FIAP GRADUAÇÃO

DIGITAL BUSINESS ENABLEMENT

Prof. MSc. Rafael Matsuyama

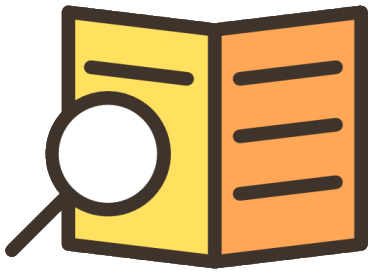
#07 – JSF INTRODUÇÃO

 *profrafael.matsuyama@fiap.com.br*  *rafaelmatsuyama*

PERCURSO

- Java Application
- Padrões de Projetos e Frameworks
- SOA e Web Services
- Web Services SOAP
- Web Services Restful
- JSF - Introdução

#07 - AGENDA



- Tipos de Frameworks Web
- Conceitos do JSF
- Criando e configurando um projeto JSF
- Componentes Visuais
- Managed Bean
- Integração com web services e JPA
- JSF com javascript e css

FRAMEWORKS WEB - TIPOS

Request-based

- Trabalha perto do **protocolo HTTP;**
- Dificuldade no reuso de código de interface;
- Maior liberdade na construção das páginas web;
- Exemplos: **Struts, Spring e WebWork;**

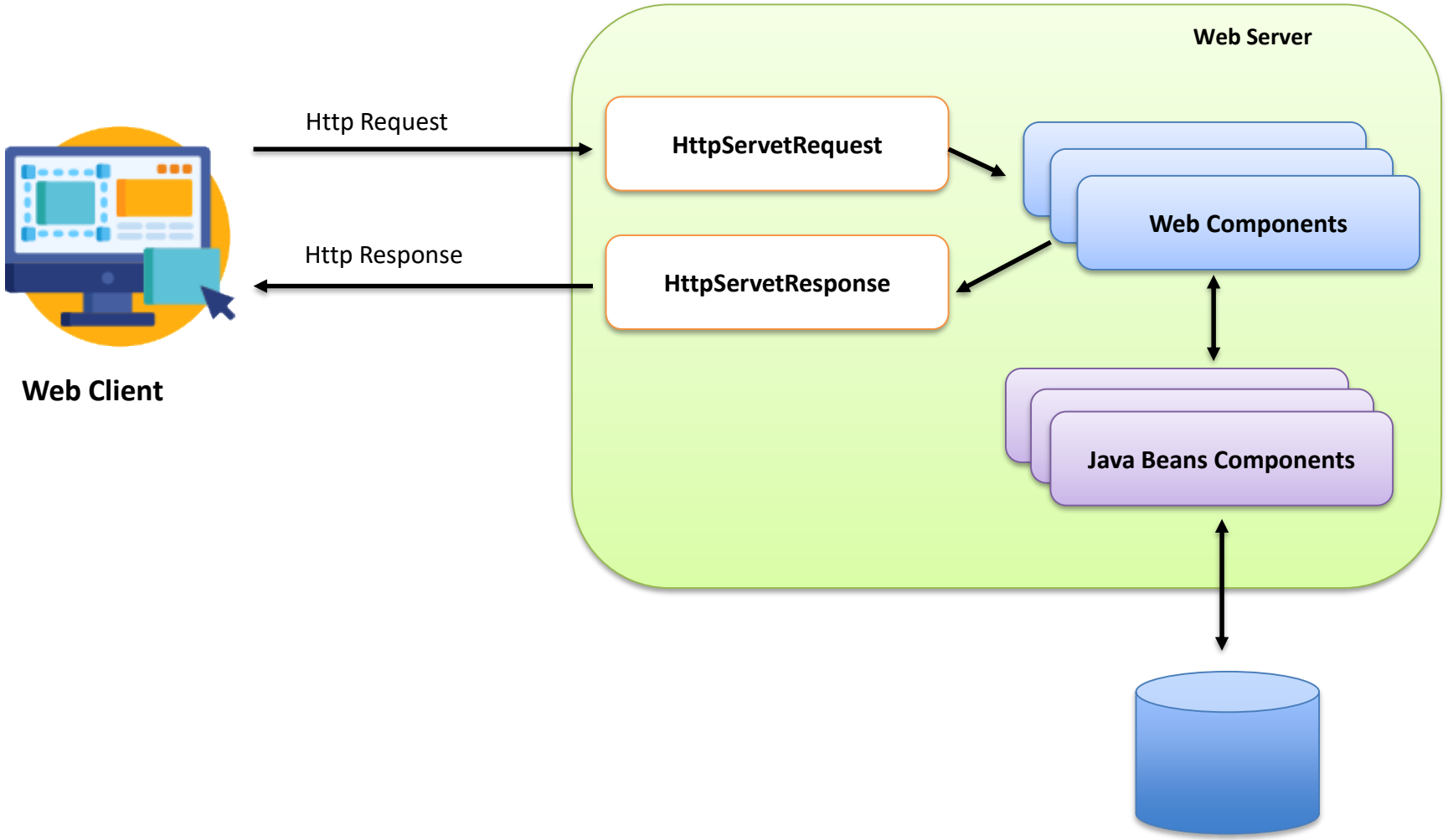
Component-based

- Modelo de desenvolvimento **parecido com desktop;**
- **Gera componentes** a partir da necessidade da tela;
- Crescimento acelerado como padrão de interface Web;
- Exemplos: **JSF, Tapestry e GWT;**

