



All Tutorials   Java ▼   Maven   Gradle   Servlet/Jsp   Spring ▼   Struts2   Hibernate   Java Web Service   JavaFX   SWT   Oracle ADF  
Android   Python   Swift   C#   C/C++   Ruby   Batch   Database ▼   Report   Client   NodeJS   OS ▼   Git   SAP   Others ▼

## Create a Simple Java Web Application Using Servlet, JSP and JDBC

View more categories:

[Java Servlet/Jsp Tutorials](#)

- 1- [Introduction](#)
- 2- [The principle when programming Servlet + JSP](#)
- 3- [View Demo of Web Application will do](#)






- 4- [📌 Prepare database](#)
- 5- [📌 Create WebApp Project](#)
- 6- [📌 Configuring the runtime environment](#)
- 7- [📌 Run application for first time](#)
- 8- [📌 Download and declare JDBC library](#)
- 9- [📌 Download and declare JSTL library](#)
- 10- [📌 Javabeans classes simulated tables in the database](#)
- 11- [📌 Database Connection Utility classes](#)
- 12- [📌 The utility class & manipulate data](#)
- 13- [📌 Create Servlet Filter connect to Database](#)
- 14- [📌 Servlet Filter reads Cookies to automatically login](#)
- 15- [📌 EncodingFilter Servlet](#)
- 16- [📌 Pages reuse](#)
- 17- [📌 Home Page](#)
- 18- [📌 Login Page - LoginServlet](#)
- 19- [📌 Product List Page](#)
- 20- [📌 Add Product Page](#)
- 21- [📌 Edit Product Page](#)
- 22- [📌 Delete Product](#)

445  
Shares



# JSP & Servlet

## Java Servlet/Jsp Tutorials

- [Installing and Configuring Tomcat Server in Eclipse](#)
- [Installing and configuring Glassfish Web Server](#)
- [Installing and configuring Oracle WebLogic Server](#)
- [Java Servlet Tutorial for Beginners](#)
- [Java Servlet Filter Tutorial](#)
- [Java JSP Tutorial for Beginners](#)
- [Java JSP Standard Tag Library \(JSTL\) Tutorial](#)
- [Install Web Tools Platform into Eclipse](#)
- [Create a simple Login application and secure pages with Java Servlet Filter](#) 
- [Create a Simple Java Web Application Using Servlet, JSP and JDBC](#) 
- [Uploading and downloading files stored to hard drive with Java Servlet](#) 

## 1- Introduction

This document is based on:

- Eclipse 4.5 MARS
- Tomcat 8.x

In this document, I will guide step by step how to create a simple web application with the combination of **Servlet + JSP + Filter + JSP EL + JDBC**. Make sure that you've mastered **Servlet, JSP** and **Filter** and **JDBC** before the start. If not, you can refer to:

### Servlet:

- [Java Servlet Tutorial for Beginners](#)

### Servlet Filter:

- [Java Servlet Filter Tutorial](#)

### JSP:

- [Java JSP Tutorial for Beginners](#)

### JSP Standard Tag Libs (JSTL)

- [Java JSP Standard Tag Library \(JSTL\) Tutorial](#)

- Uploading and downloading files from Database using Java Servlet
- Displaying Image from Database with Java Servlet
- Redirect 301 Permanent redirect in Java Servlet
- How to automatically redirect http to https in a Java Web application?
- Using Google reCAPTCHA with Java Web Application
- Run Maven Java Web Application in Tomcat Maven Plugin
- Run Maven Java Web Application in Jetty Maven Plugin
- Run background task in Java Servlet Application



Newest Documents

## JDBC

- [Java JDBC tutorial](#)

“

*NOTE: In this post I only introduce about **CRUD**, "**Login**" and "**Remember Me**" function. And do not handle the security of the pages. If you want to have an application, secure each page, please refer to the article below:*

- [Create a simple Login application and secure pages with Java Servlet Filter](#)

## 2- The principle when programming Servlet + JSP

These are the principles that you should keep in mind to be able to build a Web application using **Servlet + JSP** satisfying criteria: code is simple, easy to understand and easy to maintain.

The principles:

1. Never allow users to directly access to your JSP page.
2. JSP is only considered as the place to display interface.
3. Servlet acts as the controller of the application flows and program logical processing.
4. Open the JDBC connection and transaction management in Filter (Optional).

### According to the principle 1:

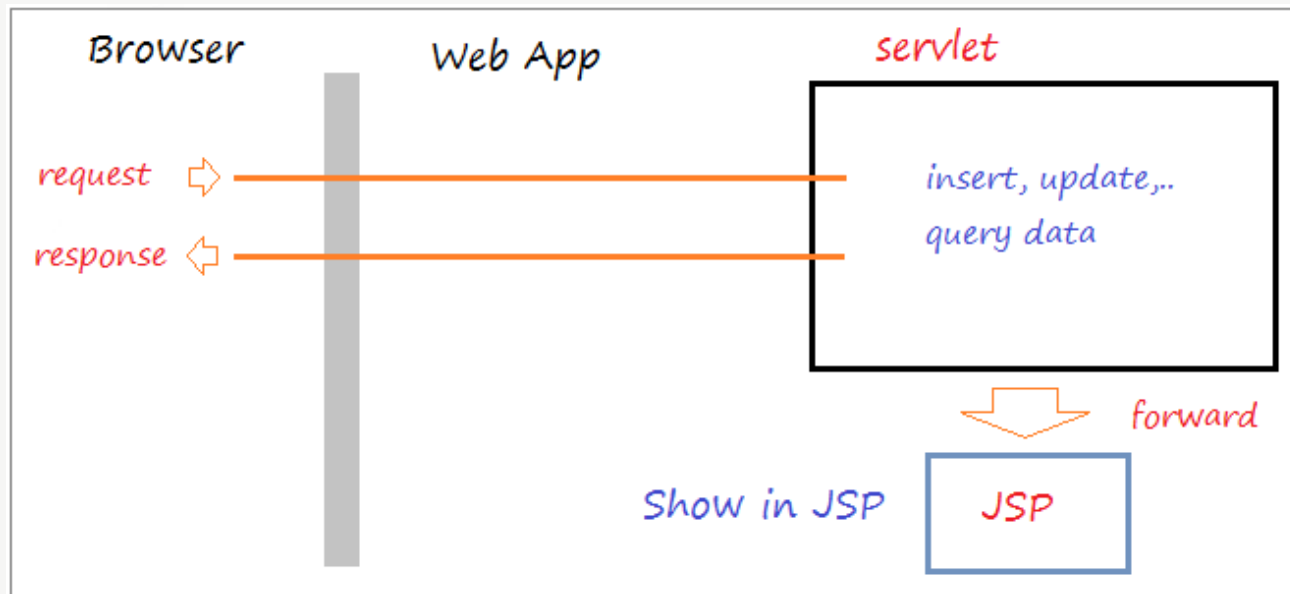
Never allow users to directly access to your JSP page, it means that all user's requests are:

- Another source of static data (images, css, js, ...)
- Or a servlet.

- [Introducing Bootstrap](#)
- [Connecting to MySQL Database using NodeJS](#)
- [NodeJS EventEmitter Tutorial](#)
- [Understanding Event Loop in NodeJS](#)
- [The concept of Callback in NodeJS](#)
- [NodeJS Modules Tutorial](#)
- [Introducing NodeJS](#)
- [Installing Atom Editor](#)
- [NodeJS Tutorial for Beginners](#)
- [Installing NodeJS on Windows](#)



Therefore, you must hide your JSP files in a place where the user can not access. For instance, set it in the **WEB-INF** folder or its subdirectories. In this example, I hide the jsp files in the **WEB-INF/views**.



When the user requests to a Servlet, it will dispose user's requirements, such insert, update and query the data, eventually forward to the JSP page to display the data. Thus, each servlet has 0 or multiple corresponding JSP pages (Usually only need 1).

## Principle 2:

JSP is only considered as the place to display data, which means that you should not handle the application logic on the JSP, such as update, insert, delete, .., and not navigate on the JSP page.

## 3- View Demo of Web Application will do

You can preview Demo Web application will do:





Start  
DEMO

## 4- Prepare database

In this document, I instruct you to work with one of 3 databases: **Oracle, MySQL or SQL Server**. You need to run scripts to create some tables and necessary data for this example.

### ORACLE:

```
1  -- Create table
2  create table USER_ACCOUNT
3  (
4  USER_NAME VARCHAR2(30) not null,
5  GENDER   VARCHAR2(1) not null,
6  PASSWORD VARCHAR2(30) not null,
7  primary key (USER_NAME)
8  );
9
10 -- Create table
11 create table PRODUCT
12 (
13 CODE VARCHAR2(20) not null,
14 NAME VARCHAR2(128) not null,
15 PRICE FLOAT not null,
16 primary key (CODE)
```



```

17 );
18
19 -- Insert data: -----
20
21 insert into user_account (USER_NAME, GENDER, PASSWORD)
22 values ('tom', 'M', 'tom001');
23
24 insert into user_account (USER_NAME, GENDER, PASSWORD)
25 values ('jerry', 'M', 'jerry001');
26
27 insert into product (CODE, NAME, PRICE)
28 values ('P001', 'Java Core', 100);
29
30 insert into product (CODE, NAME, PRICE)
31 values ('P002', 'C# Core', 90);
32
33 -- Commit
34 Commit;

```

## MYSQL:

```

1 -- Create table
2 create table USER_ACCOUNT
3 (
4 USER_NAME VARCHAR(30) not null,
5 GENDER VARCHAR(1) not null,
6 PASSWORD VARCHAR(30) not null,
7 primary key (USER_NAME)
8 );
9
10 -- Create table
11 create table PRODUCT
12 (
13 CODE VARCHAR(20) not null,
14 NAME VARCHAR(128) not null,
15 PRICE FLOAT not null,
16 primary key (CODE)
17 );
18
19 -- Insert data: -----
20
21 insert into user_account (USER_NAME, GENDER, PASSWORD)
22 values ('tom', 'M', 'tom001');
23

```



```
24 insert into user_account (USER_NAME, GENDER, PASSWORD)
25 values ('jerry', 'M', 'jerry001');
26
27 insert into product (CODE, NAME, PRICE)
28 values ('P001', 'Java Core', 100);
29
30 insert into product (CODE, NAME, PRICE)
31 values ('P002', 'C# Core', 90);
```

## SQL SERVER:

```
1  -- Create table
2  create table USER_ACCOUNT
3  (
4  USER_NAME VARCHAR(30) not null,
5  GENDER   VARCHAR(1) not null,
6  PASSWORD VARCHAR(30) not null,
7  primary key (USER_NAME)
8  );
9
10 -- Create table
11 create table PRODUCT
12 (
13 CODE VARCHAR(20) not null,
14 NAME VARCHAR(128) not null,
15 PRICE FLOAT not null,
16 primary key (CODE)
17 );
18
19 -- Insert data: -----
20
21 insert into user_account (USER_NAME, GENDER, PASSWORD)
22 values ('tom', 'M', 'tom001');
23
24 insert into user_account (USER_NAME, GENDER, PASSWORD)
25 values ('jerry', 'M', 'jerry001');
26
27 insert into product (CODE, NAME, PRICE)
28 values ('P001', 'Java Core', 100);
29
30 insert into product (CODE, NAME, PRICE)
31 values ('P002', 'C# Core', 90);
```

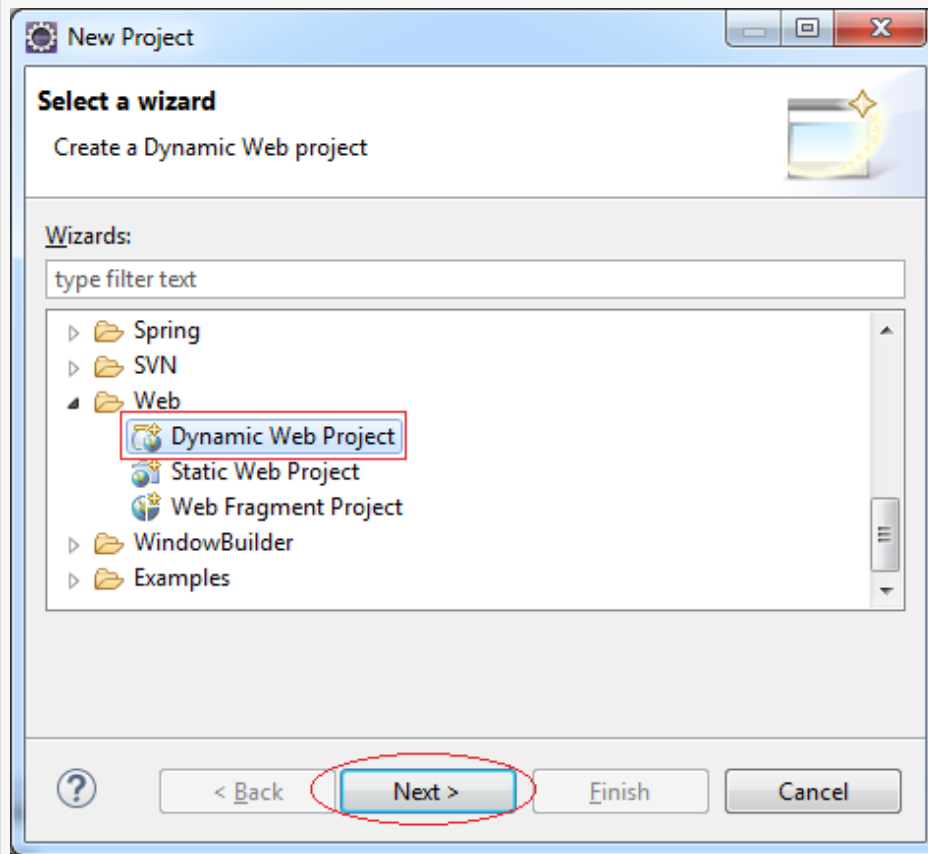





## 5- Create WebApp Project


In Eclipse select:

- File/New/Other...



 New Dynamic Web Project

---

**Dynamic Web Project** 

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

☒ Use default location

Location:

Target runtime

Dynamic web module version

Configuration

The default configuration provides a good starting point. Additional facets can later be installed to add new functionality to the project.


EAR membership

☐ Add project to an EAR

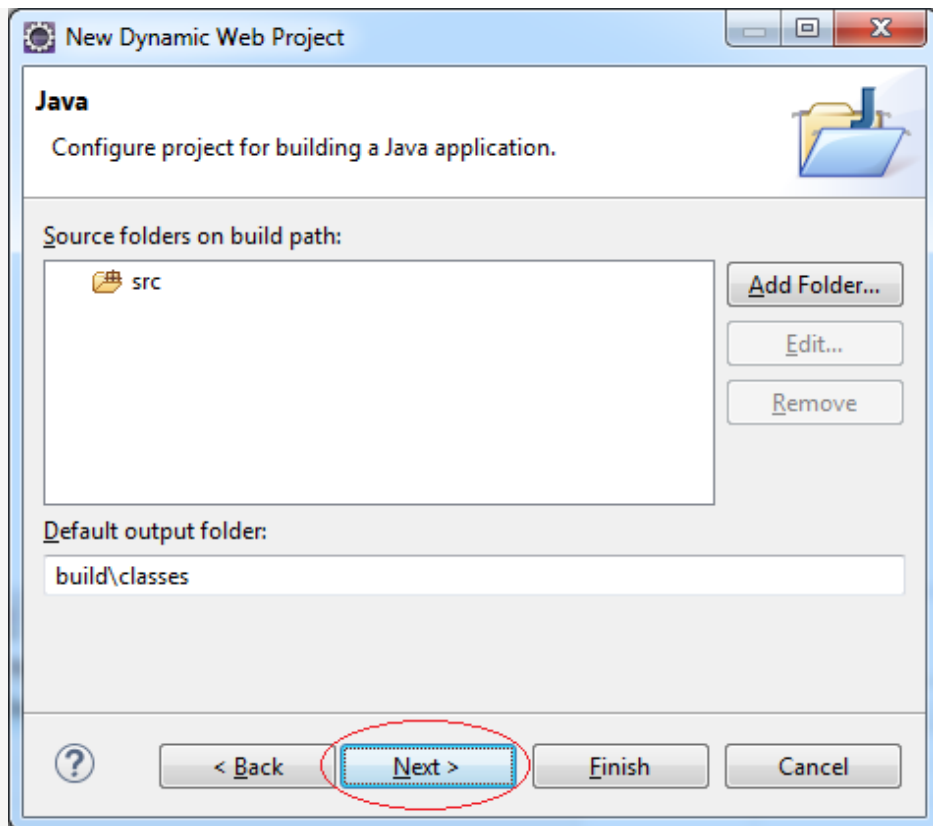
EAR project name:

