



PUCPR
GRUPO MARISTA

JavaServer Faces 2.x Exemplo Simples

Disciplina de ADS: **Tecnologia para Desenvolvimento Web**



Sumário – JavaServer Faces (JSF)

- Apresentação Geral
- Adicionando Suporte JSF a uma Aplicação Web
- Criando um Managed Bean
- Conectando o Managed Bean a Páginas
- Aplicando o Modelo de Facelets

Apresentação Geral

➤ Tutorial


- Demonstra como aplicar o **JSF 2.x** a uma **aplicação Web** utilizando o **NetBeans IDE**.
- Primeiro, é preciso configurar o ambiente de desenvolvimento, adicionando o suporte ao framework JSF 2.x. Após, será possível realizar as tarefas:
 1. Criar um **managed bean** JSF para manipular os dados solicitados,
 2. Conectar **managed bean** às páginas Web da aplicação e
 3. Converter as páginas Web em arquivos de modelo de Facelets.

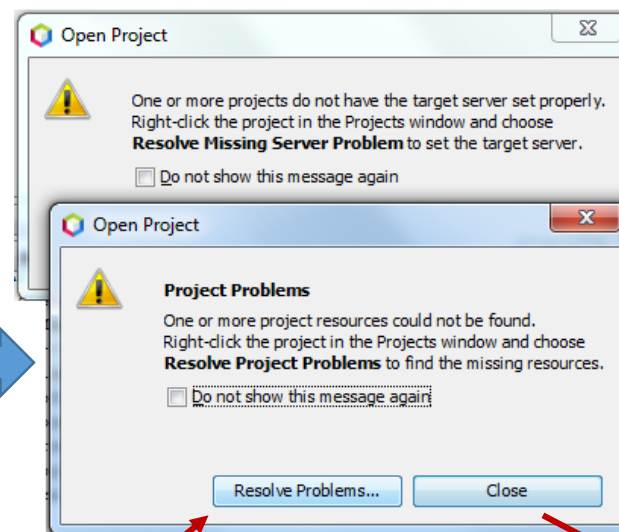
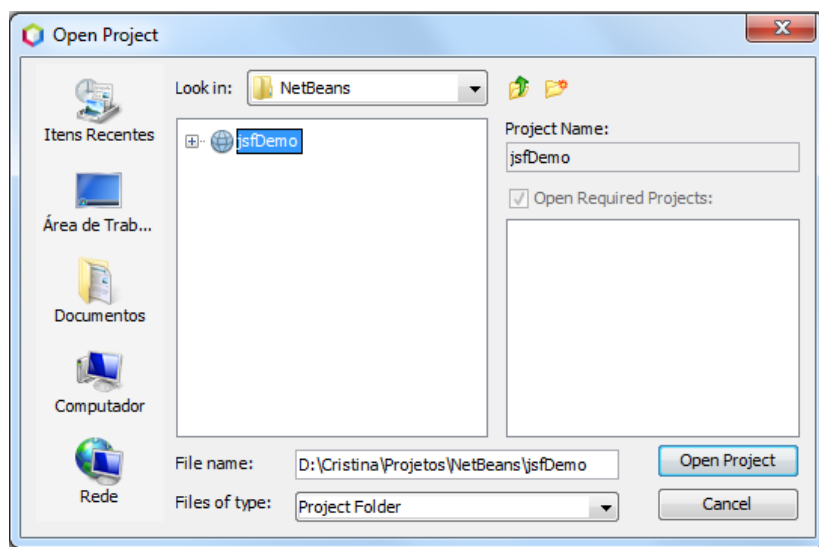
➤ Configuração do ambiente de desenvolvimento, com o framework JSF 2.x

- A instalação dos recursos para desenvolvimento Java Web estão detalhados no material sobre **Ambiente de Desenvolvimento NetBeans**

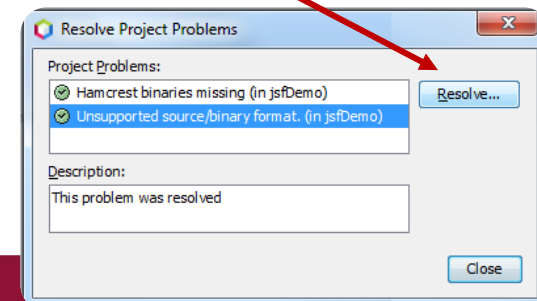
Software ou Recurso	Versão Necessária
NetBeans IDE	Apache NetBeans 11.0
JDK (Java Development Kit)	Java SE Development Kit 8 Downloads
Payara GlassFish Server 191	Payara Server 191 Full
jsfDuke - projeto de aplicação Web	Arquivo .ZIP