JSF – JAVA SERVER FACES

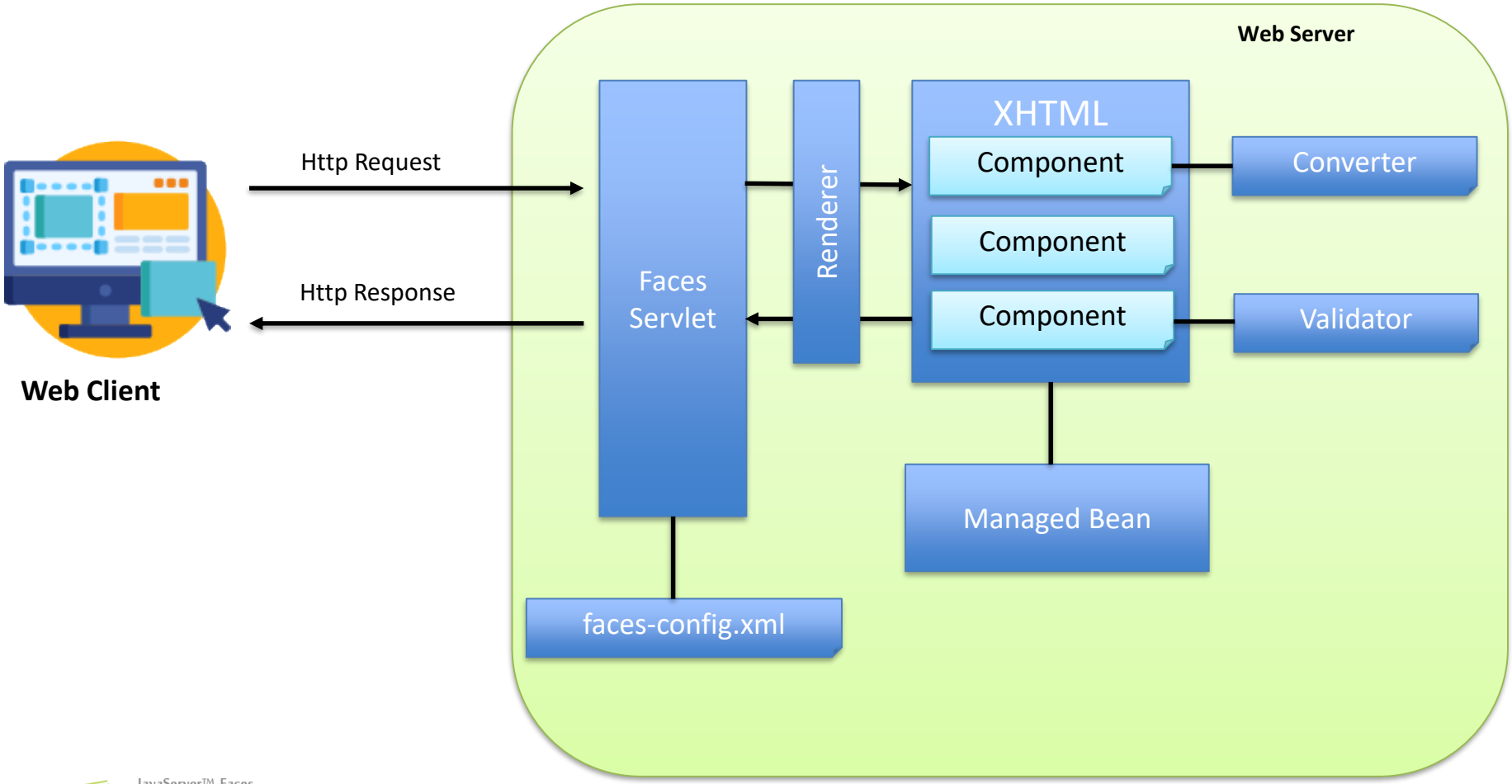
JSF

- A **especificação Java** define como **JSF deve ser implementado;**
- Permite criar aplicações Java Web através de **componentes visuais pré-prontos;**
- Não é preciso se preocupar com **Javascript e HTML;**
- Utiliza a arquitetura **MVC**;
- Um dos frameworks Java web mais utilizados no mercado;
- Produtividade;
- Implementações mais conhecidas:
  - **Oracle Mojarra Java Server Faces - Reference Implementation**
  - Apache MyFaces
  - IBM Faces



