Practice Project: Retrieving the Product Details Using the Product ID.

My GitHub link :

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

Project objective:

Creating a servlet-based application that shows a form to enter a product ID. The product ID is then validated, and product details are retrieved from the database and displayed to the user. to create a product table in MySQL and prepopulate it with data. Use JDBC to do all database processing.

Background of the problem statement:

As a part of developing an e-commerce web application, the admin backend requires a module that can retrieve product information based on the product ID.

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 ten subsections, namely:

1.1.1 Creating a dynamic web project

1.1.2 Creating a productDetail.java

1.1.3 Creating a index.html

1.1.4 Configuring web.xml

1.1.5 Building the project

1.1.6 Publishing and starting the project

1.1.7 Running the project

1.1.8 Pushing the code to GitHub 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 **Phase 2- Retrieving the Product Details Using the Product ID.**
- 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

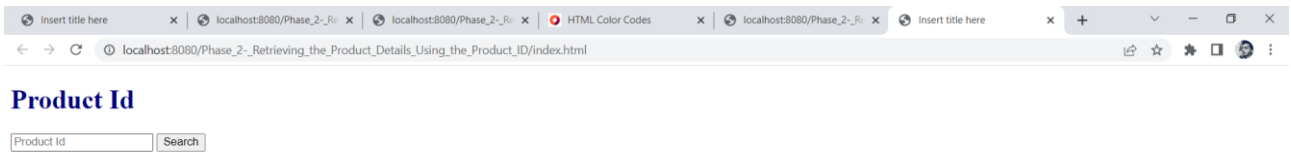
Step 1.1.2: Creating a servlet Login.java

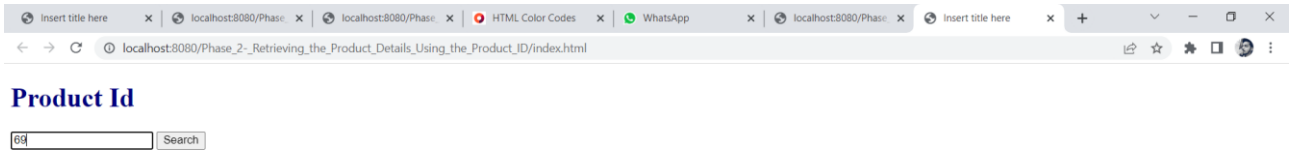
- In the Project Explorer, expand **source/main/Java**
- Right click **src** and choose **New->Servlet**
- In **Class Name,** enter **ProductDeatil** and click on **Finish**
- Enter the following code:

```
package com;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;


import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;
```

```
@WebServlet("/product")
    public class ProductDetail extends HttpServlet{
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws
            ServletException, IOException {
            String url="jdbc:mysql://localhost:3306/moulaali_db";
            String uname="root";
            String pass="Aashiyana@11";
            response.setContentType("text/html");

            String pId = request.getParameter("productid");

            PrintWriter out = response.getWriter();

            String query="select * from product where p_id=?";

            out.print("<h1>Displaying the Product Details</h1>");

            out.print("<table border='1'><tr><th>Product Id</th><th>Product
Name</th><th>Product Price</th></tr>");
    try {

             Class.forName("com.mysql.cj.jdbc.Driver");
             Connection dbCon = DriverManager.getConnection(url, uname, pass);
             PreparedStatement st= dbCon.prepareStatement(query);
             st.setString(1, pId);

             ResultSet rs =st.executeQuery();
```

```
 while(rs.next()) {

            out.print("<tr><td>");
            out.println(rs.getInt(1));
            out.print("</td>");
            out.print("<td>");
            out.print(rs.getString(2));
            out.print("</td>");
            out.print("<td>");
            out.print(rs.getInt(3));
            out.print("</td>");
            out.print("</tr>");
}
}
catch(Exception e){

          System.out.println("Some Issue : "+ e.getMessage());
}
out.print("</table>");
}
}
```





Step 1.1.3: Configuring web.xml

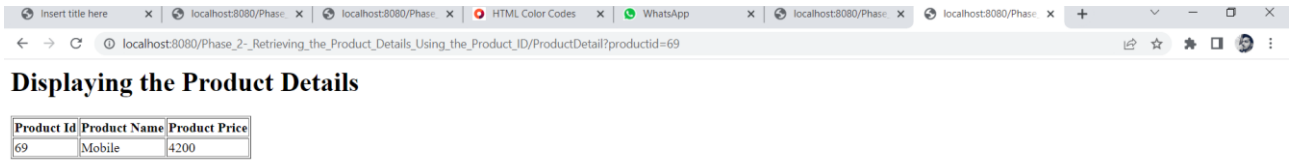
- In the Project Explorer, expand ValidationOfUserLogin->**WebContent->WEB-INF**
- Double click **web.xml** to open it in the editor
- Enter the following script:

```
<?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"
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">
  <display-name>Phase 2- Retrieving the Product Details Using the Product
ID</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
    <description></description>
    <display-name>ProductDetail</display-name>
    <servlet-name>ProductDetail</servlet-name>
    <servlet-class>com.ProductDetail</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProductDetail</servlet-name>
    <url-pattern>/ProductDetail</url-pattern>
  </servlet-mapping>
</web-app>
```

Step 1.1.4: Creating index.html

- Right click on Project and choose **New->HTML**
- In **Class Name,** enter **index.html** and click on **Finish**
- **Enter the following code:**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1 style="color:Navy;">Product Id</h1>
<form action="ProductDetail" method="get">
 <input type="text" name="productid" placeholder="Product Id">
 <button>Search</button>
</form>
</body>
      </html>
```



Step 1.1.6: Checking for servlet-api.jar

- Before building the project, we need to add servlet-api.jar to the project
- To add it to the project, follow the below mentioned steps:
  - In the Project Explorer, right click on **ProductDetails** 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 **ServletGetPost** 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:8086/ValidationOfUserLogin**

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