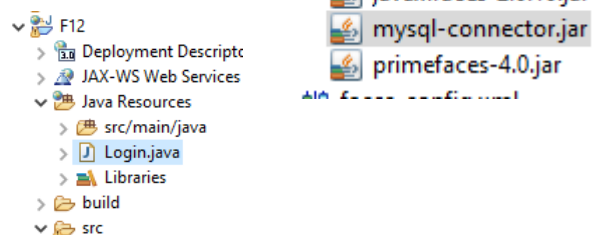


## TAREFAS

- Faz o upload da base de dados “*login\_java.sql*” para o WampServer.
- Adiciona a biblioteca *mysql-connector.jar* ao teu projeto.

- Cria uma classe para conexão à base de dados de nome “**Ligar**” (se necessário, revê a resolução da ficha de trabalho n.º1).



O conteúdo será semelhante ao seguinte:

```
// Criar uma função de nome OL que proceda à ligação à BD desejada
public Connection OL() {
    Connection con = null;
    try{
        //Indica que será utilizado o driver connector/J
        Class.forName("com.mysql.cj.jdbc.Driver");

        /* Liga ao servidor BD Local, com o utilizador root e pwd nula, acedendo à bd:
        VendasCD*/
        con = DriverManager.getConnection("jdbc:mysql://localhost/login_java", "root", "");
        System.out.println("Ligação efetuada com sucesso!");
    }
    catch(ClassNotFoundException cnfe) {
        System.out.println("erro:" + cnfe.getMessage());
    }
    catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return con;
}

public void FecharLigar(Connection con) {
    try {
        con.close();
    }
    catch (SQLException e) {
        // TODO Auto-generated catch block
        System.out.println("errofechar:" + e.getMessage());
    }
}
```

- Efetua uma cópia do teu ficheiro resposta.xhtml e dá-lhe o nome de resposta2.xhtml.
- Efetua uma cópia do teu ficheiro Login.java e dá-lhe o nome de LoginBean.java.
- Abre o ficheiro Login.java:

- Acrescenta-lhe o atributo/propriedade “id” em falta (tipo inteiro) e respetivos getter e setter (este atributo corresponde ao campo id da tabela credenciais, da BD login\_java;

- Abre o ficheiro LoginBean.java:

- Atribui os atributos nome = “loginBean” e eager=true ao ManagedBean, na anotação @ManagedBean.

```
@ManagedBean (name="loginBean", eager=true)
```

- Instancia a classe Login com propriedade do Bean e gera os getter e setter do objeto dessa classe:

```
public class LoginBean {
```

```
    private Login log = new Login();
```

- Cria um método para definir mensagens nas páginas:

```
public static void AdicionarMensagem (String mensagem, FacesMessage.Severity tipoErro) {
    FacesMessage fm = new FacesMessage(tipoErro, mensagem, null);
    FacesContext.getCurrentInstance().addMessage(null, fm);
}
```

- Cria outro método com o nome “verificaUser”, que tem como parâmetro um objeto da classe Login (para se poder aceder às propriedades dessa classe, fazendo uso dos métodos getter e setter e comunicar com a BD) e que devolva um valor booleano.

Este método servirá para verificar se as credenciais inseridas na página de login correspondem a um utilizador registado na base de dados. para escrever o código que faça a verificação das credenciais introduzidas na página de login. Caso o utilizador exista será devolvido o valor true, caso contrário, false.

Código:

```
public boolean verificaUser(Login log) {
    try{

        Ligar ligar = new Ligar();
        Connection con = ligar.OL();
        boolean i =false;

        String sql= "SELECT * FROM credenciais where user=? and pwd=?";
        PreparedStatement pstm= con.prepareStatement(sql);
        pstm.setString(1, log.getUsername());
        pstm.setString(2, log.getPwd());
        ResultSet rs = pstm.executeQuery();

        if(rs.next())
            i=true;
        ligar.FecharLigar(con);
        return i;
    }
    catch(Exception ex) {
        AdicionarMensagem("ERRO! "+ex.getMessage(),FacesMessage.SEVERITY_ERROR);
    }
    return false;
}
```

- Cria mais um método, agora com o nome VerificaBean que devolva uma string, correspondente à página que se pretende abrir. Se o utilizador existir será aberta a página de index, caso contrário uma página de resposta com mensagem de resposta.

```
public String VerificaBean() {

    if(this.verificaUser(this.log))

        return "index.xhtml";
    else

        return "resposta2.xhtml";

}
```

- Na página de index.xhtml escreve uma tag outputText cujo atributo value seja:

```
<h2><h:outputText value = "Olá #{LoginBean.log.getUsername()} " /></h2>
```

A definir no conteúdo da página.

- Na página de resposta2.xhtml escreve uma tag outputText cujo atributo value seja:

```
<h2> <h:outputText value="Utilizador não existe ou pwd incorreta!" /> </h2>
```

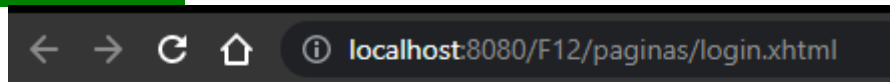
- A definir no corpo da página.

➔ Resultado final:

**Insira as suas credenciais:**

Username \*

PWD \*



Se dados incorretos:

**Utilizador não existe ou pwd incorreta!**

Se dados corretos:

**Insira as suas credenciais:**

Username \*

PWD \*



- ➔ Cria um novo ficheiro xhtml de nome “produtosInserir”,
- ➔ No ficheiro menu.xhtml, altera o nome do menuitem ‘Left’ para Inserir.
- ➔ O menu item ao ser clicado deverá abrir a página produtosInserir.xhtml

#### Bibliografia

[TutorialPointJSF. 2017:Tutorials Point \(i\) Pvt. Ltd.](#)  
[Introdução ao JSF Managed Bean | Devmedia](#)  
[JSF - Managed Beans \(tutorialspoint.com\)](#)  
<https://www.youtube.com/watch?v=sg4BmzMJ15I>  
<https://www.tutorialspoint.com/jsf/index.htm>  
<https://www.youtube.com/watch?v=OMROONn5J3c>  
[https://www.tutorialspoint.com/jsf/jsf\\_expression\\_language.htm](https://www.tutorialspoint.com/jsf/jsf_expression_language.htm)  
<https://www.javatpoint.com/jsf-jdbc-connectivity>