Referência: *http://docs.oracle.com/javaee/6/tutorial/doc*

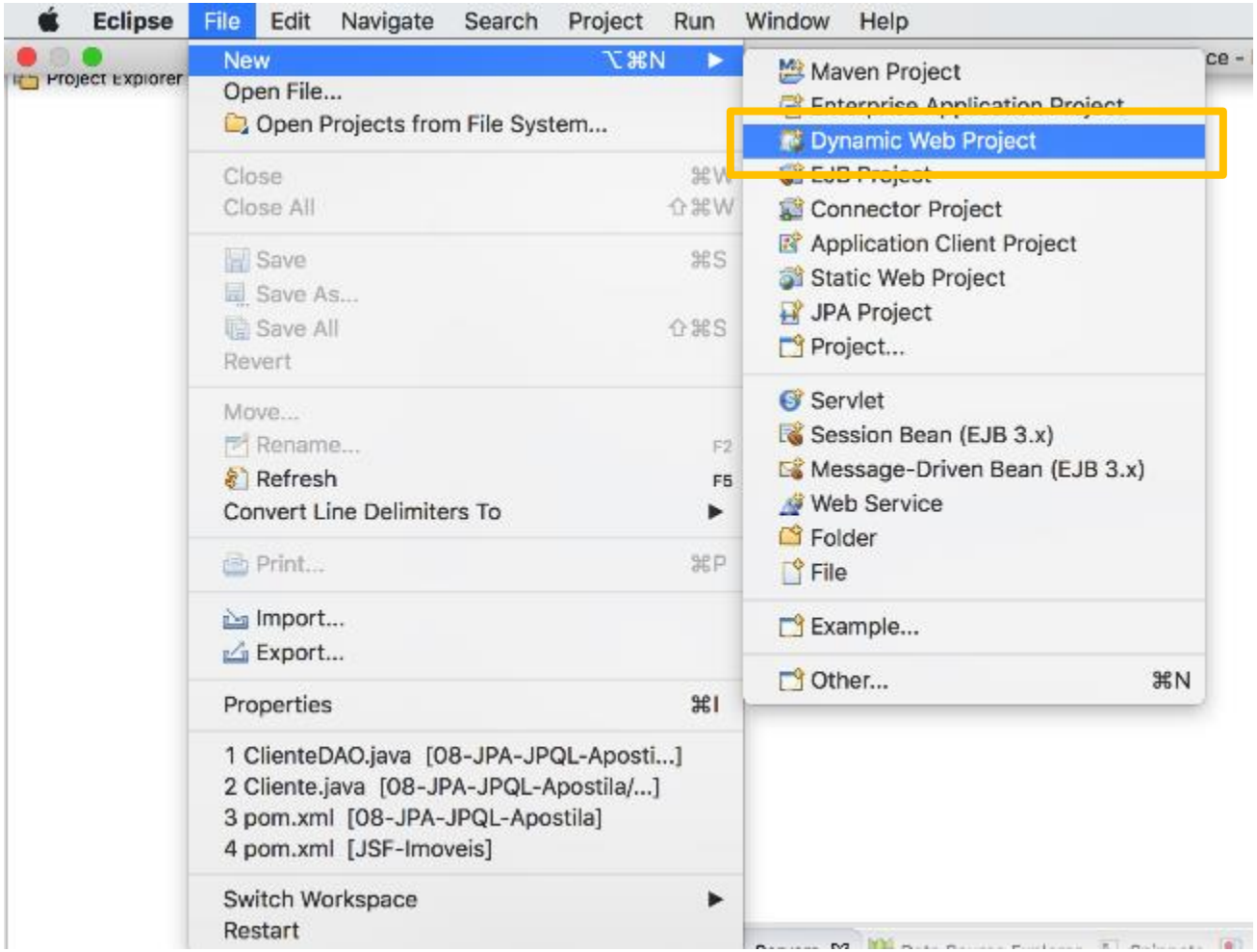
JSF - ARQUITETURA



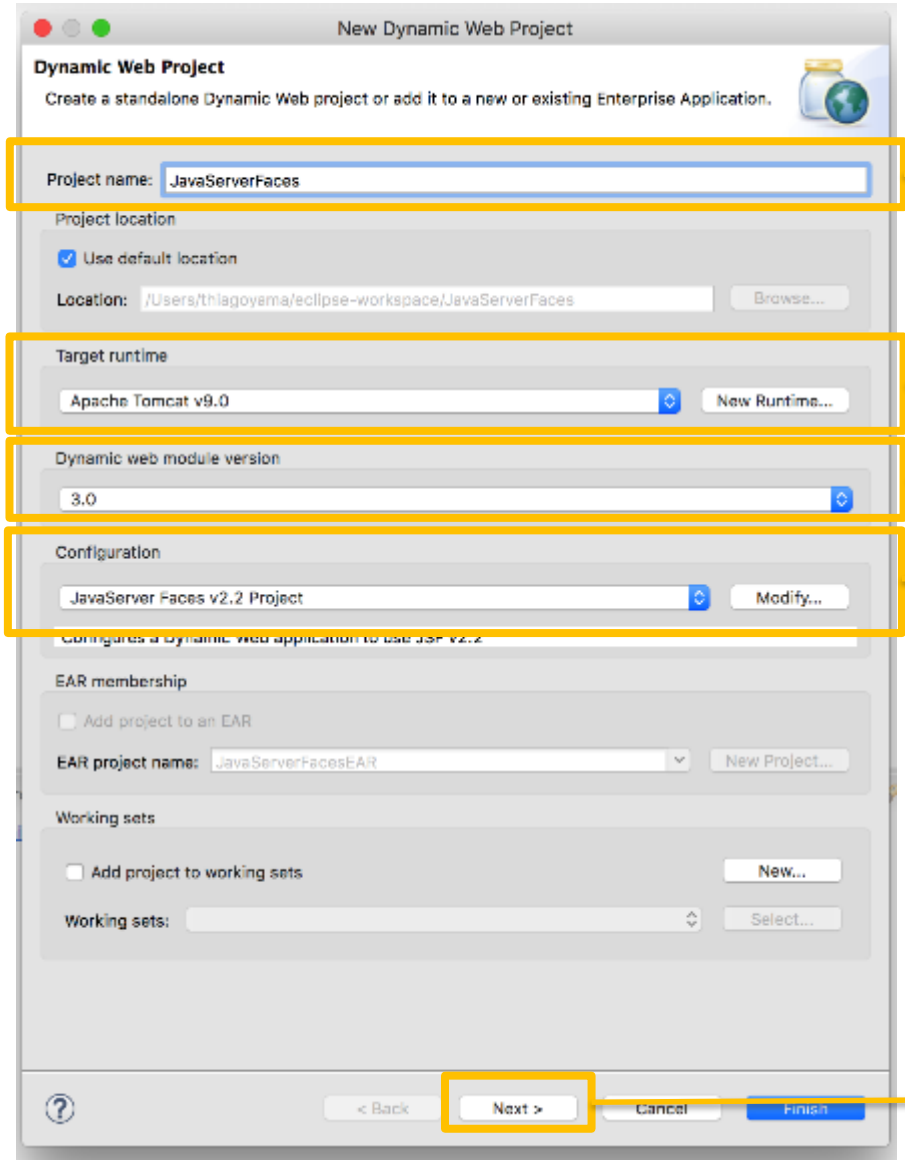
CRIANDO UM PROJETO JSF

DYNAMIC WEB PROJECT

- Crie um **Dynamic Web Project;**



PROJECT - CONFIGURAÇÃO



Configure o nome do projeto

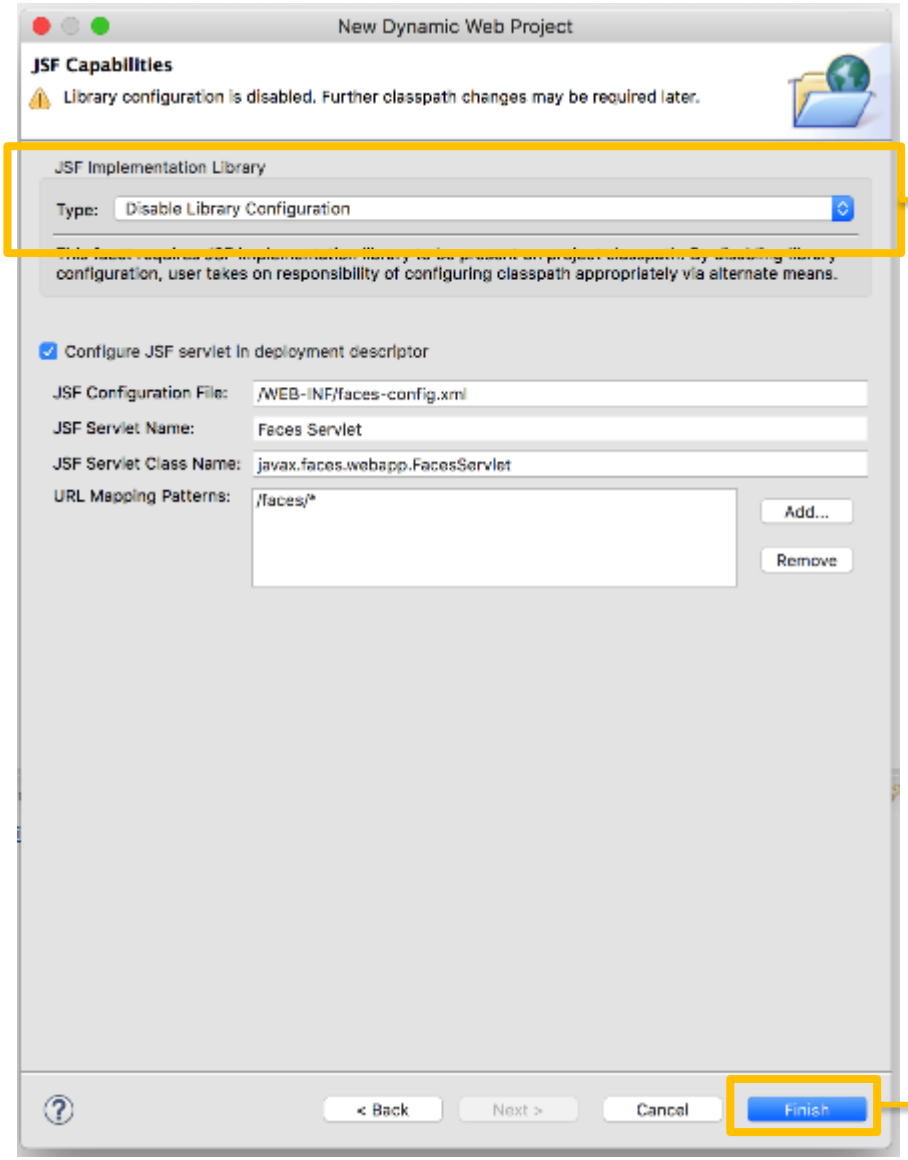
Configure o servidor **Tomcat 9**

Escolha a versão **3.0**

Escolha a opção **JavaServer Faces**

Clique em Next

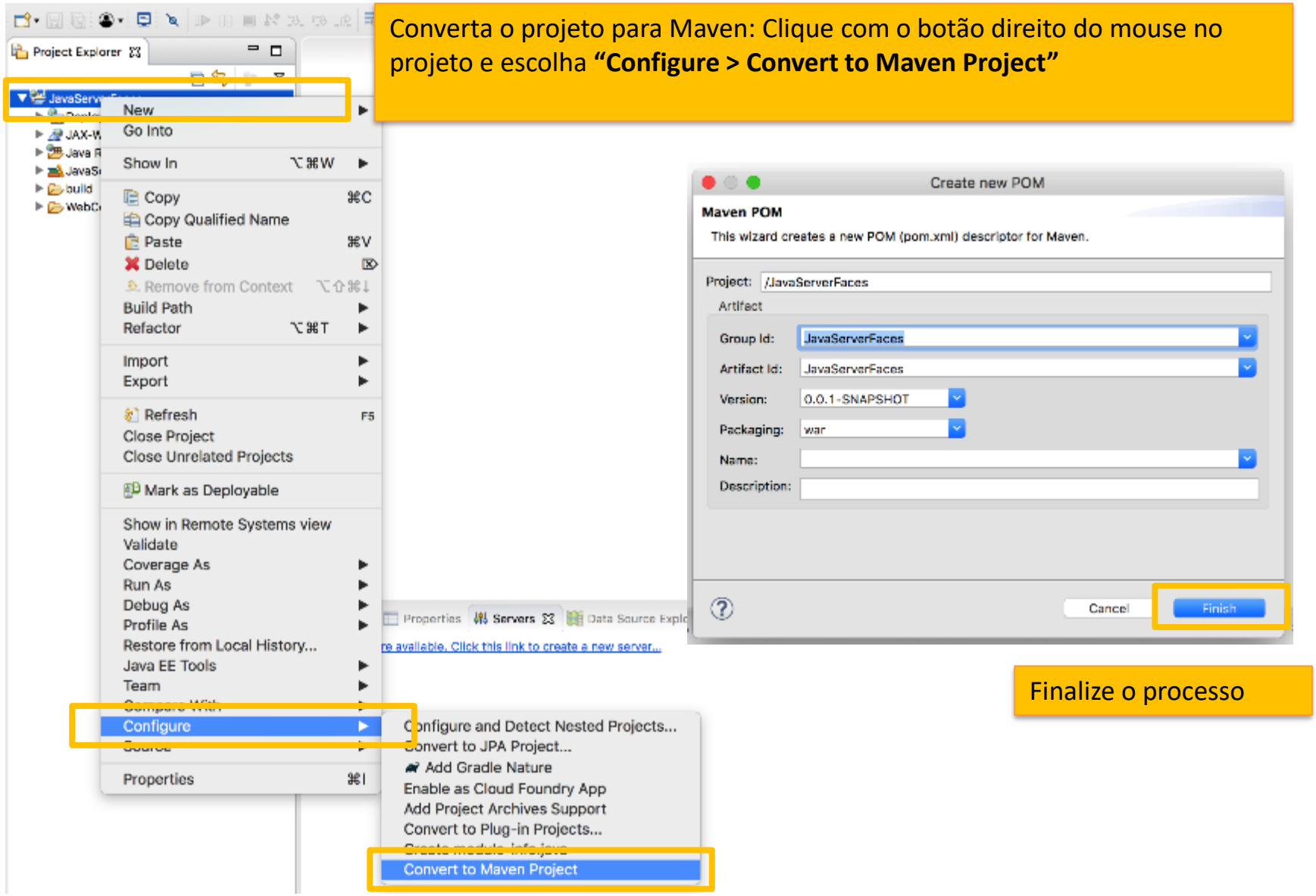
PROJETO - BIBLIOTECAS



Escolha a opção:
"**Disable Library Configuration**", para fornecer as bibliotecas do JSF posteriormente;

