

INTRODUÇÃO AO DESENVOLVIMENTO WEB

JavaServer Faces
Prof. Pablo Vargas

Tópicos Abordados

- Introdução
- JavaServer Faces
- Classe Backing Bean
- Criando páginas JSF
- Ciclo de vida de uma JSF

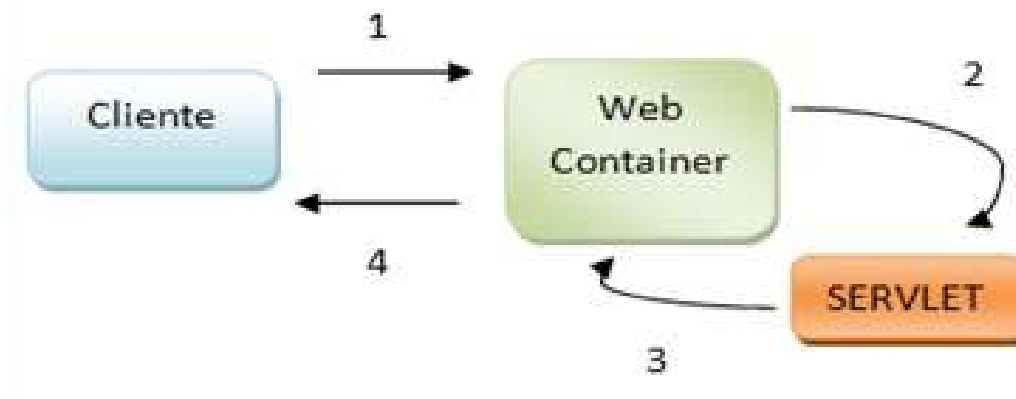
Introdução

- **O que é Servlet?**

- Surgiu com a necessidade de gerar **conteúdo dinâmico** no ano 1997 e hoje é uma tecnologia consolidada e madura.
- É uma classe Java usada para **estender as funcionalidades de um servidor**.
 - Foi a primeira e principal tecnologia da Plataforma Java para o **desenvolvimento de aplicações Web** com conteúdo dinâmico.
- Também pode ser definido como um componente semelhante um servidor, que gera dados **HTML e XML** para a camada de apresentação de uma aplicação Web.
- Processa dinamicamente requisições e respostas.

Introdução

- As aplicações são capazes de gerar conteúdo dinâmico e interagem com os clientes, utilizando o modelo requisição-resposta.
- Normalmente utilizam o protocolo HTTP e necessita de um container Web para ser executado.



Introdução

- Servlet fornece ao programador da linguagem Java uma interface para o servidor web (ou servidor de aplicação), através de uma *Application Programming Interface* (API).
- O que é API?
 - **Conjunto de rotinas e padrões** fornecidas por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação.
- Na linguagem técnica, um "servlet" é uma classe Java no Java EE que obedece à API Java Servlet.
- O que é API Java Servlet?
 - Possibilita, ao desenvolvedor, adicionar conteúdo dinâmico em um servidor web usando a plataforma Java.
 - Pertence ao pacote "javax.servlet".

Introdução

- Exemplo de API:
 - https://api.postmon.com.br/v1/cep/*cep_a_consultar*

JavaServer Pages (JSP)

- ...É um **template engine** que ajuda os desenvolvedores de software a criarem páginas web geradas dinamicamente baseadas em HTML, XML ou outros tipos de documentos.
- Lançada em 1999 pela Sun Microsystems.
- Serve para **facilitar** a criação de páginas HTML, tornando o envio e a exibição de informações um processo mais simples e organizado.
- Framework para construir visões com limitações e, por isso, esta *deprecated* atualmente substituído pelo facelets.

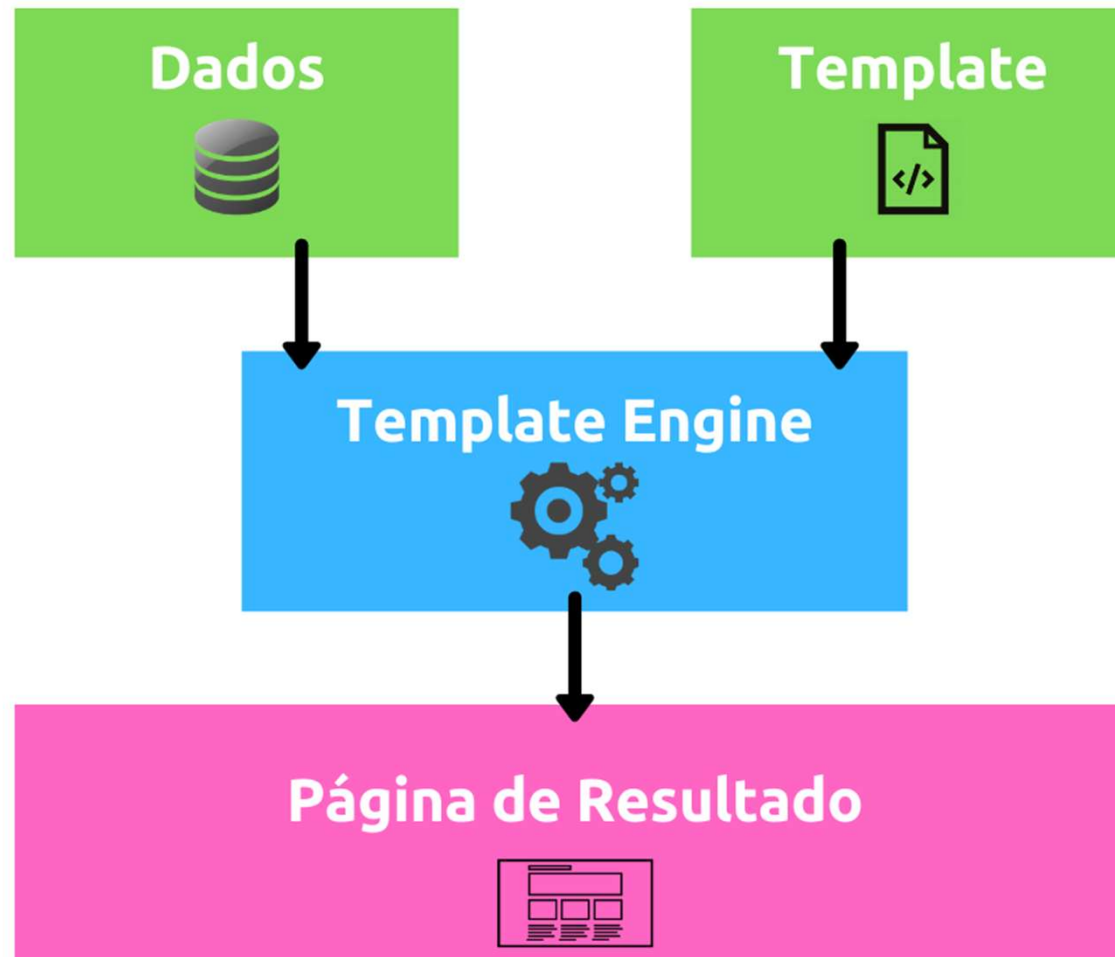
JavaServer Pages (JSP)