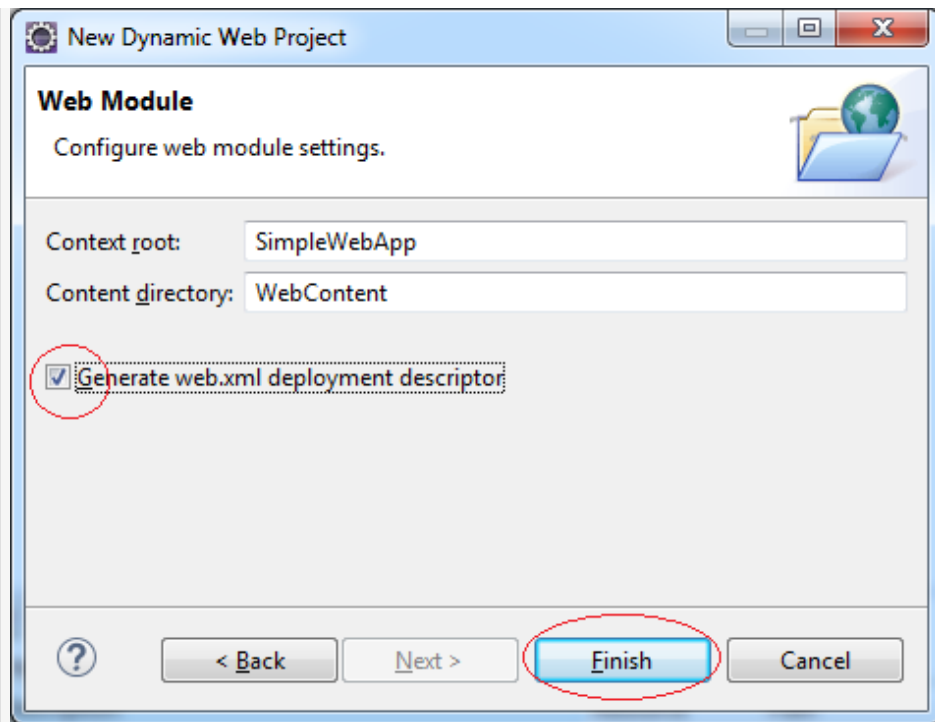
Working sets

☐ Add project to working sets



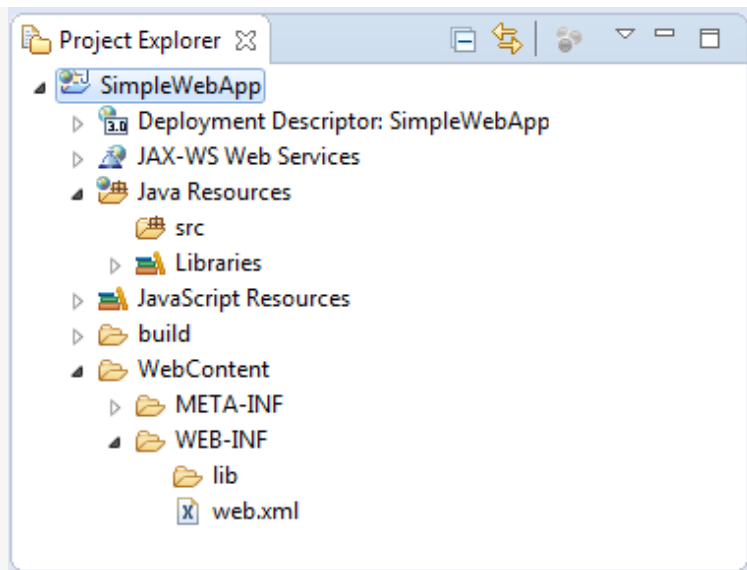




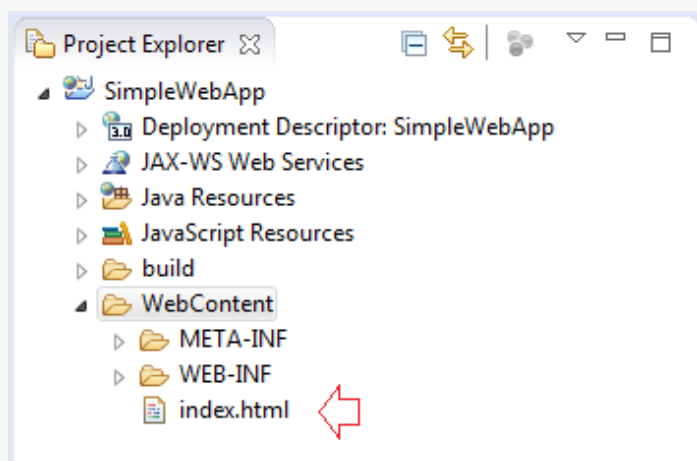


Project was created.