Finalize o processo.

PROJETO – CONVERTENDO MAVEN



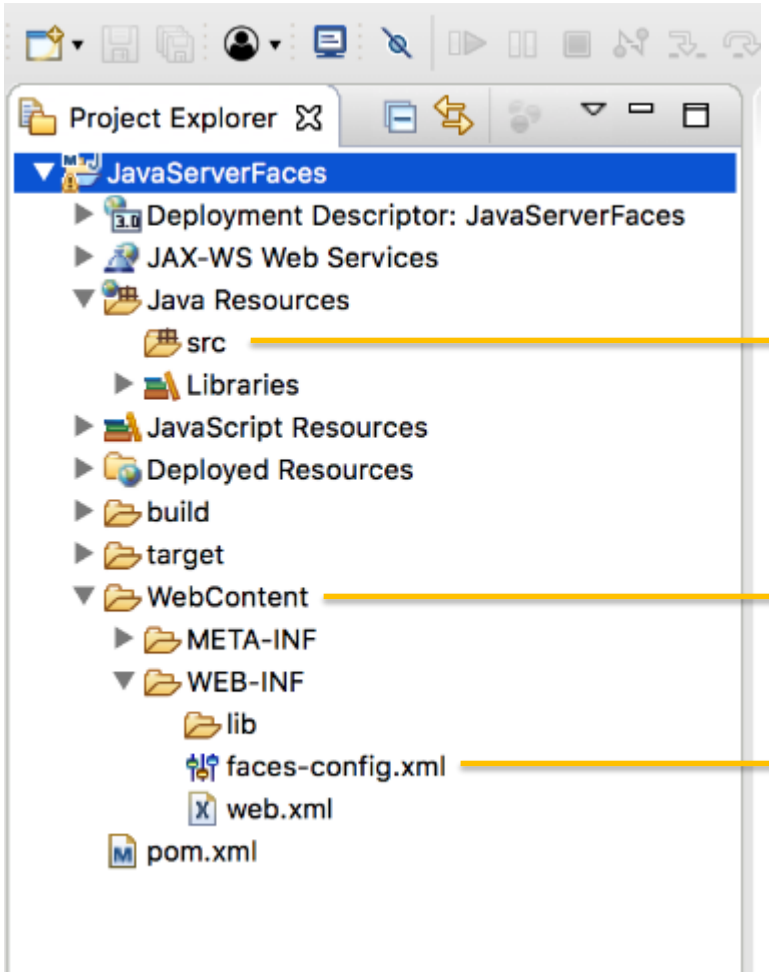
POM.XML - JSF

- Adicione as dependências no arquivo **pom.xml**

```
<dependency>
   <groupId>com.sun.faces</groupId>
   <artifactId>jsf-api</artifactId>
   <version>2.2.14</version>
</dependency>

<dependency>
   <groupId>com.sun.faces</groupId>
   <artifactId>jsf-impl</artifactId>
   <version>2.2.14</version>
</dependency>
```

JSF – ESTRUTURA DO PROJETO

- Estrutura do projeto:



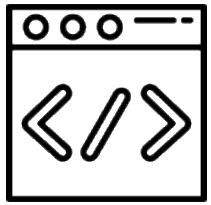
Código Java

Arquivos **Web** (Html, Css, Javascript e etc..)