- **Scriptlet** é o código Java escrito entre `<% e %>`.
 - O nome tem origem da palavra script (pedaço de código em linguagem de script) com o sufixo let, que indica algo pequeno.

```
<html>
  <body>
    <br />

    <%
      System.out.println("Tudo foi executado!");
    %>
  </body>
</html>
```

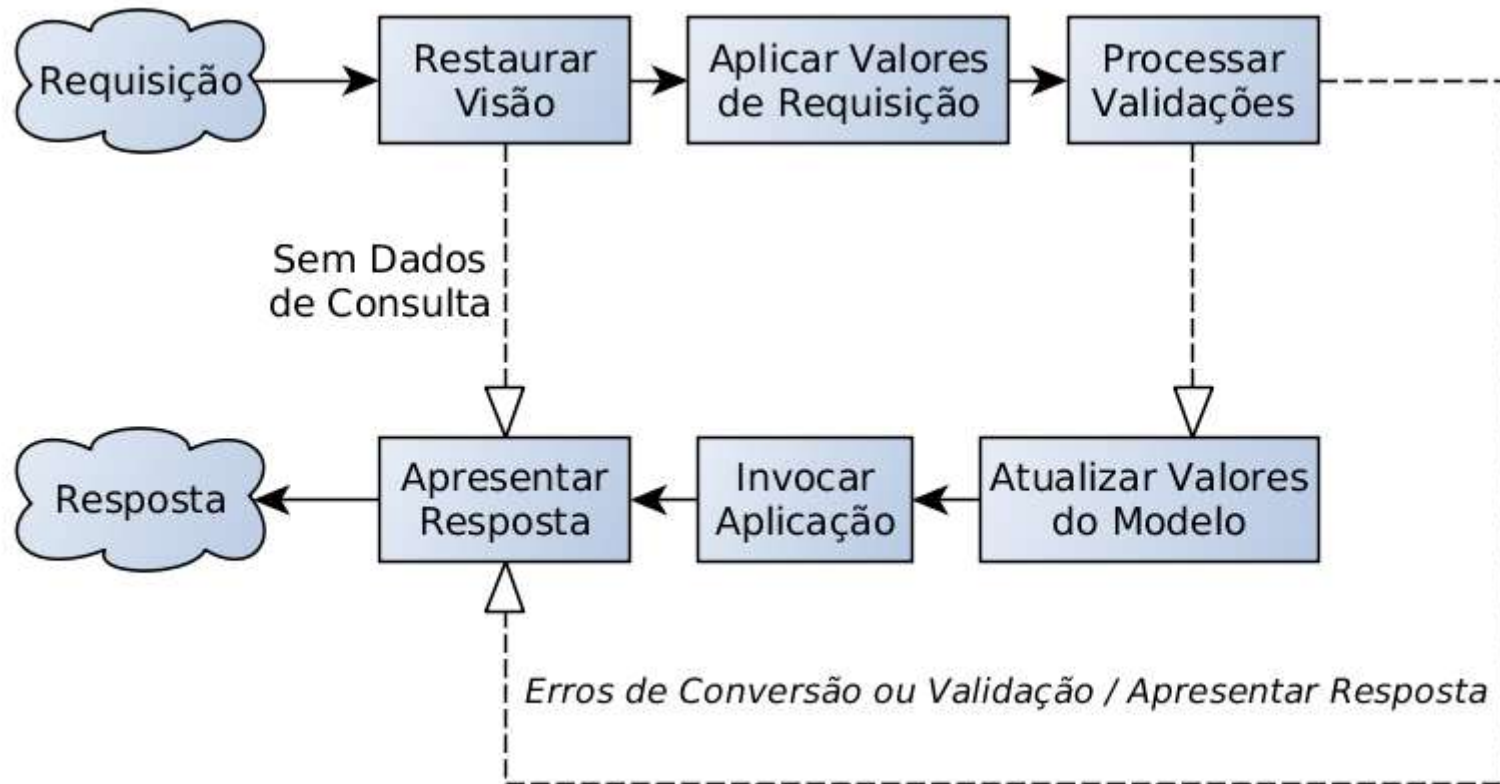

JavaServer Pages (JSP)



JavaServer Faces

- ...é uma especificação de um framework MVC baseado em Java para a construção de interfaces de usuário baseadas em componentes de aplicações web.
- Definida pela Java Community Process (JCP).
 - Entidade que decide a evolução das tecnologias Java.
- Aprovada e definida por grandes empresas (Ex: IBM, HP, Oracle, Siemens, Apache...).
- PrimeFaces e RichFaces são bibliotecas de componentes com recursos adicionais aos padrões definidos pela JSF.
- Java Standard Tag Library é um conjunto de tags de apoio para o desenvolvimento web em java.
- <https://www.youtube.com/watch?v=3Cj2mDx0-Ws>

JavaServer Faces



Ciclo de vida de uma JSF

- Restaurar visão:
 - O JSF extrai da URL solicitada o nome da pagina que deve ser exibida.
 - Se essa página ainda não tenha sido carregada, o JSF fará uma leitura da página solicitada e os componentes serão carregados em memória.
 - Se já tiver em memória, o JSF carrega os componentes.
- Aplicar Valores de Requisição
 - Depois que carregado os componentes , o JSF preencherá esses componentes com os valores que foram enviados.