Add index.html



index.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
```

```
5 <title>Simple Web Application</title>
6 </head>
7
8 <body>
9
10 <h2>Simple Login Web Application using JSP/Servlet</h2>
11
12 <ul>
13 <li><a href="home">Home</a></li>
14 <li><a href="login">Login</a></li>
15 <li><a href="productList">Product List</a>
16 </ul>
17
18 </body>
19 </html>
```

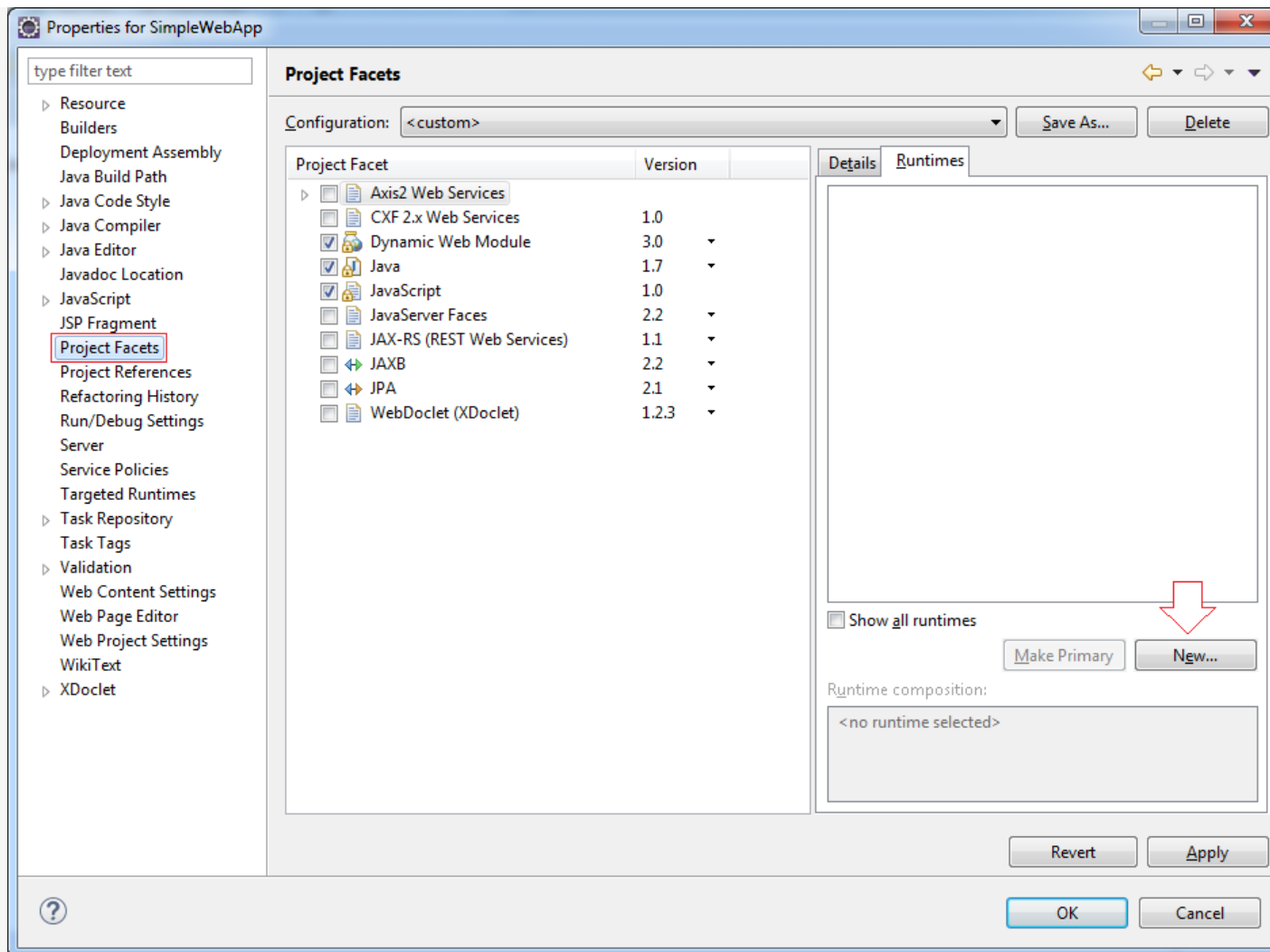
## 6- Configuring the runtime environment

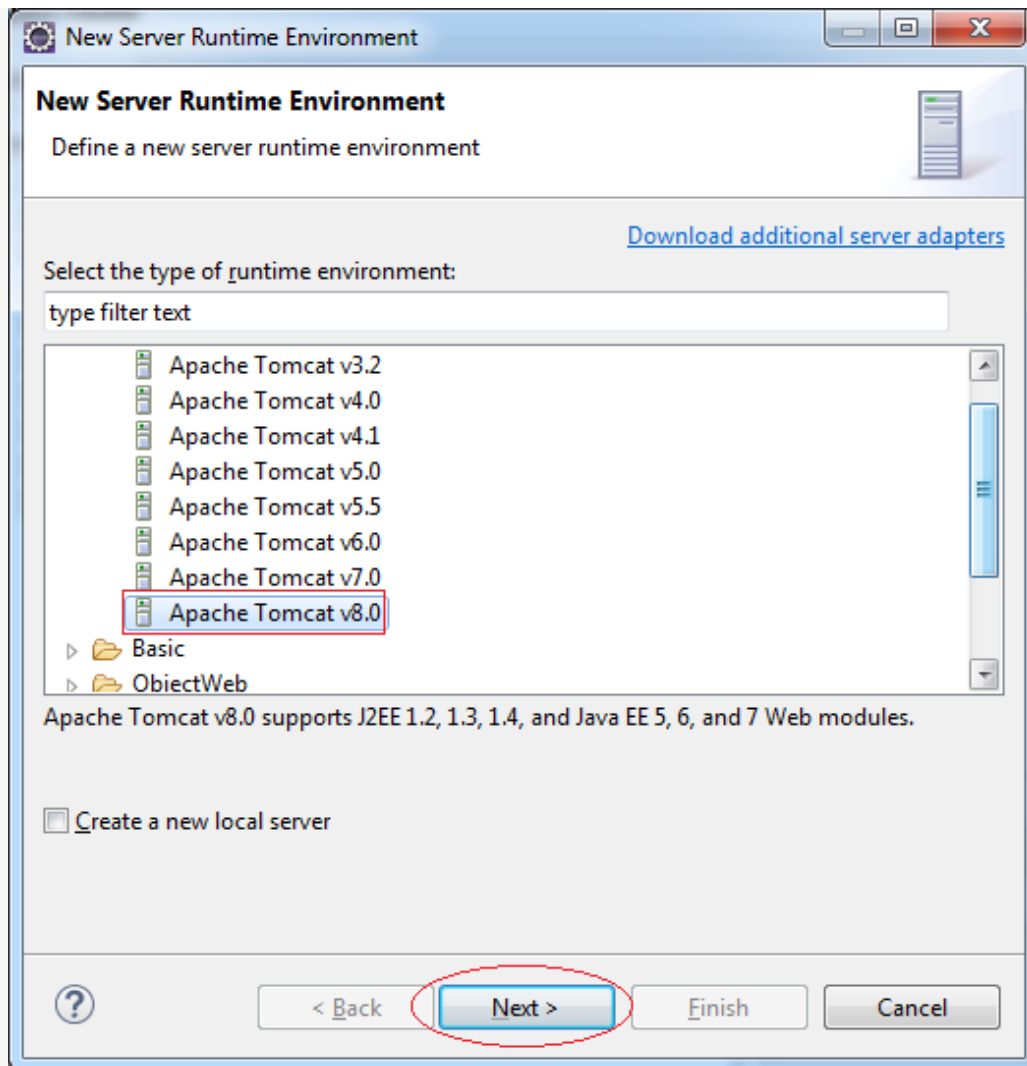
The application needs to run on a WebServer, such as **Tomcat Server**, you can refer to download and declaration instructions of **Server Tomcat** in **Eclipse** at:

- [Installing and Configuring Tomcat Server in Eclipse](#)

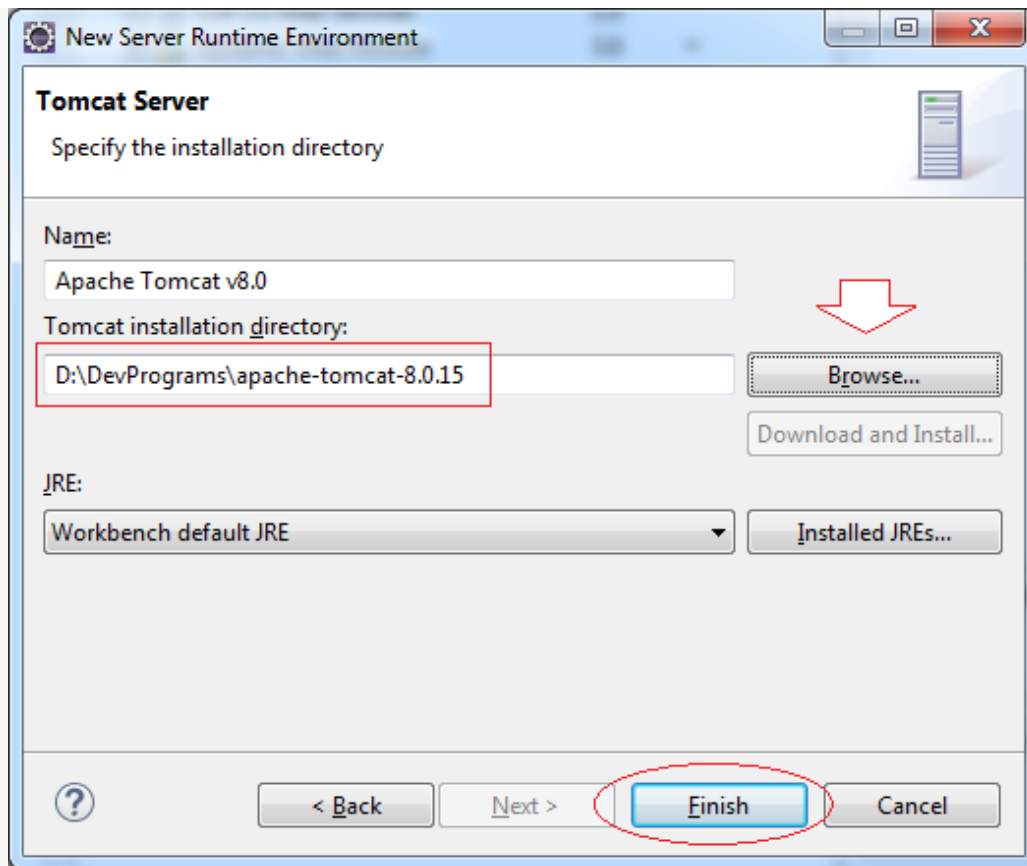
Right-click the **SimpleWebApp** select **Properties**.

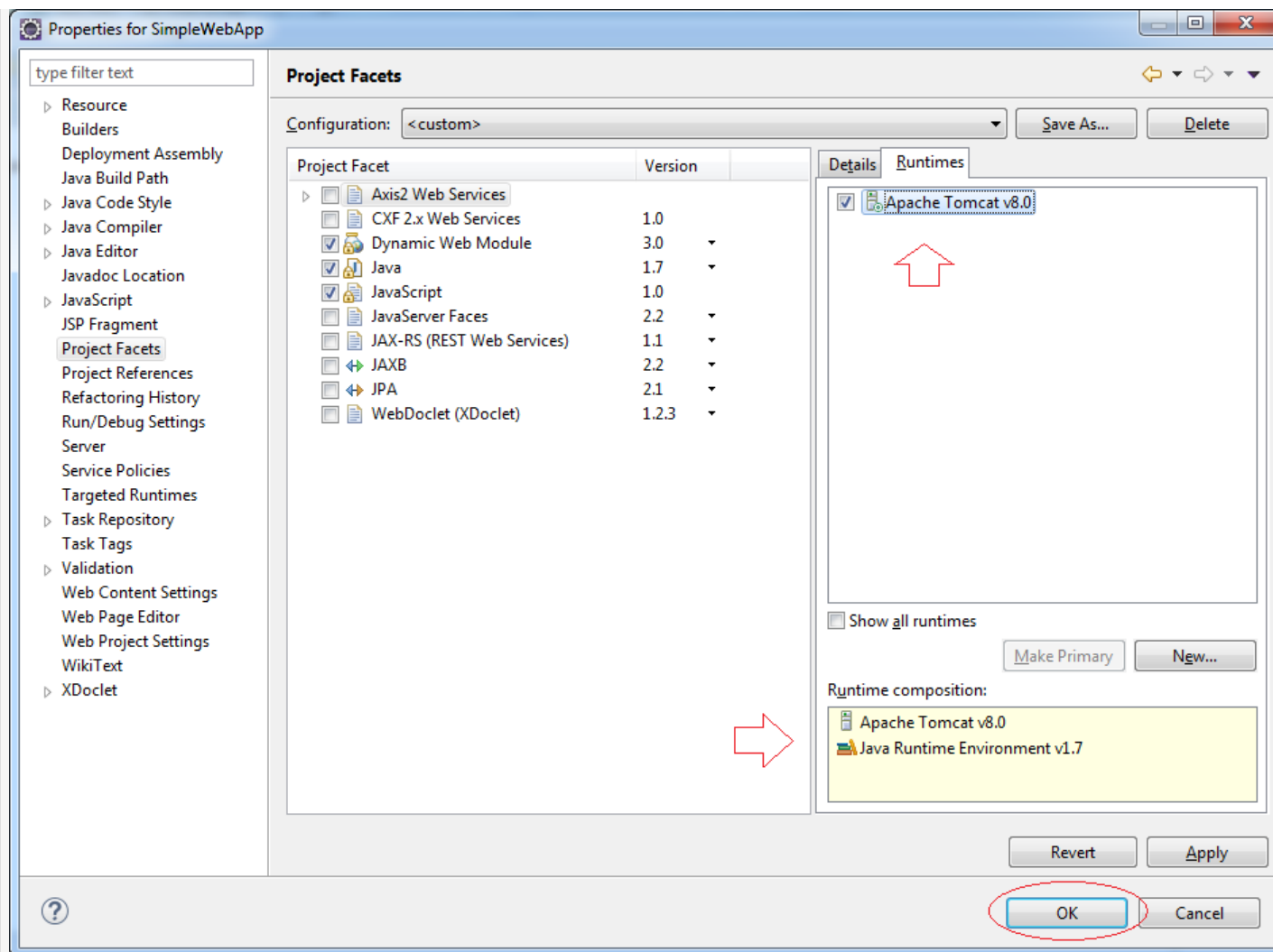










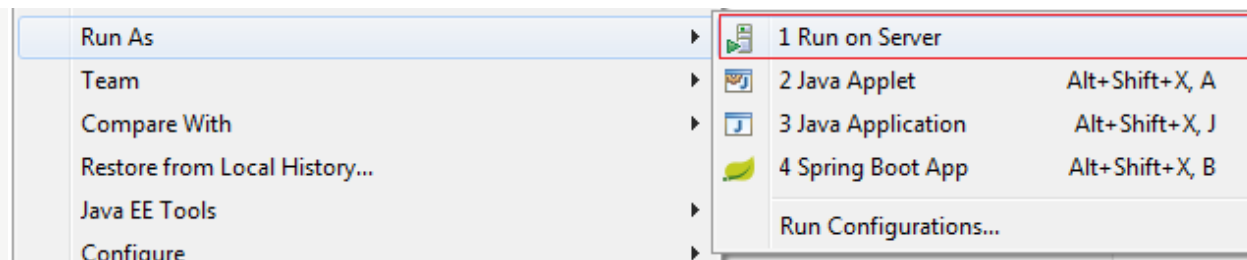


## 7- Run application for first time

Right-click on **SimpleWebApp**, select:

- Run As/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

[Download additional server adapters](#)

Select the server type:

type filter text

- Tomcat v5.5 Server
- Tomcat v6.0 Server
- Tomcat v7.0 Server
- Tomcat v8.0 Server**
- Basic
- ObjectWeb

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

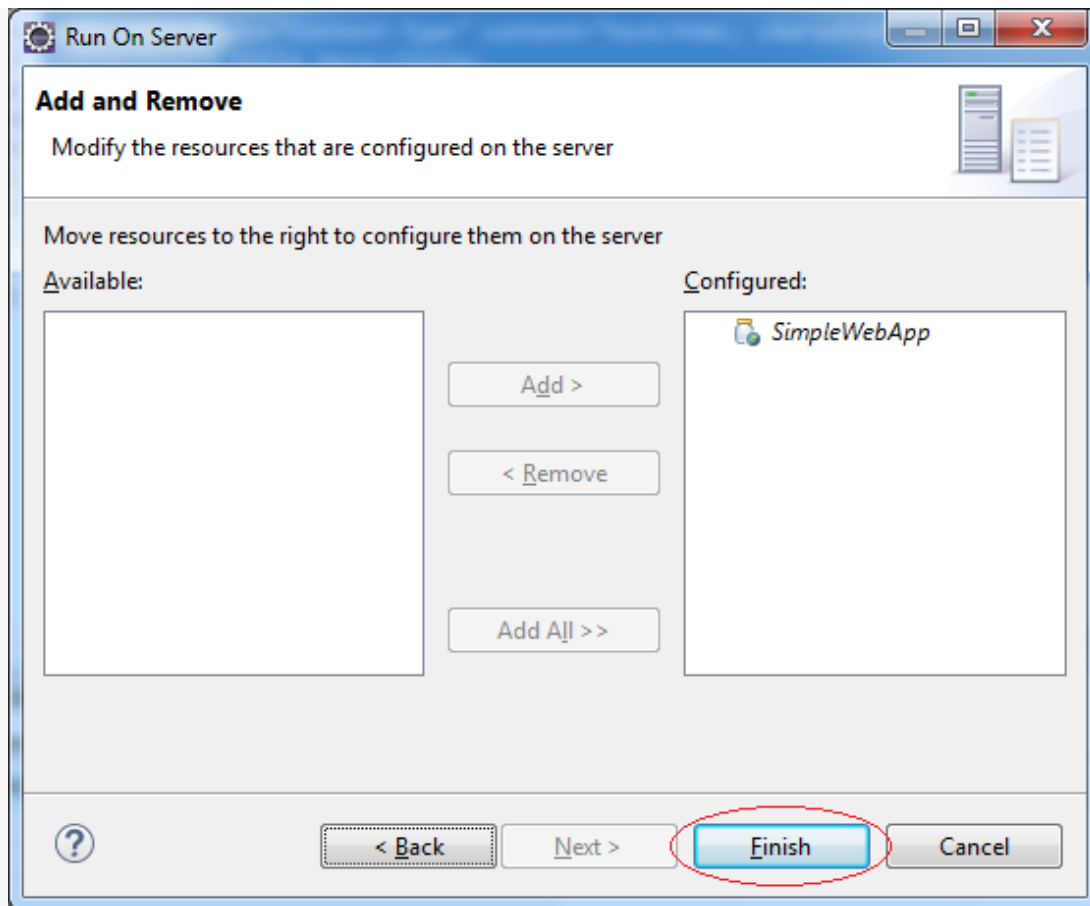
Server name: Tomcat v8.0 Server at localhost

Server runtime environment: Apache Tomcat v8.0 [Add...](#)

[Configure runtime environments...](#)

☐ Always use this server when running this project

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)



Application has been run:





OK, here everything is fine. We'll start programming a real Web application.

## 8- Download and declare JDBC library

You have to download JDBC library to driving the connection with the Database. In this document, I download both of 3 JDBC libraries for **Oracle, MySQL, SQL Server**, in practice, you only need JDBC library corresponding to the type of database you are using.

You can see download instruction of JDBC driver at:

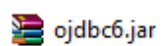
- [JDBC Driver Libraries for different types of database in Java](#)

*Or download here:*

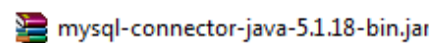
- [some-jdbc-drivers.zip](#) (MySQL + SQL Server + Oracle) o7planning link.



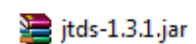
Results downloaded:



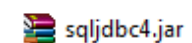
Oracle



MySQL



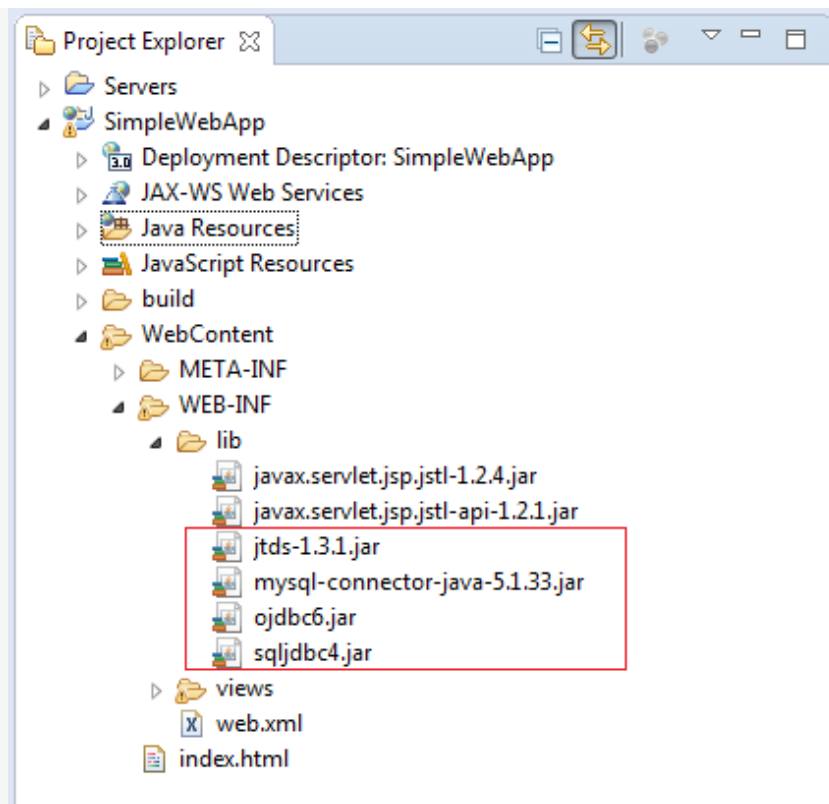
SQLServer



SQLServer

Copy these libraries into the **WEB-INF/lib**:





## 9- Download and declare JSTL library





You need to download 2 JSTL libraries to be able to use them in the JSP:

- javax.servlet.jsp.jstl-\*.jar
- javax.servlet.jsp.jstl-api-\*.jar
-  <http://mvnrepository.com/artifact/org.glassfish.web/javax.servlet.jsp.jstl>



Maven Repository: org.gla x

mvnrepository.com/artifact/org.glassfish.web/javax.servlet.jsp.jstl

Cache Implementations  
Cloud Computing  
Code Analyzers  
Collections  
Configuration Libraries  
Core Utilities  
Date and Time Utilities  
Dependency Injection  
Embedded SQL Databases  
HTML Parsers  
HTTP Clients  
I/O Utilities  
JDBC Extensions  
JDBC Pools  
JPA Implementations  
JSON Libraries

**Version** **Usages** **Type** **Date**

|       |   |         |             |
|-------|---|---------|-------------|
| 1.2.4 | 5 | release | (Jan, 2015) |
| 1.2.3 | 4 | release | (May, 2014) |
| 1.2.2 | 7 | release | (Aug, 2012) |
| 1.2.1 | 1 | release | (Dec, 2011) |

**Related Books**

A website in 3 minutes!  
Using The Grid, the world's first artificial intelligence platform that designs websites.  
"The Future"  
FAST COMPANY  
FIND OUT HOW

- <http://mvnrepository.com/artifact/javax.servlet.jsp.jstl/javax.servlet.jsp.jstl-api>

Maven Repository: javax.s...

mvnrepository.com/artifact/javax.servlet.jsp.jstl/javax.servlet.jsp.jstl-api

Google

| Version | Usages | Type    | Date        |
|---------|--------|---------|-------------|
| 1.2.1   | 30     | release | (Dec, 2011) |

This ad was based on your browsing activity. We are committed to providing you with transparency and control over the types of advertising you see.

[Set Your Ad Preferences »](#)

[The Trade Desk Privacy Policy »](#)

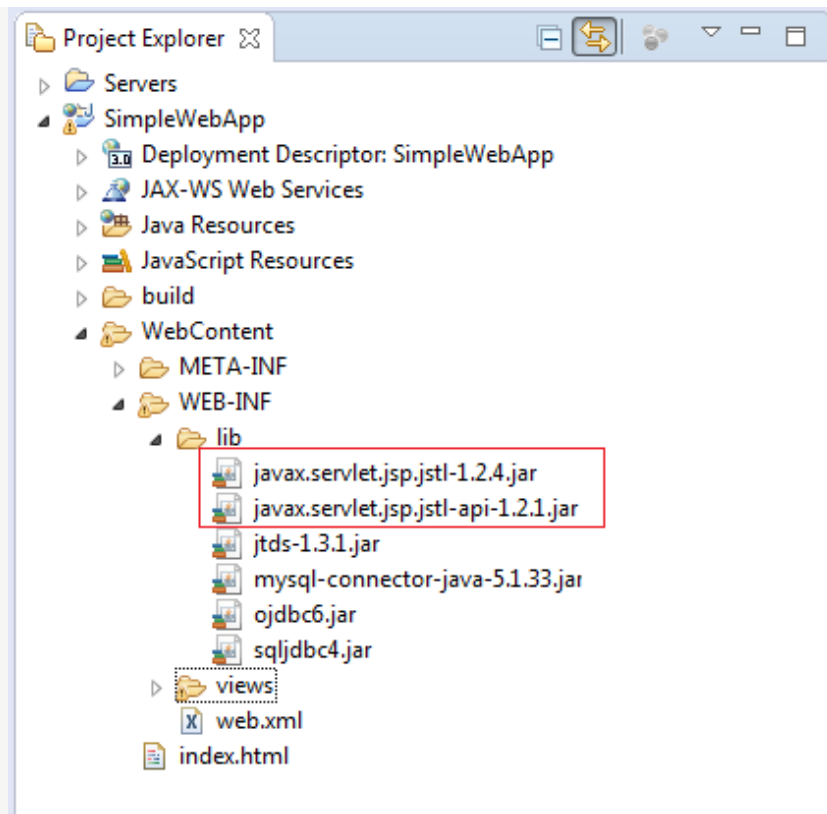
AdChoices by TRUSTe

Popular Tags

android apache api assets beans  
build build-system bytecode cache  
client codehaus config container  
database eclipse ejb esb framework

Copy 2 jar files that you just downloaded into the /WEB-INF/lib:

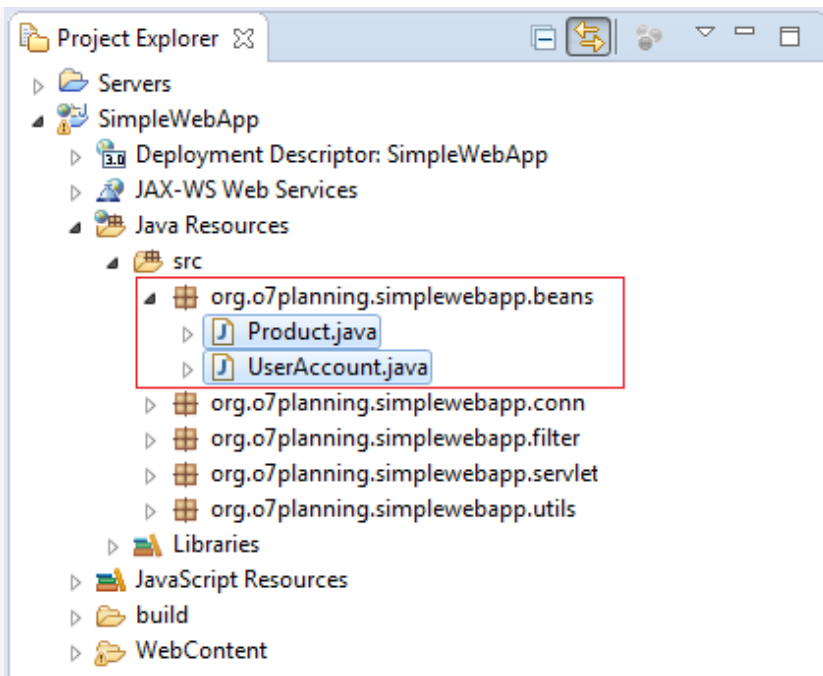




## 10- Javabean classes simulated tables in the database

Create 2 JavaBean classes, wherein each class simulated a table in the database:





### UserAccount.java

```
1 package org.o7planning.simplewebapp.beans;
2
3 public class UserAccount {
4
5     public static final String GENDER_MALE = "M";
6     public static final String GENDER_FEMALE = "F";
7
8     private String userName;
9     private String gender;
10    private String password;
11
12
13    public UserAccount() {
14    }
15
16
17    public String getUserName() {
18        return userName;
19    }
20 }
```

```
21 public void setUsername(String userName) {
22     this.userName = userName;
23 }
24
25 public String getGender() {
26     return gender;
27 }
28
29 public void setGender(String gender) {
30     this.gender = gender;
31 }
32
33 public String getPassword() {
34     return password;
35 }
36
37 public void setPassword(String password) {
38     this.password = password;
39 }
40
41 }
```

### Product.java

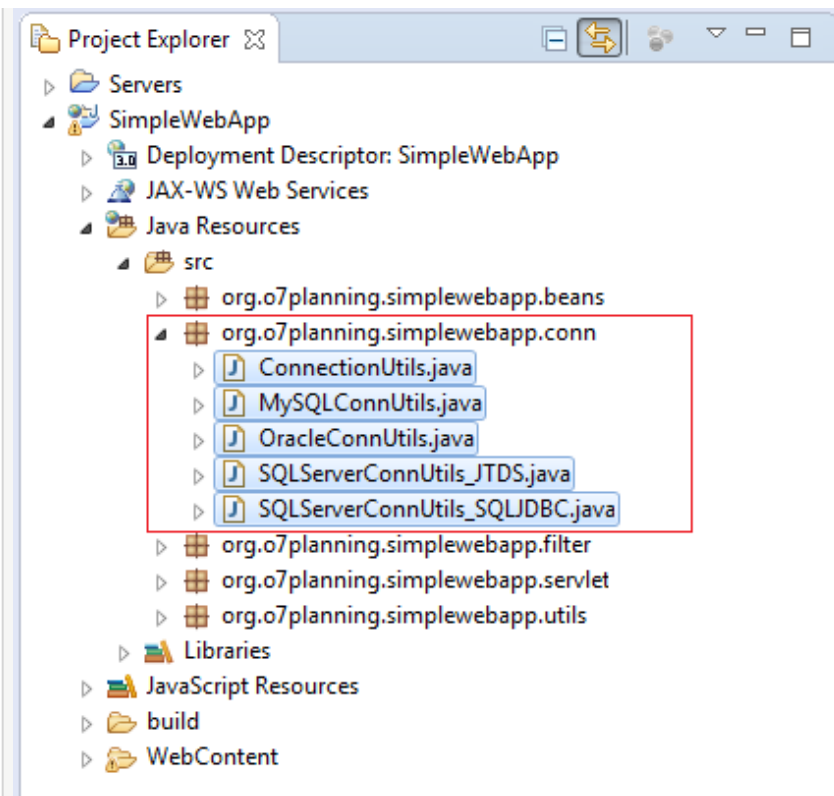
```
1 package org.o7planning.simplewebapp.beans;
2
3 public class Product {
4
5     private String code;
6     private String name;
7     private float price;
8
9     public Product() {
10
11     }
12
13     public Product(String code, String name, float price) {
14         this.code = code;
15         this.name = name;
16         this.price = price;
17     }
18
19     public String getCode() {
20         return code;
21     }
22
23     public void setCode(String code) {
```



