jsp`. It contains two input fields: "Usuário" (username) with the value "tester" and "Senha" (password) which is empty. Below the fields is a "Login" button.

The right screenshot shows a page titled "Exercite seu cérebro" (Exercise your brain). It contains the text "Tente adivinha o resultado da multiplicação dos números abaixo:" (Try to guess the result of the multiplication of the numbers below:). Below this, it displays "Primeiro Número: 6" (First Number: 6) and "Segundo Número: 93" (Second Number: 93). There is a "Resultado" (Result) field with the value "40". Below the result field are two buttons: "Mostrar Resultado" (Show Result) and "Novo Teste" (New Test). At the bottom, it says "Incorreto, que tal tentar novamente !!!" (Incorrect, how about trying again !!!).

# Criação da Funcionalidade de Login

- Necessidade de Criação de:
  - ManagedBean User associado
    - Para receber e validar os dados no servidor
  - Página JSF de Login
    - Com classe de validação do JSF associada
  - Configuração da Navegação entre as páginas
    - Arquivo faces-config.xml

# ManagedBean User.java

```
public class User {  
    private String name;  
    private String password;  
  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public String getPassword() {  
        return password;  
    }  
    public void setPassword(String password) {  
        this.password = password;  
    }  
  
    public String login(){  
        // Image here a database access to validate the users  
        if (name.equalsIgnoreCase("tester") && password.equalsIgnoreCase("tester")){  
            return "success";  
        } else {  
            return "failed";  
        }  
    }  
}
```

# Página JSF de Login

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/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">
<title>Login</title>
</head>
<body>
<f:view>
    <f:loadBundle basename="br.ufrn.dimap.card.starter.messages" var="msg" />
    <h:form>
        <h:panelGrid columns="2">
            <h:outputLabel value="#{msg.user}"></h:outputLabel>
            <h:inputText value="#{user.name}">
                <f:validator
                    validatorId="br.ufrn.dimap.card.starter.LoginValidator" />
            </h:inputText>
            <h:outputLabel value="#{msg.password}"></h:outputLabel>
            <h:inputSecret value="#{user.password}">
                </h:inputSecret>
        </h:panelGrid>
        <h:commandButton action="#{user.login}" value="#{msg.login}"></h:commandButton>
        <h:messages layout="table"></h:messages>
    </h:form>
</f:view>
</body>
</html>
```

# Classe de Validação do Login

```
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

public class LoginValidator implements Validator {
    public void validate(FacesContext context, UIComponent component,
        Object value) throws ValidatorException {

        String user = (String) value;