Ciclo de vida de uma JSF

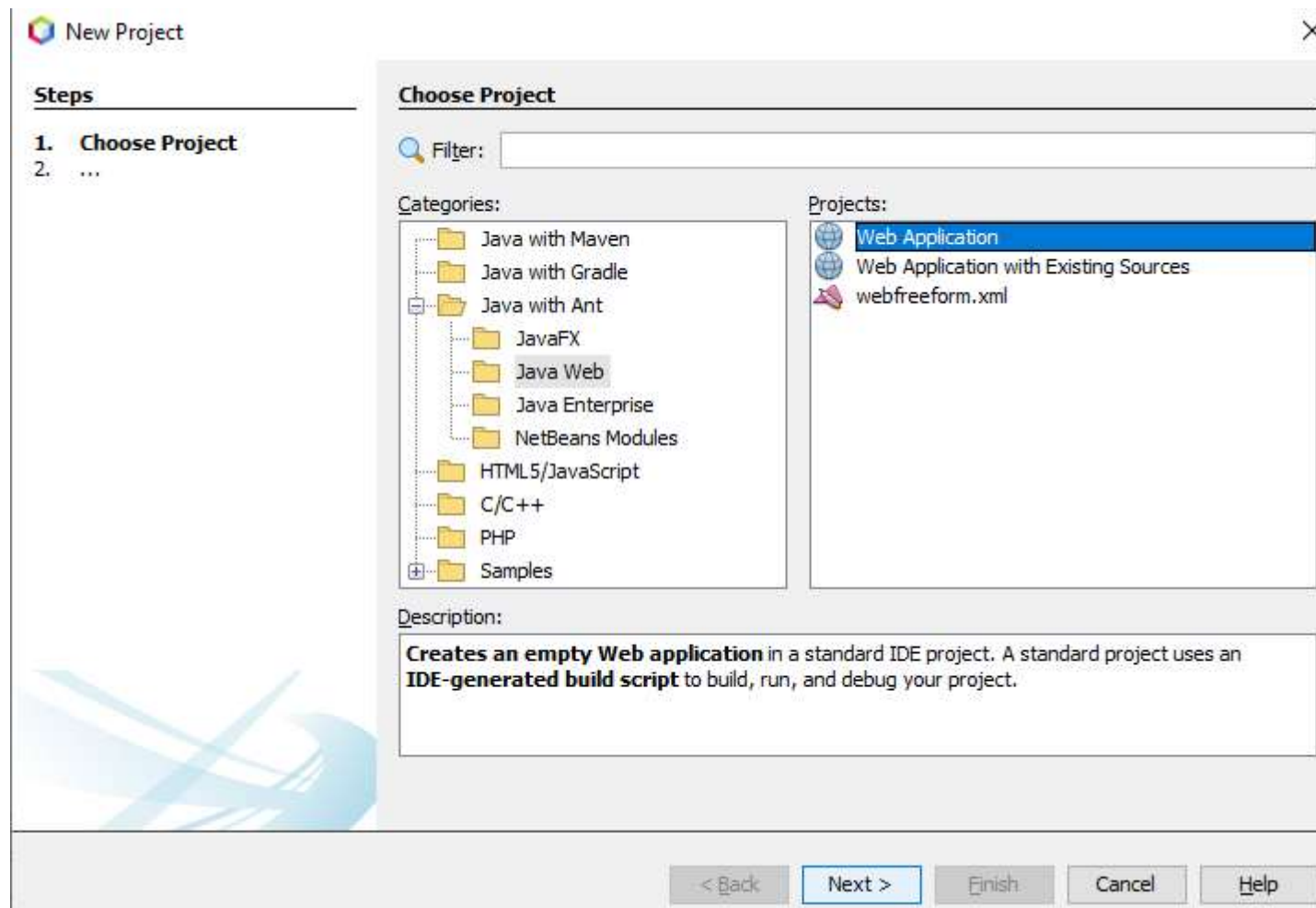
- Processar Validações:
 - O JSF valida os componentes conforme as regras definidas pelo desenvolvedor. Se a validação falhar a página será reexibida.
- Atualizar Valores do Modelo:
 - Depois de validados e registrados, os valores serão atribuídos à respectiva propriedade na classe Bean.
 - Caso ocorra algum erro de conversão do valor enviado para a propriedade é gerado uma mensagem que será enviada para exibir.

Ciclo de vida de uma JSF

- **Invoca Aplicação:**
 - É nessa fase que o JSF acionará o método da classe Bean que acionou a requisição.
- **Apresentar Resposta:**
 - A página será montada e devolvida ao navegador. O JSF solicita que cada componente gere seu próprio HTML, formando a página a ser exibida.

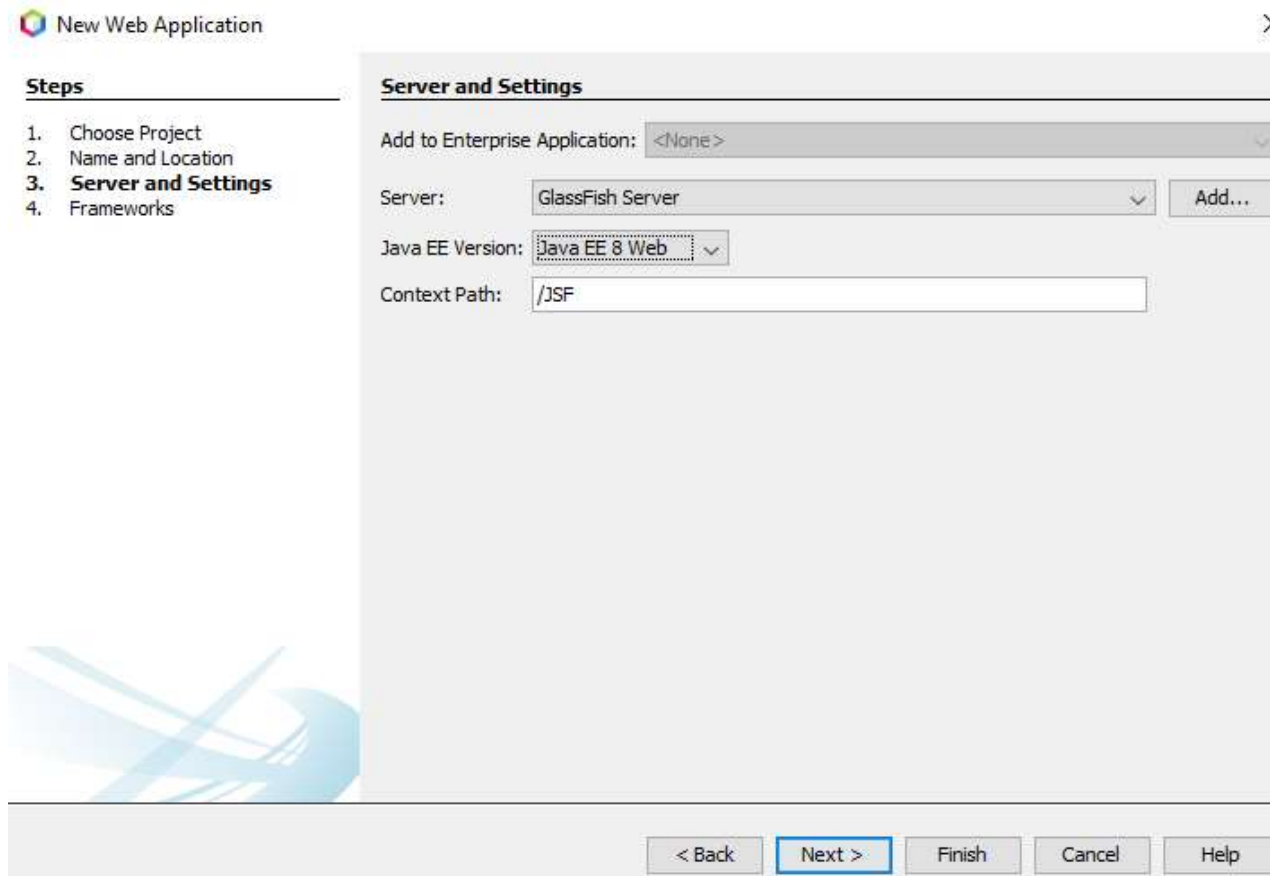
JavaServer Faces

- Crie um novo projeto do tipo Web Application



JavaServer Faces

- Coloque nome no projeto de JSF e clique em “Next >”.
- Na tela seguinte marque os seguintes parâmetros.