```
24     this.code = code;
25 }
26
27 public String getName() {
28     return name;
29 }
30
31 public void setName(String name) {
32     this.name = name;
33 }
34
35 public float getPrice() {
36     return price;
37 }
38
39 public void setPrice(float price) {
40     this.price = price;
41 }
42
43 }
```

## 11- Database Connection Utility classes





### MySQLConnUtils.java

```
1 package org.o7planning.simplewebapp.conn;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class MySQLConnUtils {
8
9     public static Connection getMySQLConnection()
10         throws ClassNotFoundException, SQLException {
11         // Note: Change the connection parameters accordingly.
12         String hostName = "localhost";
13         String dbName = "mytest";
14         String userName = "root";
15         String password = "12345";
16         return getMySQLConnection(hostName, dbName, userName, password);
17     }
18 }
```





```

17 }
18
19 public static Connection getMySQLConnection(String hostName, String dbName,
20     String userName, String password) throws SQLException,
21     ClassNotFoundException {
22
23     Class.forName("com.mysql.jdbc.Driver");
24
25     // URL Connection for MySQL:
26     // Example:
27     // jdbc:mysql://localhost:3306/simplehr
28     String connectionURL = "jdbc:mysql://" + hostName + ":3306/" + dbName;
29
30     Connection conn = DriverManager.getConnection(connectionURL, userName,
31         password);
32     return conn;
33 }
34 }

```

### OracleConnUtils.java

```

1 package org.o7planning.simplewebapp.conn;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class OracleConnUtils {
8
9     public static Connection getOracleConnection()
10         throws ClassNotFoundException, SQLException {
11
12         // Note: Change the connection parameters accordingly.
13         String hostName = "localhost";
14         String sid = "db12c";
15         String userName = "mytest";
16         String password = "12345";
17
18         return getOracleConnection(hostName, sid, userName, password);
19     }
20
21     public static Connection getOracleConnection(String hostName, String sid,
22         String userName, String password) throws ClassNotFoundException,
23         SQLException {
24
25         Class.forName("oracle.jdbc.driver.OracleDriver");
26

```



```

27 // URL Connection for Oracle
28 // Example:
29 // jdbc:oracle:thin:@localhost:1521:db11g
30 // jdbc:oracle:thin:@//HOSTNAME:PORT/SERVICENAME
31 String connectionURL = "jdbc:oracle:thin:@" + hostName + ":1521:" + sid;
32
33 Connection conn = DriverManager.getConnection(connectionURL, userName,
34     password);
35 return conn;
36 }
37 }

```

### SQLServerConnUtils\_JTDS.java

```

1 package org.o7planning.simplewebapp.conn;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class SQLServerConnUtils_JTDS {
8
9     // Connect to SQLServer
10    // (Using JDBC Driver of JTDS library)
11    public static Connection getSQLServerConnection_JTDS() //
12        throws SQLException, ClassNotFoundException {
13
14        // Note: Change the connection parameters accordingly.
15        String hostName = "localhost";
16        String sqlInstanceName = "SQLEXPRESS";
17        String database = "mytest";
18        String userName = "sa";
19        String password = "12345";
20
21        return getSQLServerConnection_JTDS(hostName, sqlInstanceName, database, userName, password);
22    }
23
24    // Connect to SQLServer, using JTDS library
25    private static Connection getSQLServerConnection_JTDS(String hostName, //
26        String sqlInstanceName, String database, String userName, String password)
27        throws ClassNotFoundException, SQLException {
28
29        Class.forName("net.sourceforge.jtds.jdbc.Driver");
30
31        // Example:
32        // jdbc:jtds:sqlserver://localhost:1433/simplehr;instance=SQLEXPRESS
33        String connectionURL = "jdbc:jtds:sqlserver://" + hostName + ":1433/" //

```



```

34         + database + ";instance=" + sqlInstanceName;
35
36     Connection conn = DriverManager.getConnection(connectionURL, userName, password);
37     return conn;
38 }
39
40 }

```

### SQLServerConnUtils\_SQLJDBC.java

```

1  package org.o7planning.simplewebapp.conn;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class SQLServerConnUtils_SQLJDBC {
8
9      // Connect to SQL Server.
10     // (Using JDBC Driver: SQLJDBC)
11     public static Connection getSQLServerConnection_SQLJDBC() //
12         throws ClassNotFoundException, SQLException {
13
14         // Note: Change the connection parameters accordingly.
15         String hostName = "localhost";
16         String sqlInstanceName = "SQLEXPRESS";
17         String database = "mytest";
18         String userName = "sa";
19         String password = "12345";
20
21         return getSQLServerConnection_SQLJDBC(hostName, sqlInstanceName, database, userName, password);
22     }
23
24     // Connect to SQLServer, using SQLJDBC Library.
25     private static Connection getSQLServerConnection_SQLJDBC(String hostName, //
26         String sqlInstanceName, String database, String userName, String password)//
27         throws ClassNotFoundException, SQLException {
28
29         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
30
31         // Example:
32         // jdbc:sqlserver://ServerIp:1433/SQLEXPRESS;databaseName=simplehr
33         String connectionURL = "jdbc:sqlserver://" + hostName + ":1433" //
34             + ";instance=" + sqlInstanceName + ";databaseName=" + database;
35
36         Connection conn = DriverManager.getConnection(connectionURL, userName, password);
37         return conn;

```

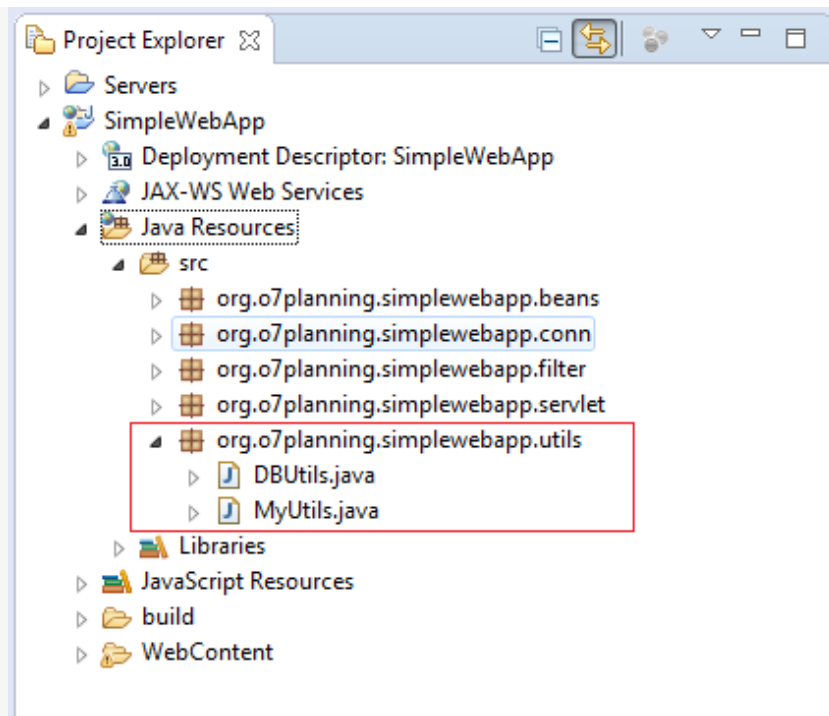


```
38 }
39
40 }
```

### ConnectionUtils.java

```
1 package org.o7planning.simplewebapp.conn;
2
3 import java.sql.Connection;
4 import java.sql.SQLException;
5
6 public class ConnectionUtils {
7
8     public static Connection getConnection()
9         throws ClassNotFoundException, SQLException {
10
11         // Here I using Oracle Database.
12         // (You can change to use another database.)
13         return OracleConnUtils.getOracleConnection();
14
15         // return OracleConnUtils.getOracleConnection();
16         // return MySQLConnUtils.getMySQLConnection();
17         // return SQLServerConnUtils_JTDS.getSQLServerConnection_JTDS();
18         // return SQLServerConnUtils_SQLJDBC.getSQLServerConnection_SQLJDBC();
19         // return PostGresConnUtils.getPostGresConnection();
20     }
21
22     public static void closeQuietly(Connection conn) {
23         try {
24             conn.close();
25         } catch (Exception e) {
26         }
27     }
28
29     public static void rollbackQuietly(Connection conn) {
30         try {
31             conn.rollback();
32         } catch (Exception e) {
33         }
34     }
35 }
```

## 12- The utility class & manipulate data



### MyUtils.java

```
1 package org.o7planning.simplewebapp.utils;
2
3 import java.sql.Connection;
4
5 import javax.servlet.ServletException;
6 import javax.servlet.http.Cookie;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9 import javax.servlet.http.HttpSession;
10
11 import org.o7planning.simplewebapp.beans.UserAccount;
12
13 public class MyUtils {
14
15     public static final String ATT_NAME_CONNECTION = "ATTRIBUTE_FOR_CONNECTION";
16
17     private static final String ATT_NAME_USER_NAME = "ATTRIBUTE_FOR_STORE_USER_NAME_IN_COOKIE";
18
19     // Store Connection in request attribute.
```

```

20 // (Information stored only exist during requests)
21 public static void storeConnection(ServletRequest request, Connection conn) {
22     request.setAttribute(ATT_NAME_CONNECTION, conn);
23 }
24
25 // Get the Connection object has been stored in attribute of the request.
26 public static Connection getStoredConnection(ServletRequest request) {
27     Connection conn = (Connection) request.getAttribute(ATT_NAME_CONNECTION);
28     return conn;
29 }
30
31 // Store user info in Session.
32 public static void storeLoggedInUser(HttpSession session, UserAccount loggedInUser) {
33     // On the JSP can access via ${loggedInUser}
34     session.setAttribute("loggedInUser", loggedInUser);
35 }
36
37 // Get the user information stored in the session.
38 public static UserAccount getLoggedInUser(HttpSession session) {
39     UserAccount loggedInUser = (UserAccount) session.getAttribute("loggedInUser");
40     return loggedInUser;
41 }
42
43 // Store info in Cookie
44 public static void storeUserCookie(HttpServletResponse response, UserAccount user) {
45     System.out.println("Store user cookie");
46     Cookie cookieUserName = new Cookie(ATT_NAME_USER_NAME, user.getUserName());
47     // 1 day (Converted to seconds)
48     cookieUserName.setMaxAge(24 * 60 * 60);
49     response.addCookie(cookieUserName);
50 }
51
52 public static String getUserNamelnCookie(HttpServletRequest request) {
53     Cookie[] cookies = request.getCookies();
54     if (cookies != null) {
55         for (Cookie cookie : cookies) {
56             if (ATT_NAME_USER_NAME.equals(cookie.getName())) {
57                 return cookie.getValue();
58             }
59         }
60     }
61     return null;
62 }
63
64 // Delete cookie.
65 public static void deleteUserCookie(HttpServletResponse response) {
66     Cookie cookieUserName = new Cookie(ATT_NAME_USER_NAME, null);
67     // 0 seconds (This cookie will expire immediately)
68     cookieUserName.setMaxAge(0);

```



```
69     response.addCookie(cookieUserName);
70 }
71
72 }
```

### DBUtils.java

```
1  package org.o7planning.simplewebapp.utils;
2
3  import java.sql.Connection;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.util.ArrayList;
8  import java.util.List;
9
10 import org.o7planning.simplewebapp.beans.Product;
11 import org.o7planning.simplewebapp.beans.UserAccount;
12
13 public class DBUtils {
14
15     public static UserAccount findUser(Connection conn, //
16         String userName, String password) throws SQLException {
17
18         String sql = "Select a.User_Name, a.Password, a.Gender from User_Account a " //
19             + " where a.User_Name = ? and a.password= ?";
20
21         PreparedStatement pstmt = conn.prepareStatement(sql);
22         pstmt.setString(1, userName);
23         pstmt.setString(2, password);
24         ResultSet rs = pstmt.executeQuery();
25
26         if (rs.next()) {
27             String gender = rs.getString("Gender");
28             UserAccount user = new UserAccount();
29             user.setUserName(userName);
30             user.setPassword(password);
31             user.setGender(gender);
32             return user;
33         }
34         return null;
35     }
36
37     public static UserAccount findUser(Connection conn, String userName) throws SQLException {
38
39         String sql = "Select a.User_Name, a.Password, a.Gender from User_Account a " //
40             + " where a.User_Name = ? ";
```



```

41
42 PreparedStatement pstmt = conn.prepareStatement(sql);
43 pstmt.setString(1, userName);
44
45 ResultSet rs = pstmt.executeQuery();
46
47 if (rs.next()) {
48     String password = rs.getString("Password");
49     String gender = rs.getString("Gender");
50     UserAccount user = new UserAccount();
51     user.setUserName(userName);
52     user.setPassword(password);
53     user.setGender(gender);
54     return user;
55 }
56 return null;
57 }
58
59 public static List<Product> queryProduct(Connection conn) throws SQLException {
60     String sql = "Select a.Code, a.Name, a.Price from Product a ";
61
62     PreparedStatement pstmt = conn.prepareStatement(sql);
63
64     ResultSet rs = pstmt.executeQuery();
65     List<Product> list = new ArrayList<Product>();
66     while (rs.next()) {
67         String code = rs.getString("Code");
68         String name = rs.getString("Name");
69         float price = rs.getFloat("Price");
70         Product product = new Product();
71         product.setCode(code);
72         product.setName(name);
73         product.setPrice(price);
74         list.add(product);
75     }
76     return list;
77 }
78
79 public static Product findProduct(Connection conn, String code) throws SQLException {
80     String sql = "Select a.Code, a.Name, a.Price from Product a where a.Code=?";
81
82     PreparedStatement pstmt = conn.prepareStatement(sql);
83     pstmt.setString(1, code);
84
85     ResultSet rs = pstmt.executeQuery();
86
87     while (rs.next()) {
88         String name = rs.getString("Name");
89         float price = rs.getFloat("Price");

```





```

90     Product product = new Product(code, name, price);
91     return product;
92 }
93 return null;
94 }
95
96 public static void updateProduct(Connection conn, Product product) throws SQLException {
97     String sql = "Update Product set Name=?, Price=? where Code=? ";
98
99     PreparedStatement pstmt = conn.prepareStatement(sql);
100
101     pstmt.setString(1, product.getName());
102     pstmt.setFloat(2, product.getPrice());
103     pstmt.setString(3, product.getCode());
104     pstmt.executeUpdate();
105 }
106
107 public static void insertProduct(Connection conn, Product product) throws SQLException {
108     String sql = "Insert into Product(Code, Name,Price) values (?, ?, ?)";
109
110     PreparedStatement pstmt = conn.prepareStatement(sql);
111
112     pstmt.setString(1, product.getCode());
113     pstmt.setString(2, product.getName());
114     pstmt.setFloat(3, product.getPrice());
115
116     pstmt.executeUpdate();
117 }
118
119 public static void deleteProduct(Connection conn, String code) throws SQLException {
120     String sql = "Delete From Product where Code= ?";
121
122     PreparedStatement pstmt = conn.prepareStatement(sql);
123
124     pstmt.setString(1, code);
125
126     pstmt.executeUpdate();
127 }
128
129 }

