ĐĂNG NHẬP VÀ ĐĂNG KÝ

THAM KHẢO

==============================

Ví dụ 1: Đăng nhập

Ví dụ 2-3: Đăng ký

THAM KHẢO THÊM

<https://netbeans.org/kb/docs/web/mysql-webapp.html?print=yes>

<http://javaandj2eetutor.blogspot.com/2014/01/login-application-using-jsp-servlet-and.html>

==============================

JSP - Servlets: Full Login Example

http://met.guc.edu.eg/OnlineTutorials/JSP%20 %20Servlets/Full%20Login%20Example.aspx

In this section we are going to discuss the implementation of a complete Login application (We received the source code of this application through Amira Thabet)

The application will

* Ask the user to input his "username" and "password"
* Check the existence of this user in the Data Base
  + If exists, Retrieve his first name and last name from the DB and display them
  + If not registered, Display "Sorry, you are not registered"

You are assumed to know about the connection between java and DB.

This Login application uses the so called "JavaBeans" and "DAOs" - standing for Data Access Objects - to handle the interactions with the DB.

About JavaBeans and DAOs

**Beans** are normal java classes containing

1. Attributes
2. Getters and Setters for such attributes (and may be some additional methods)

In our application, Beans are used to save data needed in the application (in the form of variables) . This data may be a representation of data existing in the DB, data entered by the user, or results of business logic.

**DAOs**are objects responsible for handling the interactions with the Data Source, through implementing the access mechanism required to work with the data source.

In our application the DAO is responsible for

1. Reading data from the Bean (data entered by the user) and checking its consistency with the DB - use Getters to get values of variables from the bean-
2. Retrieving data from the DB and saving it to the Bean - use Setter methods to set values to variables -

You can either go to the Next page to step by step implement the example, or download its source code from [here](http://met.guc.edu.eg/OnlineTutorials/static/article_media/jsp%20%C3%A2%E2%82%AC%E2%80%9C%20servlets/loginExample.zip). The source code is commented in details, so that can understand every statement's role.

Example's Implementation Steps

As mentioned in the application description, the user will have to enter his username and password, so first of all, we need a JSP that asks the user to input his username and password in their corresponding fields.

**To have this JSP, please follow these steps:**

* Open eclipse
* Create a new "Dynamic Web Project"
* Name it "LoginExample"
* Create the JSP
  + In the "Web Content" folder, create a new JSP
  + Name it "LoginPage"
  + Place this code
* <%@ page language="java"
* contentType="text/html; charset=windows-1256"
* pageEncoding="windows-1256"
* %>
* <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
* "http://www.w3.org/TR/html4/loose.dtd">
* <html>
* <head>
* <meta http-equiv="Content-Type" content="text/html; charset=windows-1256">
* <title>Login Page</title>
* </head>
* <body>
* <form action="LoginServlet">
* Please enter your username
* <input type="text" name="un"/><br>
* Please enter your password
* <input type="text" name="pw"/>
* <input type="submit" value="submit">
* </form>
* </body>
* </html>

As you can see; (in the LoginPage) when the user submits, the JSP calls "LoginServlet".This LoginServlet is intended to handle the Business logic associated with the request.

**Create the LoginServlet by following these steps:**

* In the "src" folder, create a new "Package"
* Name it "ExamplePackage"
* In the "ExamplePackage", create a new "Servlet"
* Name it "LoginServlet"
* Place this code
* import java.io.IOException;
* import javax.servlet.ServletException;
* import javax.servlet.http.HttpServlet;
* import javax.servlet.http.HttpServletRequest;
* import javax.servlet.http.HttpServletResponse;
* import javax.servlet.http.HttpSession;
* /\*\*
* \* Servlet implementation class LoginServlet
* \*/
* public class LoginServlet extends HttpServlet {
* public void doGet(HttpServletRequest request, HttpServletResponse response)
* throws ServletException, java.io.IOException {
* try
* {
* UserBean user = new UserBean();
* user.setUserName(request.getParameter("un"));
* user.setPassword(request.getParameter("pw"));
* user = UserDAO.login(user);
* if (user.isValid())
* {
* HttpSession session = request.getSession(true);
* session.setAttribute("currentSessionUser",user);
* response.sendRedirect("userLogged.jsp"); //logged-in page
* }
* else
* response.sendRedirect("invalidLogin.jsp"); //error page
* }

* catch (Throwable theException)
* {
* System.out.println(theException);
* }
* }
* }

The login servlet instantiates a Bean that is of type "UserBean", and then calls the DAO named "UserDAO".