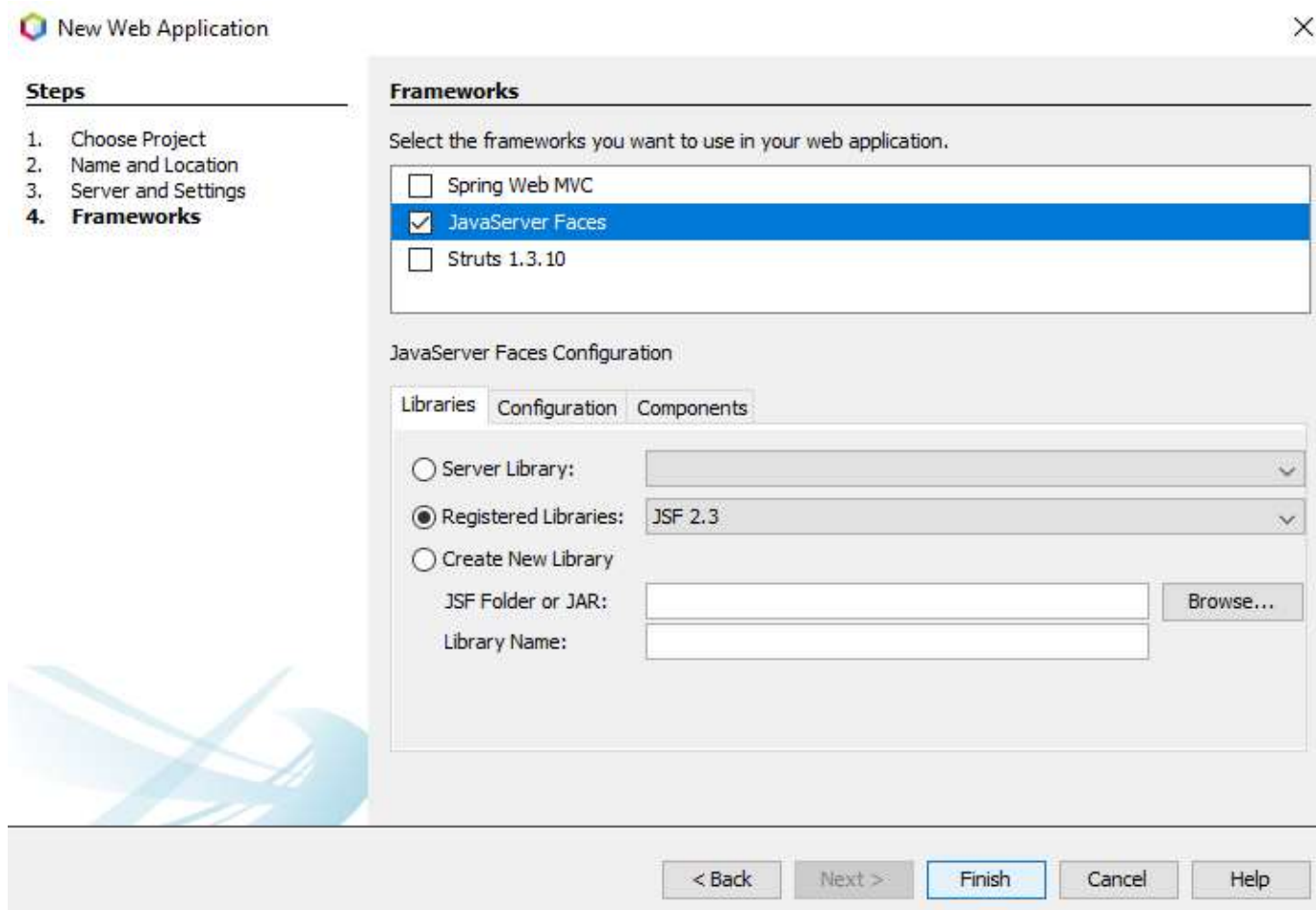
The screenshot shows the 'New Web Application' wizard in an IDE. The window title is 'New Web Application'. On the left, a 'Steps' panel lists four steps: 1. Choose Project, 2. Name and Location, 3. **Server and Settings** (highlighted), and 4. Frameworks. The main area is titled 'Server and Settings' and contains the following fields:

- 'Add to Enterprise Application:' with a dropdown menu showing '<None>'.
- 'Server:' with a dropdown menu showing 'GlassFish Server' and an 'Add...' button.
- 'Java EE Version:' with a dropdown menu showing 'Java EE 8 Web'.
- 'Context Path:' with a text input field containing '/JSF'.

At the bottom of the wizard, there are five buttons: '< Back', 'Next >' (highlighted with a blue border), 'Finish', 'Cancel', and 'Help'.

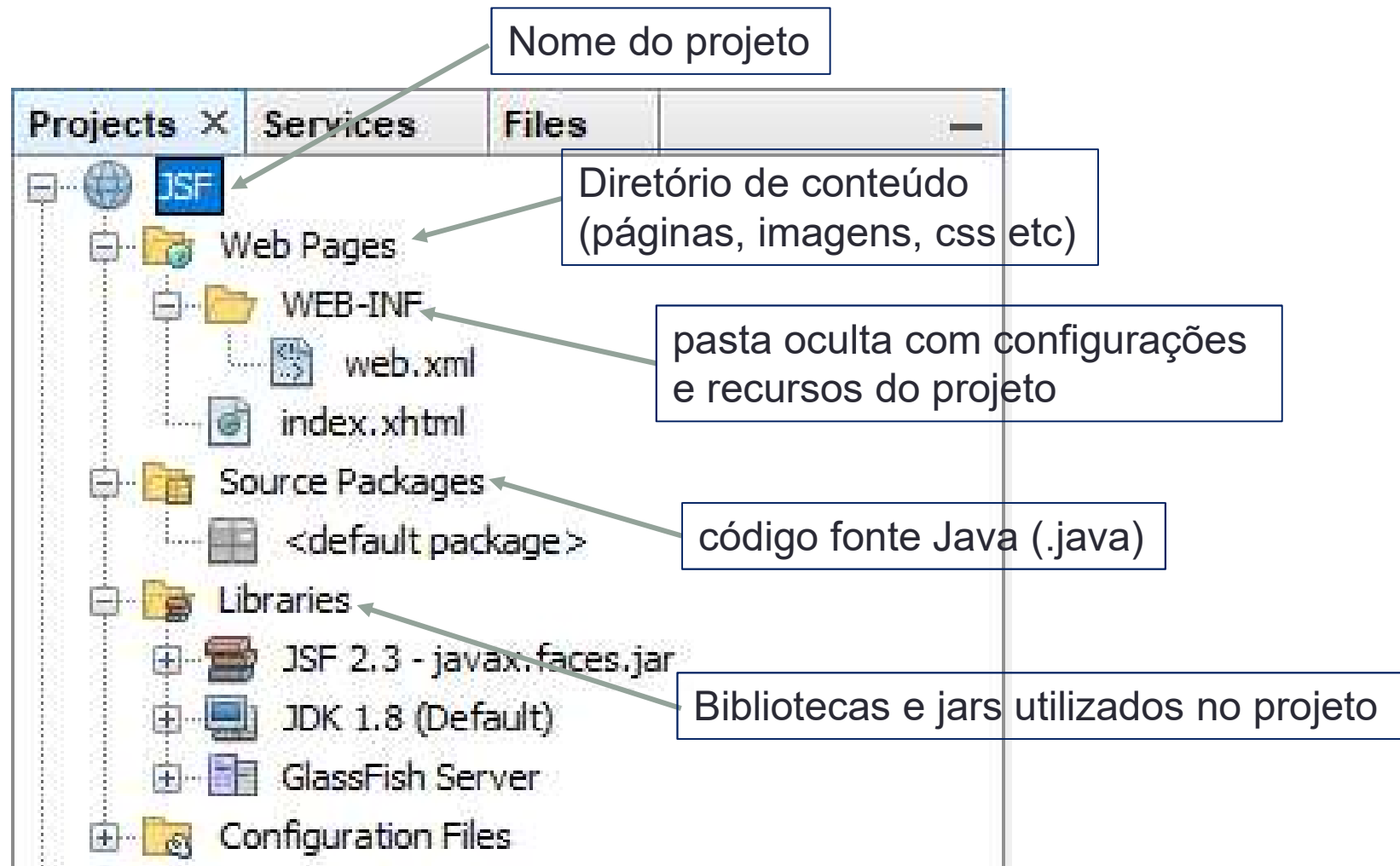
JavaServer Faces

- Na tela seguinte marque os seguintes parâmetros e clique em Finish.