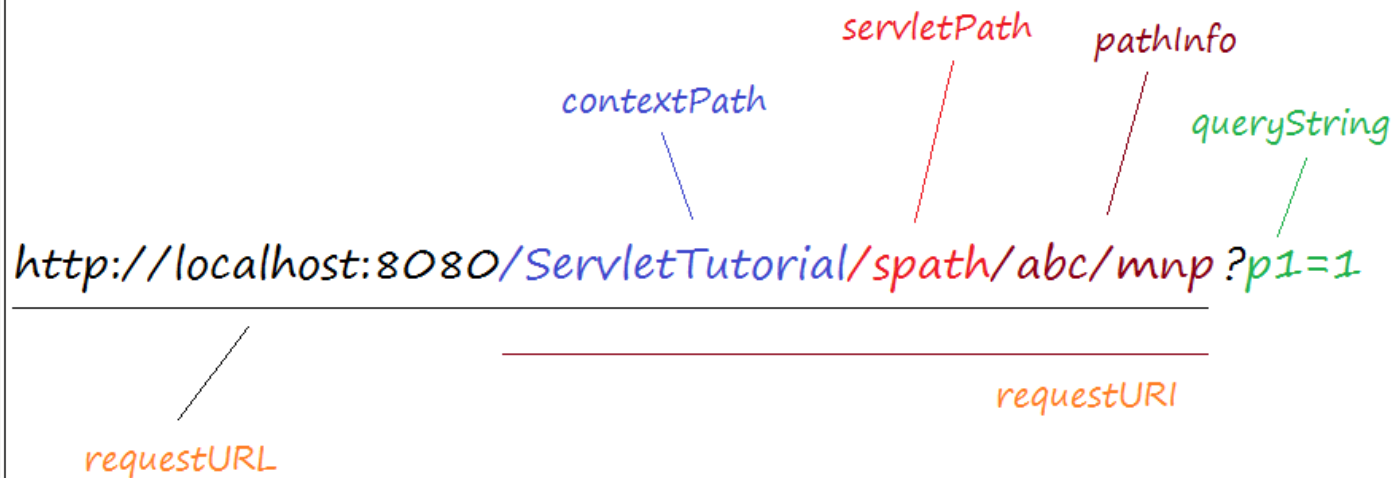
```

## 13- Create Servlet Filter connect to Database

In **JDBCFilter**, I checked which requests actually referred to a Servlet, so that you can see the picture below to find it easy to understand, it describes the relationship between the concepts of Servlet.



Servlet *url-pattern* = /spath/\*



**JDBCFilter** with *url-pattern* = /\* means that all requests of users have go through this filter.

**JDBCFilter** will check the request to ensure that it only opens JDBC connection for the necessary request, eg for Servlet, avoid opening JDBC connection to common requests like image, css, js, html

#### JDBCFilter.java

```
1 package org.o7planning.simplewebapp.filter;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.util.Collection;
6 import java.util.Map;
7
8 import javax.servlet.Filter;
9 import javax.servlet.FilterChain;
10 import javax.servlet.FilterConfig;
11 import javax.servlet.ServletException;
12 import javax.servlet.ServletRegistration;
13 import javax.servlet.ServletRequest;
14 import javax.servlet.ServletResponse;
15 import javax.servlet.annotation.WebFilter;
16 import javax.servlet.http.HttpServletRequest;
17
18 import org.o7planning.simplewebapp.conn.ConnectionUtils;
```

```

19 import org.o7planning.simplewebapp.utils.MyUtils;
20
21 @WebFilter(filterName = "jdbcFilter", urlPatterns = { "/" })
22 public class JDBCFilter implements Filter {
23
24     public JDBCFilter() {
25     }
26
27     @Override
28     public void init(FilterConfig fConfig) throws ServletException {
29
30     }
31
32     @Override
33     public void destroy() {
34
35     }
36
37     // Check the target of the request is a servlet?
38     private boolean needJDBC(HttpServletRequest request) {
39         System.out.println("JDBC Filter");
40         //
41         // Servlet Url-pattern: /spath/*
42         //
43         // => /spath
44         String servletPath = request.getServletPath();
45         // => /abc/mnp
46         String pathInfo = request.getPathInfo();
47
48         String urlPattern = servletPath;
49
50         if (pathInfo != null) {
51             // => /spath/*
52             urlPattern = servletPath + "/*";
53         }
54
55         // Key: servletName.
56         // Value: ServletRegistration
57         Map<String, ? extends ServletRegistration> servletRegistrations = request.getServletContext()
58             .getServletRegistrations();
59
60         // Collection of all servlet in your Webapp.
61         Collection<? extends ServletRegistration> values = servletRegistrations.values();
62         for (ServletRegistration sr : values) {
63             Collection<String> mappings = sr.getMappings();
64             if (mappings.contains(urlPattern)) {
65                 return true;
66             }
67         }
68     }

```



```

68     return false;
69 }
70
71 @Override
72 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
73     throws IOException, ServletException {
74
75     HttpServletRequest req = (HttpServletRequest) request;
76
77     // Only open connections for the special requests.
78     // (For example, the path to the servlet, JSP, ..)
79     //
80     // Avoid open connection for commons request.
81     // (For example: image, css, javascript,...)
82     //
83     if (this.needJDBC(req)) {
84
85         System.out.println("Open Connection for: " + req.getServletPath());
86
87         Connection conn = null;
88         try {
89             // Create a Connection.
90             conn = ConnectionUtils.getConnection();
91             // Set auto commit to false.
92             conn.setAutoCommit(false);
93
94             // Store Connection object in attribute of request.
95             MyUtils.storeConnection(request, conn);
96
97             // Allow request to go forward
98             // (Go to the next filter or target)
99             chain.doFilter(request, response);
100
101             // Invoke the commit() method to complete the transaction with the DB.
102             conn.commit();
103         } catch (Exception e) {
104             e.printStackTrace();
105             ConnectionUtils.rollbackQuietly(conn);
106             throw new ServletException();
107         } finally {
108             ConnectionUtils.closeQuietly(conn);
109         }
110     }
111     // With commons requests (images, css, html, ..)
112     // No need to open the connection.
113     else {
114         // Allow request to go forward
115         // (Go to the next filter or target)
116         chain.doFilter(request, response);

```



```
117     }
118
119 }
120
121 }
```

## 14- Servlet Filter reads Cookies to automatically login

In case, the user logged in and remembered information in previous access (for example the day before). And now the user return, this Filter will check the Cookie information stored by the browser and automatic Login.

### CookieFilter.java

```
1 package org.o7planning.simplewebapp.filter;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.sql.SQLException;
6
7 import javax.servlet.Filter;
8 import javax.servlet.FilterChain;
9 import javax.servlet.FilterConfig;
10 import javax.servlet.ServletException;
11 import javax.servlet.ServletRequest;
12 import javax.servlet.ServletResponse;
13 import javax.servlet.annotation.WebFilter;
14 import javax.servlet.http.HttpServletRequest;
15 import javax.servlet.http.HttpSession;
16
17 import org.o7planning.simplewebapp.beans.UserAccount;
18 import org.o7planning.simplewebapp.utils.DBUtils;
19 import org.o7planning.simplewebapp.utils.MyUtils;
20
21 @WebFilter(filterName = "cookieFilter", urlPatterns = { "/*" })
22 public class CookieFilter implements Filter {
23
24     public CookieFilter() {
25     }
26
27     @Override
28     public void init(FilterConfig fConfig) throws ServletException {
29     }
30
31 }
```



```

32  @Override
33  public void destroy() {
34
35  }
36
37  @Override
38  public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
39      throws IOException, ServletException {
40      HttpServletRequest req = (HttpServletRequest) request;
41      HttpSession session = req.getSession();
42
43      UserAccount userInSession = MyUtils.getLoginUser(session);
44      //
45      if (userInSession != null) {
46          session.setAttribute("COOKIE_CHECKED", "CHECKED");
47          chain.doFilter(request, response);
48          return;
49      }
50
51      // Connection was created in JDBCFilter.
52      Connection conn = MyUtils.getStoredConnection(request);
53
54      // Flag check cookie
55      String checked = (String) session.getAttribute("COOKIE_CHECKED");
56      if (checked == null && conn != null) {
57          String userName = MyUtils.getUserNameInCookie(req);
58          try {
59              UserAccount user = DBUtils.findUser(conn, userName);
60              MyUtils.storeLoginUser(session, user);
61          } catch (SQLException e) {
62              e.printStackTrace();
63          }
64          // Mark checked Cookies.
65          session.setAttribute("COOKIE_CHECKED", "CHECKED");
66      }
67
68      chain.doFilter(request, response);
69  }
70
71  }

```

## NOTE:



*JDBCFilter & CookieFilter have the same url-pattern =/\*, you must be configured to ensure that JDBCFilter is executed first. Therefore, you need to declare the order in web.xml (There is no way to declare the order by Annotation).*

```
1 <filter-mapping>
2   <filter-name>jdbcFilter</filter-name>
3   <url-pattern>/*</url-pattern>
4 </filter-mapping>
5
6 <filter-mapping>
7   <filter-name>cookieFilter</filter-name>
8   <url-pattern>/*</url-pattern>
9 </filter-mapping>
```

See full **web.xml**:

#### web.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns="http://java.sun.com/xml/ns/javaee"
4   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5     http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
6   id="WebApp_ID" version="3.0">
7   <display-name>SimpleWebApp</display-name>
8
9
10  <filter-mapping>
11    <filter-name>jdbcFilter</filter-name>
12    <url-pattern>/*</url-pattern>
13  </filter-mapping>
14
15  <filter-mapping>
16    <filter-name>cookieFilter</filter-name>
17    <url-pattern>/*</url-pattern>
18  </filter-mapping>
19
20  <welcome-file-list>
21    <welcome-file>home</welcome-file>
22
23    <welcome-file>index.html</welcome-file>
24  </welcome-file-list>
```

```
25
26 </welcome-file-list>
27
28
29 </web-app>
```

## 15- EncodingFilter Servlet

### EncodingFilter.java

```
1 package org.o7planning.simplewebapp.filter;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.util.Collection;
6 import java.util.Map;
7
8 import javax.servlet.Filter;
9 import javax.servlet.FilterChain;
10 import javax.servlet.FilterConfig;
11 import javax.servlet.ServletException;
12 import javax.servlet.ServletRegistration;
13 import javax.servlet.ServletRequest;
14 import javax.servlet.ServletResponse;
15 import javax.servlet.annotation.WebFilter;
16 import javax.servlet.http.HttpServletRequest;
17
18 import org.o7planning.simplewebapp.conn.ConnectionUtils;
19 import org.o7planning.simplewebapp.utils.MyUtils;
20
21 @WebFilter(filterName = "encodingFilter", urlPatterns = { "/*" })
22 public class EncodingFilter implements Filter {
23
24     public EncodingFilter() {
25     }
26
27     @Override
28     public void init(FilterConfig fConfig) throws ServletException {
29
30     }
31
32     @Override
33     public void destroy() {
34
35     }
```





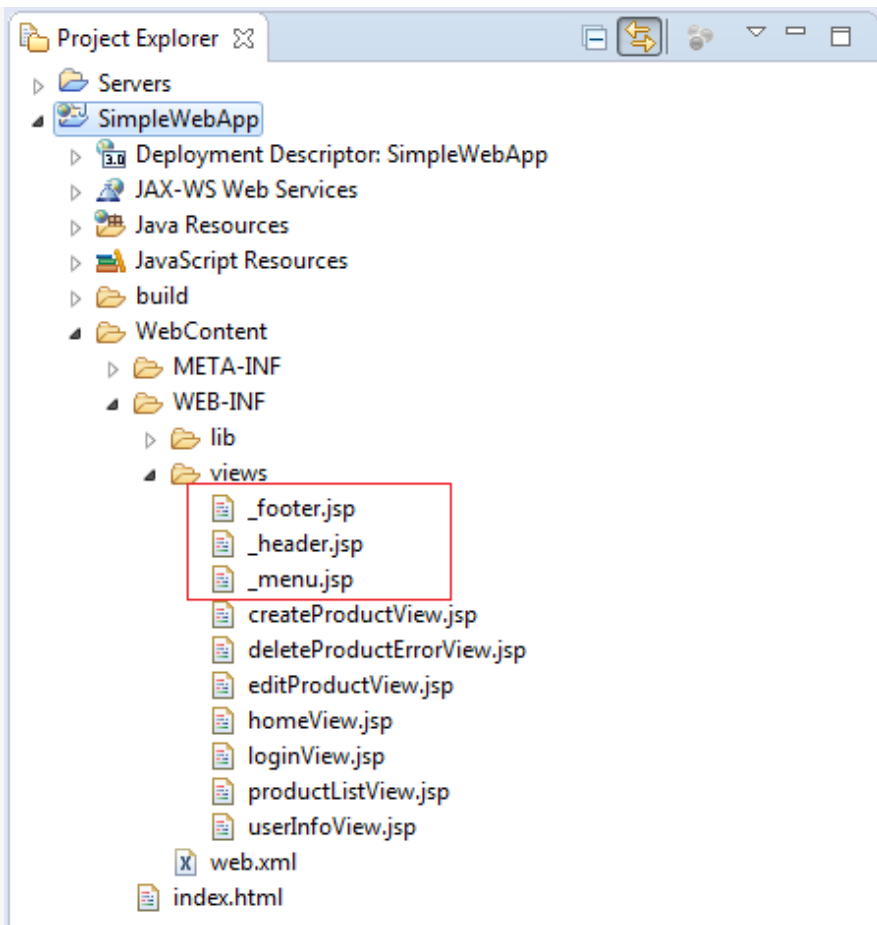
```
35 }
36
37 @Override
38 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
39     throws IOException, ServletException {
40     request.setCharacterEncoding("UTF-8");
41
42     chain.doFilter(request, response);
43 }
44
45 }
```

## 16- Pages reuse

Some JSP pages will be used to embed into other JSP page at Runtime, through the use of:

```
1 <jsp:include page="_header.jsp"></jsp:include>
2 <jsp:include page="_menu.jsp"></jsp:include>
3
4 <jsp:include page="_footer.jsp"></jsp:include>
```





### /WEB-INF/views/\_header.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <div style="background: #E0E0E0; height: 55px; padding: 5px;">
4   <div style="float: left">
5     <h1>My Site</h1>
6   </div>
7
8   <div style="float: right; padding: 10px; text-align: right;">
9
10    <!-- User store in session with attribute: loggedUser -->
11    Hello <b>${loggedUser.userName}</b>
```

```
12 <br/>
13 Search <input name="search">
14
15 </div>
16
17 </div>
```

### /WEB-INF/views/\_menu.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <div style="padding: 5px;">
5
6   <a href="${pageContext.request.contextPath}/">Home</a>
7   |
8   <a href="${pageContext.request.contextPath}/productList">Product List</a>
9   |
10  <a href="${pageContext.request.contextPath}/userInfo">My Account Info</a>
11  |
12  <a href="${pageContext.request.contextPath}/login">Login</a>
13
14 </div>
```

### /WEB-INF/views/\_footer.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <div
5   style="background: #E0E0E0; text-align: center; padding: 5px; margin-top: 10px;">
6
7   @Copyright o7planning.org
8
9 </div>
```

## 17- Home Page



When entering the default path, eg enter the site's domain name it will display the home page (Case *contextPath = ""*), you need to declare your home page in *<welcome-file-list>* of *web.xml*

Link below is showing the content of the page *index.html*

- <http://localhost:8081/SimpleWebApp/>

You need to design a home page as a **JSP** page to display dynamic information instead of a **html** page that contains static information.

#### web.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns="http://java.sun.com/xml/ns/javaee"
4   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5     http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
6   id="WebApp_ID" version="3.0">
7   <display-name>SimpleWebApp</display-name>
8
9
10  <filter-mapping>
11    <filter-name>jdbcFilter</filter-name>
12    <url-pattern>/*</url-pattern>
13  </filter-mapping>
```

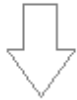


```
14
15 <filter-mapping>
16   <filter-name>cookieFilter</filter-name>
17   <url-pattern>/*</url-pattern>
18 </filter-mapping>
19
20 <welcome-file-list>
21   <welcome-file>home</welcome-file>
22   <welcome-file>index.html</welcome-file>
23
24 </welcome-file-list>
25
26 </web-app>
```

Models of Home Page:



<http://localhost:8080/SimpleWebApp/>



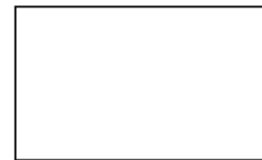
web.xml

```
<welcome-file-list>
  <welcome-file>home</welcome-file>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

/home



HomeServlet



forward



/WEB-INF/views/homeView.jsp

### HomeServlet.java

```
1 package org.o7planning.simplewebapp.servlet;
2
3 import java.io.IOException;
4
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 @WebServlet(urlPatterns = { "/home" })
13 public class HomeServlet extends HttpServlet {
14     private static final long serialVersionUID = 1L;
15 }
```



```

16 public HomeServlet() {
17     super();
18 }
19
20 @Override
21 protected void doGet(HttpServletRequest request, HttpServletResponse response)
22     throws ServletException, IOException {
23
24
25     // Forward to /WEB-INF/views/homeView.jsp
26     // (Users can not access directly into JSP pages placed in WEB-INF)
27     RequestDispatcher dispatcher = this.getServletContext().getRequestDispatcher("/WEB-INF/views/homeView.jsp");
28
29     dispatcher.forward(request, response);
30
31 }
32
33 @Override
34 protected void doPost(HttpServletRequest request, HttpServletResponse response)
35     throws ServletException, IOException {
36     doGet(request, response);
37 }
38
39 }

```

### /WEB-INF/views/homeView.jsp