* Our UserBean is a class representing the User table in our Database (where each column in the user table has a corresponding instance variable with a setter and a getter method).
* The DAO, as said before, contains methods needed to communicate with the data source. In our example, the only needed method is the login method that checks if the username and password inputted by the user are valid or not.

**Before implementing the DAO, you need to prepare your Data Source.**

1. Create a table in your DB
2. Name it users
3. Create the columns: 'FirstName', 'LastName', 'username', and 'password'
4. Refer to your DB as a data source from "Administrative tools" in Control Panel

**Please follow these steps to implement the Bean and the DAO**

* Create the "UserBean" class
  1. In the " ExamplePackage ", create a new "Class"
  2. Name it "UserBean"
  3. Place this code
  5. public class UserBean {
  7. private String username;
  8. private String password;
  9. private String firstName;
  10. private String lastName;
  11. public boolean valid;

  14. public String getFirstName() {
  15. return firstName;
  16. }
  17. public void setFirstName(String newFirstName) {
  18. firstName = newFirstName;
  19. }
  21. public String getLastName() {
  22. return lastName;
  23. }
  24. public void setLastName(String newLastName) {
  25. lastName = newLastName;
  26. }
  28. public String getPassword() {
  29. return password;
  30. }
  31. public void setPassword(String newPassword) {
  32. password = newPassword;
  33. }

  36. public String getUsername() {
  37. return username;
  38. }
  39. public void setUserName(String newUsername) {
  40. username = newUsername;
  41. }
  43. public boolean isValid() {
  44. return valid;
  45. }
  46. public void setValid(boolean newValid) {
  47. valid = newValid;
  48. }
  49. }

* Create the "UserDAO" class
  1. In the " ExamplePackage ", create a new "Class"
  2. Name it "UserDAO"
  3. Place this code

import java.text.\*;

import java.util.\*;

import java.sql.\*;

public class UserDAO

{

static Connection currentCon = null;

static ResultSet rs = null;

public static UserBean login(UserBean bean) {

//preparing some objects for connection

Statement stmt = null;

String username = bean.getUsername();

String password = bean.getPassword();

String searchQuery =

"select \* from users where username='"

+ username

+ "' AND password='"

+ password

+ "'";

// "System.out.println" prints in the console; Normally used to trace the process

System.out.println("Your user name is " + username);

System.out.println("Your password is " + password);

System.out.println("Query: "+searchQuery);

try

{

//connect to DB

currentCon = ConnectionManager.getConnection();

stmt=currentCon.createStatement();

rs = stmt.executeQuery(searchQuery);

boolean more = rs.next();

// if user does not exist set the isValid variable to false

if (!more)

{

System.out.println("Sorry, you are not a registered user! Please sign up first");

bean.setValid(false);

}

//if user exists set the isValid variable to true

else if (more)

{

String firstName = rs.getString("FirstName");

String lastName = rs.getString("LastName");

System.out.println("Welcome " + firstName);

bean.setFirstName(firstName);

bean.setLastName(lastName);

bean.setValid(true);

}

}

catch (Exception ex)

{

System.out.println("Log In failed: An Exception has occurred! " + ex);

}

//some exception handling

finally

{

if (rs != null) {

try {

rs.close();

} catch (Exception e) {}

rs = null;

}

if (stmt != null) {

try {

stmt.close();

} catch (Exception e) {}

stmt = null;

}

if (currentCon != null) {

try {

currentCon.close();

} catch (Exception e) {

}

currentCon = null;

}

}

return bean;

}

}

The DAO uses a class named "ConnectionManager" to get a connection with the DB.

**Create the "ConnectionManager" class:**

In the " ExamplePackage ", create a new "Class"

Name it "ConnectionManager"

Place this code

import java.sql.\*;

import java.util.\*;

public class ConnectionManager {

static Connection con;

static String url;

public static Connection getConnection()

{

try

{

String url = "jdbc:odbc:" + "DataSource";

// assuming "DataSource" is your DataSource name

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

try

{

con = DriverManager.getConnection(url,"username","password");

// assuming your SQL Server's username is "username"

// and password is "password"

}

catch (SQLException ex)

{

ex.printStackTrace();

}

}

catch(ClassNotFoundException e)

{

System.out.println(e);

}

return con;

}

}

1. In the pasted code, Replace
   * "DataSource" with your Data Source name
   * "username" with your SQL Server username
   * "password" with your SQL Server password