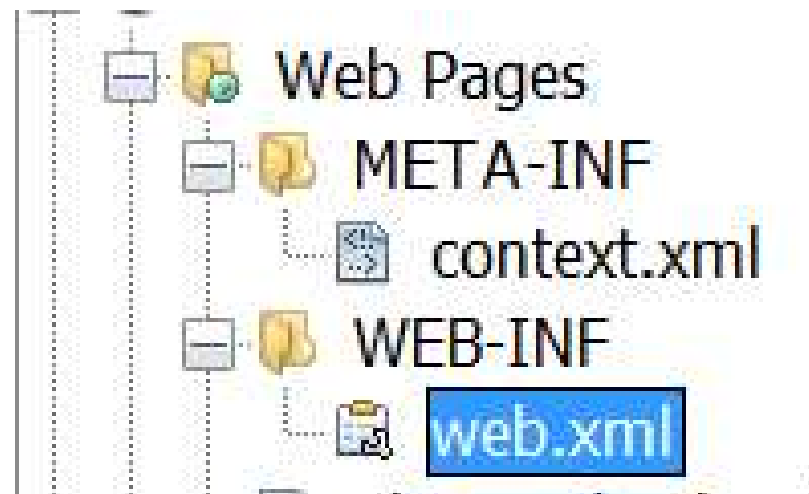
The screenshot shows the 'New Web Application' wizard in the Eclipse IDE. The window title is 'New Web Application'. On the left, a 'Steps' pane lists the progression: 1. Choose Project, 2. Name and Location, 3. Server and Settings, and 4. **Frameworks**. The main area is titled 'Frameworks' and contains the instruction 'Select the frameworks you want to use in your web application.' Below this is a list of frameworks: 'Spring Web MVC' (unchecked), 'JavaServer Faces' (checked and highlighted in blue), and 'Struts 1.3.10' (unchecked). Underneath is the 'JavaServer Faces Configuration' section with three tabs: 'Libraries', 'Configuration', and 'Components'. The 'Libraries' tab is active, showing three radio button options: 'Server Library:' (with a dropdown menu), 'Registered Libraries:' (selected, with a dropdown menu showing 'JSF 2.3'), and 'Create New Library'. Below these are input fields for 'JSF Folder or JAR:' and 'Library Name:', with a 'Browse...' button next to the first field. At the bottom of the wizard are five buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), 'Cancel', and 'Help'.

JavaServer Faces



JavaServer Faces

- Instalação e Configuração:
 - Localize o arquivo “**web.xml**” dentro da pasta “**WEB-INF**”.



JavaServer Faces

- Instalação e Configuração:
 - Modifique o arquivo de acordo com o código abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <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>*.xhtml</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>index.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```

JavaServer Faces

- **Mais sobre o url-pattern:**
 - Oferece a flexibilidade de disponibilizar uma servlet através de várias URLs de um caminho
 - Exemplo: o código abaixo fará com que qualquer endereço acessado dentro de “/” seja interpretado pela sua servlet:

```
<servlet-mapping>  
    <servlet-name>Faces Servlet</servlet-name>  
    <url-pattern>/</url-pattern>  
</servlet-mapping>
```

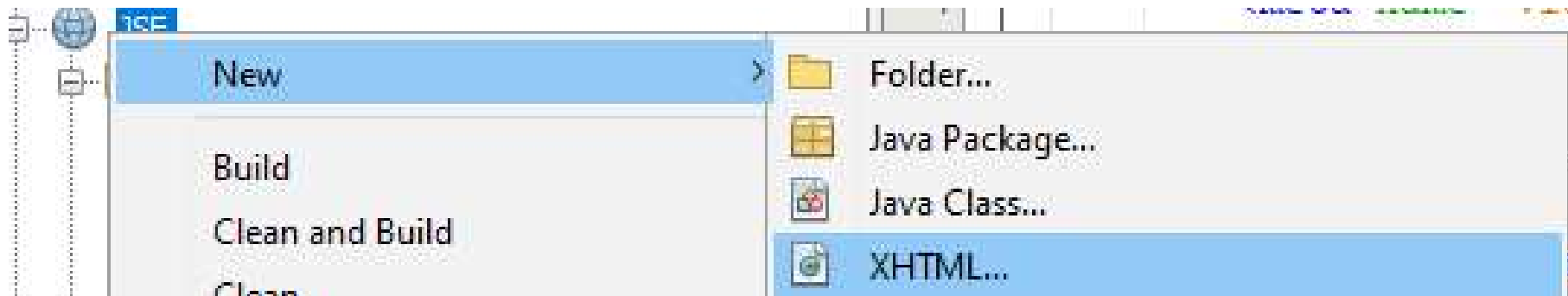
JavaServer Faces

- **Mais sobre o url-pattern:**
 - Você ainda pode configurar "extensões" para as suas servlets.
 - Exemplo: o mapeamento abaixo fará com que sua servlet seja chamada por qualquer requisição que termine com .jsf:

```
<servlet-mapping>  
  <servlet-name>Faces Servlet</servlet-name>  
  <url-pattern>*.jsf</url-pattern>  
</servlet-mapping>
```

JavaServer Faces

- Testando:
 - Clique com o botão esquerdo em “**Web Pages**”.
 - Vá em “**New**” e clique em “**XHTML...**”



JavaServer Faces

- Testando:
 - No campo “XHTML File Name:” escreva “**olaMundo**” e clique em “**Finish**”.



New XHTML

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

XHTML File Name:

Project:

Location:

Folder:

Created File:

JavaServer Faces

- Testando:
 - Digite o código abaixo.

```
<?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://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Teste Inicial JSF</title>
  </h:head>
  <h:body>
    <h:outputText value="Olá Mundo!" />
  </h:body>
</html>
```