```



1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="UTF-8">
7     <title>Home Page</title>
8   </head>
9   <body>
10
11     <jsp:include page="_header.jsp"></jsp:include>
12     <jsp:include page="_menu.jsp"></jsp:include>
13
14     <h3>Home Page</h3>
15
16     This is demo Simple web application using jsp, servlet & Jdbc. <br><br>
17     <b>It includes the following functions:</b>
18     <ul>
19       <li>Login</li>
20       <li>Storing user information in cookies</li>

```



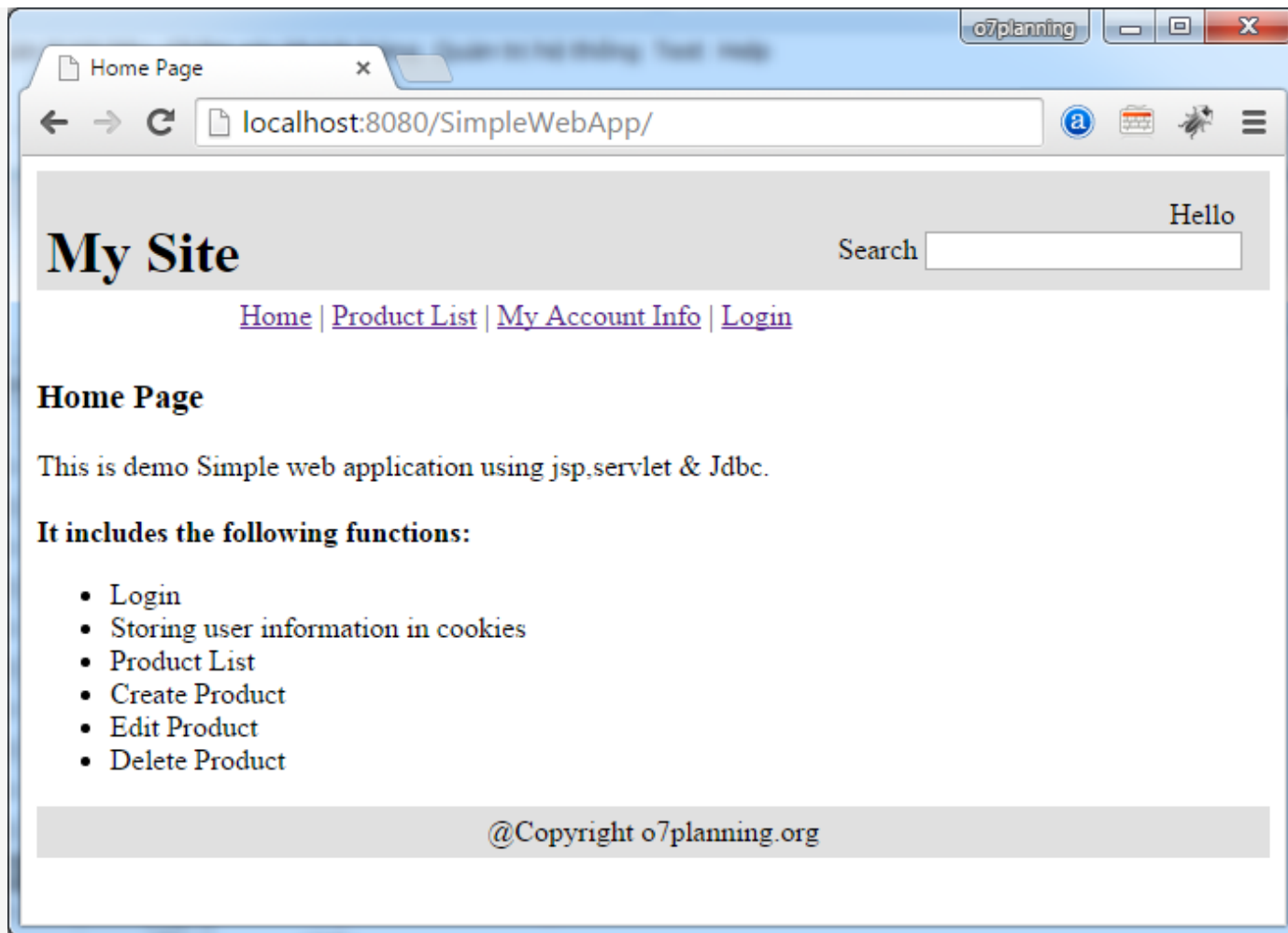
```
21     <li>Product List</li>
22     <li>Create Product</li>
23     <li>Edit Product</li>
24     <li>Delete Product</li>
25 </ul>
26
27 <jsp:include page="_footer.jsp"></jsp:include>
28
29 </body>
30 </html>
```

Rerun your application, and try two URLs:

-  <http://localhost:8081/SimpleWebApp/>
-  <http://localhost:8081/SimpleWebApp/home>



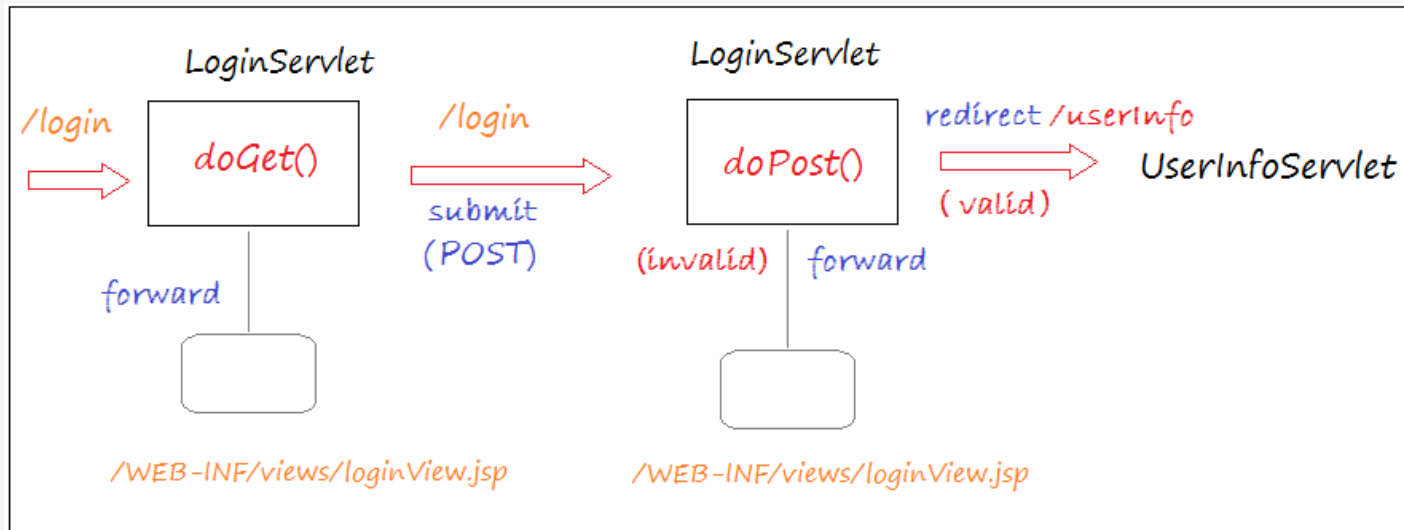




NOTE: For whatever reason, the link <http://localhost:8080/SimpleWebApp/> is still displayed content of *index.html*, you can delete or rename the *index.html* file, for example, you can change it into *\_index.html*

## 18- Login Page - LoginServlet

This is a model of Login function:



### LoginServlet.java

```
1 package org.o7planning.simplewebapp.servlet;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.sql.SQLException;
6
7 import javax.servlet.RequestDispatcher;
8 import javax.servlet.ServletException;
9 import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import javax.servlet.http.HttpSession;
14
15 import org.o7planning.simplewebapp.beans.UserAccount;
16 import org.o7planning.simplewebapp.utils.DBUtils;
17 import org.o7planning.simplewebapp.utils.MyUtils;
18
19 @WebServlet(urlPatterns = { "/login" })
```

```

20 public class LoginServlet extends HttpServlet {
21     private static final long serialVersionUID = 1L;
22
23     public LoginServlet() {
24         super();
25     }
26
27     // Show Login page.
28     @Override
29     protected void doGet(HttpServletRequest request, HttpServletResponse response)
30         throws ServletException, IOException {
31
32         // Forward to /WEB-INF/views/loginView.jsp
33         // (Users can not access directly into JSP pages placed in WEB-INF)
34         RequestDispatcher dispatcher //
35             = this.getServletContext().getRequestDispatcher("/WEB-INF/views/loginView.jsp");
36
37         dispatcher.forward(request, response);
38
39     }
40
41     // When the user enters userName & password, and click Submit.
42     // This method will be executed.
43     @Override
44     protected void doPost(HttpServletRequest request, HttpServletResponse response)
45         throws ServletException, IOException {
46         String userName = request.getParameter("userName");
47         String password = request.getParameter("password");
48         String rememberMeStr = request.getParameter("rememberMe");
49         boolean remember = "Y".equals(rememberMeStr);
50
51         UserAccount user = null;
52         boolean hasError = false;
53         String errorString = null;
54
55         if (userName == null || password == null || userName.length() == 0 || password.length() == 0) {
56             hasError = true;
57             errorString = "Required username and password!";
58         } else {
59             Connection conn = MyUtils.getStoredConnection(request);
60             try {
61                 // Find the user in the DB.
62                 user = DBUtils.findUser(conn, userName, password);
63
64                 if (user == null) {
65                     hasError = true;
66                     errorString = "User Name or password invalid";
67                 }
68             } catch (SQLException e) {

```



```

69         e.printStackTrace();
70         hasError = true;
71         errorString = e.getMessage();
72     }
73 }
74 // If error, forward to /WEB-INF/views/login.jsp
75 if (hasError) {
76     user = new UserAccount();
77     user.setUserName(userName);
78     user.setPassword(password);
79
80     // Store information in request attribute, before forward.
81     request.setAttribute("errorString", errorString);
82     request.setAttribute("user", user);
83
84     // Forward to /WEB-INF/views/login.jsp
85     RequestDispatcher dispatcher //
86         = this.getServletContext().getRequestDispatcher("/WEB-INF/views/loginView.jsp");
87
88     dispatcher.forward(request, response);
89 }
90 // If no error
91 // Store user information in Session
92 // And redirect to userInfo page.
93 else {
94     HttpSession session = request.getSession();
95     MyUtils.storeLoggedInUser(session, user);
96
97     // If user checked "Remember me".
98     if (remember) {
99         MyUtils.storeUserCookie(response, user);
100     }
101     // Else delete cookie.
102     else {
103         MyUtils.deleteUserCookie(response);
104     }
105
106     // Redirect to userInfo page.
107     response.sendRedirect(request.getContextPath() + "/userInfo");
108 }
109 }
110 }
111 }

```

### UserInfoservlet.java

```

1 package org.o7planning.simplewebapp.servlet;

```



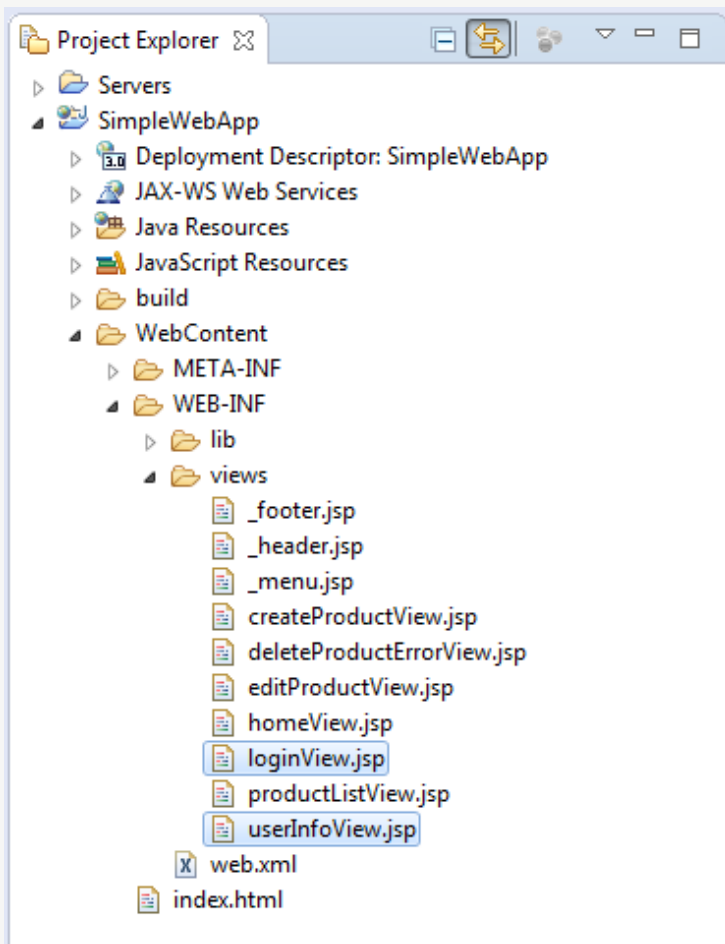
```

2
3 import java.io.IOException;
4
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import javax.servlet.http.HttpSession;
12
13 import org.o7planning.simplewebapp.beans.UserAccount;
14 import org.o7planning.simplewebapp.utils.MyUtils;
15
16 @WebServlet(urlPatterns = { "/userInfo" })
17 public class UserInfoServlet extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19
20     public UserInfoServlet() {
21         super();
22     }
23
24     @Override
25     protected void doGet(HttpServletRequest request, HttpServletResponse response)
26         throws ServletException, IOException {
27         HttpSession session = request.getSession();
28
29         // Check User has logged on
30         UserAccount loggedInUser = MyUtils.getLoggedInUser(session);
31
32         // Not logged in
33         if (loggedInUser == null) {
34             // Redirect to login page.
35             response.sendRedirect(request.getContextPath() + "/login");
36             return;
37         }
38         // Store info to the request attribute before forwarding.
39         request.setAttribute("user", loggedInUser);
40
41         // If the user has logged in, then forward to the page
42         // /WEB-INF/views/userInfoView.jsp
43         RequestDispatcher dispatcher //
44             = this.getServletContext().getRequestDispatcher("/WEB-INF/views/userInfoView.jsp");
45         dispatcher.forward(request, response);
46
47     }
48
49     @Override
50     protected void doPost(HttpServletRequest request, HttpServletResponse response)

```



```
51     throws ServletException, IOException {  
52     doGet(request, response);  
53     }  
54  
55 }
```



**/WEB-INF/views/loginView.jsp**

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"  
2   pageEncoding="UTF-8"%>  
3 <!DOCTYPE html>
```

```

4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>Login</title>
8 </head>
9 <body>
10 <jsp:include page="_header.jsp"></jsp:include>
11 <jsp:include page="_menu.jsp"></jsp:include>
12
13 <h3>Login Page</h3>
14 <p style="color: red;">${errorString}</p>
15
16
17 <form method="POST" action="${pageContext.request.contextPath}/login">
18 <table border="0">
19 <tr>
20 <td>User Name</td>
21 <td><input type="text" name="userName" value= "${user.userName}" /> </td>
22 </tr>
23 <tr>
24 <td>Password</td>
25 <td><input type="text" name="password" value= "${user.password}" /> </td>
26 </tr>
27 <tr>
28 <td>Remember me</td>
29 <td><input type="checkbox" name="rememberMe" value= "Y" /> </td>
30 </tr>
31 <tr>
32 <td colspan="2">
33 <input type="submit" value= "Submit" />
34 <a href="${pageContext.request.contextPath}/">Cancel</a>
35 </td>
36 </tr>
37 </table>
38 </form>
39
40 <p style="color:blue;">User Name: tom, password: tom001 or jerry/jerry001</p>
41
42 <jsp:include page="_footer.jsp"></jsp:include>
43 </body>
44 </html>
45