Finally, we are done with the logic and accessing the DB. So back to the interface, we need two JSPs; one for the valid login and another for the invalid. The two JSPs are

* userLogged.jsp: Displays a message to welcome the user, using his first and last names (retrieved from the DB)
* invalidLogin.jsp: Displays a message to inform the user that he is not a registered user

**Steps to create the "userLogged" JSP**

1. In the "WebContent" folder, create a new "JSP"
2. Name it "userLogged"
3. Place this code
4. <%@ page language="java"
5. contentType="text/html; charset=windows-1256"
6. pageEncoding="windows-1256"
7. import="ExamplePackage.UserBean"
8. %>
10. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
11. "http://www.w3.org/TR/html4/loose.dtd">
12. <html>
13. <head>
14. <meta http-equiv="Content-Type"
15. content="text/html; charset=windows-1256">
16. <title> User Logged Successfully </title>
17. </head>
19. <body>
20. <center>
21. <% UserBean currentUser = (UserBean (session.getAttribute("currentSessionUser"));%>
23. Welcome <%= currentUser.getFirstName() + " " + currentUser.getLastName() %>
24. </center>
25. </body>
27. </html>

**Steps to create the "invalidLogin" JSP**

In the "WebContent" folder, create a new "JSP"

Name it "invalidLogin"

Place this code

<%@ page language="java"

contentType="text/html; charset=windows-1256"

pageEncoding="windows-1256"

%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type"

content="text/html; charset=windows-1256">

<title>Invalid Login</title>

</head>

<body>

<center>

Sorry, you are not a registered user! Please sign up first

</center>

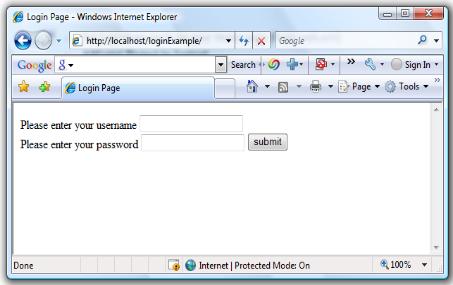
</body>

</html>

**Run the application** , If you do not know any of the following steps, please check Steps 5-8 in the JSP Example

* Set LoginPage.jsp to be your Home page (from web.xml)
* Add your Project to Tomcat
* Start Tomcat
* Go to http://Localhost/loginExample
* Test your project

**This should be the result**

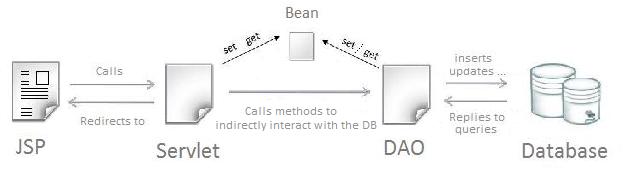






Now your project should be working; if you would like to take a look at how the process went, go to the next page.

The process flow

This diagram basically shows the process flow in our application

1. The user enters his username and password in the fields displayed by the JSP - LoginPage.jsp -
2. When the user submits, the servlet responsible for handling the request is called - LoginServlet -
3. The Servlet is responsible for calling the appropriate method in the DAO so that it can indirectly interact with the DB.

It is also responsible for setting and updating data saved in the bean, which will be used later by the DAO.

In our application,

* + The LoginServlet creates a new instant of the UserBean and fills it with the username and the password entered by the user. The DAO will use this bean later to compare between the user input and the DB data
  + The Servlet calls the "login" method in the "UserDAO" to start performing its task

1. The Login method, in the DAO, is responsible for checking whether the data entered by the user exists in the DB or not.

In addition, it has to update the Bean's data that will be used later by the servlet.

In our application,

* + The DAO uses the ConnectionManager class to get the DB connection
  + Query the DB (asks the DB to search for a user having certain username and password ) and checks,
    - If the ResultSet is empty, this means that the username and password were invalid (not in the DB).
    - If the ResultSet is not empty, this means that the username and password were valid.
  + Updates the UserBean.
    - In case of valid username and password, the DAO fills the bean with the rest of the user's information that will need to be displayed later by the JSP (first and last names).

In addition, it sets the "valid" attribute of the bean to true.