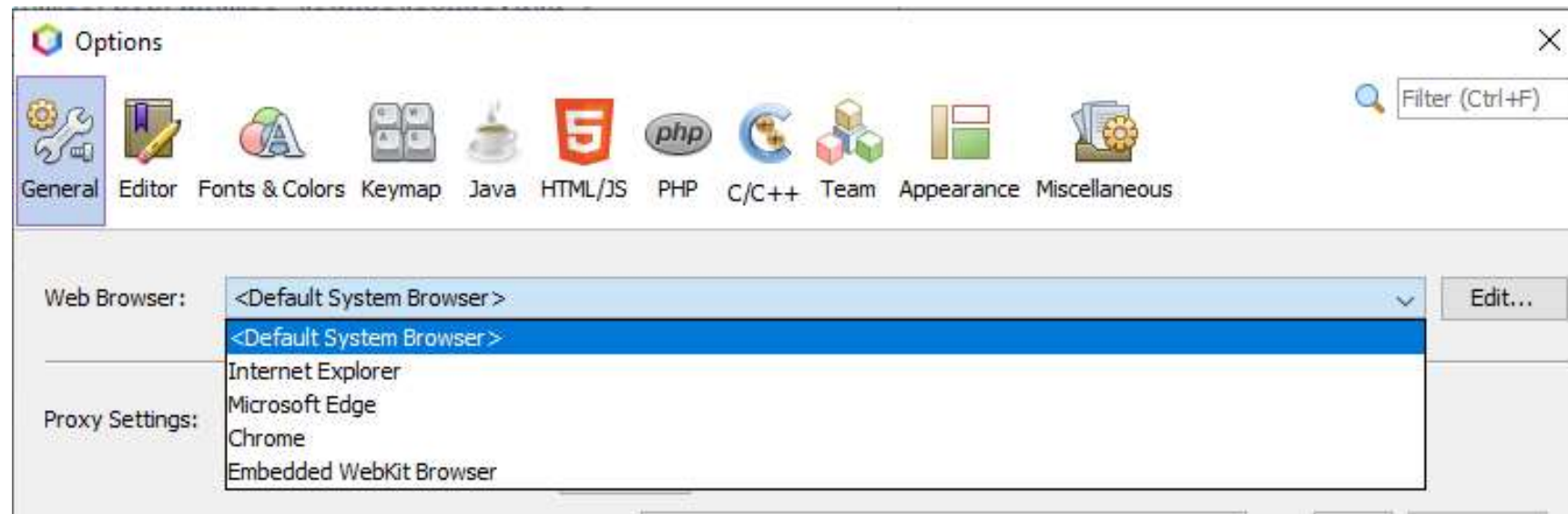
JavaServer Faces

- Testando:
 - Selecione o Projeto na aba “**Projects**” e aperte F6 no teclado.
 - O Apache/GlassFish iniciará e o Navegador abrirá uma janela.
 - Modifique o endereços para
“**localhost:8080/JSF/olaMundo.xhtml**”.



NetBeans

- Configurar navegador para rodar a aplicação Web
 - Clique em “Tools” e depois em “Options”.
 - Na Aba General, escolha em Web Browser o seu navegador preferido.



Classe Backing Bean

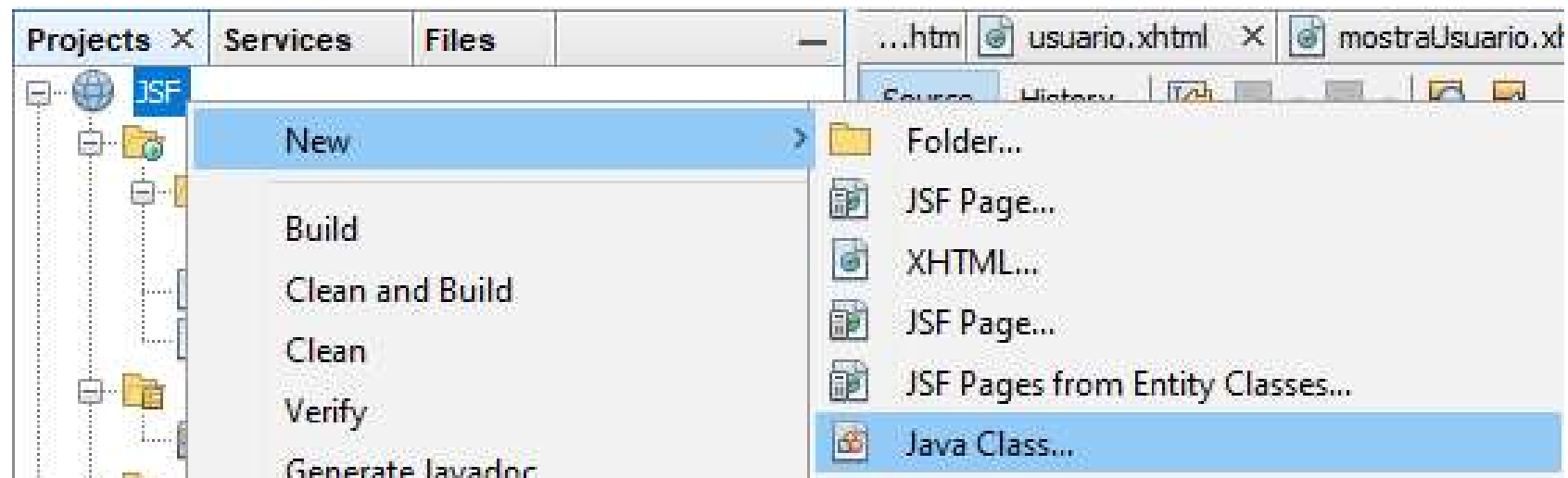
- “...é uma classe Java normal, que suportará todo o funcionamento das páginas JSF.”(Luckow)
- Separa regras de funcionamento do layout da página.
- Fornece os dados a ser exibidos e as operações a ser executadas.
- É ele o responsável por fazer a ligação entre a visão (as páginas XHTML) e o modelo do nosso sistema.
- É uma classe *Java* que responde a pedidos e gera o estado dos componentes *JSF*.
- Os *backing beans* mais conhecidos em *Java* são os “*managed beans*” (suportado pelo Tomcat) e os “*named beans*”.

Classe Backing Bean

- **Managed Bean:** representa o *Controller* no padrão MVC. Caracteriza-se consoante o seu âmbito (*scope*). Os mais importantes são:
 - **Application:** o *bean* persiste na aplicação estando disponível para todas as sessões;
 - **Session:** o *bean* existe por utilizador; é útil para dados referentes ao que utilizador está a fazer durante um período de utilização;
 - **View:** semelhante ao *session*, mas por *tab*; é como que se cada *tab* do *browser* representasse uma sessão diferente;
 - **Request:** a única gestão de estado que existe dura apenas o pedido e por isso é o mais leve.