```

/WEB-INF/views/userInfoView.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>

```



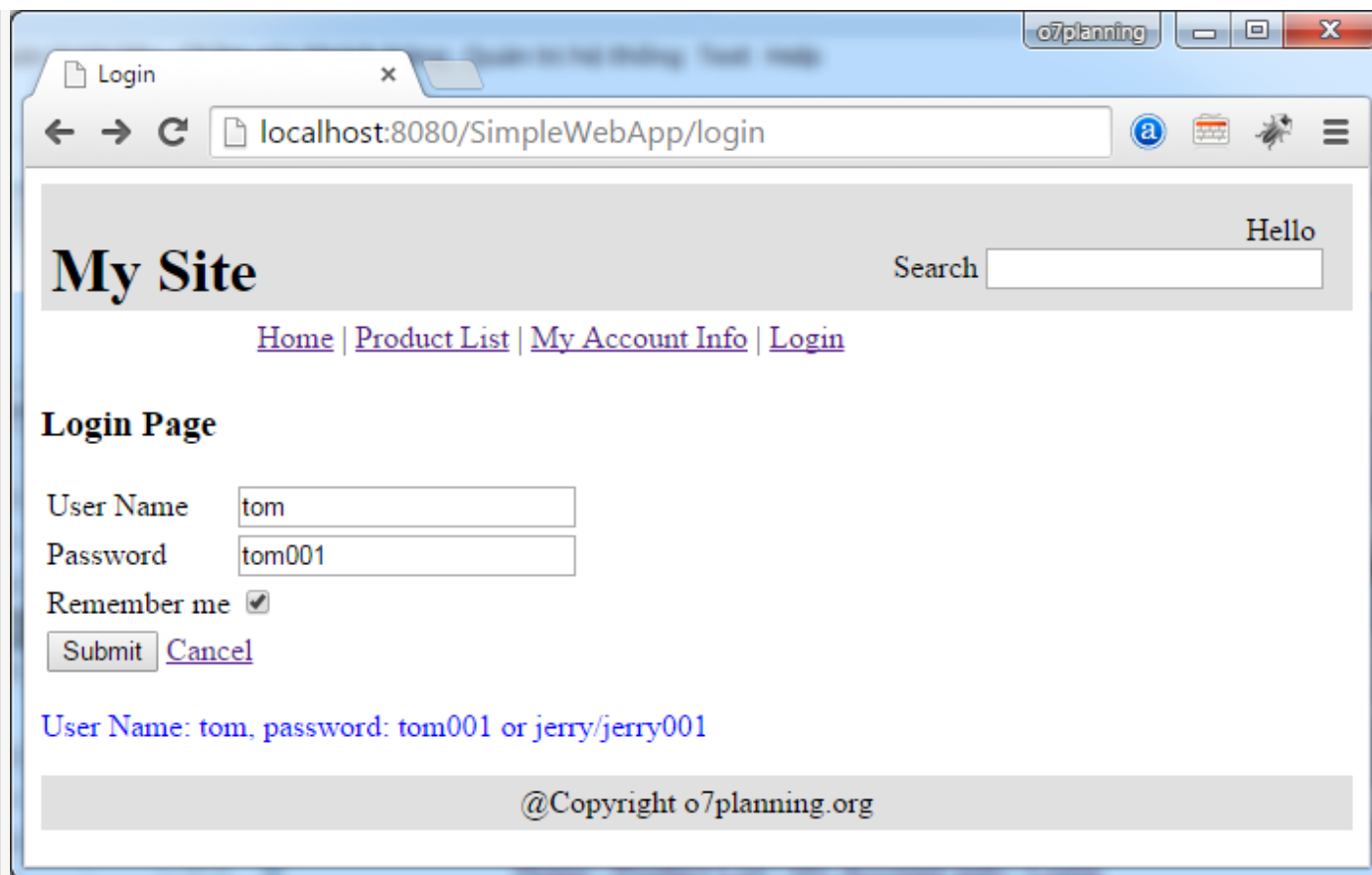
```
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7   <title>User Info</title>
8 </head>
9 <body>
10
11   <jsp:include page="_header.jsp"></jsp:include>
12   <jsp:include page="_menu.jsp"></jsp:include>
13
14   <h3>Hello: ${user.userName}</h3>
15
16   User Name: <b>${user.userName}</b>
17   <br />
18   Gender: ${user.gender } <br />
19
20   <jsp:include page="_footer.jsp"></jsp:include>
21
22 </body>
23 </html>
```

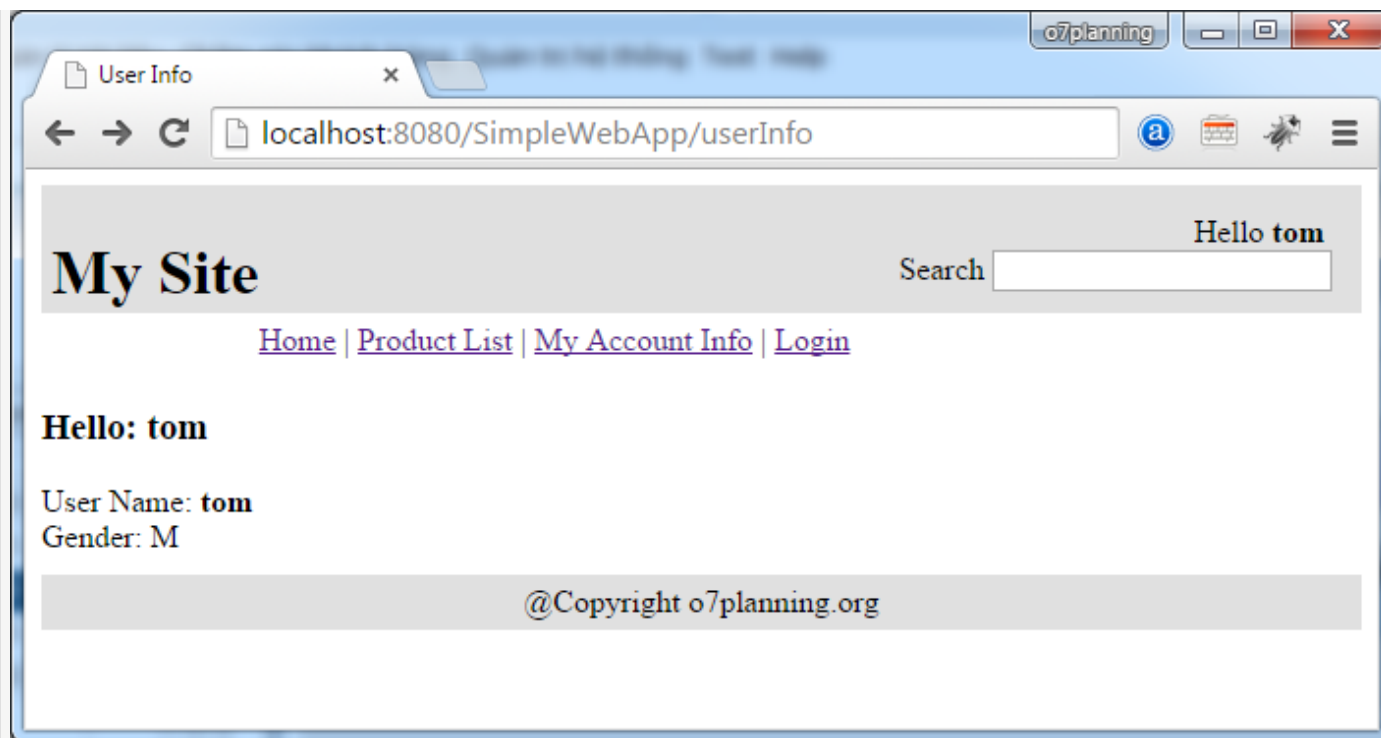
Running your application:

- <http://localhost:8080/SimpleWebApp/login>









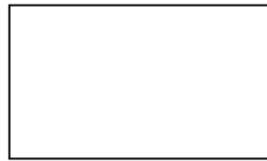
## 19- Product List Page

Model:



## ProductListServlet

/productList



forward



/WEB-INF/views/productListView.jsp

### ProductListServlet.java

```
1 package org.o7planning.simplewebapp.servlet;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.sql.SQLException;
6 import java.util.List;
7
8 import javax.servlet.RequestDispatcher;
9 import javax.servlet.ServletException;
10 import javax.servlet.annotation.WebServlet;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14
15 import org.o7planning.simplewebapp.beans.Product;
16 import org.o7planning.simplewebapp.utils.DBUtils;
17 import org.o7planning.simplewebapp.utils.MyUtils;
18
19 @WebServlet(urlPatterns = { "/productList" })
20 public class ProductListServlet extends HttpServlet {
21     private static final long serialVersionUID = 1L;
```



```

22
23 public ProductListServlet() {
24     super();
25 }
26
27 @Override
28 protected void doGet(HttpServletRequest request, HttpServletResponse response)
29     throws ServletException, IOException {
30     Connection conn = MyUtils.getStoredConnection(request);
31
32     String errorString = null;
33     List<Product> list = null;
34     try {
35         list = DBUtils.queryProduct(conn);
36     } catch (SQLException e) {
37         e.printStackTrace();
38         errorString = e.getMessage();
39     }
40     // Store info in request attribute, before forward to views
41     request.setAttribute("errorString", errorString);
42     request.setAttribute("productList", list);
43
44     // Forward to /WEB-INF/views/productListView.jsp
45     RequestDispatcher dispatcher = request.getServletContext()
46         .getRequestDispatcher("/WEB-INF/views/productListView.jsp");
47     dispatcher.forward(request, response);
48 }
49
50 @Override
51 protected void doPost(HttpServletRequest request, HttpServletResponse response)
52     throws ServletException, IOException {
53     doGet(request, response);
54 }
55
56 }

```

### /WEB-INF/views/productListView.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Product List</title>
9 </head>

```



```

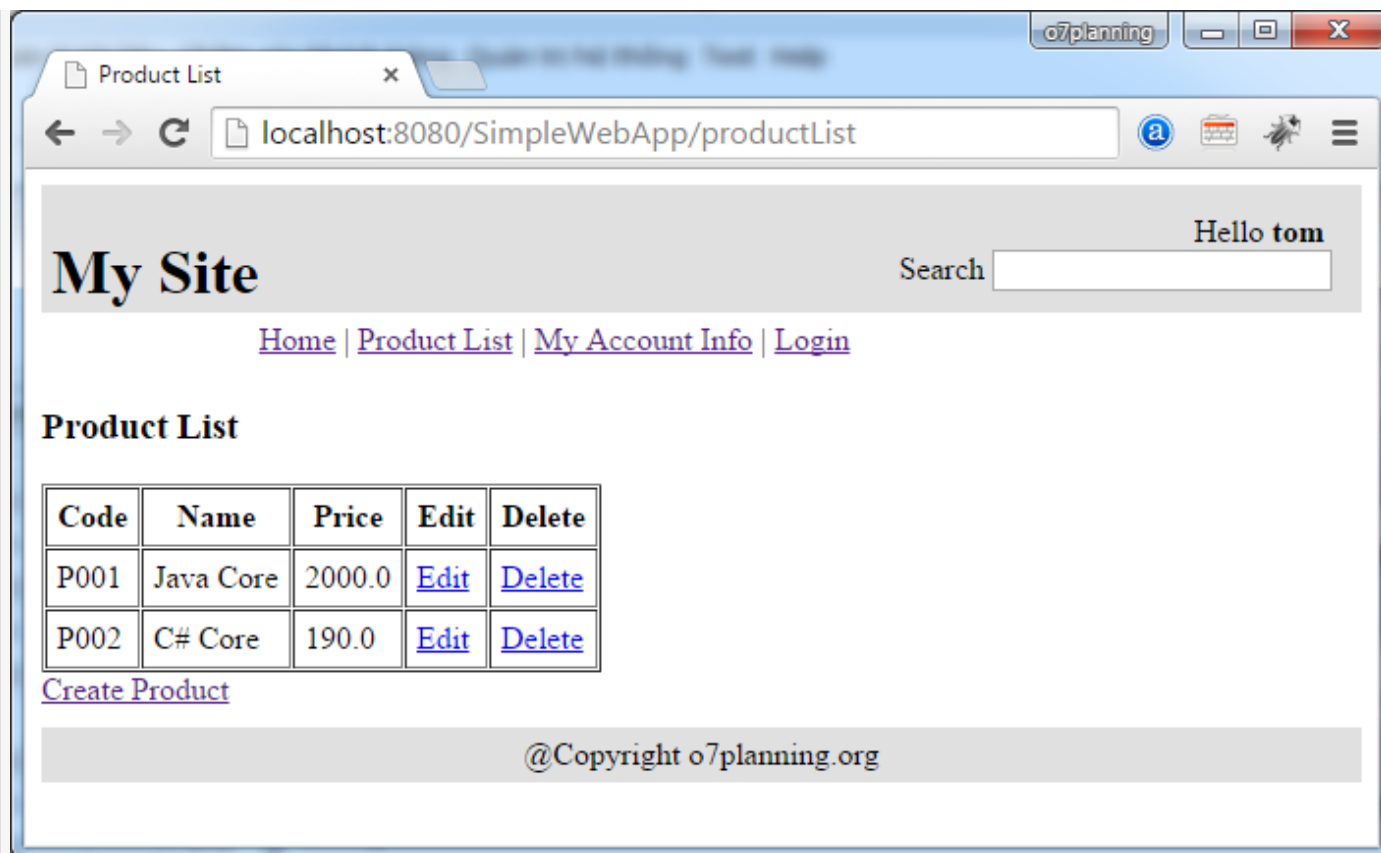
10 <body>
11
12 <jsp:include page="_header.jsp"></jsp:include>
13 <jsp:include page="_menu.jsp"></jsp:include>
14
15 <h3>Product List</h3>
16
17 <p style="color: red;">${errorString}</p>
18
19 <table border="1" cellpadding="5" cellspacing="1" >
20   <tr>
21     <th>Code</th>
22     <th>Name</th>
23     <th>Price</th>
24     <th>Edit</th>
25     <th>Delete</th>
26   </tr>
27   <c:forEach items="${productList}" var="product" >
28     <tr>
29       <td>${product.code}</td>
30       <td>${product.name}</td>
31       <td>${product.price}</td>
32       <td>
33         <a href="editProduct?code=${product.code}">Edit</a>
34       </td>
35       <td>
36         <a href="deleteProduct?code=${product.code}">Delete</a>
37       </td>
38     </tr>
39   </c:forEach>
40 </table>
41
42 <a href="createProduct" >Create Product</a>
43
44 <jsp:include page="_footer.jsp"></jsp:include>
45
46 </body>
47 </html>

```

Rerun Application:

- <http://localhost:8080/SimpleWebApp/productList>

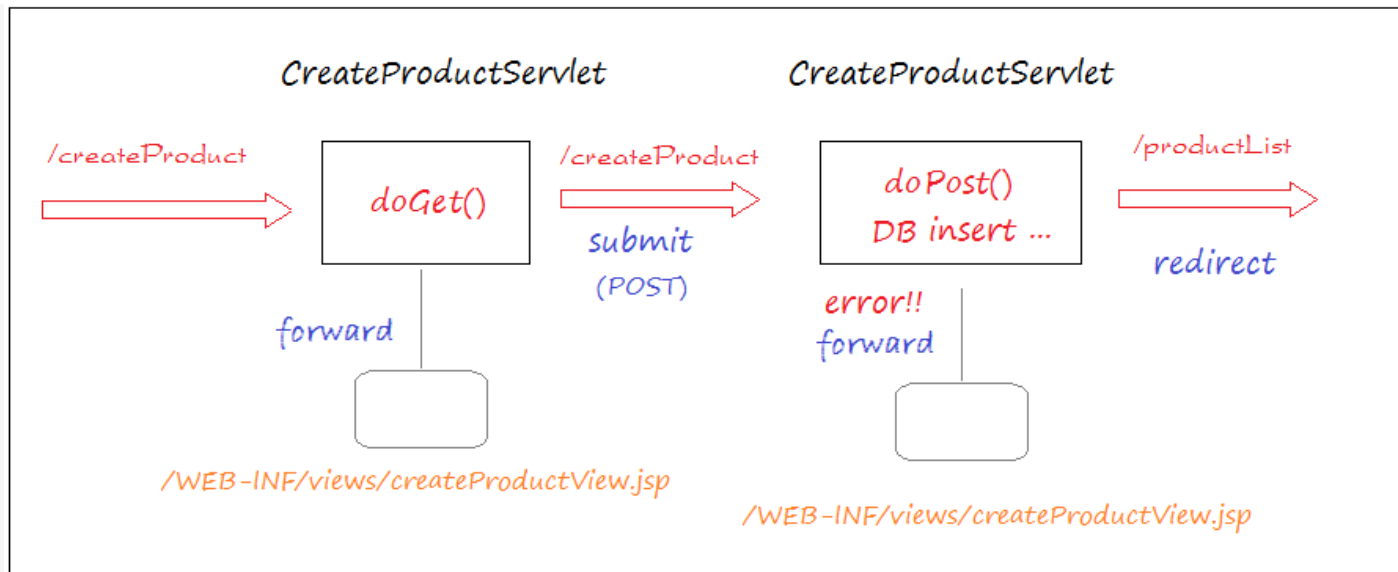




## 20- Add Product Page

Model:





### CreateProductServlet.java

```

1 package org.o7planning.simplewebapp.servlet;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.sql.SQLException;
6
7 import javax.servlet.RequestDispatcher;
8 import javax.servlet.ServletException;
9 import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import org.o7planning.simplewebapp.beans.Product;
15 import org.o7planning.simplewebapp.utils.DBUtils;
16 import org.o7planning.simplewebapp.utils.MyUtils;
17
18 @WebServlet(urlPatterns = { "/createProduct" })
19 public class CreateProductServlet extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21
22     public CreateProductServlet() {
23         super();
24     }

```

```

25
26 // Show product creation page.
27 @Override
28 protected void doGet(HttpServletRequest request, HttpServletResponse response)
29     throws ServletException, IOException {
30
31     RequestDispatcher dispatcher = request.getServletContext()
32         .getRequestDispatcher("/WEB-INF/views/createProductView.jsp");
33     dispatcher.forward(request, response);
34 }
35
36 // When the user enters the product information, and click Submit.
37 // This method will be called.
38 @Override
39 protected void doPost(HttpServletRequest request, HttpServletResponse response)
40     throws ServletException, IOException {
41     Connection conn = MyUtils.getStoredConnection(request);
42
43     String code = (String) request.getParameter("code");
44     String name = (String) request.getParameter("name");
45     String priceStr = (String) request.getParameter("price");
46     float price = 0;
47     try {
48         price = Float.parseFloat(priceStr);
49     } catch (Exception e) {
50     }
51     Product product = new Product(code, name, price);
52
53     String errorString = null;
54
55     // Product ID is the string literal [a-zA-Z_0-9]
56     // with at least 1 character
57     String regex = "\\w+";
58
59     if (code == null || !code.matches(regex)) {
60         errorString = "Product Code invalid!";
61     }
62
63     if (errorString == null) {
64         try {
65             DBUtils.insertProduct(conn, product);
66         } catch (SQLException e) {
67             e.printStackTrace();
68             errorString = e.getMessage();
69         }
70     }
71
72     // Store information to request attribute, before forward to views.
73     request.setAttribute("errorString", errorString);

```





```

74 request.setAttribute("product", product);
75
76 // If error, forward to Edit page.
77 if (errorString != null) {
78     RequestDispatcher dispatcher = request.getServletContext()
79         .getRequestDispatcher("/WEB-INF/views/createProductView.jsp");
80     dispatcher.forward(request, response);
81 }
82 // If everything nice.
83 // Redirect to the product listing page.
84 else {
85     response.sendRedirect(request.getContextPath() + "/productList");
86 }
87 }
88
89 }