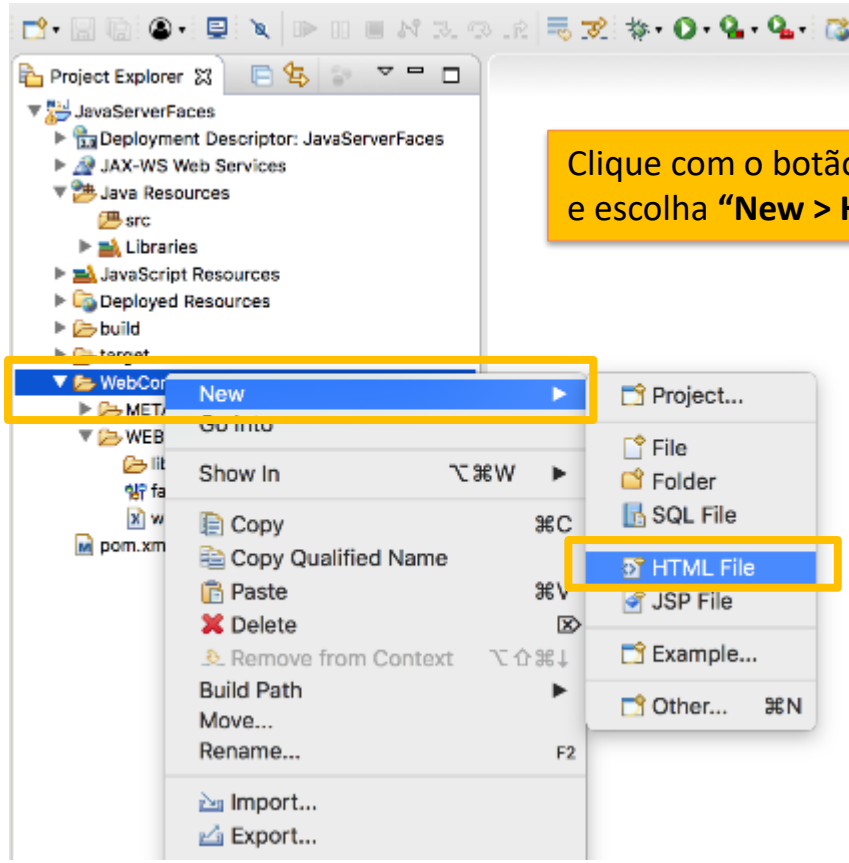
Arquivo de configuração do JSF

PÁGINA JSF

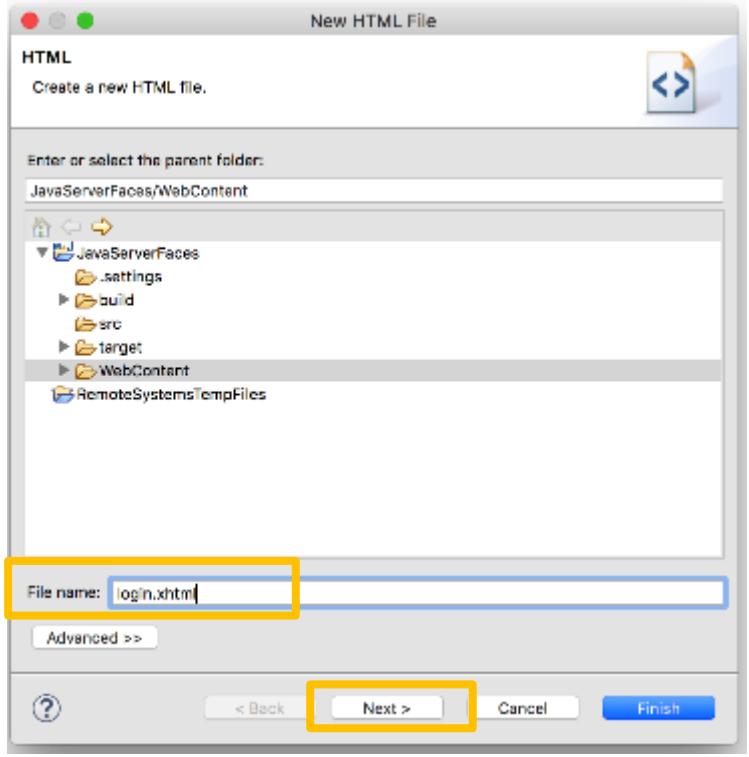
- Recomenda-se utilizar o nome da página **sempre com minúsculo,** exemplos **cadastro.xhtml, reposicao-estoque.xhtml;**
- **Não utilizar acentos** ou espaços em branco em nomes de páginas;
- As páginas comunicam-se com as classes Java por meio de **Expression Language (EL);**
- **EL** são formadas por **#{nome-managed-bean.nome-propriedade};**



JSF – CRIANDO UMA PÁGINA



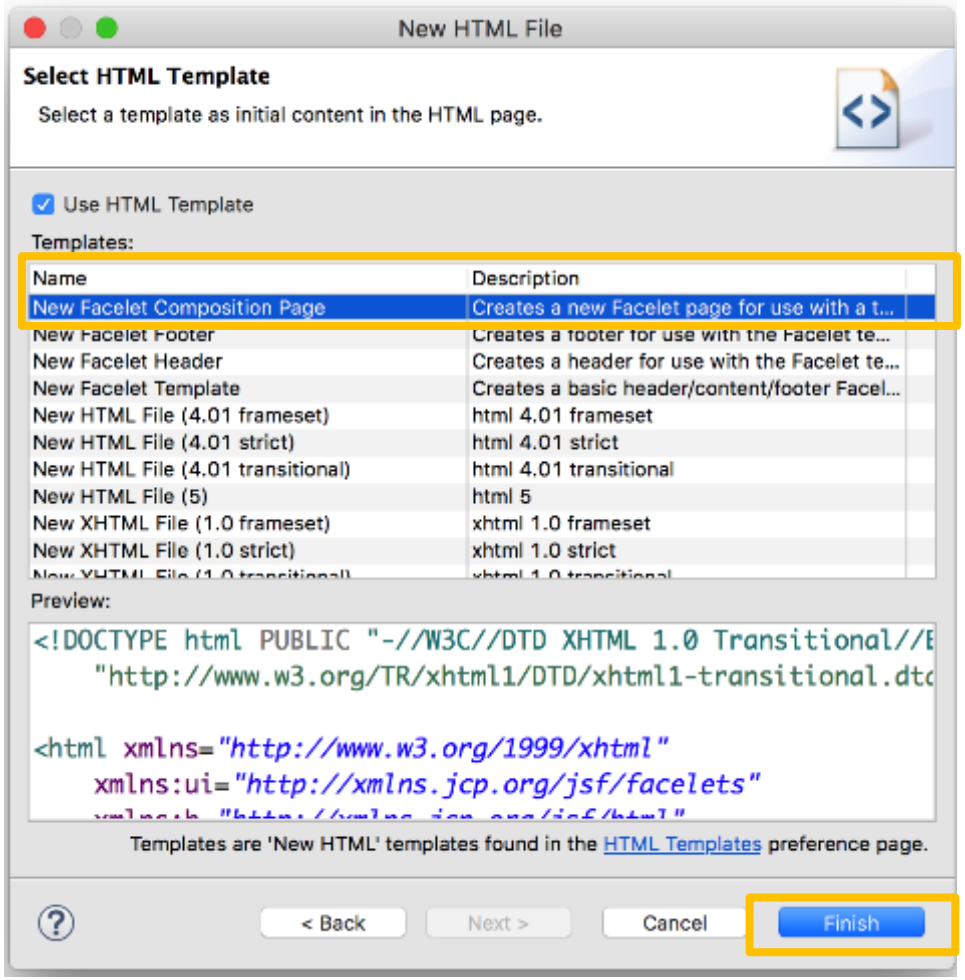
Clique com o botão direito do mouse na pasta WebContent
e escolha **"New > HTML File"**



Configure o nome da página com a **extensão .xhtml** e clique em **Next**

JSF – DEFININDO UM TEMPLATE

- Neste passo definimos um **código inicial** para a página (template);



Escolha o template **"New Facelet Composition Page"** e finalize o processo.