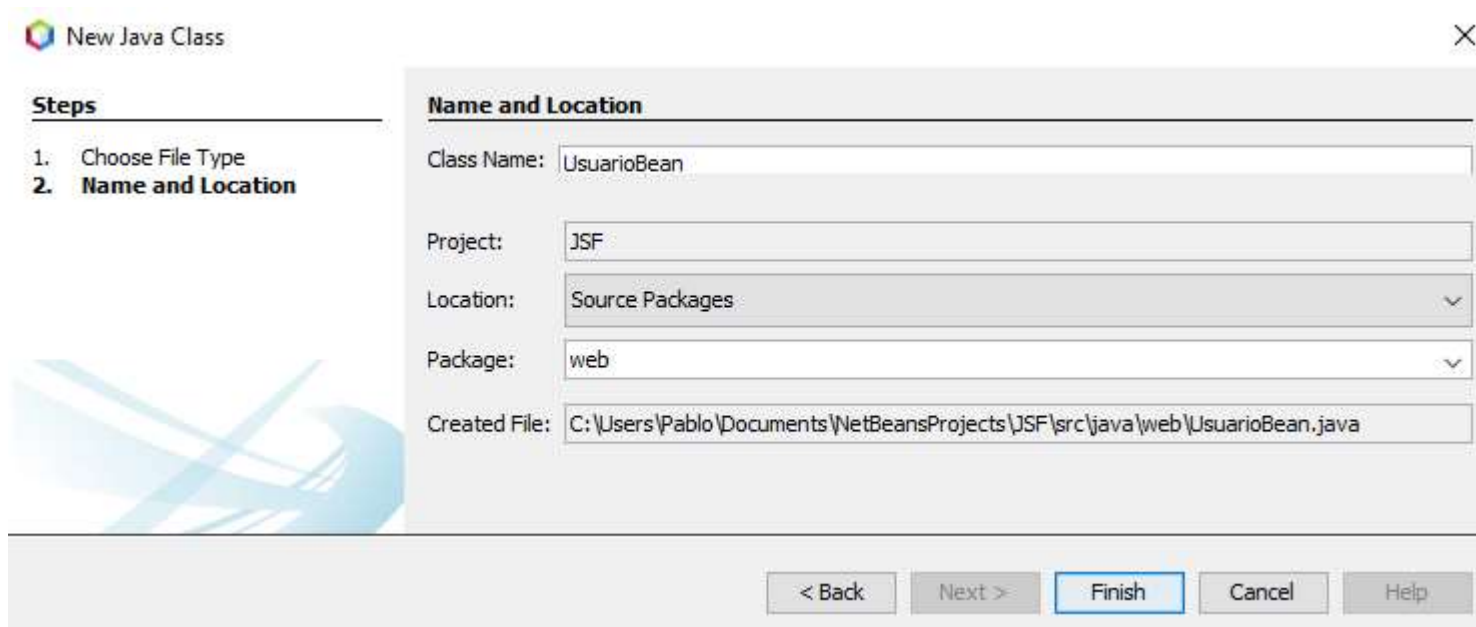
Classe Backing Bean

- Criando a Classe Bean:
 - Clique com o botão direito do mouse no projeto e selecione a opção da figura abaixo.



Classe Backing Bean

- Criando a Classe Bean:
 - Mude o “Nome da Classe:” e o “Pacote:” de acordo com a figura e clique em “Finalizar”.



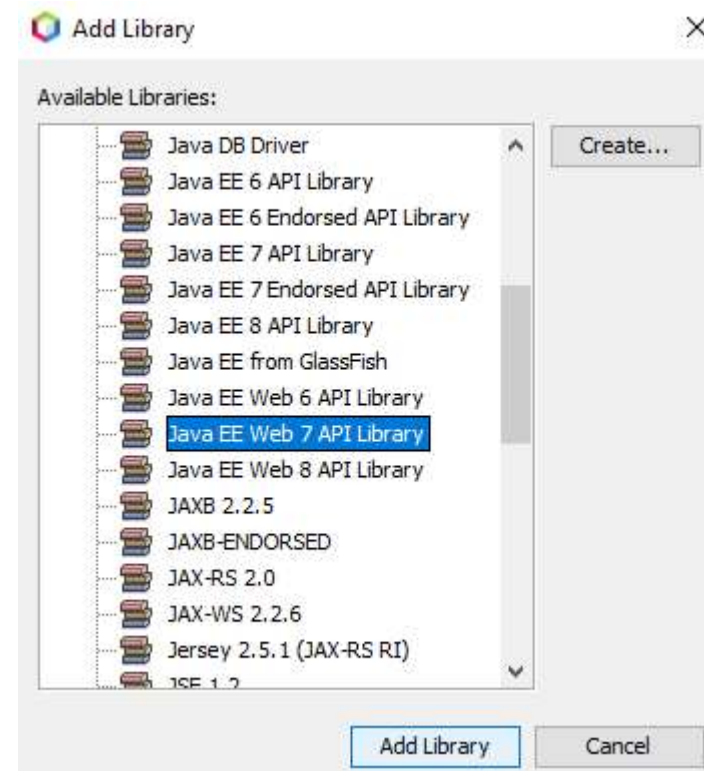
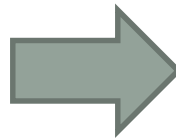
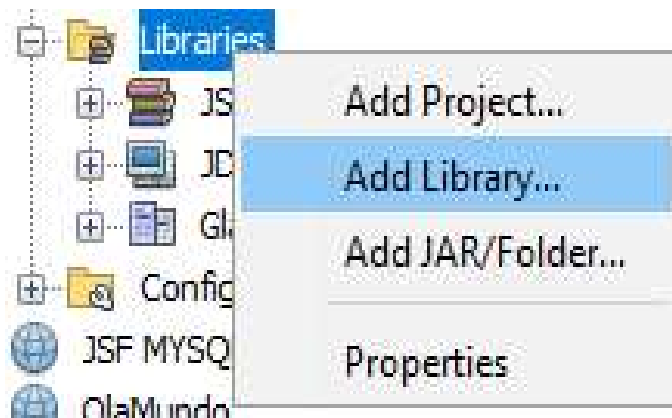
The screenshot shows the 'New Java Class' dialog box in NetBeans. The 'Steps' panel on the left indicates that step 2, 'Name and Location', is the current step. The 'Name and Location' panel contains the following fields:

- Class Name:** UsuarioBean
- Project:** JSF
- Location:** Source Packages (dropdown menu)
- Package:** web (dropdown menu)
- Created File:** C:\Users\Pablo\Documents\NetBeansProjects\JSF\src\java\web\UsuarioBean.java

At the bottom of the dialog, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), 'Cancel', and 'Help'.

Classe Backing Bean

- Criando a Classe Bean:
 - Antes de inserir o código da Classe Bean adicione a biblioteca “Java EE Web 7 API Library”



Classe Backing Bean

- Criando a Classe Bean:
 - Adicione as propriedades e métodos abaixo no código da Classe Bean.

```
public class UsuarioBean {
    private String nome;
    private String email;
    private String senha;
    private String confirmaSenha;

    //Gerar os gets e sets de todas os atributos da classe (alt + insert)

    public String novo(){
        return "usuario";
    }

    public String salvar(){
        FacesContext context = FacesContext.getCurrentInstance();
        if (!this.senha.equalsIgnoreCase(this.confirmaSenha)){
            context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Senha confirmada incorretamente", ""));
            return "usuario";
        }
        return "mostraUsuario";
    }
}
```

Classe Backing Bean

- Mapeamento da Classe Bean:
 - Adicione os trechos destacados em vermelho no início da classe.

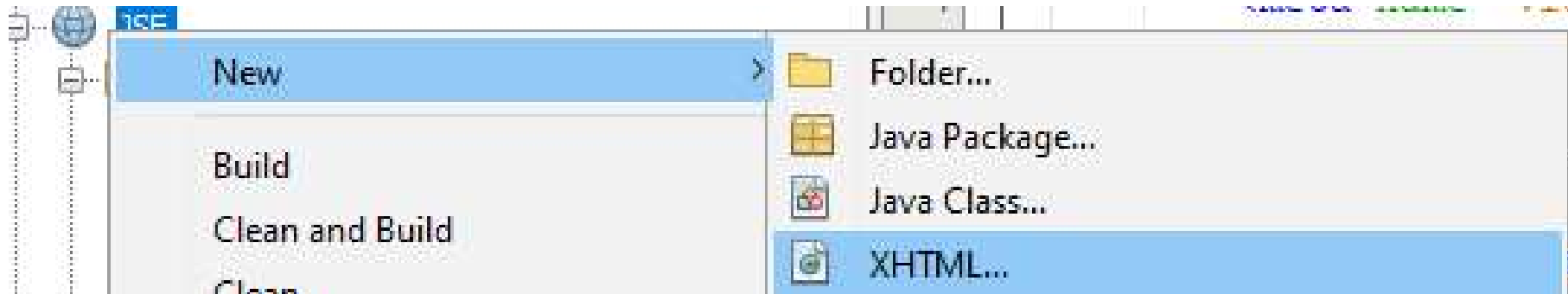
```
[-] import java.io.Serializable;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;

[-] /**
 *
 * @author Pablo
 */

@Named
@SessionScoped
public class UsuarioBean implements Serializable {
    private String nome;
    private String email;
    private String senha;
    private String confirmaSenha;
```

Criando páginas JSF

- Página de cadastro do usuário:
 - Clique com botão direito na pasta “Páginas Web”, em seguida vá em “Novo” e depois clique em “XHTML...”.