Adicionando Suporte JSF a uma Aplicação Web

- Copie o projeto jsfDuke na pasta de projetos (workspace) do NetBeans
- Clique no botão  para Abrir o Projeto na barra de ferramentas principal do NetBeans



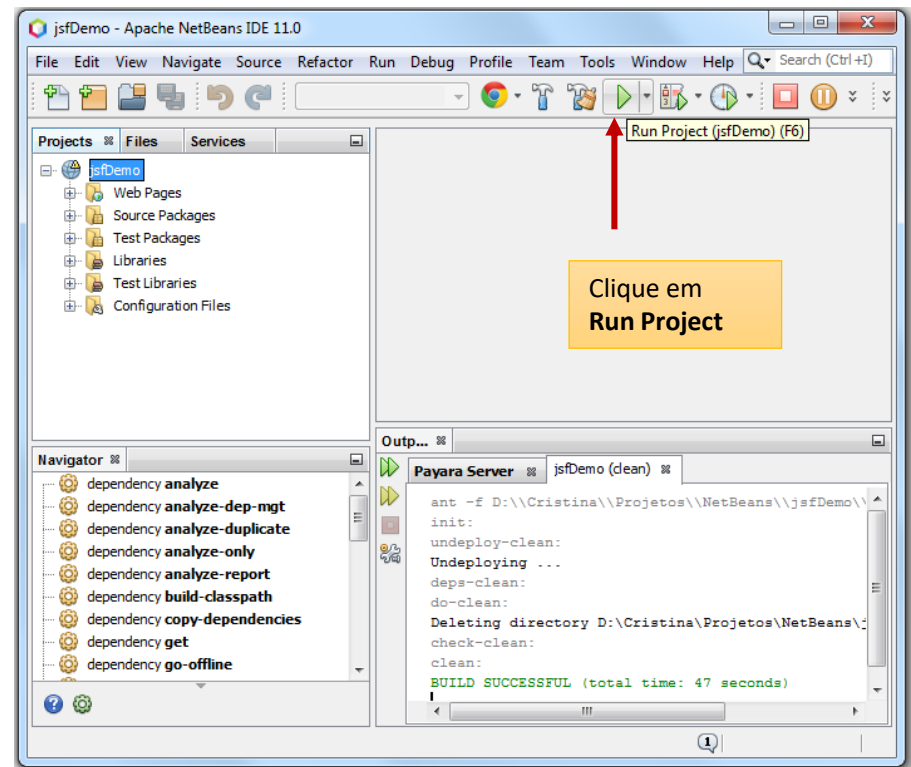
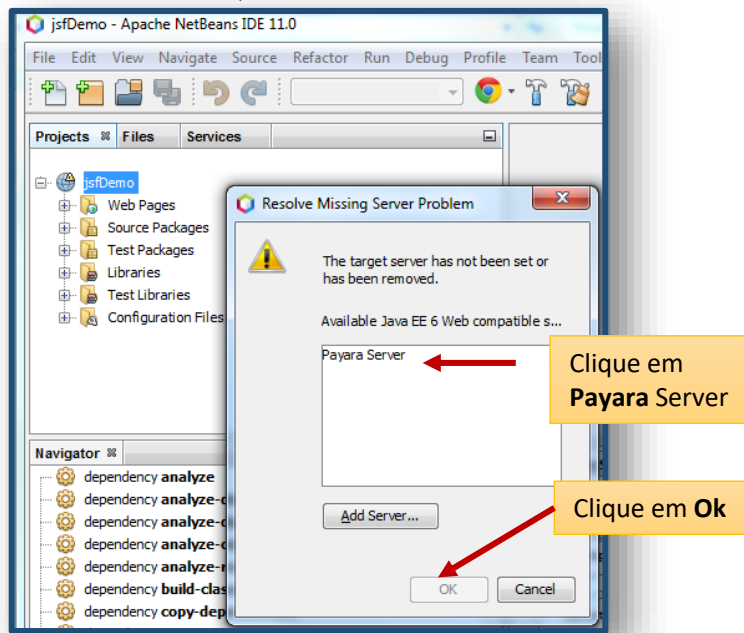
Clique no botão “**Resolve Problems**” para acertar a referência às bibliotecas **JUnit** quando abrir o projeto NetBeans.