PÁGINA JSF - EXEMPLO

```
<!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:ui="http://xmlns.jcp.org/jsf/facelets"
xmlns:h="http://xmlns.jcp.org/jsf/html"
xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>
            <title>Login - JSF</title>
</h:head>
<h:body>
            <h:form>
               <h:outputLabel value="Login"/>
               <h:inputText value="#{loginBean.login}"/>
               <h:commandButton action="#{loginBean.logar}" value="Entrar"/>
            </h:form>
</h:body>
</html>
```

MANAGED BEAN

- Captura as **ações e valores** preenchidos na página web;
- Usar **padrão de nomenclatura** de classe **Java** para criar **Managed Bean;**
- É recomendando utilizar o sufixo **Bean** no final do nome da classe, exemplos: **LoginBean** e **ClienteBean**;
- A classe deve ser anotada com **@ManagedBean** do pacote **javax.faces.bean.ManagedBean**

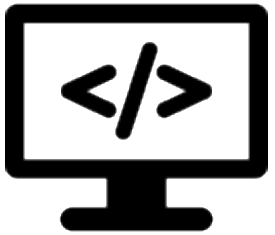
MANAGED BEAN - EXEMPLO

```
import javax.faces.bean.ManagedBean;


@ManagedBean
public class LoginBean {

  private String login;
  private String senha;


  public  void logar() {

    if (“rafael".equals(getLogin()) && "password".equals(getSenha()))
       System.out.println("Usuario logado: " + getLogin());
    else
       System.out.println("Usuario com senha incorreta");

  }

  // gets e sets

}
```

COMPONENTES VISUAIS

COMPONENTES VISUAIS JSF

- No **JSF** o componente responsável por gerar o **HTML final** chama-se **RenderKit**;
- No desenvolvimento de formulários é possível misturar **objetos de diferentes bibliotecas.** Estas bibliotecas ficam organizadas em subgrupos, também conhecidas como **taglibs**;
- As **2 taglibs** básicas no desenvolvimento com JSF são:
  - **JSF Core**
  - **JSF HTML**
- Os objetos em JSF possuem uma **uniformidade** no modo de interação com o **Managed Bean**;



COMPONENTES VISUAIS JSF

- Componentes visuais do JSF são elementos como **inputText, commandButton, outputText, dataTable** utilizados para criar **interface com o usuário;**
- Existem **2 categorias** de componentes JSF:
  - **Simple Components (Componentes Simples) -** são componentes individuais, que não precisam estar em conjunto com outros componentes para funcionar, por exemplo: **inputText, outputText e commandButton;**
  - **Compound Components (Componentes Compostos) -** são componentes compostos por outros componentes dentro dele, como por exemplo uma tabela **(dataTable),** que também é composto colunas;

CAIXAS DE TEXTO

- **<h:inputText> -** Campo de **texto;**
- **<h:inputTextarea> -** Campo de **área de texto;**
- **<h:inputSecret> -** Campo de **senha**;

XHTML

```
<h:inputText value="#{loginBean.user}"/>
```

```
@ManagedBean
public class LoginBean {

  private String user;

  //get e set
}
```

Java

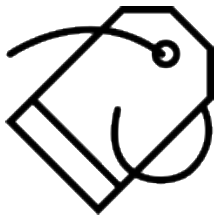
Na página XHTML, a classe é referenciada com a **primeira letra em minúscula**, na **expression language #{}**

LABEL

- **<h:outputLabel> -** Cria um elemento de **label**;

XHTML

```
<h:outputLabel value="Login" for="user"/>

<h:inputText value="#{loginBean.user}" id="user"/>
```



O **for** e o **id** devem possuir o mesmo valor para que o campo e o label possam ser "ligados"

BOTÕES

- **<h:commandButton>** - define um botão;
- **<h:commandLink>** - define um link;
- Acionam um **método** no **Managed Bean**, enviando os dados de um **formulário HTML** para o servidor;

XHTML

```
<h:form>
  <h:commandButton actionListener="#{loginBean.logar}" value="Login" />
</h:form>
```

```
@ManagedBean
public class LoginBean {

  public void logar(){
    //Lógica de logar
  }
}
```

Java

Esse componente deve estar dentro do **<h:form></h:form>** para funcionar;

CHECKBOX

- **<h:selectBooleanCheckbox> -** Cria um campo de **checkbox**;
- Esse campo é associado a uma **propriedade** do tipo **boolean;**

```java
@ManagedBean
public class CadastroBean {

  private boolean termo;

  //get e set
}
```

Java

Concordo com os termos

```xhtml
<h:selectBooleanCheckbox value="#{cadastroBean.termo}"  id="termo"/>
<h:outputLabel value="Concordo com os termos" for="termo"/>
```

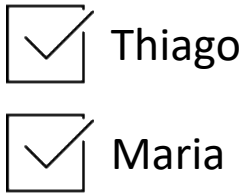
XHTML

CHECKBOX - VÁRIOS

- **<h:selectManyCheckbox>** - Define um **conjunto de campos de checkbox**, as opções podem ser definidas de forma **estática** e/ou **dinâmica;**
  - **<f:selectItem>** - Define **cada opção** (Estático);
  - **<f:selectItems>** - Define **dinâmicamente** as opções;

Java

```
@ManagedBean
public class CadastroBean {

  private List<Integer> listaSelecionados;
  //get e set
}
```




```
<h:selectManyCheckbox value="#{cadastroBean.listaSelecionados}">
  <f:selectItem itemLabel="Thiago" itemValue="1"/>
  <f:selectItem itemLabel="Maria" itemValue="2"/>
</h:selectManyCheckbox>
```

XHTML

CHECKBOX – VALORES DINÂMICOS

```xhtml
<h:selectManyCheckbox value="#{cadastroBean.listaSelecionados}">
  <f:selectItems value="#{cadastroBean.usuarios}"
    var="usuario"
    itemLabel="#{usuario.nome}"
    itemValue="#{usuario.id}"/>
</h:selectManyCheckbox>
```

XHTML

```java
@ManagedBean
public class CadastroBean{


  private List<Integer> listaSelecionados;


  public List<Usuario> getUsuarios(){
    UsuarioBO bo = new UsuarioBO();
    return bo.getAll();
  }
  //get e set
}
```

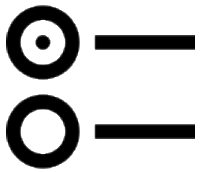
Java

value – lista de itens;
var – variável para cada um dos itens;
itemLabel – atributo exibido na tela;
itemValue – atributo enviado para o servidor se a opção for escolhida;

COMPONENTES DE SELEÇÃO ÚNICA

- **<h:selectOneRadio> -** Define um conjunto de **campos de radio;**
- **<h:selectOneMenu> -** Define um **select list** (combobox);
- **<h:selectOneListbox> -** Define um **select list**, mas exibe todas as opções;

- Os **campos são parecidos**, permitem a escolha de uma **única opção;**
- Se diferem na forma que são **exibidos na tela**;



RADIO BUTTONS

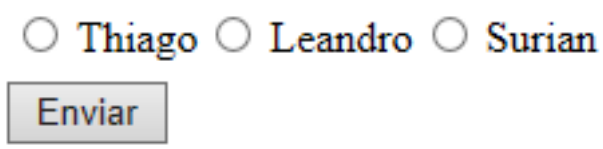
- **<h:selectOneRadio>** - Define radio buttons;

```xhtml
<h:selectOneRadio value="#{cadastroBean.usuarioSelecionado}">
  <f:selectItems value="#{cadastroBean.usuarios}"
    var="usuario"
    itemLabel="#{usuario.nome}"
    itemValue="#{usuario.id}"/>
</h:selectOneRadio>
```