* + - Otherwise, the DAO sets the "valid" attribute of the bean to false

Now we know if the user was registered or not

1. Finally, the Servlet will check the validity of the user (by reading the valid attribute of the bean) and redirect to the appropriate JSP .
   * If valid, the servlet will
     + Add the bean as an attribute to the session. The bean will be used by the JSP to display the user's first and last names
     + Redirect to â€œuserLogged.jspâ€ - That will welcome the user
   * If invalid, the servlet will redirect to â€œinvalidLogin.jspâ€ - That will ask the user to sign up

**REGISTERING**

[**http://www.javawebtutor.com/articles/servlets/servlet\_db\_example.php**](http://www.javawebtutor.com/articles/servlets/servlet_db_example.php)

We are going to develop a web application in which User can register and then login to the application.We are going to maintain user information in DB.We are using mysql database, so we need to create a table first as given below.We use Eclipse IDE for Java EE Developers which includes tools for creating Java EE and Web applications.

**CREATE** **TABLE** USERDETAILS

( NAME VARCHAR(100),

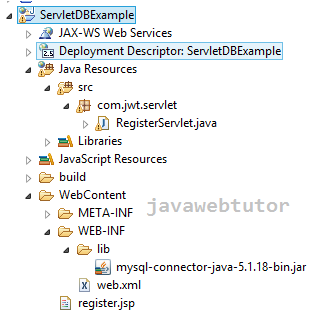
PASSWORD VARCHAR(50),

EMAIL VARCHAR(50),

**LANGUAGE** VARCHAR(50)

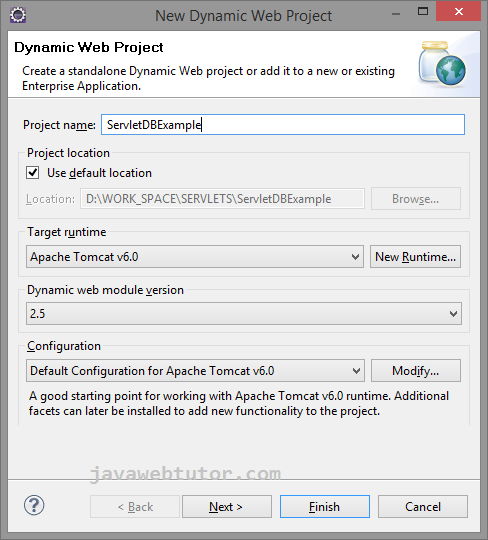
);

## Directory Structure Of Project



## Steps to create Registration form in servlet

Create a dynamic Web Project in Eclipse and assign the name of project as ServletDBExample as shown below.



Now create a jsp file inside WebContent directory of your project and provide the name of jsp file as register.jsp and add following code into this file.

## register.jsp

In this page,user will provide inputs using text fields and combobox. The information entered by the user is forwarded to RegisterServlet, which is responsible to store the data into the database.

[?](http://www.javawebtutor.com/articles/servlets/servlet_db_example.php)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <%@ page language="java" contentType="text/html; charset=ISO-8859-1"      pageEncoding="ISO-8859-1"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "<http://www.w3.org/TR/html4/loose.dtd>">  <html>  <body>      <form action="register" method="post">            Name:<input type="text" name="userName"/><br/>          Password:<input type="password" name="password"/><br/>          Email Id:<input type="text" name="email" /><br/>          Language: <select name="language">              <option>Hindi</option>              <option>English</option>              <option>French</option>          </select> <br/>          <input type="submit" value="Submit"/>        </form>  </body>  </html> |

## RegisterServlet.java

Now create a package com.jwt.servlet and in that package create a class RegisterServlet and add following code into this. This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic.

[?](http://www.javawebtutor.com/articles/servlets/servlet_db_example.php)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | package com.jwt.servlet;    import java.io.IOException;  import java.io.PrintWriter;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;    import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;    public class RegisterServlet extends HttpServlet {      public void doPost(HttpServletRequest request, HttpServletResponse response)              throws ServletException, IOException {            response.setContentType("text/html");          PrintWriter out = response.getWriter();            String n = request.getParameter("userName");          String p = request.getParameter("password");          String e = request.getParameter("email");          String c = request.getParameter("language");            try {              Class.forName("com.mysql.jdbc.Driver");              Connection con = DriverManager.getConnection(                      "jdbc:<mysql://localhost:3306/servlet>", "root", "mukesh");                PreparedStatement ps = con                      .prepareStatement("insert into USERDETAILS values(?,?,?,?)");                ps.setString(1, n);              ps.setString(2, p);              ps.setString(3, e);              ps.setString(4, c);                int i = ps.executeUpdate();              if (i > 0)                  out.print("You are successfully registered...");            } catch (Exception e2) {              System.out.println(e2);          }            out.close();      }    } |