Resolvendo “missing problems”

- Clique com o botão direito do mouse sobre o projeto jsfDemo para resolver a falta de definição do servidor

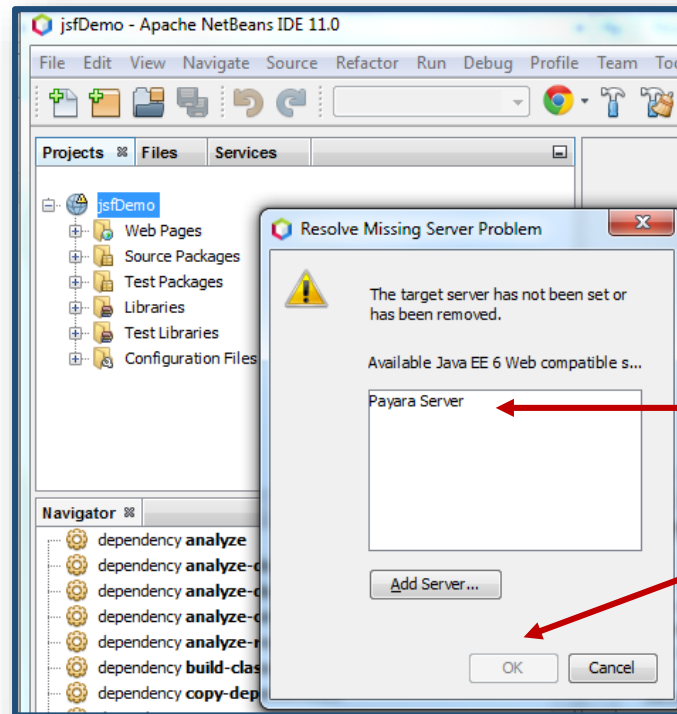
Resolve Missing Server Problem...



Resolvendo “missing problems”

- Clique com o **botão direito do mouse** sobre o projeto **jsfDemo** para resolver a falta de definição do servidor

Resolve Missing Server Problem...

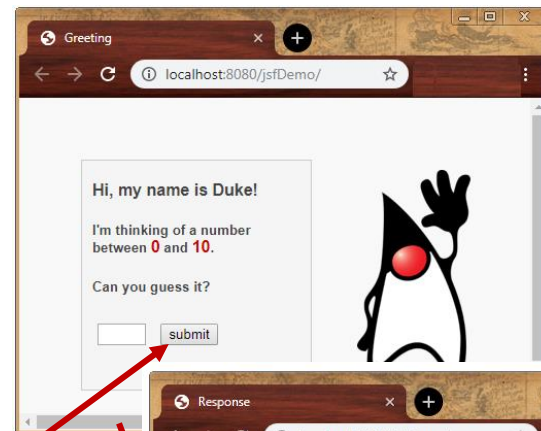
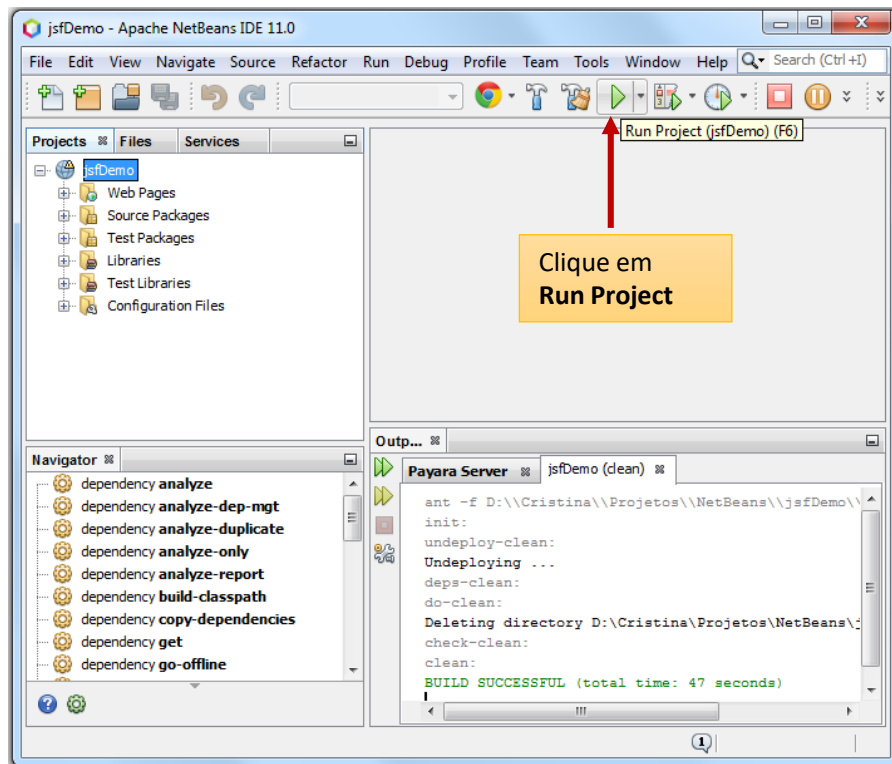


Clique em
Payara Server

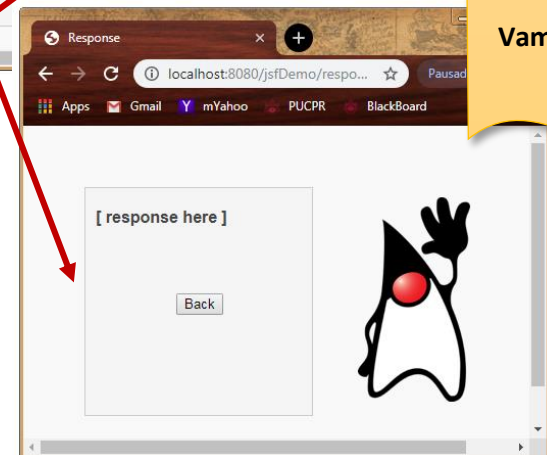
Clique em **Ok**

Projeto executando no Navegador

- Execute o projeto em um browser: clique com o botão direito do mouse no **jsfDemo**, na janela Projetos e selecione **Run** ou clique no Botão Executar Projeto na barra de ferramentas principal.
- O projeto é encapsulado e implantado no **Payara Server** e o navegador é aberto para exibir a página **index.xhtml** de Greetings (boas vindas).



Clique no botão
"Submit"
A página de
resposta
(**response.xhtml**)
é exibida,
e é estática

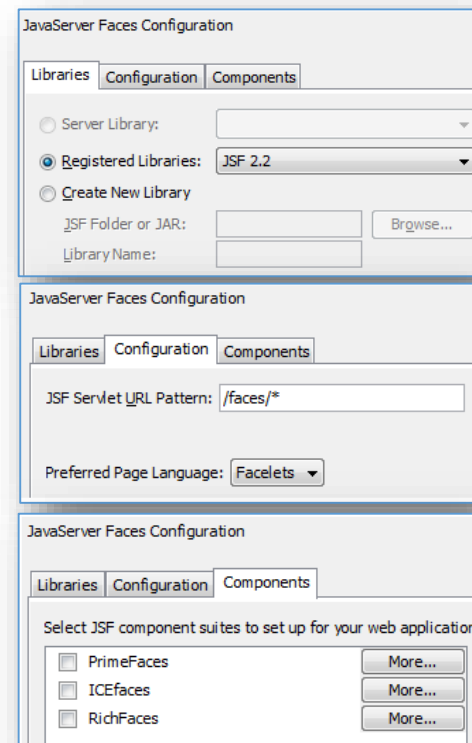
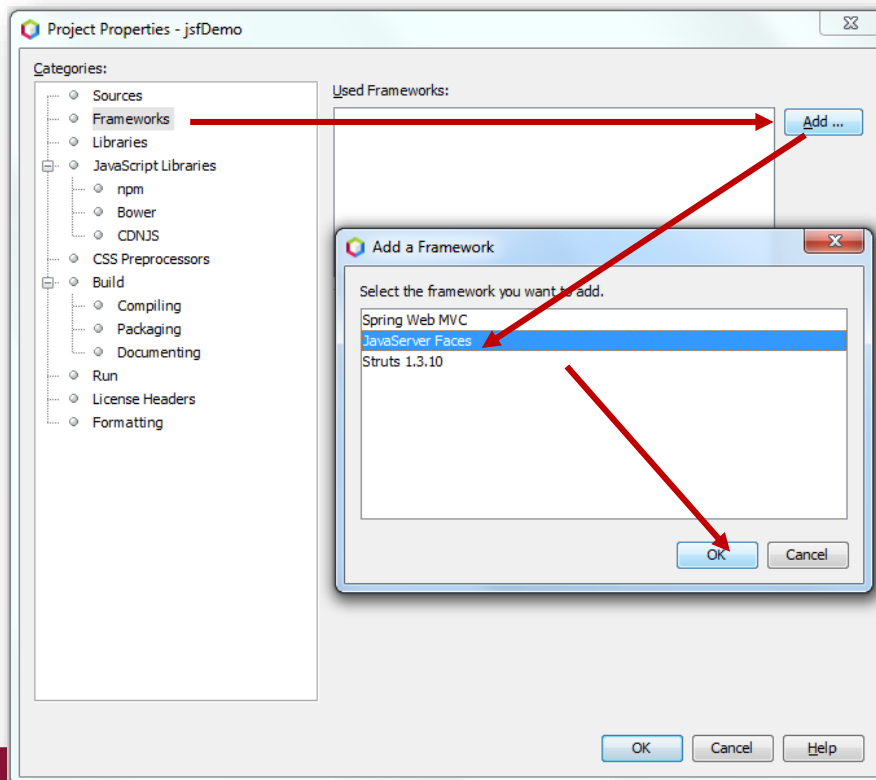


Até agora, as
páginas de
index.xhtml e
response.xhtml
são estáticas,
ou seja, **não s**
são geradas
dinamicamente.

**Vamos alterar
isso!**

JSF no Projeto

- Primeiro, vamos adicionar o recurso de JSF ao Projeto:
- Clique com o botão direito do mouse no projeto **jsfDemo** e escolha **Propriedades**, que abrir a janela Propriedades do Projeto.
- Selecione a categoria **Frameworks** e, em seguida clique no botão **Adicionar**.
- Selecione **JavaServer Faces**, e na caixa de diálogo **Adicionar um Framework**. Clique em **OK**.



Após selecionar **JavaServer Faces**, diversas opções de configuração ficarão disponíveis.

Clique em **OK** para concluir as alterações e sair da janela **Propriedades** do Projeto.

Conferindo o **web.xml**

- Depois de adicionar o suporte JSF ao seu projeto, o descritor de implantação **web.xml** do projeto é modificado para que tenha a aparência a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" 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-app_3_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>/faces/*</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```

Importante:

confirme se **web.xml**
contém só uma
entrada **<welcome-file>**



Criando um Managed Bean

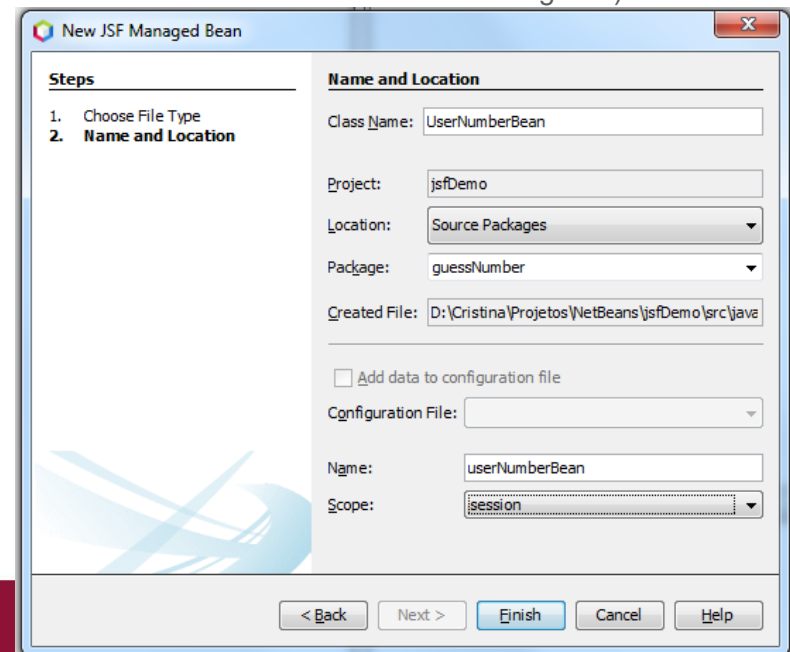
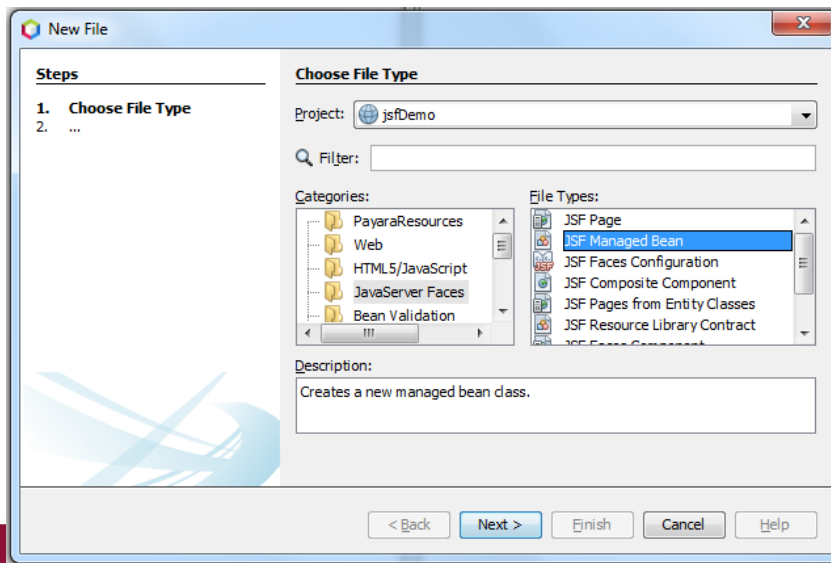
- Podemos usar os **Managed Beans** do JSF para:
 - processar dados do usuário e
 - retê-los entre as solicitações.
- Um **Managed Bean** é um **POJO** (**Plain Old Java Object**, ou Objeto Java Simples Antigo) que pode ser utilizado para armazenar dados e é gerenciado pelo contêiner (por exemplo, o **Payara Server** ou o **GlassFish Server**), utilizando o framework JSF.

Um **POJO** é essencialmente uma classe Java que contém um construtor público sem argumentos e está em conformidade com as convenções de nomenclatura do JavaBeans.

- Para tornar nossas páginas de estáticas para dinâmicas, precisaremos de um mecanismo que determine que o número inserido pelo usuário corresponde ao número aleatório selecionado atualmente, e também que ele retorne uma view apropriada para esse resultado.
- Para isso, utilizaremos o assistente de **Managed Beans** para criar um bean para essa finalidade.
- As páginas de Facelets que você criará na próxima seção precisarão acessar o **número digitado pelo usuário** e a **resposta gerada**. Para ativar esta opção, adicione as propriedades **userNumber** (número digitado) e **response** (resposta gerada) ao **Managed Bean**.

Criando um Managed Bean utilizando o Assistente

- Na janela **Projects**, clique com o botão direito do mouse no projeto **jsfDemo** e selecione **New > JSF Managed Bean**. (Se o **JSF Managed Bean** não estiver listado, selecione **Others**. Em seguida, selecione a opção **JSF Managed Bean** na categoria **JavaServer Faces**. Clique em **Próximo**).
- No assistente, informe o seguinte:
 - **Nome da Classe:** UserNumberBean
 - **Pacote:** guessNumber
 - **Nome:** UserNumberBean
 - **Escopo:** Session (os objetos e atributos do **managed bean** são mantidos durante a sessão do navegador)



Managed Bean criado

- Ao código gerado automaticamente, acrescente as **propriedades**:

```
Integer randomInt;  
Integer userNumber;  
String response;
```

- Acrescente os **métodos**:

```
Integer getUserNumber();  
Integer getUserNumber();  
public String getResponse();
```

- Acrescente as **Importações** (Alt-Shift-I):

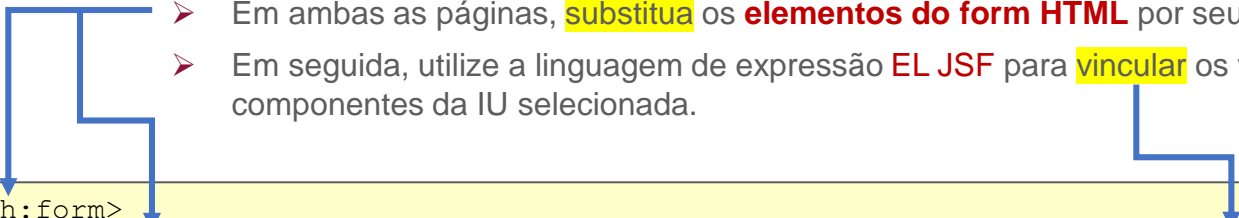
```
java.util.Random;  
javax.faces.context.FacesContext;  
javax.servlet.http.HttpSession;
```

```
package guessNumber;  
  
import javax.inject.Named;  
import javax.enterprise.context.SessionScoped;  
import java.io.Serializable;  
import java.util.Random;  
import javax.faces.context.FacesContext;  
import javax.servlet.http.HttpSession;  
  
@Named(value = "userNumberBean")  
@SessionScoped  
public class UserNumberBean implements Serializable {  
  
    Integer randomInt;  
    Integer userNumber;  
    String response;  
  
    public UserNumberBean() {  
        Random randomGR = new Random();  
  
        randomInt = randomGR.nextInt(10);  
        System.out.println("Duke's number: " + randomInt);  
    }  
  
    public Integer getUserNumber() {  
        return userNumber;  
    }  
  
    public Integer getUserNumber() {  
        this.userNumber = userNumber;  
    }  
  
    public String getResponse() {  
        if ((userNumber != null) && (userNumber.compareTo(randomInt) == 0)) {  
            //invalidate user session  
            FacesContext context = FacesContext.getCurrentInstance();  
            HttpSession session =  
                (HttpSession) context.getExternalContext().getSession(false);  
            session.invalidate();  
  
            return "Yay! You got it!";  
        } else {  
            return "<p>Sorry, " + userNumber + " isn't it.</p>"  
                + "<p>Guess again...</p>";  
        }  
    }  
}
```

Conecte o Managed Bean às Páginas

- Iremos explorar como é possível utilizar **UserNumberBean** e suas propriedades em páginas Web.
- O JSF permite isso utilizando a sua **linguagem de expressão** (EL-Expression Language).
- A EL é utilizada para vincular os valores da propriedade aos componentes da IU do JSF nas páginas Web da aplicação.
- Ações para as páginas **index.xhtml** e **response.xhtml**.

- Em ambas as páginas, **substitua** os **elementos do form HTML** por seus equivalente **JSF**.
- Em seguida, utilize a linguagem de expressão **EL JSF** para **vincular** os valores da propriedade aos componentes da IU selecionada.



```
<h:form>
  <h:inputText id="userNumber" size="2" maxlength="2" value="#{UserNumberBean.userNumber}"/>
  <h:commandButton id="submit" value="submit" action="response"/>
</h:form>
```

Importante: A **EL JSF** utiliza a sintaxe **#{}** , Onde são especificados o nome do **managed bean** e sua **propriedade**, separados por um ponto (**.**). Quando os dados do **form** forem enviados ao servidor, o **valor será salvo automaticamente na propriedade `userNumber`** utilizando o **setter** da propriedade (**`setUserNumber()`**). Além disso, quando a página for solicitada e um valor para **`userNumber`** já tiver sido definido, o valor será exibido automaticamente no componente **`inputText`** renderizado.>

Managed Bean acessado pela index.xhtml

- **Substitua** os elementos do form HTML por seus equivalente JSF
- Utilize a EL JSF para **vincular** os valores da propriedade aos componentes da IU selecionada
- **Importante:** verifique as alterações do código modificado, em relação ao original fornecido do exercício

index.xhtml



```
<!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>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link href="css/styleSheet.css" rel="stylesheet" type="text/css" />
    <h:outputStylesheet name="css/styleSheet.css" />
    <title>Greeting</title>
  </h:head>
  <body>
    <div id="mainContainer">
      <div id="left" class="subContainer greyBox">
        <h4>Hi, my name is Duke!</h4>
        <h5>I'm thinking of a number
          <br/>
          between
          <span class="highlight">0</span> and
          <span class="highlight">10</span>.</h5>
        <h5>Can you guess it?</h5>
        <h:form>
          <h:inputText id="userNumber" size="2" maxlength="2"
            value="#{UserNumberBean.userNumber}" />
          <h:commandButton id="submit" value="submit"
            action="response" />
        </h:form>
      </div>
      <div id="right" class="subContainer">
        <h:graphicImage url="/duke.png" alt="Duke waving" />
      </div>
    </div>
  </body>
</html>
```

Managed Bean acessado pela response.xhtml

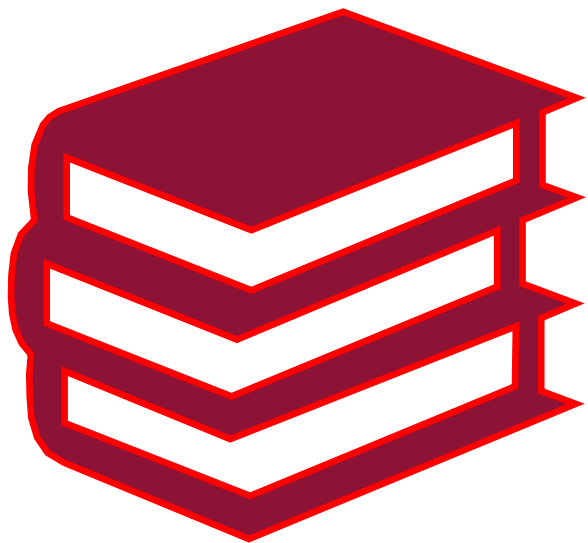
- **Substitua** os elementos do form HTML por seus equivalente JSF
- Utilize a EL JSF para **vincular** os valores da propriedade aos componentes da IU selecionada
- **Importante:** verifique as alterações do código modificado, em relação ao original fornecido do exercício

response.xhtml



```
<!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>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link href="css/stylesheet.css" rel="stylesheet" type="text/css" />
    <h:outputStylesheet name="css/stylesheet.css" />
    <title>Response</title>
  </h:head>
  <body>
    <div id="mainContainer">
      <div id="left" class="subContainer greyBox">
        <h4><h:outputText escape="false"
          value="#{UserNumberBean.response}" /></h4>

        <h:form prependId="false">
          <h:commandButton id="backButton" value="Back"
            action="index" />
        </h:form>
      </div>
      <div id="right" class="subContainer">
        <h:graphicImage url="/duke.png" alt="Duke waving" />
      </div>
    </div>
  </body>
</html>
```

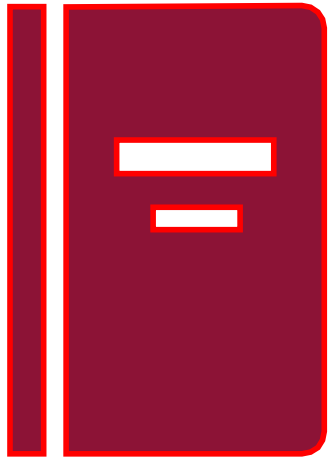



Glossário

Glossário

- **POJO (Plain Old Java Object, ou Objeto Java Simples Antigo)**: é essencialmente uma classe Java que contém um construtor público sem argumentos, e está em conformidade com as convenções de nomenclatura do JavaBeans. Um **Managed Bean** é um POJO.
- **Glassfish Server**: é um servidor de aplicação Web e/ou Corporativas de código aberto (open source) liderado pela Sun Microsystems para a plataforma Java EE. Sua versão proprietária é chamada Sun GlassFish Enterprise Server.
- **Payara Server**: é um servidor de aplicações Web e/ou Corporativas de código aberto (open source) derivado do GlassFish Server Open Source Edition. Foi criado em 2014 pela C2B2 Consulting como substituto do **GlassFish** depois que a Oracle anunciou que estava descontinuando o suporte comercial para o GlassFish.
- **Session Scope**: todos os objetos e atributos vinculados ao **Managed Bean**, sobrevivem durante toda a sessão do usuário. A sessão é definida pelo vínculo do navegador do usuário com o servidor. Desta forma, se usuário abrir dois navegadores, ele estará criando duas sessões com o servidor..





Referências

Referências

- Material baseado no tutorial NetBeans em
 - Introdução ao JavaServer Faces 2.x
 - https://netbeans.org/kb/docs/web/jsf20-intro_pt_BR.html#staticPage



PUCPR

GRUPO MARISTA

PROFESSOR CONTEUDISTA

Cristina Verçosa Pérez Barrios de Souza, Profa. Dra.

© PUCPR. Todos os direitos reservados.

Nenhum texto pode ser reproduzido sem prévia autorização.