        if (!user.equalsIgnoreCase("tester")) {
            FacesMessage message = new FacesMessage();
            message.setDetail("User " + user + " does not exists");
            message.setSummary("Login Incorrect");
            message.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(message);
        }
    }
}
```

# Configuração da Validação do Login

The screenshot shows the NetBeans IDE with the 'faces-config.xml' file open. The 'Component' window is active, displaying the configuration for a 'Validator' component. The 'General' tab is selected, showing fields for 'Display Name', 'Description', 'Validator ID\*', and 'Validator Class\*'. The 'Attributes' tab is also visible, showing a table for defining attributes.

**Component**

- Components
- Converters
- Render Kits
- Validators
  - The following Validators are defined:
    - Validator
      - Add
      - Remove

**General**

This section describes general information of this Validator

Display Name: LoginValidator

Description: LoginValidator

Validator ID\*: br.ufrn.dimap.card.starter.LoginValidator

Validator Class\*: br.ufrn.dimap.card.starter.LoginValidator [Browse...](#)

**Attributes**

This section lists all the attributes.

Name	Class	Defa...Value	Sugg...Value	

[Add...](#)  
[Edit...](#)  
[Remove](#)

Introduction Overview Navigation Rule ManagedBean **Component** Others Source



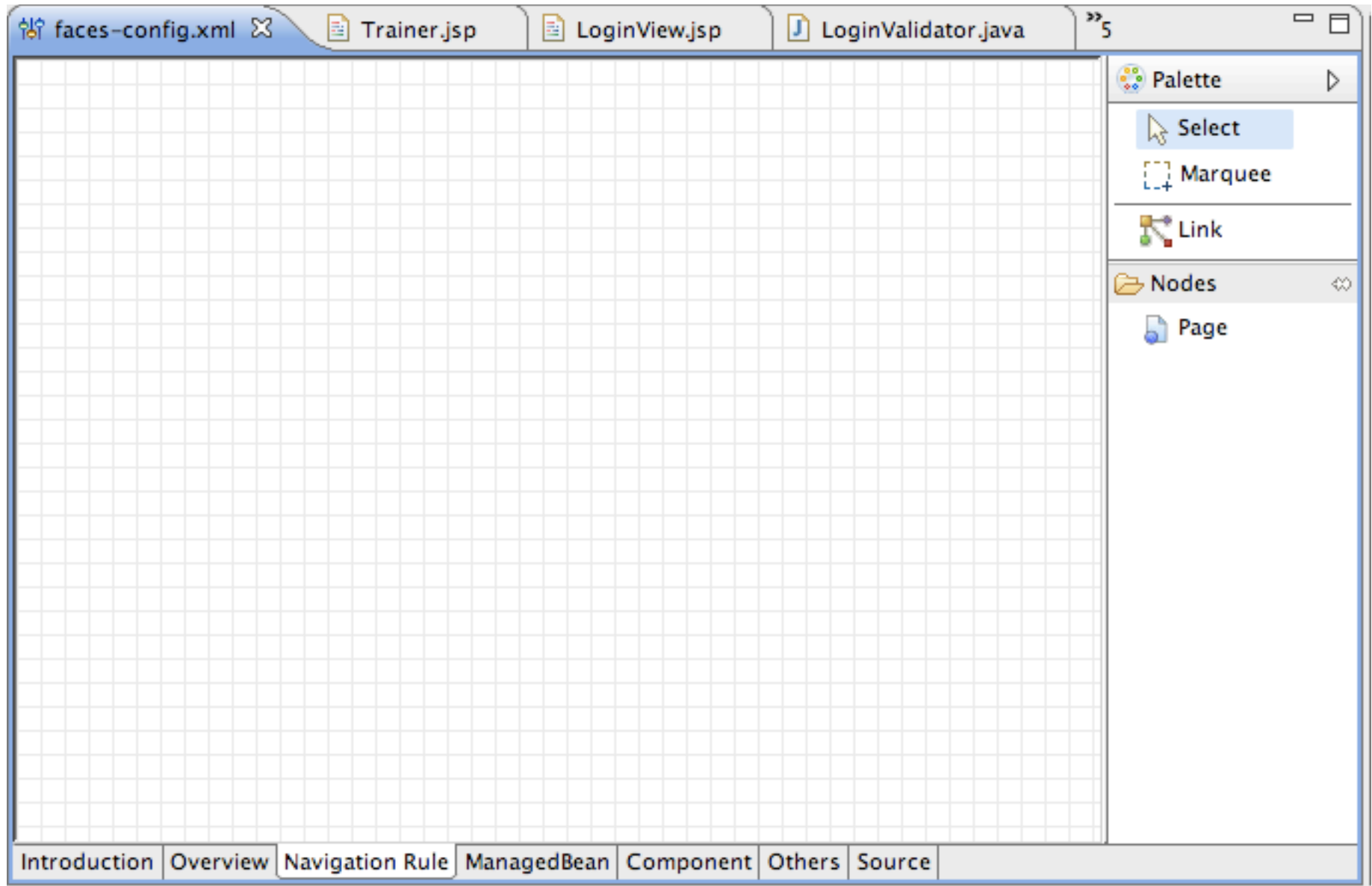
# Validator no faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
  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/web-facesconfig_1_2.xsd"
  version="1.2">
  <managed-bean>
    <managed-bean-name>cardController</managed-bean-name>
    <managed-bean-class>br.ufrn.dimap.controller.CardController</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>card</property-name>
      <property-class>br.ufrn.dimap.card.model.Card</property-class>
      <value>#{card}</value>
    </managed-property>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>card</managed-bean-name>
    <managed-bean-class>br.ufrn.dimap.card.model.Card</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
  </managed-bean>
  <managed-bean>
    <managed-bean-name>user</managed-bean-name>
    <managed-bean-class>br.ufrn.dimap.card.model.User</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
  <validator>
    <description>LoginValidator</description>
    <display-name>LoginValidator</display-name>
    <validator-id>br.ufrn.dimap.card.starter.LoginValidator</validator-id>
    <validator-class>br.ufrn.dimap.card.starter.LoginValidator</validator-class>
  </validator>
</faces-config>
```