XHTML

```java
@ManagedBean
public class CadastroBean{


  private int usuarioSelecionado;


  public List<Usuario> getUsuarios(){
    UsuarioBO bo = new UsuarioBO();
    return bo.getAll();
  }
  //get e set
}
```

Java



SELECT LIST

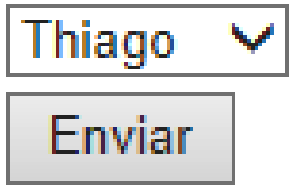
- **<h:selectOneMenu>** - Define um select list;

```
<h:selectOneMenu value="#{cadastroBean.usuarioSelecionado}">
  <f:selectItems value="#{cadastroBean.usuarios}"
    var="usuario"
    itemLabel="#{usuario.nome}"
    itemValue="#{usuario.id}"/>
</h:selectOneMenu>
```

XHTML

```
@ManagedBean
public class CadastroBean{


  private int usuarioSelecionado;


  public List<Usuario> getUsuarios(){
    UsuarioBO bo = new UsuarioBO();
    return bo.getAll();
  }
  //get e set
}
```

Java



SELECT LIST - ABERTO

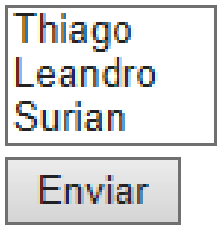
- **<h:selectOneListBox>** - Define um select list aberto;

```xhtml
<h:selectOneListBox value="#{cadastroBean.usuarioSelecionado}">
  <f:selectItems value="#{cadastroBean.usuarios}"
    var="usuario"
    itemLabel="#{usuario.nome}"
    itemValue="#{usuario.id}"/>
</h:selectOneListBox>
```

XHTML

Java

```java
@ManagedBean
public class CadastroBean{


  private int usuarioSelecionado;


  public List<Usuario> getUsuarios(){
    UsuarioBO bo = new UsuarioBO();
    return bo.getAll();
  }
  //get e set
}
```



COMPONENTES - PROPRIEDADES

- **readonly -** Campo **somente de leitura**;
- **rendered –** Define se o componente **será exibido ou não**;

```xhtml
<h:inputText  value="#{produtoBean.produto.id}"
  readonly="true"
  rendered="#{produtoBean.produto.id != 0}"/>


<h:commandButton
  rendered="#{usuario.perfil == 1}"
  value="Cadastrar"
  action="#{clienteBean.cadastrar}"/>
```

XHTML

MENSAGENS

- Para **exibir mensagens** na tela, crie um objeto do tipo **FacesMessage** no **controller,** informando a mensagem e depois adicione o **objeto no contexto**. **Na página**, coloque a tag **<h:messages/>** para exibir a mensagem.

```java
FacesMessage mensagem = new FacesMessage("Usuário cadastrado");
FacesContext.getCurrentInstance().addMessage( null , mensagem);
```

Java

```xhtml
<h:messages/>
```

XHTML

Dica: use o código para manter as mensagens no contexto após um **redirect**:

```
FacesContext.getCurrentInstance().getExternalContext().getFlash().setKeepMessages(true);
```

TABELAS

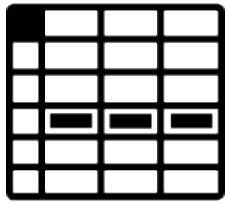
- **<h:dataTable> -** Define uma **tabela**;
- **<h:column> -** Define uma **coluna** de uma tabela;
- **<f:facet> -** Utilizado para definir **cabeçalho**, **rodapés** e o **título** de cada coluna;

Lista de elementos

Variável que recebe **cada item** da lista

```
<h:dataTable value="#{cursosBean.cursos}" var="curso">

</h:dataTable>
```

XHTML



TABELAS - EXEMPLO

```xhtml
<h:dataTable value="#{produtoBean.produtos}" var="prod" >
  <h:column>
    <f:facet name="header">Código</f:facet>
    #{prod.codigo}
  </h:column>
  <h:column>
    <f:facet name="header">Descrição</f:facet>
    #{prod.descricao}
  </h:column>
  <h:column>
    <f:facet name="header">Preço</f:facet>
    #{prod.preco}
  </h:column>
  <h:column>
    <f:facet name="header">Quantidade</f:facet>
    #{prod.qtd}
  </h:column>
</h:dataTable>
```

XHTML

TABELAS - PARÂMETROS

- É possível **enviar valores** para uma **outra página** ou para um **método** quando o usuário **clicar em um botão**;
- Tag **<setPropertyActionListerner>** possui os atributos: **value,** valor que será enviado e **target**, atributo que recebe o valor definido em **value**;

Nome da página de destino

```
<h:column>
  <h:form>
        <h:commandButton value="Alterar" action="alterar-produto">
              <f:setPropertyActionListener value="#{prod}"
                    target="#{produtoBean.produto}" />
        </h:commandButton>
        <h:commandButton action="#{produtoBean.excluir(prod.codigo)}"
              value="Excluir" />
  </h:form>
</h:column>
```

XHTML

Parâmetro que será enviado para o método no **ManagedBean**

FORMATAÇÃO DE CAMPOS

- **Formatação de valores** podem ser aplicados em campos de input e output, utilizando as tags **<f:convertXXX>**

```xhtml
<h:outputText value="#{p.quantidade}">
  <f:convertNumber pattern= "#,##" />
</h:outputText>
<h:outputText value="#{p.preco}">
  <f:convertNumber currencyCode="BRL" type="currency"/>
</h:outputText>
```

XHTML

- Para **formatar uma data**, caso o atributo seja do tipo **Calendar**, é necessário utilizar o **.time**, para recuperar o objeto Date do Calendar:

```xhtml
<h:outputText value="#{p.dataHoje.time}">
   <f:convertDateTime timeZone="America/Sao_Paulo" />
</h:outputText>
```

XHTML

Mais informações: http://www.roseindia.net/jsf/convertDateTime.shtml

MANAGED BEAN

MANAGED BEAN

- É uma classe **Java Bean gerenciado** pelo **framework JSF**;
- Responsável por intermediar a comunicação **entre as páginas e o modelo;**
- **Possui atributos e gets e sets**, que são utilizados pelas “páginas” para recuperar/atribuir valores nos atributos;
- Recomendado implementar a **interface Serializable**, para escopos maiores do que **Request**;
- Podemos anotar um método com **@PostConstruct** para que seja executado **após o Managed Bean** for **instanciado** pelo framework;

MANAGED BEAN - ESCOPOS

O **Managed Bean** possui alguns escopos, sendo os principais:

- **Request**: conteúdo do Managed Bean visível somente no fluxo request-response;
- **View**: conteúdo do Managed Bean visível enquanto a página estiver em processamento. Escopo maior que Request e menor que Session;
- **Session**: conteúdo do Managed Bean visível durante toda a sessão do usuário;
- **Application**: conteúdo do Managed Bean para todas as sessões/usuários do servidor;

MANAGED BEAN - EXEMPLO

Java

```java
import java.io.Serializable;
import javax.annotation.PostConstruct;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;

@ManagedBean
@ViewScoped
public class LoginBean implements Serializable{

  private String login;
  private String senha;

  /* get e set's */

  @PostConstruct
  public void init() {
            /* inicializando os objetos */
  }
}
```