## web.xml

Map the servlet in web.xml file

[?](http://www.javawebtutor.com/articles/servlets/servlet_db_example.php)

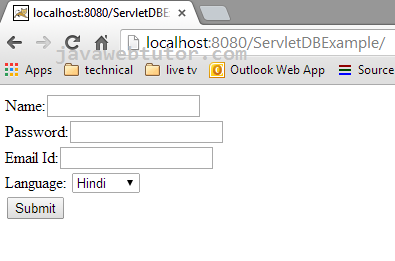
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"      xmlns="<http://java.sun.com/xml/ns/javaee>" xmlns:web="<http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd>"      xsi:schemaLocation="<http://java.sun.com/xml/ns/javaee> <http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd>"      id="WebApp\_ID" version="2.5">      <servlet>          <servlet-name>Register</servlet-name>          <servlet-class>com.jwt.servlet.RegisterServlet</servlet-class>      </servlet>        <servlet-mapping>          <servlet-name>Register</servlet-name>          <url-pattern>/register</url-pattern>      </servlet-mapping>        <welcome-file-list>          <welcome-file>register.jsp</welcome-file>      </welcome-file-list>    </web-app> |

Note:-We need to add mysql connector jar in the lib folder of the project for getting the MYSQL DB connection using java.

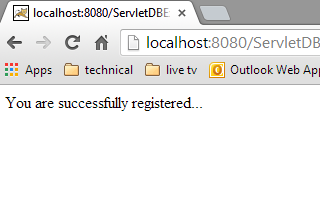
**Run the Application**

Right Click on the project and select Run as -> Run on Server and select the Tomcat Server and add your project in the server and click Finish.

**Output in Browser**



After providing the value when user clicks on Submit button user will be redirected to success page as shown below.



## Related Articles

1. [How to Create Servlet Example In Eclipse](http://www.javawebtutor.com/articles/servlets/servlet_example_in_eclipse.html)
2. [Installation Of Tomcat Server in Eclipse](http://www.javawebtutor.com/articles/servlets/installing_apache_tomcat_in_eclipse.html)
3. [JSP & Database Example](http://www.javawebtutor.com/articles/jsp/jspdbaccess.html)
4. [JSP Example in Eclipse](http://www.javawebtutor.com/articles/jsp/jspeginecl.html)
5. [How to Create Struts Application in Eclipse](http://www.javawebtutor.com/articles/struts/struts_application_in_eclipse.html)

**LOGIN**

<http://javaandj2eetutor.blogspot.com/2014/01/login-application-using-jsp-servlet-and.html>

### [Login application using jsp servlet and mysql database](http://javaandj2eetutor.blogspot.com/2014/01/login-application-using-jsp-servlet-and.html)

Today we are going to create a simple web login application using JSP, servlet and mysql database. In order to create an application we are going to use the following software.

1. MySql database
2. Eclipse IDE
3. Tomcat server

Firstly, lets create a database and a table in mysql. Turn on the database connection and open the mysql command prompt and paste the below code.

create database form;

use form;

CREATE  TABLE `form`.`login` (

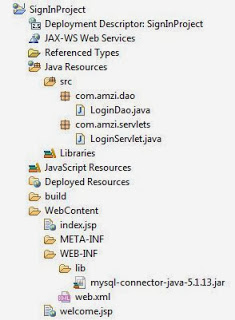
  `user` VARCHAR(20) NOT NULL ,

  `password` VARCHAR(20) NOT NULL ,

  PRIMARY KEY (`user`) );

INSERT INTO `form`.`login` (`user`, `password`) VALUES ('Admin', 'passw0rd');

Now, open up the Eclipse IDE and create a dynamic web project and create the project structure as per the screen shot below.

[](http://2.bp.blogspot.com/-HgnQTstr_SU/UsfDf56-C5I/AAAAAAAAAKU/jM7RzQESDUw/s1600/projectStructure.JPG)

Lets create the front end with two basic jsp pages.  
  
**index.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Login Application</title>

</head>

<body>

    <form action="loginServlet" method="post">

        <fieldset style="width: 300px">

            <legend> Login to App </legend>

            <table>

                <tr>

                    <td>User ID</td>

                    <td><input type="text" name="username" required="required" /></td>

                </tr>

                <tr>

                    <td>Password</td>

                    <td><input type="password" name="userpass" required="required" /></td>

                </tr>

                <tr>

                    <td><input type="submit" value="Login" /></td>

                </tr>

            </table>

        </fieldset>

    </form>

</body>

</html>

**welcome.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Welcome <%=session.getAttribute("name")%></title>

</head>

<body>

    <h3>Login successful!!!</h3>

    <h4>

        Hello,

        <%=session.getAttribute("name")%></h4>

</body>

</html>

Now, lets create the login DAO which will enable us to connect our login application with mysql database and execute the query to the DB.  
  