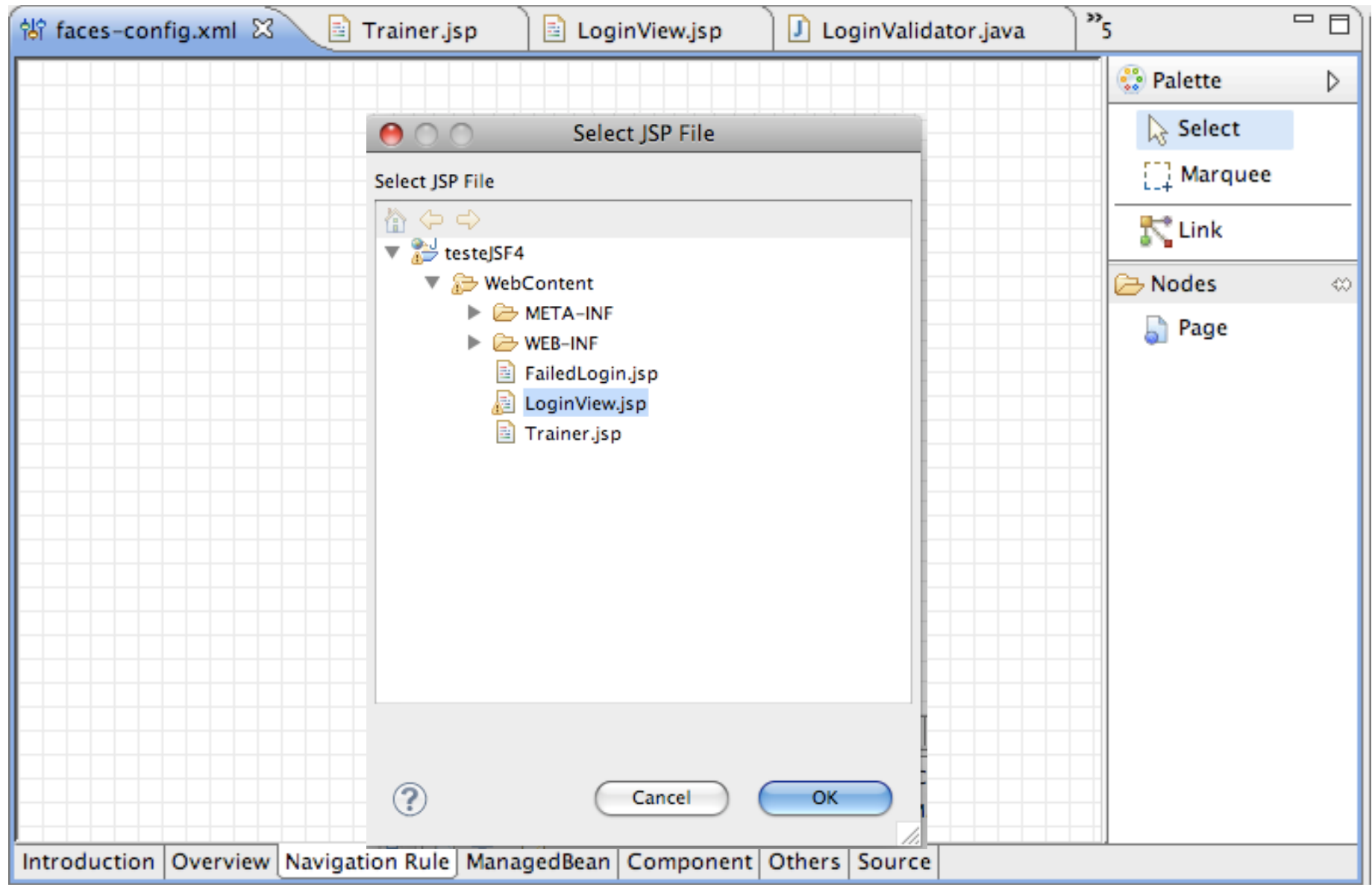
# Configuração da Navegação entre as Páginas

- Editor do faces-config.xml permite a edição da navegação entre páginas JSF
- Informações são automaticamente atualizadas no arquivo de configuração