INTEGRAÇÃO JSF COM JPA E WEB SERVICES

INTEGRAÇÃO JSF E WEB SERVICES

- Um projeto **JSF** pode trabalhar como **Web Service SOAP Provider** ou **Requester;**
- O mesmo se aplica para **Web Services Restful**, onde o sistema com JSF pode **consumir** ou **disponibilizar** serviços;
- O Web Service no **JSF** segue as mesmas regras que foram utilizadas com outros tipos de canais (desktop e texto);

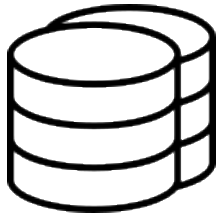
JSF E WS - DEPENDÊNCIAS

- Para consumir um web services **restful** com **JSF**, adicione as dependências do **Jersey client** no arquivo pom.xml

```
<dependency>
        <groupId>com.sun.jersey</groupId>
        <artifactId>jersey-client</artifactId>
        <version>1.8</version>
</dependency>
<dependency>
        <groupId>org.glassfish.jersey.media</groupId>
        <artifactId>jersey-media-json-jackson</artifactId>
        <version>2.25.1</version>
</dependency>
```

JSF COM JPA

- Um projeto **JSF** pode utilizar **JPA** para acesso ao banco de dados;
- Para isso, precisamos **adicionar as bibliotecas** do **JPA/Hibernate,** que podem ser através do **maven** e/ou da pasta "**WEB-INF/lib";**
- O arquivo **persistence.xml** deve estar dentro da pasta "**Java Resources/src/META-INF";**
- O **JPA no JSF** segue as **mesmas regras**, ou seja é recomendado utilizar os padrões DAO, Singleton e etc..

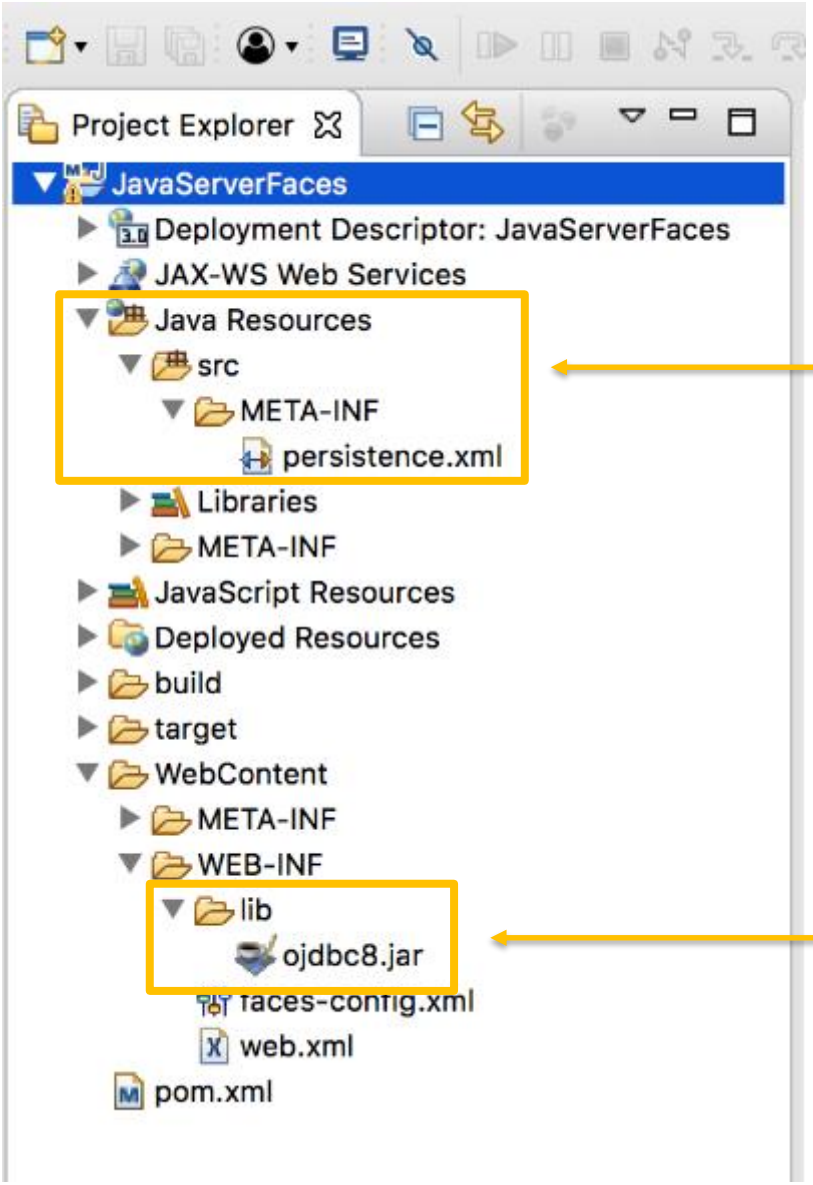


JSF E JPA - DEPENDÊNCIAS

- Adicione a dependência do **Hibernate** no arquivo pom.xml

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.2.12.Final</version>
</dependency>
```

JSF E JPA - DEPENDÊNCIAS



Coloque a pasta **META-INF** com o arquivo **persistence.xml**

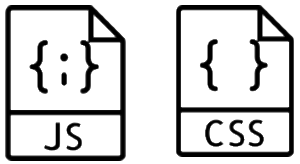
Adicione o **driver** do banco de dados no diretório **WebContent/WEB-INF/lib**

JSF COM JAVASCRIPT E CSS

JSF COM JS E CSS

- É possível gerar o conteúdo misturando elementos **dinâmicos** e elementos **estáticos** através das tags:
  - **<h:outputScript>**
  - **<h:oututStylesheet>**

- Por padrão, recursos estáticos devem ficar no diretório **resources**.

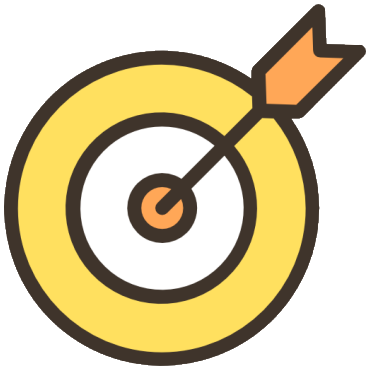


JSF E CSS - EXEMPLO

- O atributo **library** define o **diretório** dentro da pasta resources;
- Já o atributo **name** define o **nome** do arquivo;


```
<html>
 <h:head>
  <title>FIAP</title>
  <h:outputScript library=“js" name=“funcoes.js" />
  <h:outputStylesheet library="css" name=“main.css" />
 </h:head>
 <h:body>

 <h:graphicImage library="imagens" name="logo.png" id="logo“/>

 </h:body>
</html>
```

VOCÊ APRENDEU...



- Sobre os tipos de **Frameworks Web** e **JSF**;
- Como **criar** e **configurar** um **projeto JSF**;
- Trabalhar com **componentes visuais** como **campos de formulários, botões e tabelas**;
- Implementar um **Managed Bean**, definir seu **escopo** e o **método de inicialização**;
- Realizar integração com **web services** e **JPA**;
- Utilizar **javascript** e **css** nas páginas JSF;

"Aquele que não luta pelo futuro que quer, deve aceitar o futuro que vier"