Criando páginas JSF

- Página de cadastro do usuário:
 - Digite o nome do arquivo como “usuário” e depois clique em “Finish”.
 - Adicione a taglib do JSF

```
<?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://xmlns.jcp.org/jsf/html">
```

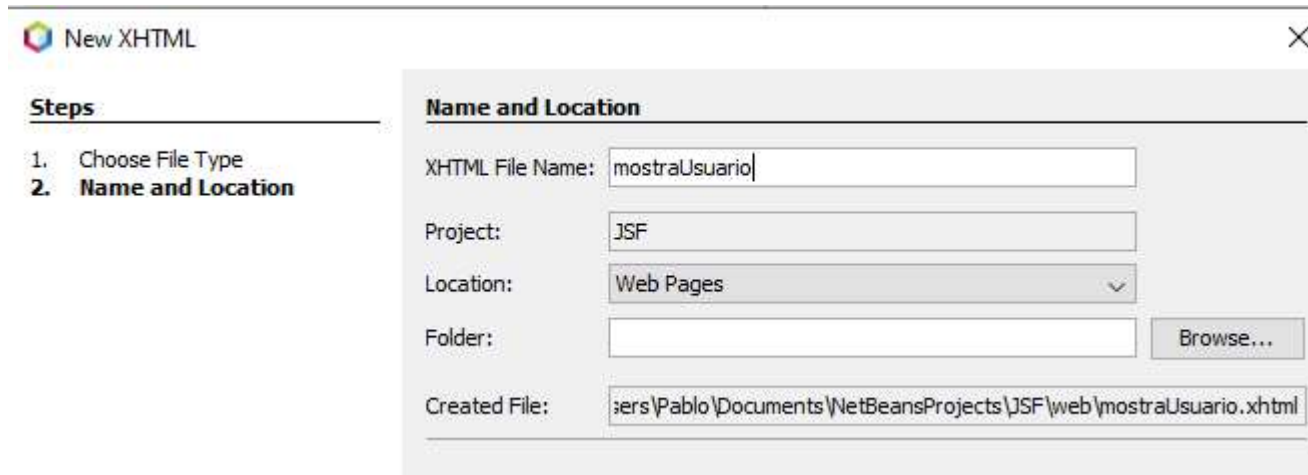
Criando páginas JSF

- Página de cadastro do usuário:
 - Digite dentro das tags <h:head> e <h:body> o seguinte código.

```
<h:head>
  <title>Cadastro de Usuários</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</h:head>
<h:body>
  <h1>Cadastro Usuário</h1>
  <hr/>
  <h:form>
    <h:messages/>
    <h:panelGrid columns="2">
      <h:outputLabel value="Nome:" for="nome"/>
      <h:inputText id="nome" label="nome" value="#{usuarioBean.nome}" required="true"/>
      <h:outputLabel value="e-mail:" for="email"/>
      <h:inputText id="email" label="e-mail" value="#{usuarioBean.email}"/>
      <h:outputLabel value="Senha:" for="senha"/>
      <h:inputSecret id="senha" label="Senha" value="#{usuarioBean.senha}" required="true"/>
      <h:outputLabel value="Confirmar Senha:" for="confirmarsenha"/>
      <h:inputSecret id="confirmarsenha" label="Confirmar Senha"
        value="#{usuarioBean.confirmaSenha}" required="true"/>
      <h:outputText/>
      <h:commandButton action="#{usuarioBean.salvar}" value="Salvar"/>
    </h:panelGrid>
  </h:form>
  <hr/>
</h:body>
```

Criando páginas JSF

- Página que mostra o usuário:
 - Crie um novo arquivo “XHTML...” com as configurações abaixo.
 - Adicione a taglib do JSF (`xmlns:h="http://xmlns.jcp.org/jsf/html"`).



The screenshot shows the 'New XHTML' dialog box in NetBeans IDE. The dialog has a title bar with the text 'New XHTML' and a close button. On the left, there is a 'Steps' section with two steps: '1. Choose File Type' and '2. Name and Location'. The 'Name and Location' section is active and contains the following fields:

- XHTML File Name:** A text box containing 'mostraUsuario'.
- Project:** A text box containing 'JSF'.
- Location:** A dropdown menu showing 'Web Pages'.
- Folder:** A text box, currently empty, with a 'Browse...' button next to it.
- Created File:** A text box showing the full path: 'ers\Pablo\Documents\NetBeansProjects\JSF\web\mostraUsuario.xhtml'.

Criando páginas JSF

- Página que mostra o usuário:
 - Insira o código abaixo nas tags <h:head> e <h:body> .

```
<h:head>
  <title>Usuário Cadastrado</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</h:head>
<h:body>
  <h1> Usuário cadastrado </h1>
  <hr/>
  Nome: <h:outputText value="#{usuarioBean.nome}"/><br/>
  E-mail: <h:outputLink value="mailto:#{usuarioBean.email}">
    <h:outputText value="#{usuarioBean.email}"/>
  </h:outputLink><br/>
  Senha: <h:outputText value="#{usuarioBean.senha}"/><br/>
  <hr/>
  <h:form>
    <h:commandLink action="#{usuarioBean.novo}" value="Inicio"/>
  </h:form>
</h:body>
```

Criando páginas JSF

- Página inicial:
 - Abra o arquivo “index.xhtml” da pasta “Web Pages” e modifique seu código html de acordo com o código abaixo.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Primeira APP JAVA WEB</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  </h:head>
  <h:body>
    <h1>Gerenciador Pessoal</h1>
    <h:form>
      <h:commandLink action="#{usuarioBean.novo}">Novo Usuário</h:commandLink>
    </h:form>
  </h:body>
</html>
```


Extra

- <https://www.caelum.com.br/apostila-java-web/servlets/#a-estrutura-de-diretrios>
- Diferença entre CGI (Common Gateway Interface) e Servlet
 - <https://pt.gadget-info.com/difference-between-cgi>
- Ciclo JSF
 - <https://www.youtube.com/watch?v=hbeQHFNLcJs>

Atividades

- I-Crie um campo data de nascimento no cadastro de usuário e exiba esse campo na pagina mostraUsuario.xhtml.
- II-Crie na página index.xhtml um link para pagina de cadastro de fornecedor.
 - Crie a página de cadastro do fornecedor, com os seguintes atributos: cnpj, nome e cidade.
 - Crie uma página para mostrar o fornecedor.