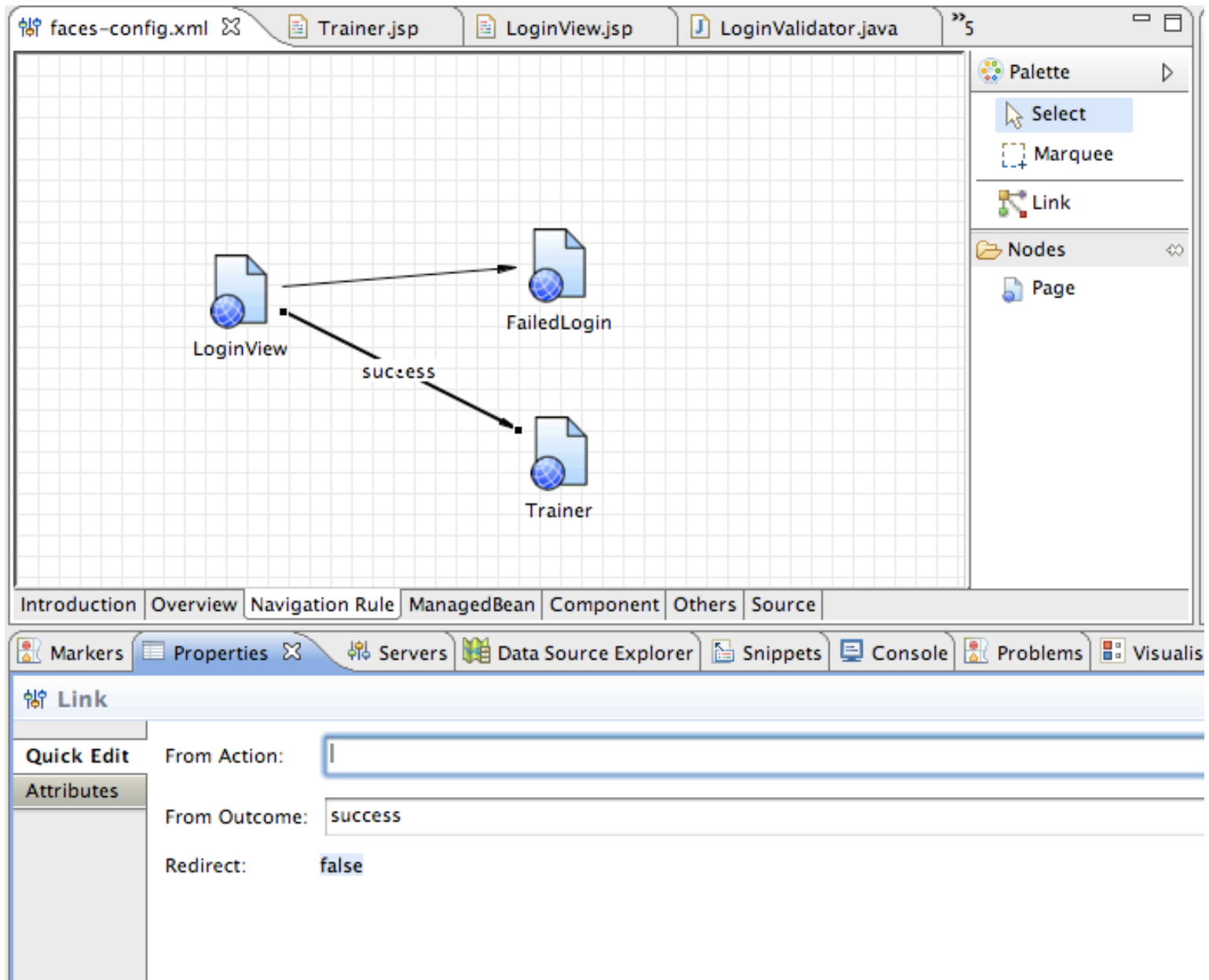
# Configuração da Navegação entre as Páginas



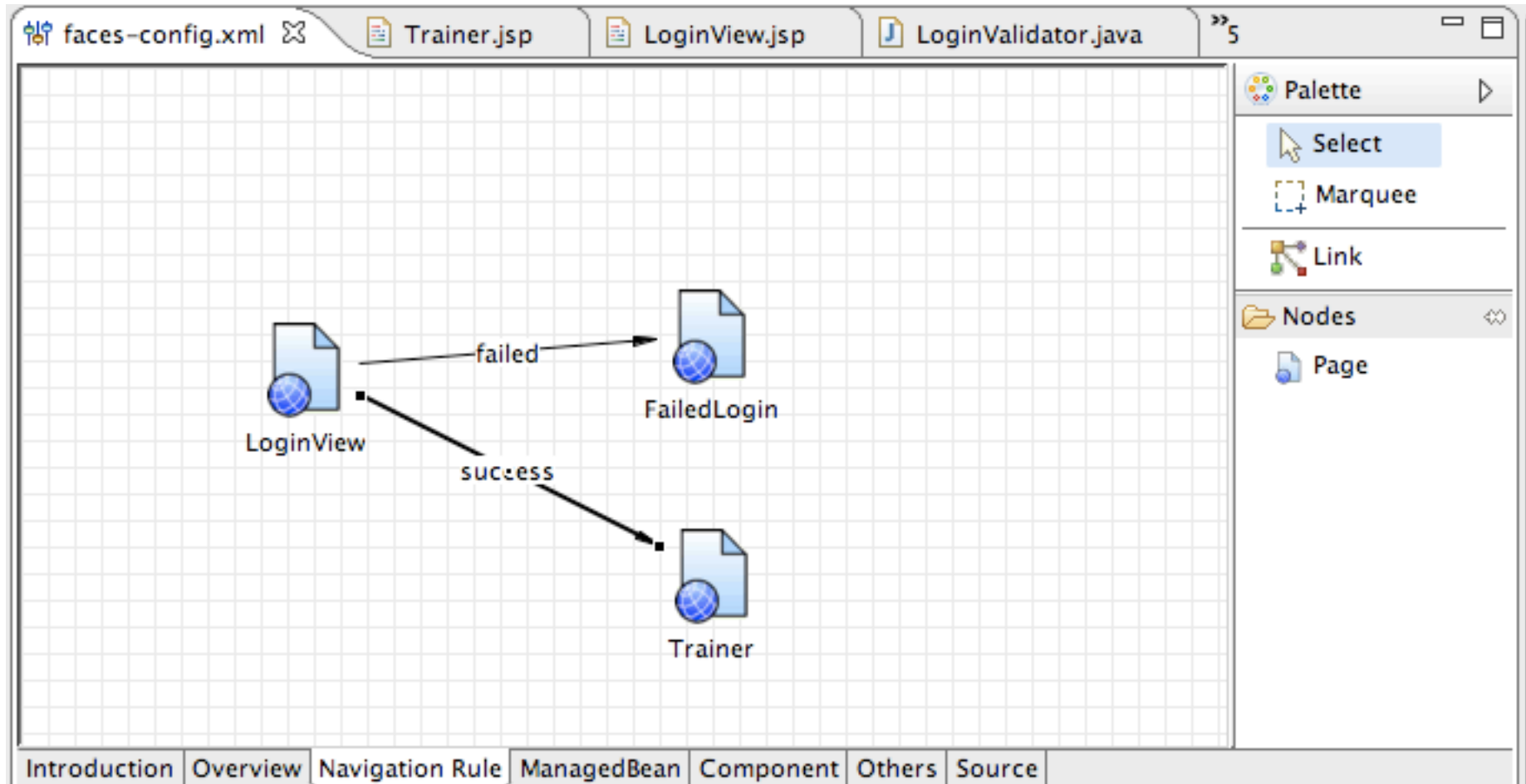
# Configuração da Navegação entre as Páginas



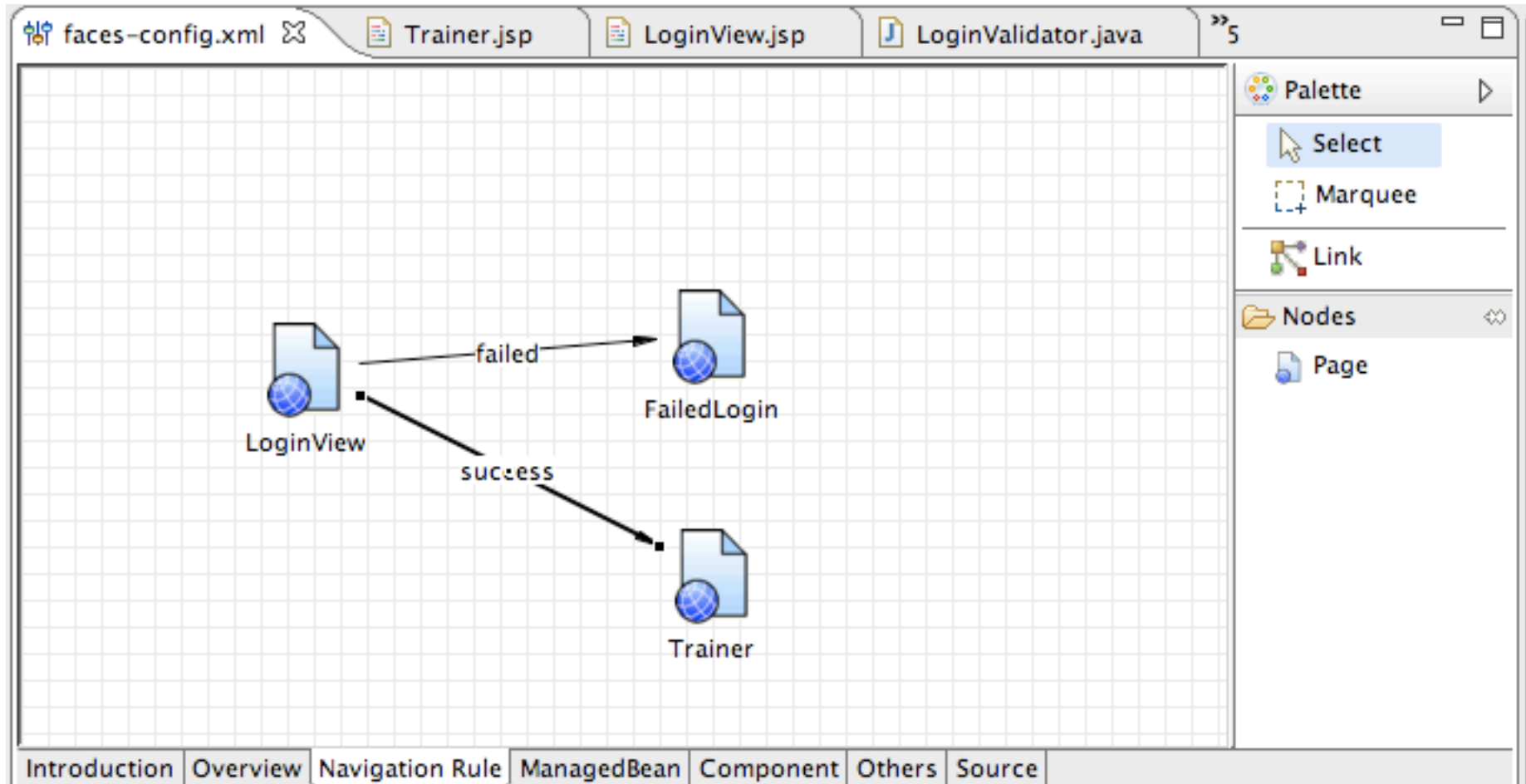
# Configuração da Navegação entre as Páginas



# Configuração da Navegação entre as Páginas



# Configuração da Navegação entre as Páginas



# Validator no faces-config.xml

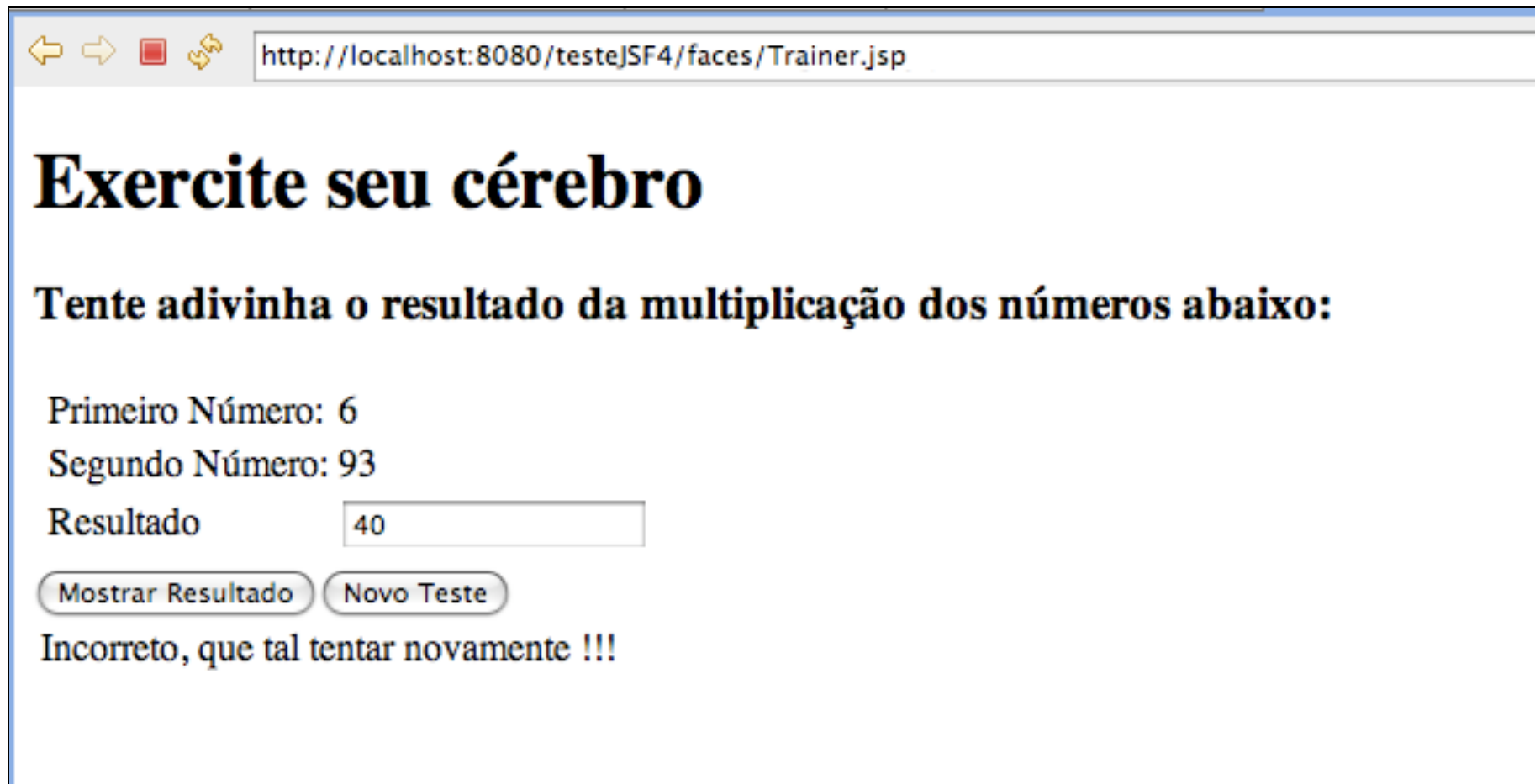
```
<managed-bean>
  <managed-bean-name>user</managed-bean-name>
  <managed-bean-class>br.ufrn.dimap.card.model.User</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
<navigation-rule>
  <display-name>LoginView</display-name>
  <from-view-id>/LoginView.jsp</from-view-id>
  <navigation-case>
    <from-outcome>failed</from-outcome>
    <to-view-id>/FailedLogin.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <display-name>LoginView</display-name>
  <from-view-id>/LoginView.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/Trainer.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
<validator>
  <description>LoginValidator</description>
  <display-name>LoginValidator</display-name>
  <validator-id>br.ufrn.dimap.card.starter.LoginValidator</validator-id>
  <validator-class>br.ufrn.dimap.card.starter.LoginValidator</validator-class>
</validator>
</faces-config>
```



