Project: Adding a new product in the database

My GitHub link:

https://github.com/moulaalihujare/Simplilearn.git

Development Environment

- Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
- Apache Tomcat Server v9.0

    JRE: OpenJDK Runtime Environment 11.0.2

    This lab has the following subsections :

    1.1.1   Creating a dynamic web project

    1.1.2   Creating an HTML page

    1.1.3   Creating a servlet Product.java

    1.1.4   Creating a servlet ProductAdd.java

    1.1.5   Creating a servlet Register.java

    1.1.6   Configuring web.xml

    1.1.7   Checking for servlet-api.jar

    1.1.8   Building the project

    1.1.9   Publishing and starting the project

    1.1.10 Running the project

    1.1.11 Pushing the code to git repositories.

Step 1.1.1: Creating a dynamic web project

- Open Eclipse
- Go the **File** menu. Choose **New->Dynamic Web Project**
- Enter the project name as **AddingNewProject**. Click on **Next**
- Enter nothing in the next screen and click on **Next**
- Check the checkbox **Generate web.xml deployment descriptor** and click on **Finish**
- This will create the project files in the Project Explorer

Step 1.1.2: Creating an HTML page

- In the Project Explorer, expand the project **AddingNewProject**

- Expand **WebContent**. Right click on **WebContent**. Choose **New->JSP File**

- Enter the filename as index.jsp and click on **Finish**

- Enter the following code:

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!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=UTF-8">
<title>Product Registration</title>
</head>
<body>
<form action="Register" method="post">
<table>
<tr><td>Product ID: </td><td><input type="text" name="id"></td></tr>
<tr><td>Product Name: </td><td><input type="text" name="name"></td></tr>
<tr><td>Product Cost: </td><td><input type="text" name="cost"></td></tr>
<tr><td>Quantity: </td><td><input type="text" name="quantity"></td></tr>
<tr><td></td><td><input type="submit" value="Add Product"></td></tr>
</table>
</form>
</body>
    </html>
```

Step 1.1.3: Creating a servlet Product.java

- Right click **src** and choose **New->Servlet**

- In **Class Name,** enter **Product** and click on **Finish**

- Enter the following code:

```java
package com;

public class Product {

    private String id, name, cost, quantity;


    public Product() {
        super();
    }

    public Product(String id, String name, String cost, String quantity) {
        super();
        this.id = id;
        this.name = name;
        this.cost = cost;
        this.quantity = quantity;
    }

    public String getid() {
        return id;
    }

    public void setid(String id) {
        this.id = id;
    }
```

```java
    public String getname() {
        return name;
    }

    public void setname(String name) {
        this.name = name;
    }

    public String getcost() {
        return cost;
    }

    public void setcost(String cost) {
        this.cost = cost;
    }

    public String getquantity() {
        return quantity;
    }

    public void setquantity(String quantity) {
        this.quantity = quantity;
    }


}
```

Step 1.1.4: Creating a servlet ProductAdd.java

- Right click **src** and choose **New->Servlet**

- In **Class Name,** enter **ProductAdd** and click on **Finish**

- Enter the following code:

package com;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class ProductAdd {

    private String dbUrl =

    "jdbc:mysql://localhost:3306/moulaali_db";private String

    dbUname = "root";

```java
private String dbPassword = "Aashiyana@11";
private String dbDriver = "com.mysql.cj.jdbc.Driver";


public void loadDriver(String dbDriver)
{
	try {
			Class.forName(dbDriver);
	} catch (ClassNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
	}
}


public Connection getConnection()
{
	Connection con = null;
	try {
			con = DriverManager.getConnection(dbUrl, dbUname, dbPassword);
	} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
	}
	return con;
}



public String insert(Product member)
{
	loadDriver(dbDriver);
	Connection con =  getConnection();
	String result = "Data Insert Successfully";
	String sql = "insert into addprodut values(?,?,?,?)";
```

```java
            PreparedStatement ps;

            try {

            ps = con.prepareStatement(sql);

            ps.setString(1, member.getid());

            ps.setString(2, member.getname());

            ps.setString(3, member.getcost());

            ps.setString(4, member.getquantity());

            ps.executeUpdate();

            } catch (SQLException e) {

                    // TODO Auto-generated catch block

                    e.printStackTrace();

                    result = "Data insertion failed";

            }

            return result;

    }


}
```

Step 1.1.5: Creating a servlet Register.java

- Right click **src** and choose **New->Servlet**
- In **Class Name,** enter **Register** and click on **Finish**
- Enter the following code:

```java
package com;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


@WebServlet("/Register")
```

```java
public class Register extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Register() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        // TODO Auto-generated method stub

        response.getWriter().append("Served at: ").append(request.getContextPath());
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

        String id = request.getParameter("id");
        String name = request.getParameter("name");
        String cost = request.getParameter("cost");
        String quantity = request.getParameter("quantity");

        Product member = new Product(id, name, cost, quantity);
        ProductAdd rDao = new ProductAdd();
```

```
            String result = rDao.insert(member);

            response.getWriter().print(result);

    }

}
```

Step 1.1.6: Configuring web.xml

- In the Project Explorer, expand **AddingNewProduct->WebContent->WEB-INF**

- Double click **web.xml** to open it in the editor

- Enter the following script:

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>AddProduct</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
  </web-app>
```

Step 1.1.6: Checking for servlet-api.jar

- Before building the project, we need to add servlet-api.jar to the project

- Servlet-api.jar file is already present in your practice lab. (Refer FSD: Lab Guide - Phase 2)

- To add it to the project, follow the below mentioned steps:

  ◦ In the Project Explorer, right click on **AddNewProduct** and choose **Properties**

  ◦ Select **Java Build Path** from the options on the left

  ◦ Click on **Libraries** tab on the right

  ◦ Under **ClassPath,** expand the node that says **Apache Tomcat**

  ◦ If there is an existing entry for **servlet-api.jar,** then click on **Cancel** and exit the window

  ◦ If it is not there, then click on **Classpath** entry and click on **Add External JARs** button on the right

  ◦ From the file list, select **servlet-api.jar** file and click **Ok**

  ◦ Click on **Apply and Close**

Step 1.1.7: Building the project

- From the **Project** menu at the top, click on **Build**
- If any compile errors are shown, fix them as required

Step 1.1.8: Publishing and starting the project

- If you do not see the **Servers** tab near the bottom of the IDE, go to the Window menu and click **Show View->Servers**
- Right click on the **Server** entry and choose **Add and Remove**
- Click the **Add** button to move **AddNewProject** from the **Available** list to the **Configured** List
- Click **Finish**
- Right click on the **Server** entry and click on **Publish**
- Right click on the **Server** entry and click on **Start**
- This will start the server

Step 1.1.9: Running the project

- To run the project, open a web browser and type: **http://localhost:8080/ServletGetPost**

Step 1.1.10: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m "Changes have been committed."

- Push the files to the folder you initially created using the following command:

git push -u origin master

Output:

Enter the product details: