




Java WebDeveloper - Aula 06a - Noite Seg, Qua, Sex - Noite



Tema da aula:

Projeto JSF, Serializable, Script, JDBC, MySql, Dao, Configuração JSF, Lista, Tema PrimeFaces, Panel, Datalist

—  Prof Edson Belém - profedsonbelem@gmail.com (mailto:profedsonbelem@gmail.com) 🕒 Quarta, Abr 08, 2020

JavaServer Faces



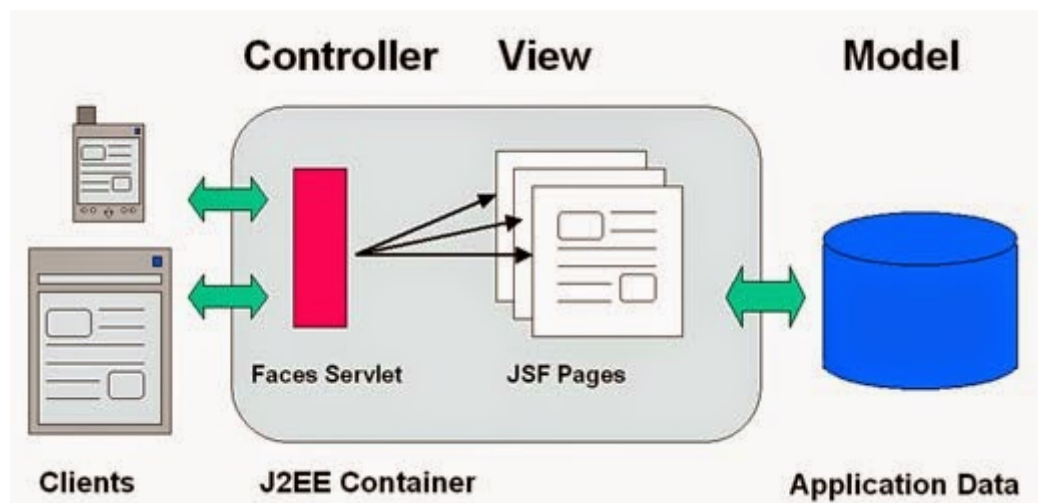
JavaServer Faces (JSF) é uma especificação Java para a construção de interfaces de usuário baseadas em componentes para aplicações web. Possui um modelo de programação dirigido a eventos, abstraindo os detalhes da manipulação dos eventos e organização dos componentes, permitindo que o programador se concentre na lógica da aplicação.

Foi formalizada como um padrão através do Java Community Process e faz parte da Java Platform, Enterprise Edition. JSF 2 utiliza Facelets como seu sistema de template padrão. Outras tecnologias da camada de visão, como XUL também podem ser empregadas. Em contraste, JSF 1.x utiliza JavaServer Pages (JSP) como seu sistema de template padrão.

Histórico

O JavaServer Faces ganhou expressão na versão 1.1 quando implementado pela comunidade utilizando a especificação 127 do Java Community Process, evidenciando maturidade e segurança. Hoje ele está na versão 2.0 da especificação 252 do JCP. A fundação Apache vem realizando esforços na implementação da especificação através do projeto MyFaces. O reconhecimento do trabalho é visto por diversas empresas, tanto é que a Oracle doou os fontes do ADF Faces, conjunto de mais de 100 componentes JSF, para o projeto MyFaces que o denominará de Trinidad. O JSF é atualmente considerado pela comunidade Java como a última palavra em termos de desenvolvimento de aplicações Web utilizando Java, resultado da experiência e maturidade adquiridas com o JSP/Servlet (Model1), Model2 (MVC) e Struts.

Como trabalha

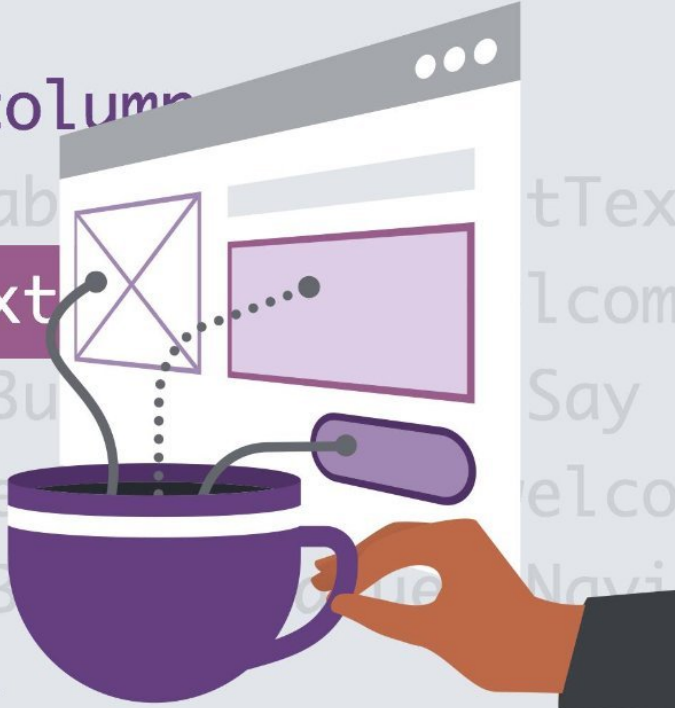


JavaServer Faces é baseada em um modelo de desenho de IU (interface de usuário) baseada em componentes, usando arquivos XML chamados de modelos de visão ou Facelets views. Os pedidos são processados pelo FacesServlet, que carrega o modelo de visão adequado, constrói uma árvore de componentes, processa os eventos e apresenta a resposta, normalmente na linguagem HTML, para o cliente. O estado de componentes de interface do usuário e outros objetos de interesse de escopo, é salvo no final de cada pedido em um processo chamado stateSaving (nota: transiente true) e restaurado na próxima criação desta visão. Objetos e estados podem ser salvos ou no cliente ou no servidor.

Características

```
<h:form>  
  <h:panelGrid column  
    <h:outputLab  
    <h:inputText  
    <h:commandBu  
    <h:outputTe  
    <h:commandB  
</h:panelGrid>
```

```
<h:form>  
  <h:panelGrid column  
    <h:outputLab  
    <h:inputText  
    <h:commandBu  
    <h:outputTe  
    <h:commandB  
</h:panelGrid>
```

An illustration of a hand pouring coffee from a purple cup into a web browser window. The browser window has a grey header with three white dots. Inside the window, there is a purple square with a white 'X' and a purple rectangle with a white dot. A dotted line connects the 'X' to the dot. A purple coffee spoon is shown pouring coffee from the cup into the browser window. The background is light grey with faint, repeating text: 'tTex', 'lcom', 'Say', 'elco', 'Navi'.

```
<h:form>
  <h:panelGrid column
    <h:outputLab
    <h:inputText
    <h:commandBu
    <h:outputTe
    <h:commandB
  </h:panelGrid>
```

```
<h:form>
  <h:panelGrid column
    <h:outputLab
    <h:inputText
    <h:commandBu
    <h:outputTe
    <h:commandB
  </h:panelGrid>
```

```
<h:form>
  <h:panelGrid column
    <h:outputLab
    <h:inputText
    <h:commandBu
    <h:outputTe
    <h:commandB
  </h:panelGrid>
```

```
<h:form>  
  <h:panelGrid columns="2">  
    <h:outputLabel value="Name" />  
    <h:inputText value="" />  
    <h:commandButton value="Submit" />  
  </h:panelGrid>  
</h:form>
```

```
<h:form>
  <h:panelGrid column
    <h:outputLab
    <h:inputText
    <h:commandBu
    <h:outputTe
    <h:commandB
  </h:panelGrid>
```

```
<h:form>
  <h:panelGrid column
    <h:outputLab
    <h:inputText
    <h:commandBu
    <h:outputTe
    <h:commandB
  </h:panelGrid>
```

- ```
<h:form>
 <h:panelGrid column
 <h:outputLab
 <h:inputText
 <h:commandBu
 <h:outputTe
 <h:commandB
 </h:panelGrid>
```
- 

```
<h:form>
 <h:panelGrid columns="2">
 <h:outputLabel value="Name" />
 <h:inputText value="" />
 <h:commandButton value="Submit" />
 </h:panelGrid>
</h:form>
```

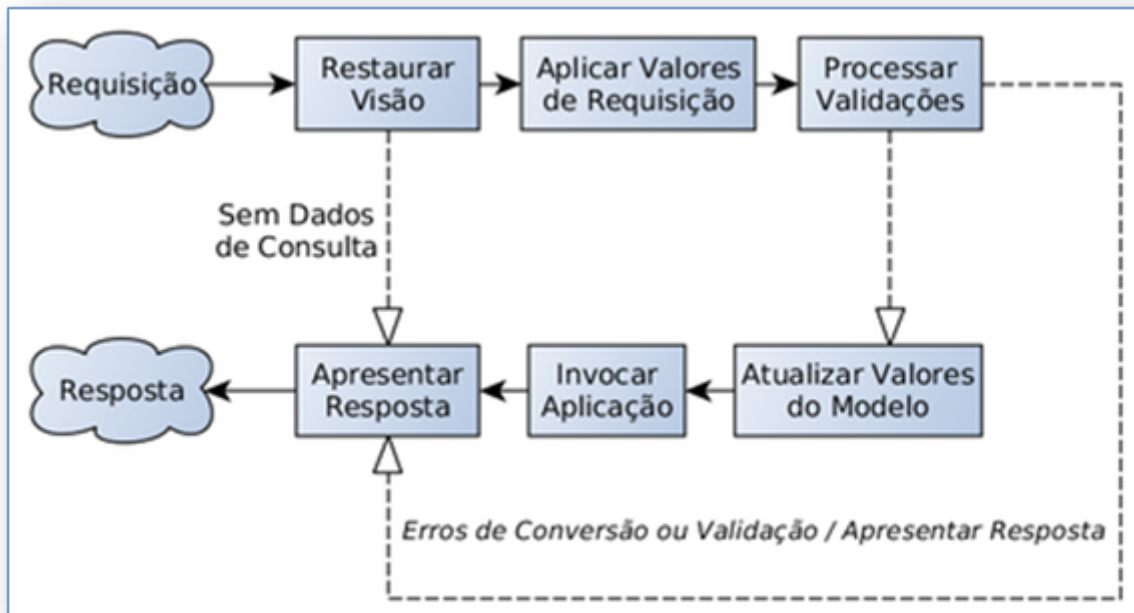


Diagrama mostrando o fluxo do ciclo de vida do JSF.

Ciclo de vida do JSF é o nome dado à sequência de processamento realizada na implementação JSF para a geração das visões. O JSF, diferente de outros frameworks, possui um processamento de requisição dividido em seis fases:

1. **Restauração da visão** - o objetivo principal desta fase é construir a árvore de componentes. Ela utiliza o template e cria a árvore inicial através da análise da requisição. Após isto, ela salva o estado da árvore no objeto FacesContext. Nas requisições subsequentes, ela cria a árvore do estado e procede a execução do resto do ciclo de vida.
2. **Aplicação dos valores de requisição** - o JSF pega cada componente da árvore começando com a raiz e a cria ou recupera do objeto FacesContext. Cada componente na árvore gerencia seus próprios valores e toma-os dos parâmetros, cookies e cabeçalhos da requisição HTTP.
3. **Validações de processo** - o JSF realiza a conversão e validação sobre todos os componentes começando com o raiz. O valor submetido de cada componente é convertido em um objeto e validado chamando-se o validador registrado. O JSF salva o valor submetido. Se ocorrer um erro durante a conversão ou validação, o ciclo de vida escapa diretamente para a fase de "apresentação da resposta".
4. **Atualização de valores de modelo** - durante esta fase, o valor do componente é passado para o modelo através da atualização das propriedades dos backing beans.
5. **Invocação da aplicação** - a manipulação de evento para cada ação e ouvidor de ação é executada começando com o(s) ouvidor(es) de ação e então a chamada do método de ação.

## Recursos visuais

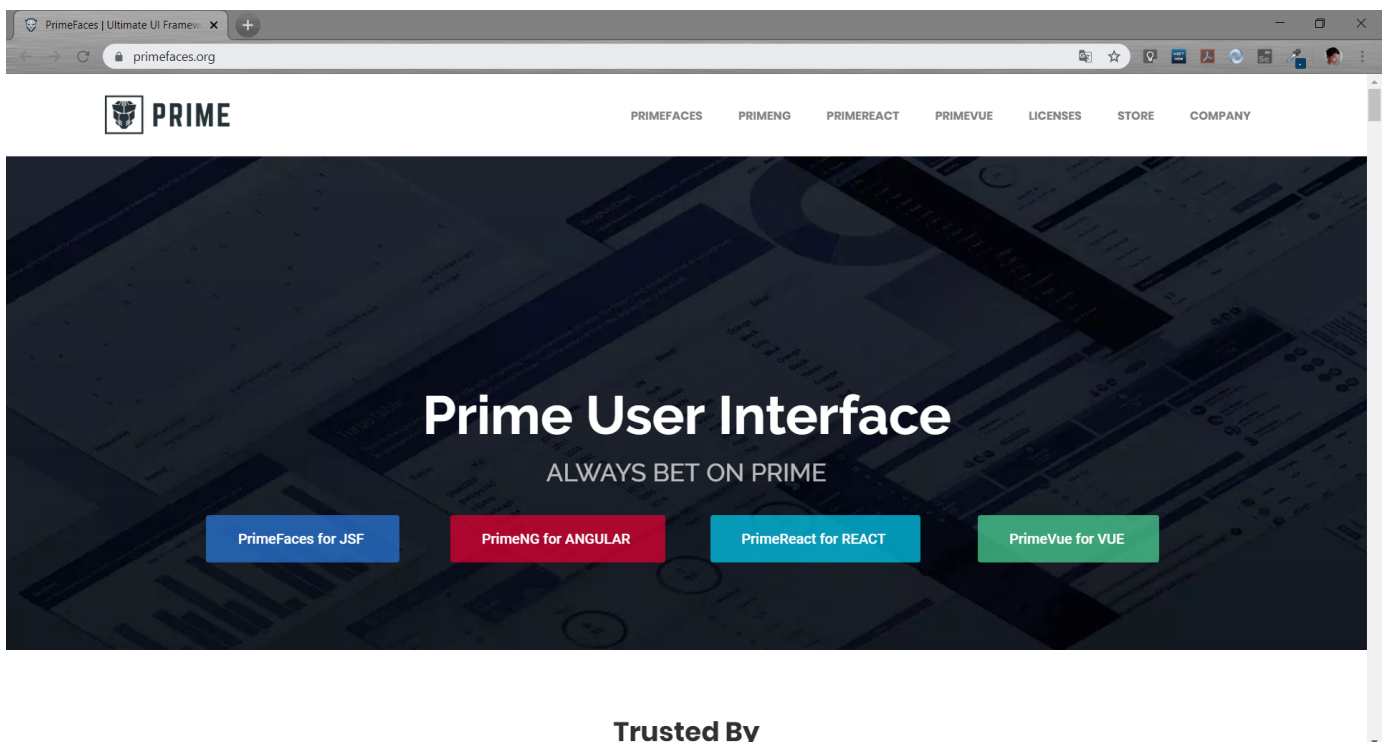
O framework JSF permite a inserção, via IDE, de:

- Folhas de estilo (CSS);
- Comandos em JavaScript;
- Metodologia Ajax.

## JSF inclui

- Suporte a internacionalização e acessibilidade;
- Um conjunto padrão de componentes de interface de usuário que possibilitam validação padronizada;
- Duas bibliotecas de etiqueta ("tag libraries") especiais do JavaServer Pages (JSP) para expressar a interface do JavaServer Faces dentro de uma página JSP;
- Um modelo de eventos do lado servidor ("server-side event model");
- Gerência de estados;
- Managed Beans;
- Linguagem de Expressão Unificada ("Unified Expression Language") para JSP 2.0 e JSF 1.2.

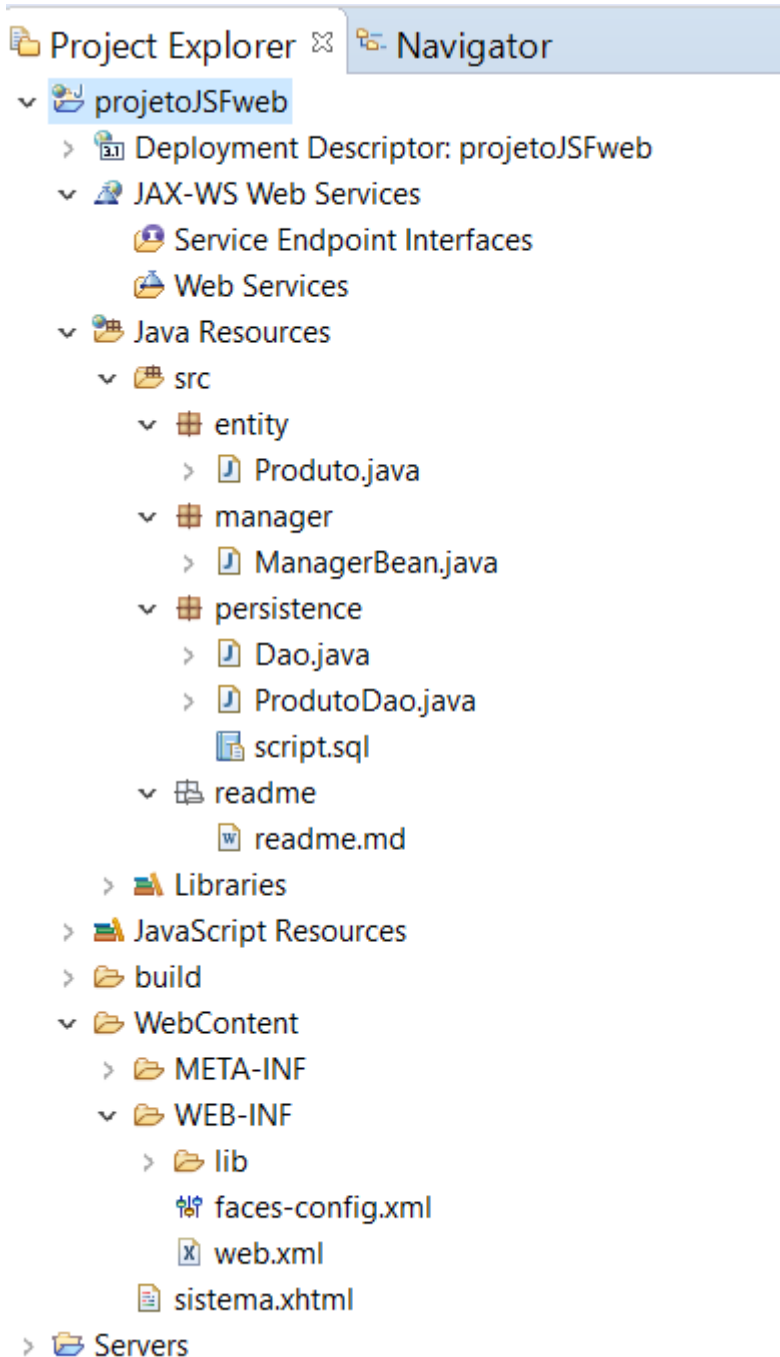
## PrimeFaces (<http://primefaces.org/>)



→ **Showcase do PrimeFaces** (<https://www.primefaces.org/showcase/>)  
<https://www.primefaces.org/showcase/> (<https://www.primefaces.org/showcase/>)

→ **Temas** (<https://www.primefaces.org/themes/>)  
<https://www.primefaces.org/themes/> (<https://www.primefaces.org/themes/>)

👁 **Estrutura do projeto depois de finalizado:**



## Bibliotecas utilizadas (JSF):

---

▼ 📁 lib

- 📄 all-themes-1.0.10.jar
- 📄 commons-beanutils-1.8.0.jar
- 📄 commons-collections-3.2.1.jar
- 📄 commons-digester-2.0.jar
- 📄 commons-logging-1.1.1.jar
- 📄 javax.faces-2.0.11.jar
- 📄 jstl.jar
- 📄 mysql-connector-java-5.1.23-bin.jar
- 📄 primefaces-5.0-sources.jar
- 📄 primefaces-5.0.jar
- 📄 standard.jar



## Readme.md (<http://Readme.md>)

JSF (Concorrente ==> JEE Servlet)(EspecificaÃ§Ã£o)

# Parte um

# Configuracao

#pacotes

entity, manager, persistence (view)

view --> Manger

tela --> Classe

xhtml

##



## Produto.java

```
1 package entity;
2
3 import java.io.Serializable;
4
5 public class Produto implements Serializable {
6
7 private static final long serialVersionUID = 1L;
8 private Integer id;
9 private String nome;
10 private String autor;
11 private Double preco;
12
13 public Produto() {
14 }
15
16 public Produto(Integer id, String nome, String autor, Double preco) {
17 this.id = id;
18 this.nome = nome;
19 this.autor = autor;
20 this.preco = preco;
21 }
22
23 @Override
24 public String toString() {
25 return "Produto [id=" + id + ", nome=" + nome + ", autor="
26 + autor + ", preco=" + preco + "]";
27 }
28
29 public Integer getId() {
30 return id;
31 }
32
33 public void setId(Integer id) {
34 this.id = id;
35 }
36
37 public String getNome() {
38 return nome;
39 }
40
41 public void setNome(String nome) {
42 this.nome = nome;
43 }
44
45 public String getAutor() {
46 return autor;
47 }
48
49 public void setAutor(String autor) {
50 this.autor = autor;
51 }
52
53 public Double getPreco() {
54 return preco;
```



```

55 }
56
57 public void setPreco(Double preco) {
58 this.preco = preco;
59 }
60
61 }

```

## script.sql

```

1 mysql -u root -p
2
3 create database BDJSFUM;
4 use BDJSFUM;
5
6 create table produto(
7 id int primary key auto_increment,
8 nome varchar (50) unique ,
9 autor varchar (50),
10 preco double
11);
12
13 show tables;
14
15 insert into produto (id,nome,autor,preco) values
16 (null, 'Java Certificacao', 'kathy sierra',200),
17 (null, 'Java OO', 'Douglas Rocha',80),
18 (null, 'Java EE', 'Cleiton Sampaio',70);
19
20 select * from produto;

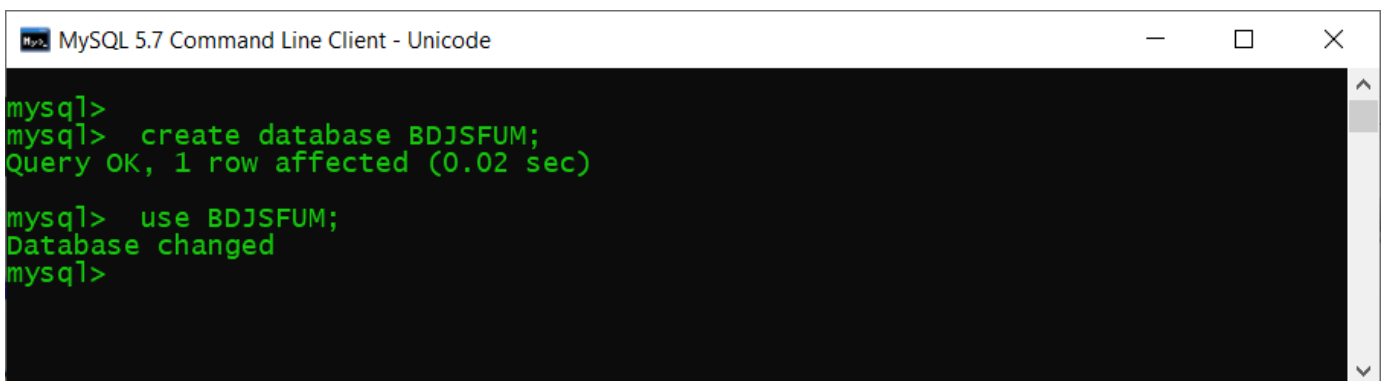
```

 Visualizando no banco.

```

1 create database BDJSFUM;
2 use BDJSFUM;

```



```

MySQL 5.7 Command Line Client - Unicode
mysql>
mysql> create database BDJSFUM;
Query OK, 1 row affected (0.02 sec)

mysql> use BDJSFUM;
Database changed
mysql>

```

```

1 create table produto(
2 id int primary key auto_increment,
3 nome varchar (50) unique ,
4 autor varchar (50),
5 preco double
6);
7
8 show tables;

```

```

mysql>
mysql> create table produto(
-> id int primary key auto_increment,
-> nome varchar (50) unique ,
-> autor varchar (50),
-> preco double
->);
Query OK, 0 rows affected (0.08 sec)

mysql>
mysql> show tables;
+-----+
| Tables_in_bdjsfum |
+-----+
| produto |
+-----+
1 row in set (0.00 sec)

```

```

1 insert into produto (id,nome,autor,preco) values
2 (null, 'Java Certificacao', 'kathy sierra',200),
3 (null, 'Java OO', 'Douglas Rocha',80),
4 (null, 'Java EE', 'Cleiton Sampaio',70);
5
6 select * from produto;

```

```

mysql>
mysql>
mysql> insert into produto (id,nome,autor,preco) values
-> (null, 'Java Certificacao', 'kathy sierra',200),
-> (null, 'Java OO', 'Douglas Rocha',80),
-> (null, 'Java EE', 'Cleiton Sampaio',70);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
mysql> select * from produto;
+----+-----+-----+-----+
| id | nome | autor | preco |
+----+-----+-----+-----+
1	Java Certificacao	kathy sierra	200
2	Java OO	Douglas Rocha	80
3	Java EE	Cleiton Sampaio	70
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```



## Dao.java

```
1 package persistence;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7
8 public class Dao {
9
10 Connection con;
11 PreparedStatement stmt;
12 ResultSet rs;
13
14 public void open() throws Exception {
15 Class.forName("com.mysql.jdbc.Driver");
16 con = DriverManager.getConnection
17 ("jdbc:mysql://localhost:3306/BDJSFUM", "root", "coti");
18 }
19
20 public void close() throws Exception {
21 con.close();
22 }
23
24 }
```



## ProdutoDao.java

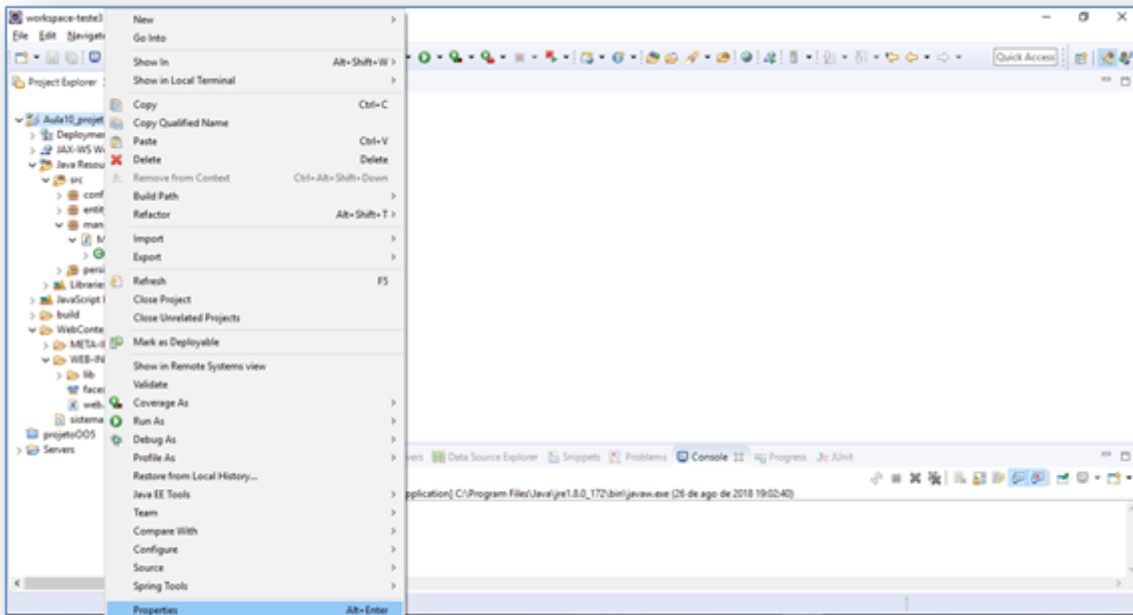
```

1 package persistence;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import entity.Produto;
7
8 public class ProdutoDao extends Dao {
9
10 public void create(Produto p) throws Exception {
11 open();
12 stmt = con.prepareStatement("insert into produto values
13 (null,?,?,?)");
14 stmt.setString(1, p.getNome());
15 stmt.setString(2, p.getAutor());
16 stmt.setDouble(3, p.getPreco());
17 stmt.execute();
18 close();
19 }
20
21 public List<Produto> findAll() throws Exception {
22 open();
23 stmt = con.prepareStatement("select * from produto");
24 rs = stmt.executeQuery();
25 List<Produto> lista = new ArrayList<Produto>();
26 while (rs.next()) {
27 Produto p = new Produto();
28 p.setId(rs.getInt(1));
29 p.setNome(rs.getString(2));
30 p.setAutor(rs.getString(3));
31 p.setPreco(rs.getDouble(4));
32 lista.add(p);
33 }
34 close();
35 return lista;
36 }
37
38 public static void main(String[] args) {
39 try {
40 Produto p = new Produto(null, "Nodejs", "belem", 20.);
41 ProdutoDao dao = new ProdutoDao();
42 dao.create(p);
43
44 dao.findAll().stream().forEach(x -> {
45 System.out.println(x.getNome() + ", "
46 + x.getPreco());
47 });
48 } catch (Exception ex) {
49 ex.printStackTrace();
50 }
51 }
52 }
53
54 }

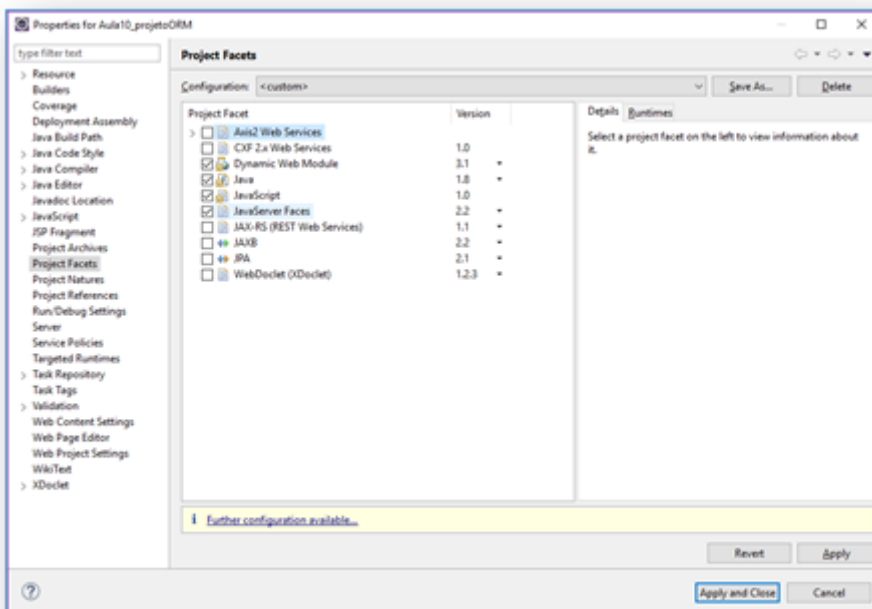
```


⚠ Para habilitar o JSF no projeto:

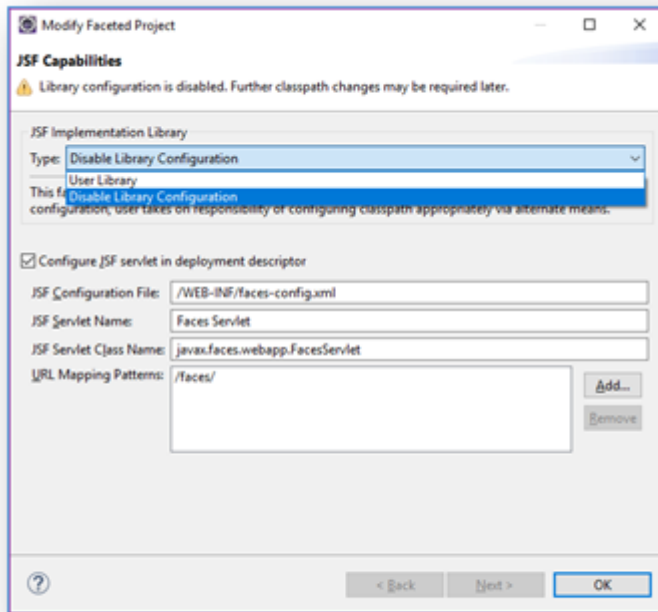
✎ Clicar no projeto com o botão direito.




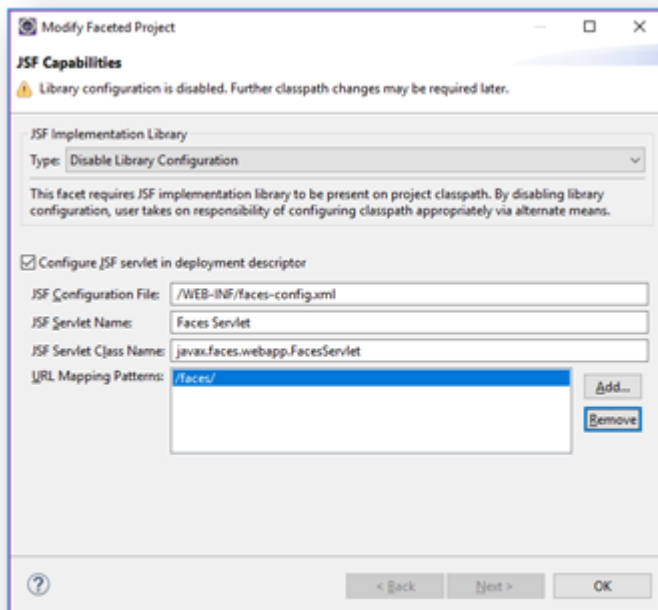
✎ Clicar em "project facets" ➡ marcar "java server faces" -> clicar no link "further configuration"




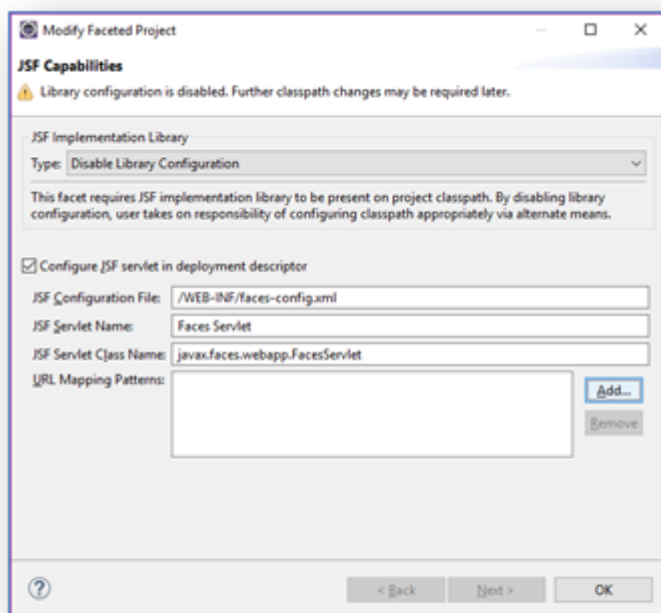
 Clicar em type e selecionar "disable library"




 Clicar em "url mapping" → clicar em "remove"

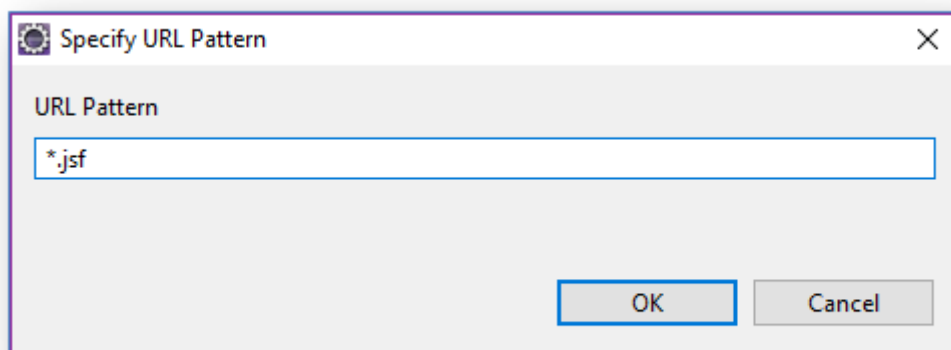


 Clicar em "add"




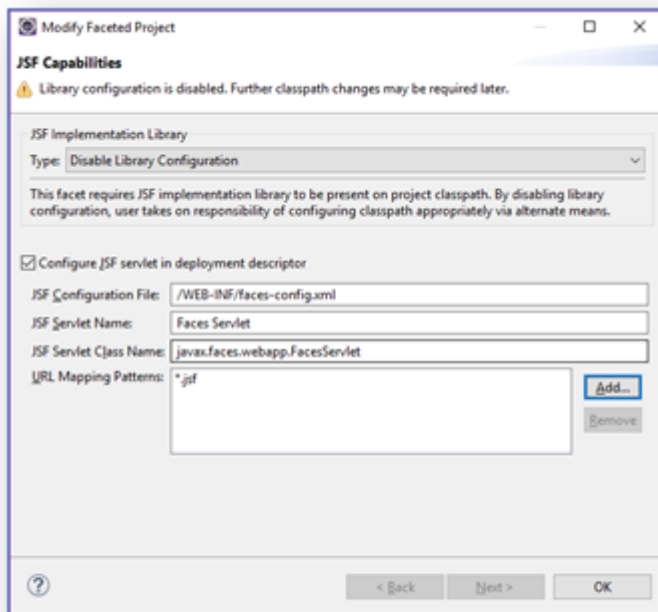
The screenshot shows the 'Modify Faceted Project' dialog box with the 'JSF Capabilities' tab selected. A warning message at the top states: 'Library configuration is disabled. Further classpath changes may be required later.' Below this, the 'JSF Implementation Library' section shows 'Type: Disable Library Configuration'. A note explains that this facet requires JSF implementation library to be present on project classpath. The 'Configure JSF servlet in deployment descriptor' checkbox is checked. The 'JSF Configuration File' is set to '/WEB-INF/faces-config.xml'. The 'JSF Servlet Name' is 'Faces Servlet' and the 'JSF Servlet Class Name' is 'javax.faces.webapp.FacesServlet'. The 'URL Mapping Patterns' list is empty, with an 'Add...' button next to it. At the bottom are '< Back', 'Next >', and 'OK' buttons.


 Escrever a "url pattern" → ok

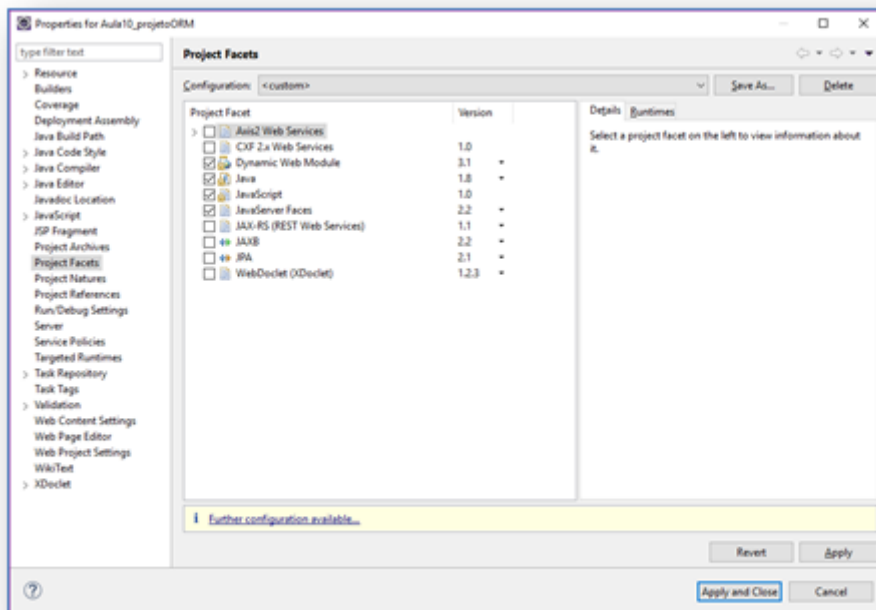


The screenshot shows the 'Specify URL Pattern' dialog box. The 'URL Pattern' text field contains the text '\*jsf'. At the bottom are 'OK' and 'Cancel' buttons.

 Clicar em "ok"



 Clicar em "apply and close"



 ManagerBean.java



```

1 package manager;
2
3 import java.util.List;
4
5 import javax.faces.bean.ManagedBean;
6 import javax.faces.bean.RequestScoped;
7
8 import entity.Produto;
9 import persistence.ProdutoDao;
10
11 @ManagedBean(name = "mb")
12 @RequestScoped
13 public class ManagerBean {
14
15 // unidade e listr
16 // #{mb.produto.nome}
17
18 private Produto produto; // entrada
19 private List<Produto> produtos; // lista
20
21 public ManagerBean() {
22 this.produto = new Produto();
23 }
24
25 public Produto getProduto() {
26 return produto;
27 }
28
29 public void setProduto(Produto produto) {
30 this.produto = produto;
31 }
32
33 public List<Produto> getProdutos() {
34 try {
35 produtos = new ProdutoDao().findAll();
36 } catch (Exception ex) {
37
38 }
39 return produtos;
40 }
41
42 public void setProdutos(List<Produto> produtos) {
43 this.produtos = produtos;
44 }
45 // #{mb.produtos}
46 }

```

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3 xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5 http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6 id="WebApp_ID" version="3.1">
7 <display-name>projetoJSFweb</display-name>
8 <welcome-file-list>
9 <welcome-file>index.html</welcome-file>
10 <welcome-file>index.htm</welcome-file>
11 <welcome-file>index.jsp</welcome-file>
12 <welcome-file>default.html</welcome-file>
13 <welcome-file>default.htm</welcome-file>
14 <welcome-file>default.jsp</welcome-file>
15 </welcome-file-list>
16 <servlet>
17 <servlet-name>Faces Servlet</servlet-name>
18 <servlet-class>javax.faces.webapp.FacesServlet
19 </servlet-class>
20 <load-on-startup>1</load-on-startup>
21 </servlet>
22 <servlet-mapping>
23 <servlet-name>Faces Servlet</servlet-name>
24 <url-pattern>*.jsf</url-pattern>
25 </servlet-mapping>
26
27 <context-param>
28 <param-name>primefaces.THEME</param-name>
29 <param-value>blitzer</param-value>
30 </context-param>
31
32 </web-app>

```






sistema.xhtml


```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4 xmlns:ui="http://java.sun.com/jsf/facelets"
5 xmlns:h="http://java.sun.com/jsf/html"
6 xmlns:f="http://java.sun.com/jsf/core"
7 xmlns:p="http://primefaces.org/ui">
8 <h:head>
9 </h:head>
10
11 <h:body>
12 <p:panel header="Consulta dos Livros">
13 <h:form id="form2">
14 <p:dataList value="#{mb.produtos}" var="item">
15 #{item.nome} , #{item.preco}
16 </p:dataList>
17 </h:form>
18 </p:panel>
19 </h:body>
20
21
22 </html>
```



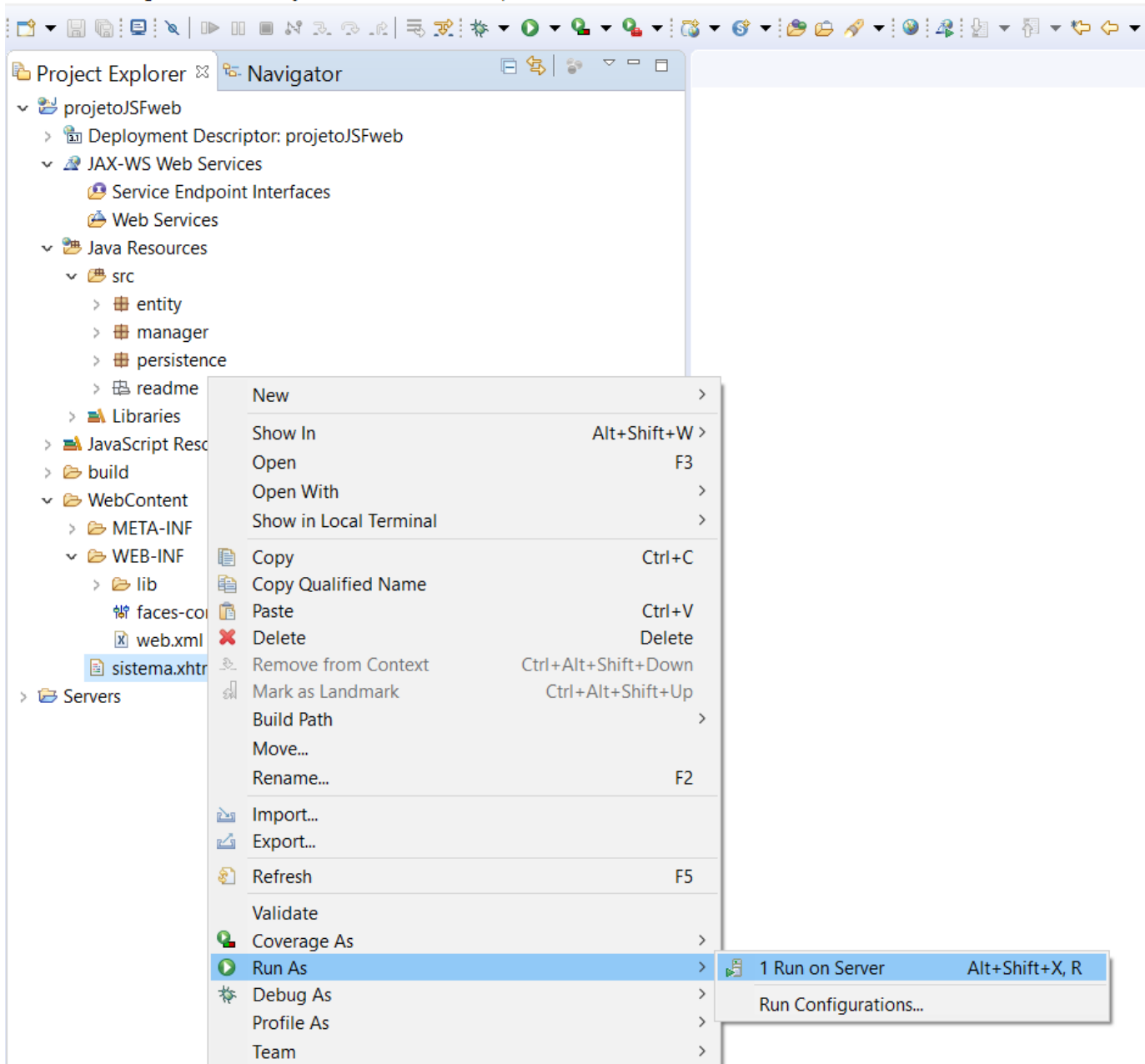
**Para rodar a página:**

---

 Clique na classe com o botão direito  run as  run on server

 workspace20195 - Eclipse IDE

File Edit Navigate Search Project Run Window Help



The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays the project structure for 'projetoJSFweb'. The Navigator on the right shows the 'sistema.xhtml' file selected. A context menu is open over the 'sistema.xhtml' file, listing various actions. The 'Run As' option is highlighted, and a sub-menu is visible showing '1 Run on Server' and 'Run Configurations...'. The 'Run on Server' option is also highlighted in the sub-menu.

Project Explorer


- projetoJSFweb
  - Deployment Descriptor: projetoJSFweb
  - JAX-WS Web Services
    - Service Endpoint Interfaces
    - Web Services
  - Java Resources
    - src
      - entity
      - manager
      - persistence
      - readme
    - Libraries
    - JavaScript Resources
    - build
    - WebContent
      - META-INF
      - WEB-INF
        - lib
        - faces-config.xml
        - web.xml
        - sistema.xhtml
  - Servers

Navigator

Context Menu:

- New
- Show In (Alt+Shift+W)
- Open (F3)
- Open With
- Show in Local Terminal
- Copy (Ctrl+C)
- Copy Qualified Name
- Paste (Ctrl+V)
- Delete
- Remove from Context (Ctrl+Alt+Shift+Down)
- Mark as Landmark (Ctrl+Alt+Shift+Up)
- Build Path
- Move...
- Rename... (F2)
- Import...
- Export...
- Refresh (F5)
- Validate
- Coverage As
- Run As
  - 1 Run on Server (Alt+Shift+X, R)
  - Run Configurations...
- Debug As
- Profile As
- Team

Selecionar o tomcat ➡ Finish

 Run On Server

**Run On Server**  
Select which server to use

How do you want to select the server?

☒ Choose an existing server


☐ Manually define a new server

Select the server that you want to use:

| Server                          | State   |
|---------------------------------|---------|
| ▼ localhost                     |         |
| Tomcat v8.0 Server at localhost | Stopped |

Apache Tomcat v8.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules. [Columns...](#)

☐ Always use this server when running this project

 < Back Next > **Finish** Cancel

Aguardar abrir o browser

 <http://localhost:8080/projetoJSFweb/sistema.jsf>

localhost:8080/projetoJSFweb/si: x +

localhost:8080/projetoJSFweb/sistema.jsf

**Consulta dos Livros**

- Java Certificacao , 200.0
- Java OO , 80.0
- Java EE , 70.0