# Criação do Avaliador Matemático

- Necessidade de Criação de:
  - ManagedBean Card associado
    - Para receber e validar os dados no servidor
  - Classe Controlador para tratar manipulação de dados
  - Página JSF do Avaliador

# Tela Gráfica do Avaliador Matemático



A screenshot of a web browser window displaying a math quiz interface. The browser's address bar shows the URL `http://localhost:8080/testeJSF4/faces/Trainer.jsp`. The page content includes a main heading, a challenge prompt, two given numbers, a result input field with the value 40, two buttons for 'Mostrar Resultado' and 'Novo Teste', and a feedback message indicating the answer is incorrect.

**Exercite seu cérebro**

**Tente adivinha o resultado da multiplicação dos números abaixo:**

Primeiro Número: 6  
Segundo Número: 93

Resultado

Incorreto, que tal tentar novamente !!!

# ManagedBean Card.java

```
import java.util.Random;

public class Card {
    private int left;
    private int right;

    public Card() {
        Random random = new Random();
        int i = 0;
        int j = 0;
        do {
            i = random.nextInt(10);
        } while (i <= 4);

        do {
            j = random.nextInt(100);
        } while (j <= 20);

        left = i;
        right = j;
    }

    public int getLeft() {
        return left;
    }
    public void setLeft(int left) {
        this.left = left;
    }
    public int getRight() {
        return right;
    }
    public void setRight(int right) {
        this.right = right;
    }
}
```

```
import javax.faces.application.FacesMessage;
import javax.faces.component.UIPanel;
import javax.faces.context.FacesContext;

import br.ufrn.dimap.card.model.Card;

public class CardController {
    private Card card;
    private UIPanel resultPanel;
    private int result;

    public CardController() {
    }

    public String checkResult() {
        FacesContext context = FacesContext.getCurrentInstance();
        resultPanel.setRendered(true);

        if (checkOperation()) {
            context.addMessage(null, new FacesMessage(
                FacesMessage.SEVERITY_INFO, "Parabéns, você acertou !!!", null));
        } else {
            context.addMessage(null, new FacesMessage(
                FacesMessage.SEVERITY_INFO, "Incorreto, que tal tentar novamente !!!", null));
        }
        return null;
    }

    private boolean checkOperation() {
        return (card.getLeft() * card.getRight() == result);
    }

    public UIPanel getResultPanel() {
        return resultPanel;
    }
}
```

```
public void setResultPanel(UIPanel resultPanel) {
    this.resultPanel = resultPanel;
}

public int getResult() {
    return result;
}

public void setResult(int result) {
    this.result = result;
}

public String next() {
    FacesContext context = FacesContext.getCurrentInstance();
    if (checkOperation()){
        resultPanel.setRendered(false);
        card = new Card();
        return null;
    } else {
        context.addMessage(null, new FacesMessage(
            FacesMessage.SEVERITY_INFO, "Incorrect", null));
    }
    return null;
}

public Card getCard() {
    return card;
}

public void setCard(Card card) {
    this.card = card;
}
}
```

# Registro dos ManagedBeans

- Cada um dos ManagedBeans deve ser registrado no arquivo faces-config.xml
- O editor visual do Eclipse WTP pode ser usado com tal finalidade
- Necessário configurar o Card para ser injetado automaticamente no CardController

# Criação do Avaliador Matemático

- Necessidade de Criação de:
  - ManagedBean Card associado
    - Para receber e validar os dados no servidor
  - Classe Controlador para tratar manipulação de dados
  - Página JSF do Avaliador