**LoginDao.java**

package com.amzi.dao;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class LoginDao {

    public static boolean validate(String name, String pass) {

        boolean status = false;

        Connection conn = null;

        PreparedStatement pst = null;

        ResultSet rs = null;

        String url = "jdbc:mysql://localhost:3306/";

        String dbName = "form";

        String driver = "com.mysql.jdbc.Driver";

        String userName = "root";

        String password = "password";

        try {

            Class.forName(driver).newInstance();

            conn = DriverManager

                    .getConnection(url + dbName, userName, password);

            pst = conn

                    .prepareStatement("select \* from login where user=? and password=?");

            pst.setString(1, name);

            pst.setString(2, pass);

            rs = pst.executeQuery();

            status = rs.next();

        } catch (Exception e) {

            System.out.println(e);

        } finally {

            if (conn != null) {

                try {

                    conn.close();

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

            if (pst != null) {

                try {

                    pst.close();

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

            if (rs != null) {

                try {

                    rs.close();

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

        }

        return status;

    }

}

Now, we are going to create the servlet which will capture the input parameter from the jsp and validate it against the LoginDao.  
  
**LoginServlet.java**

package com.amzi.servlets;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.amzi.dao.LoginDao;

public class LoginServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;

    public void doPost(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        String n=request.getParameter("username");

        String p=request.getParameter("userpass");

        HttpSession session = request.getSession(false);

        if(session!=null)

        session.setAttribute("name", n);

        if(LoginDao.validate(n, p)){

            RequestDispatcher rd=request.getRequestDispatcher("welcome.jsp");

            rd.forward(request,response);

        }

        else{

            out.print("<p style=\"color:red\">Sorry username or password error</p>");

            RequestDispatcher rd=request.getRequestDispatcher("index.jsp");

            rd.include(request,response);

        }

        out.close();

    }

}

Finally, lets configure the web.xml file for servlet and welcome file configuration.  
  
**web.xml**

<?xml version="1.0" encoding="UTF-8"?>

<web-app

    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"

    version="2.5">

    <servlet>

        <servlet-name>login</servlet-name>

        <servlet-class>com.amzi.servlets.LoginServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>login</servlet-name>

        <url-pattern>/loginServlet</url-pattern>

    </servlet-mapping>

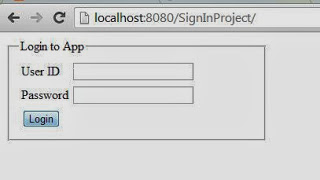
    <welcome-file-list>

        <welcome-file>index.jsp</welcome-file>

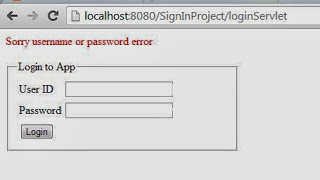
    </welcome-file-list>

</web-app>

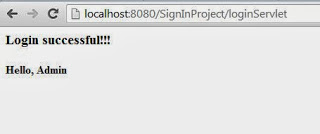
That's all it takes to create a simple web login application. This is not the ideal login application as it needs lot of modification such as security etc. Anyways, for now let's run the project on the tomcat server and test the application.

[](http://1.bp.blogspot.com/-qbNAaY0CgU4/UsfGQ3Bcx0I/AAAAAAAAAKg/uvIcb3NVQj8/s1600/index.JPG)

Now, let's input the invalid data and check the result.

[](http://1.bp.blogspot.com/-WxtrU5mPMHU/UsfG1xfVFFI/AAAAAAAAAKo/UXXVlbIpRZ0/s1600/invalid.JPG)

Now, I am using the correct credentials which will match with the database result and redirect the page to the welcome file.  
  
Note: The URL is internally redirecting the page to the welcome.jsp file since we are using the forward method. If instead we would have used redirect method, in that case we can see the request URL in the browser.

[](http://1.bp.blogspot.com/-PoPwQ9j5fAo/UsfIDk1CxrI/AAAAAAAAAK0/_z_rEDisUr0/s1600/success.JPG)

[Download Code](https://app.box.com/s/k77yip8h75i0298ujm84)