```

### /WEB-INF/views/createProductView.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4
5 <!DOCTYPE html>
6 <html>
7   <head>
8     <meta charset="UTF-8">
9     <title>Create Product</title>
10  </head>
11  <body>
12
13    <jsp:include page="_header.jsp"></jsp:include>
14    <jsp:include page="_menu.jsp"></jsp:include>
15
16    <h3>Create Product</h3>
17
18    <p style="color: red;">${errorString}</p>
19
20    <form method="POST" action="${pageContext.request.contextPath}/createProduct">
21      <table border="0">
22        <tr>
23          <td>Code</td>
24          <td><input type="text" name="code" value="${product.code}" /></td>
25        </tr>
26        <tr>
27          <td>Name</td>
28          <td><input type="text" name="name" value="${product.name}" /></td>

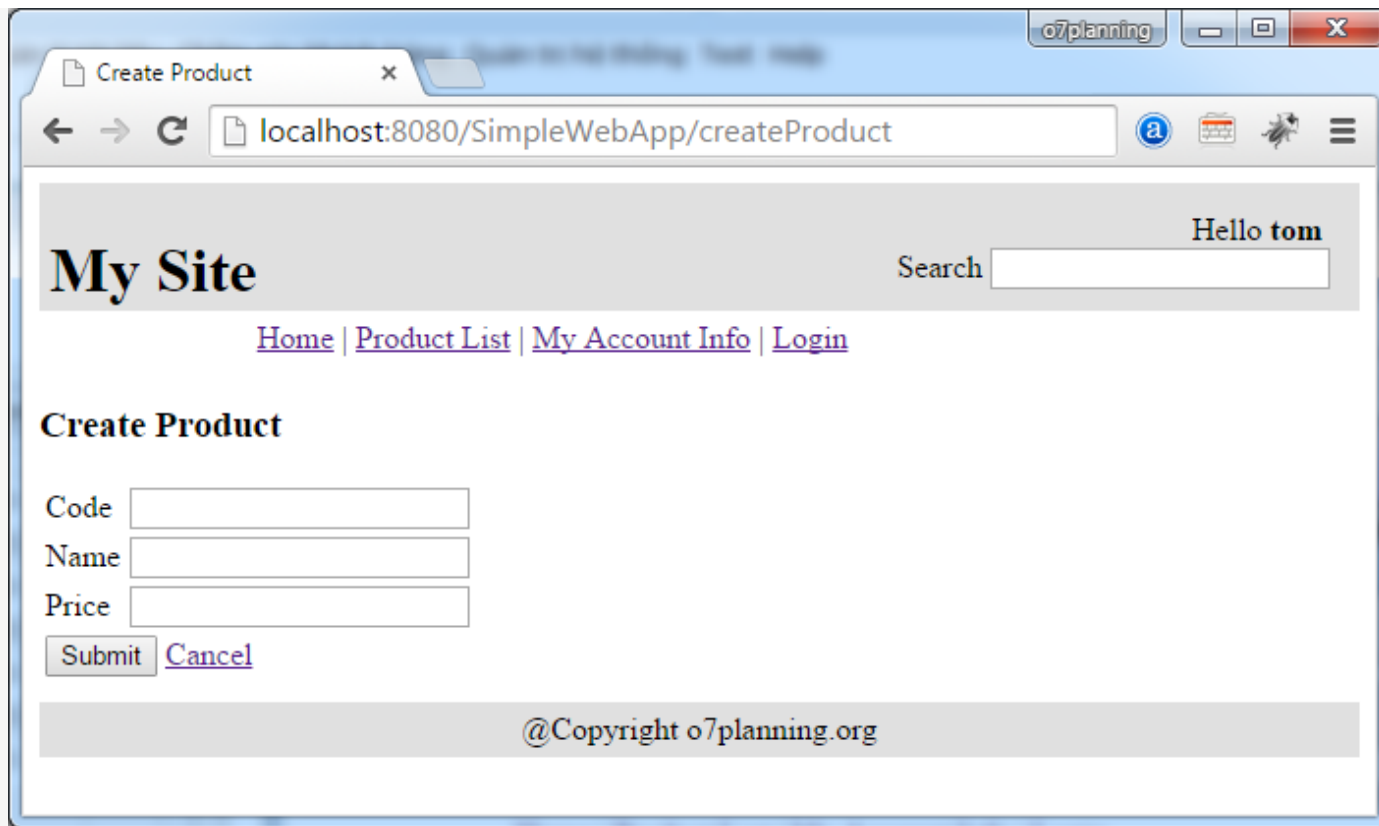
```



```
29     </tr>
30     <tr>
31         <td>Price</td>
32         <td><input type="text" name="price" value="${product.price}" /></td>
33     </tr>
34     <tr>
35         <td colspan="2">
36             <input type="submit" value="Submit" />
37             <a href="productList">Cancel</a>
38         </td>
39     </tr>
40 </table>
41 </form>
42
43 <jsp:include page="_footer.jsp"></jsp:include>
44
45 </body>
46 </html>
```

- <http://localhost:8080/SimpleWebApp/createProduct>

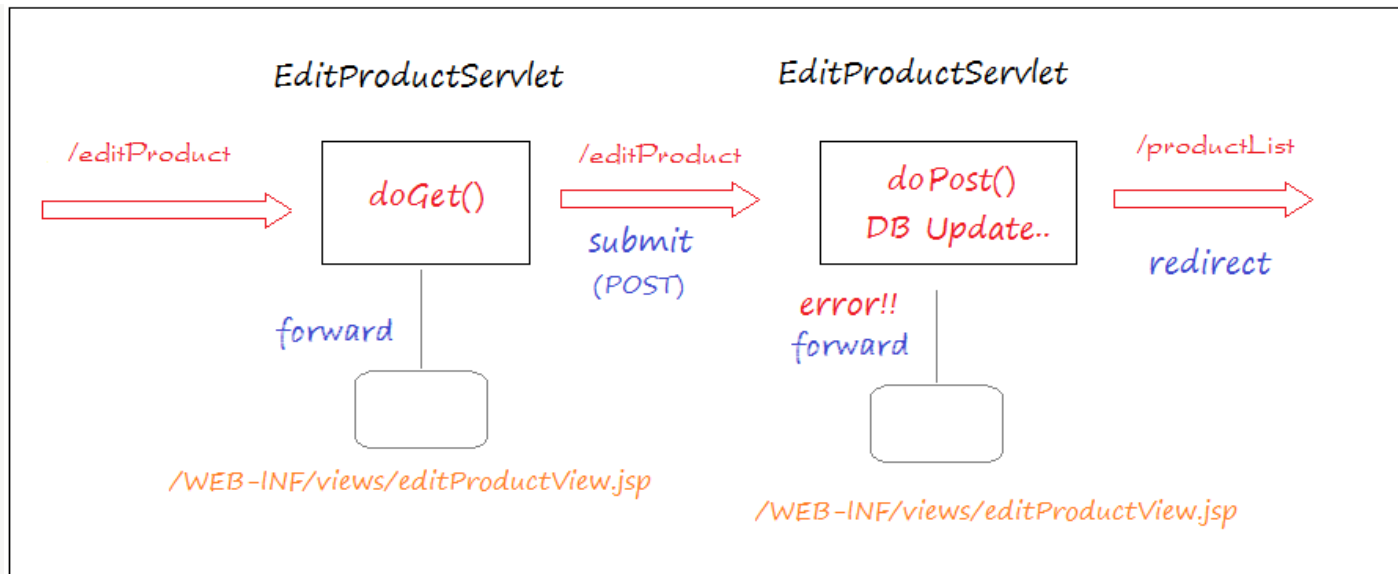




## 21- Edit Product Page

Model:





### EditProductServlet.java

```

1 package org.o7planning.simplewebapp.servlet;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.sql.SQLException;
6
7 import javax.servlet.RequestDispatcher;
8 import javax.servlet.ServletException;
9 import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import org.o7planning.simplewebapp.beans.Product;
15 import org.o7planning.simplewebapp.utils.DBUtils;
16 import org.o7planning.simplewebapp.utils.MyUtils;
17
18 @WebServlet(urlPatterns = { "/editProduct" })
19 public class EditProductServlet extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21
22     public EditProductServlet() {
23         super();
24     }
  
```

```

25
26 // Show product edit page.
27 @Override
28 protected void doGet(HttpServletRequest request, HttpServletResponse response)
29     throws ServletException, IOException {
30     Connection conn = MyUtils.getStoredConnection(request);
31
32     String code = (String) request.getParameter("code");
33
34     Product product = null;
35
36     String errorString = null;
37
38     try {
39         product = DBUtils.findProduct(conn, code);
40     } catch (SQLException e) {
41         e.printStackTrace();
42         errorString = e.getMessage();
43     }
44
45     // If no error.
46     // The product does not exist to edit.
47     // Redirect to productList page.
48     if (errorString != null && product == null) {
49         response.sendRedirect(request.getServletPath() + "/productList");
50         return;
51     }
52
53     // Store errorString in request attribute, before forward to views.
54     request.setAttribute("errorString", errorString);
55     request.setAttribute("product", product);
56
57     RequestDispatcher dispatcher = request.getServletContext()
58         .getRequestDispatcher("/WEB-INF/views/editProductView.jsp");
59     dispatcher.forward(request, response);
60
61 }
62
63 // After the user modifies the product information, and click Submit.
64 // This method will be executed.
65 @Override
66 protected void doPost(HttpServletRequest request, HttpServletResponse response)
67     throws ServletException, IOException {
68     Connection conn = MyUtils.getStoredConnection(request);
69
70     String code = (String) request.getParameter("code");
71     String name = (String) request.getParameter("name");
72     String priceStr = (String) request.getParameter("price");
73     float price = 0;

```



```

74     try {
75         price = Float.parseFloat(priceStr);
76     } catch (Exception e) {
77     }
78     Product product = new Product(code, name, price);
79
80     String errorString = null;
81
82     try {
83         DBUtils.updateProduct(conn, product);
84     } catch (SQLException e) {
85         e.printStackTrace();
86         errorString = e.getMessage();
87     }
88     // Store information to request attribute, before forward to views.
89     request.setAttribute("errorString", errorString);
90     request.setAttribute("product", product);
91
92     // If error, forward to Edit page.
93     if (errorString != null) {
94         RequestDispatcher dispatcher = request.getServletContext()
95             .getRequestDispatcher("/WEB-INF/views/editProductView.jsp");
96         dispatcher.forward(request, response);
97     }
98     // If everything nice.
99     // Redirect to the product listing page.
100    else {
101        response.sendRedirect(request.getContextPath() + "/productList");
102    }
103 }
104
105 }

```

### /WEB-INF/views/editProductView.jsp

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4  <!DOCTYPE html>
5  <html>
6  <head>
7  <meta charset="UTF-8">
8  <title>Edit Product</title>
9  </head>
10 <body>
11
12 <jsp:include page="_header.jsp"></jsp:include>

```



```

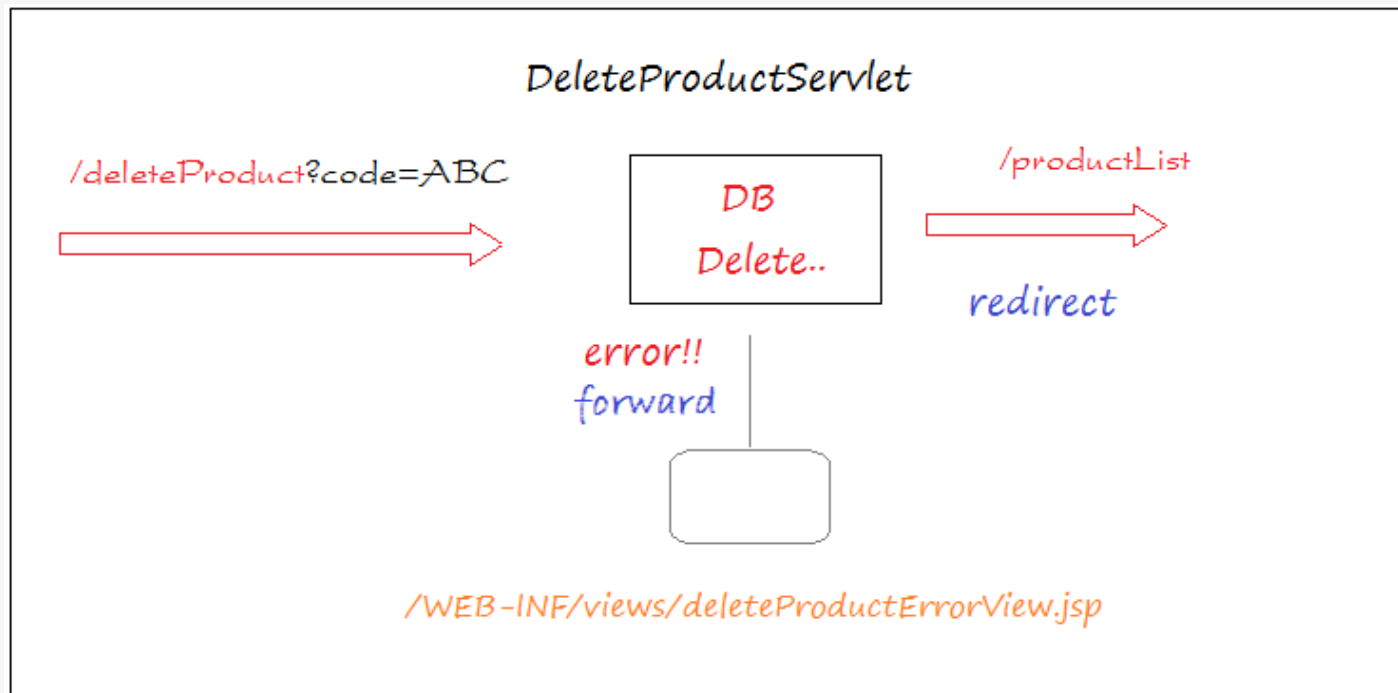
13 <jsp:include page="_menu.jsp"></jsp:include>
14
15 <h3>Edit Product</h3>
16
17 <p style="color: red;">${errorString}</p>
18
19 <c:if test="${not empty product}">
20   <form method="POST" action="${pageContext.request.contextPath}/editProduct">
21     <input type="hidden" name="code" value="${product.code}" />
22     <table border="0">
23       <tr>
24         <td>Code</td>
25         <td style="color:red;">${product.code}</td>
26       </tr>
27       <tr>
28         <td>Name</td>
29         <td><input type="text" name="name" value="${product.name}" /></td>
30       </tr>
31       <tr>
32         <td>Price</td>
33         <td><input type="text" name="price" value="${product.price}" /></td>
34       </tr>
35       <tr>
36         <td colspan = "2">
37           <input type="submit" value="Submit" />
38           <a href="${pageContext.request.contextPath}/productList">Cancel</a>
39         </td>
40       </tr>
41     </table>
42   </form>
43 </c:if>
44
45 <jsp:include page="_footer.jsp"></jsp:include>
46
47 </body>
48 </html>

```

## 22- Delete Product



Model:



#### DeleteProductServlet.java

```
1 package org.o7planning.simplewebapp.servlet;  
2  
3 import java.io.IOException;  
4 import java.sql.Connection;
```



```

5  import java.sql.SQLException;
6
7  import javax.servlet.RequestDispatcher;
8  import javax.servlet.ServletException;
9  import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import org.o7planning.simplewebapp.utils.DBUtils;
15 import org.o7planning.simplewebapp.utils.MyUtils;
16
17 @WebServlet(urlPatterns = { "/deleteProduct" })
18 public class DeleteProductServlet extends HttpServlet {
19     private static final long serialVersionUID = 1L;
20
21     public DeleteProductServlet() {
22         super();
23     }
24
25     @Override
26     protected void doGet(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28         Connection conn = MyUtils.getStoredConnection(request);
29
30         String code = (String) request.getParameter("code");
31
32         String errorString = null;
33
34         try {
35             DBUtils.deleteProduct(conn, code);
36         } catch (SQLException e) {
37             e.printStackTrace();
38             errorString = e.getMessage();
39         }
40
41         // If has an error, redirect to the error page.
42         if (errorString != null) {
43             // Store the information in the request attribute, before forward to views.
44             request.setAttribute("errorString", errorString);
45             //
46             RequestDispatcher dispatcher = request.getServletContext()
47                 .getRequestDispatcher("/WEB-INF/views/deleteProductErrorView.jsp");
48             dispatcher.forward(request, response);
49         }
50         // If everything nice.
51         // Redirect to the product listing page.
52         else {
53             response.sendRedirect(request.getContextPath() + "/productList");

```



```

54     }
55
56 }
57
58 @Override
59 protected void doPost(HttpServletRequest request, HttpServletResponse response)
60     throws ServletException, IOException {
61     doGet(request, response);
62 }
63
64 }

```

### /WEB-INF/views/deleteProductErrorView.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Delete Product</title>
9 </head>
10
11 <body>
12
13   <jsp:include page="_header.jsp"></jsp:include>
14   <jsp:include page="_menu.jsp"></jsp:include>
15
16   <h3>Delete Product</h3>
17
18   <p style="color: red;">${errorString}</p>
19   <a href="productList">Product List</a>
20
21   <jsp:include page="_footer.jsp"></jsp:include>
22
23 </body>
24 </html>

```

View more categories:



o7planning.org

