CIS 764 Tutorial: Log-in Application

Javier Ramos Rodriguez

Purpose

This tutorial shows you how to create a small web application that checks the user name and password.

Overview

This tutorial will show how to use the EJB 3.0 specification and the Java Server Faces (JSF) in Jdeveloper. We will show that JSF are very useful to validate the inputs.

Scenario

In this tutorial we'll create a persistence object Student from a table STUDENT. We will create a session bean to manage the entity object (entity bean) and we'll use JSF as View-Controller.

Prerequisites

You'll need to have access to an Oracle Database and the Jdeveloper 10g IDE. You also need a connection to the database.
Step 1. Creating the STUDENT table.

1. Open the *Jdeveloper* and on the navigator select connections. Then, go to the Oracle database connection and right-click on *SQLworksheet*.

2. In the SQL worksheet window, copy and paste this SQL code:

```sql
create table STUDENT ( login varchar2(50) primary key,
passwd varchar2(50) NOT NULL,
name varchar2(90) NOT NULL,
email varchar2(90) NOT NULL);
```
3. Click on the “Execute SQL Statement” button and the query will be executed.

Step 2. Configuring the Application

In this section we will create the persistence object using entity beans.

1. In JDeveloper, click the Applications tab. In the Applications navigator, right-click Applications and select New Application from the shortcut menu.
2. In the Create Application dialog box, enter *JSFLoginTutorial* as the application name, and select the *JSF, JSP, EJB template* option. Click OK.
3. In the JSFLoginTutorial application, right-click in ViewController project and then click on properties.

4. Choose Dependences tab and click on the Model.
Step 3. Creating the Persistence Model

1. In the Applications navigator, right-click the JSFLoginTutorial node and choose the New option.

2. In the New Gallery dialog box, expand the Business Tier node in Categories. In the Items list, select CMP Entity Beans from tables. Click OK.
3. Click **Next** on the Welcome page of the "*Create CMP Entity Beans from Tables*" wizard. In Step 1 of 5, select the **Enterprise JavaBeans 3.0 (J2EE 5.0)** option, and then click Next.

![Create CMP Entity Beans from Tables - Step 1 of 5: Select EJB Version](image)

4. In **Step 2** of 5, select the Oracle connection as the connection name.

![Create CMP Entity Beans from Tables - Step 2 of 5: Database Connection Details](image)
5. In step 3 of 5, click the **Query button**, and then select the STUDENT table from the Available list and shuttle them to the Selected list. Click Next.

6. In step 4 of 5, enter model as the package name. Click Next.
7. Click Next in step 5 of 5 and then Finish to create the entity beans.
Step 4. Creating the Business Model

1. Right-click the **JSFLoginTutorial** project node in the Applications navigator and select the **New option** from the context menu. Open the Business Tier category and choose the Session Bean item. Click OK.
2. Click Next on the Welcome page of the Create Enterprise JavaBean wizard. In step 1 of 3, enter **StudentSessionEJB** as the EJB name. Leave the options unchanged, and then click Next.

3. Leave everything unchanged in steps 2 and 3 and click Finish.
4. Open the `StudentSessionEJB` file in the navigator and add the next method.

```java
public boolean checkLogin(String login, String passwd)
    throws NamingException {
    return getEntityManager()
        .createQuery("select object(o) from Student o where
            ((o.login LIKE :login) AND (o.passwd LIKE :passwd))")
        .setParameter("login", login)
        .setParameter("passwd", passwd).getResultList().isEmpty();
}
```

This will query the database to check if the login and password given by the user are in the database.

5. Open the `StudentSessionEJB` interface and add the methods declarations.

```java
boolean checkLogin(String login, String password) throws NamingException;
```

```java
public boolean checkLogin(String login, String passwd)
    throws NamingException;
    System.out.println(login + passwd);
    return getEntityManager()
        .createQuery("select object(o) from Student o where
            ((o.login LIKE :login) AND (o.passwd LIKE :passwd))")
        .setParameter("login", login)
        .setParameter("passwd", passwd).getResultList().isEmpty();
}
```

```java
boolean checkLogin(String login, String password) throws NamingException;
```
6. Right-click in **StudentSessionEJBBean** and click on **New Sample Java Client**.
7. In Client Class name write `model.ModelFacade` and click ok.

