**Retrieving the Product Details Using the Product ID.**

**DESCRIPTION**

**Project objective:**  
Create 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. You need 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
* MySQL Connector for Java 8.0.16

Creating a database in MySQL and a table in it

* Login to the MySQL command line console
* Type **CREATE DATABASE ecommerce** and press **Enter**
* Type **USE ecommerce** and press **Enter**
* Type **CREATE TABLE eproduct (ID bigint primary key auto\_increment, name varchar(100), price decimal(10,2), date\_added timestamp default now())** and press **Enter**
* We will now add some rows into the table
* Type **INSERT INTO eproduct(name, ‘HP Laptop ABC’, 12000)** and press **Enter**
* Type **INSERT INTO eproduct(name, ‘Acer Laptop ABC’, 14000)** and press **Enter**
* Type **INSERT INTO eproduct(name, ‘Lenovo Laptop ABC’, 12000)** and press **Enter**
* Type **SELECT \* from eproduct** and press **Enter** to confirm that the rows have been added
* Type **EXIT** to exit the MySQL command console

Adding the jar files for MySQL connection for Java

* Take **mysql-connector-java.jar** and add it to the project’s **WebContent/WEB-INF/lib** folder

Creating an HTML page index.html

* In the Project Explorer, expand the project **Product\_Details**
* Expand **WebContent**. Right click on **WebContent**. Choose **New->HTML File**
* Enter the filename as **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="background-color:*powderblue*;">user credentials</h1>

<form action=*"productid"* method=*"post"*>

<input type=*"text"* name=*"productid"* placeholder=*"enter the productid"*>

<button>Submit</button>

</form>

</body>

</html>

Creating a DBConnection class to initiate a JDBC connection in code

* In the Project Explorer, expand **Product\_Details->Java Resources**
* Right click on **src** and choose **New->Servlet**
* In **Package,** enter **com.servlets** and in **Name** enter **SearchProduct** and click on **Finish**
* Enter the following code:

**package** com.servlets;

**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.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** SearchProduct **extends** HttpServlet{

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

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

String uname="root";

String pass="password";

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>quantity</th><th>bill</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>");

}

}

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 **Product\_Details** 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 on **Ok**
  + Click on **Apply and Close**

Building the project

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

Publishing and starting the project

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

Running the project

* To run the project, open a web browser and type: [**http://localhost:8080/**](http://localhost:8080/ServletConcept)**Product\_Details/SearchProducts/**

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**