# Criação da Página JSF do Avaliador

```
<body>
<h1> Exercite seu cérebro</h1>
<h3>Tente adivinha o resultado da multiplicação dos números abaixo: </h3>
<f:view>
  <f:loadBundle basename="br.ufrn.dimap.card.starter.messages" var="msg" />
  <h:form>
    <h:panelGrid columns="3">
      <h:panelGrid columns="2">
        <h:outputLabel value="#{msg.left}"></h:outputLabel>
        <h:outputLabel id="left" value="#{cardController.card.left}"></h:outputLabel>

        <h:outputLabel value="#{msg.right}"></h:outputLabel>
        <h:outputLabel id="right" value="#{cardController.card.right}">
</h:outputLabel>

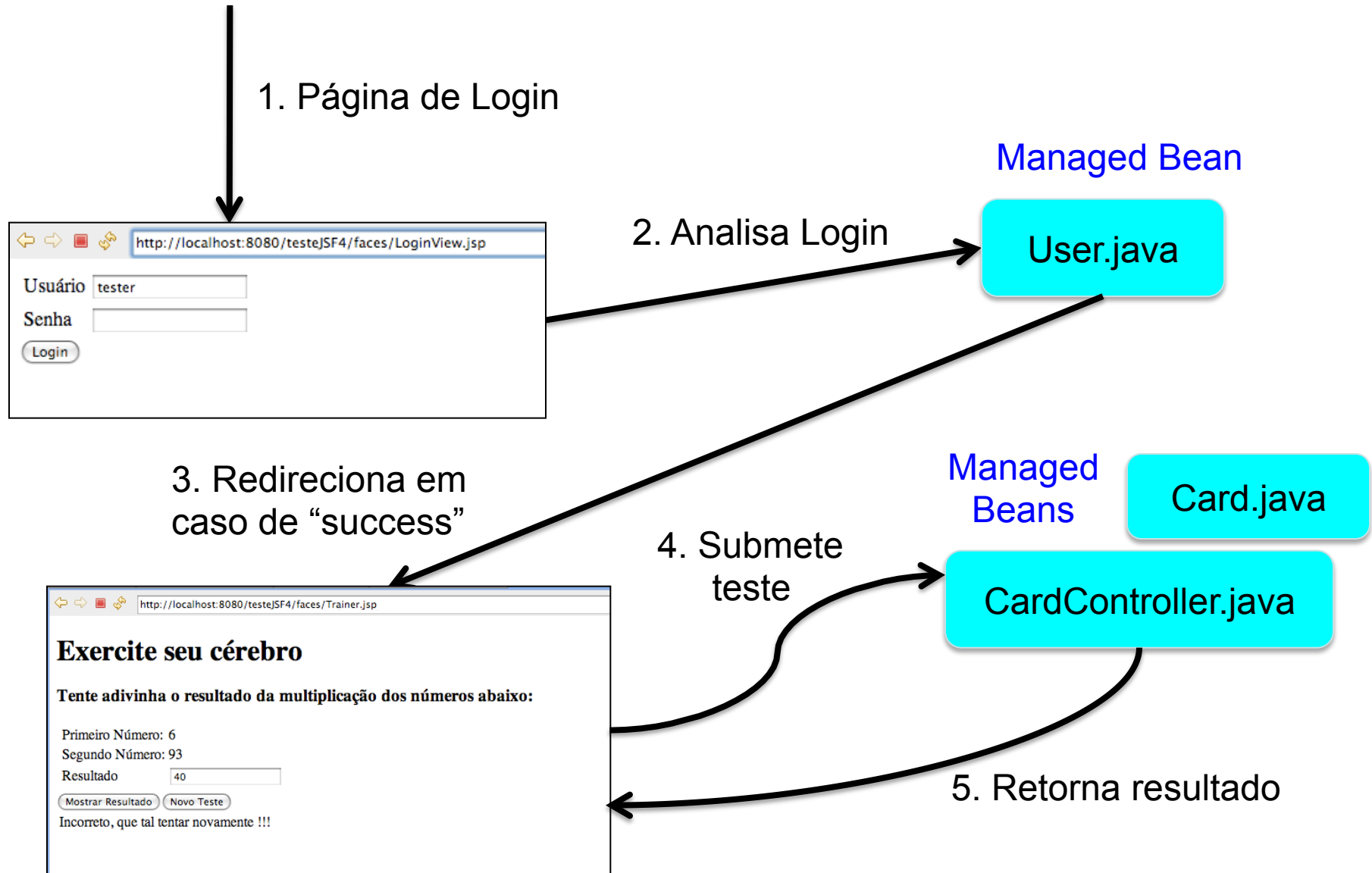
        <h:outputLabel value="#{msg.result}"></h:outputLabel>
        <h:inputText id="result" value="#{cardController.result}"></h:inputText>
      </h:panelGrid>

    </h:panelGrid>
    <h:commandButton action="#{cardController.checkResult}"
      value="#{msg.show}"></h:commandButton>
    <h:commandButton action="#{cardController.next}" value="#{msg.next}" type="submit"></h:commandButton>
    <h:messages layout="table"></h:messages>

    <h:panelGroup binding="#{cardController.resultPanel}" rendered="false">
      <h:message for="result"></h:message>
    </h:panelGroup>
  </h:form>
</f:view>
</body>
```



# Resumindo



# Exercício

- Implementar um Crud com 3 camadas (GUI, Negócio, Dados) que possam ser acessadas pela camada de GUI via JSF e Servlets

# Referências

- Tutorial JEE da Sun
- Tutorial para uso de JSF 1.2 no Eclipse  
<http://www.vogella.com/tutorials/JavaServerFaces/article.html>
- Tutorial para uso de JSF 2  
<http://www.coreservlets.com/JSF-Tutorial/jsf2/>