8. Delete the main method and create the next method that is going to check if the login and password are stored in the database.

```java
public boolean checkLogin(String login, String password) {
    boolean check = true;

    try {
        final Context context = getInitialContext();
        StudentFacade studentFacade =
            (StudentFacade) context.lookup("StudentFacade");

        check = studentFacade.checkLogin(login, password);

        // true empty results
    } catch (NamingException e) {
        e.printStackTrace();
    }

    return !check;
}
```
Step 5. Create the Control-Flow Diagram

1. In the application navigator, select the ViewController project and in the WenContent\WEB_INF folder open the faces-config.xml file.

2. In the editor, drag and drop a JSF page from the component palette. Change the name to login.jsp. Then, drag and drop another JSF page to the diagram and change the name to main.jsp. Finally, drag and drop the JSF Navigation Case component from the palette and connect the login.jsp page to the main.jsp page. Change the name to go_main.
Step 6. Create the JSF pages.

1. Double-click in the `login.jsp` page in the diagram. The Create JSF JSP Wizard Welcome screen displays. Click Next to continue. In step 1, leave everything unchanged.

2. In Step 2, choose Automatically Bind Components Using a Newly Created Managed Bean.
3. In **Step 3**, choose not to use error page.

4. In **Step 4**, choose the default libraries, JSF Core and JSF HTML.
5. In **Step 5**, Choose the page title. Click next and then finish.
Step 7. Edit the JSP page.

1. The JSP editor will appear. In the top of the page enter some text:
   Welcome. Format the text to H1 by clicking the left-hand dropdown list at
   the top of the Visual Editor, and choosing Heading 1. Then, apply a CSS
   style sheet. Select the CSS page in the Components section in the top
   right of your screen. Then drag the JDeveloper style sheet onto your new
   JSP. You should see an immediate change in the appearance of you
   page.

2. In the Component palette choose JSF HTML and click on Panel Grid.
   In the new window choose 2 in number of columns and click finish.
3. Now drag and drop the next components one after the other as shown in the figure: OutputLabel, InputText, OutputLabel, InputSecret, CommandButton and Messages.

4. Change the outputLabel1 name to Login and the outputLabel2 name to Password. Also, change the commandButton name to Submit, to do this go to the property inspector window and change the value label.
Step 8. Implementing the behavior

1. Double click in the Submit button. The Login.java file will open to edit the action. Copy and paste the next code:

```java
public String commandButton1_action() {
    boolean check = false;
    String  message = new String();
    String  login   = inputText1.getValue().toString();
    String  passwd  = inputSecret1.getValue().toString();

    if ((login.length() < 4) || (passwd.length() < 4)) {
        message = "Invalid inputs";
    } else {
        ModelFacade check_log = new ModelFacade();

        if (!check_log.checkLogin(login, passwd)) {
            message = "Wrong username or password";
        } else {
            check = true;
        }
    }

    FacesContext.getCurrentInstance().addMessage(null,
                                                    new FacesMessage(message));

    if (check) {
        return "go_main";
    } else {
        return "failure";
    }
}
```

This code will check that the inputs are valid and also will ask the model to check if the username and password are valid.
2. Import the `model.ModelFacade` package.

```java
public String commandButton1_action() {
    boolean check = false;
    String message = new String();
    String login = inputText1.getValue().toString();
    String passwd = inputSecret1.getValue().toString();

    if ((login.length() < 4) || (passwd.length() < 4)) {
        message = "Invalid inputs";
    } else {
        ModelFacade check_log = new ModelFacade();

        if (!check_log.checkLogin(login, passwd)) {
            message = "Wrong username or password";
        } else {
            check = true;
        }
    }

    FacesContext.getCurrentInstance().addMessage(null,
        new FacesMessage(message));

    if (check) {
        return "go_main";
    } else {
        return "failure";
    }
}
```
**Step 9.** Create the main page and run the application.

1. Double click in `main.jsp` in the JSF navigation flow diagram. Create the JSP page like before and it will open in the editor. Enter some text and apply the style sheet.

   Welcome

2. Right-Click in the `login.jsp` page and then run.
Main Page:

**Welcome**

Login: 
Password: 
[Submit]