Retrieving the Product Details Using the Product ID

DESCRIPTION

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.

You must use the following:

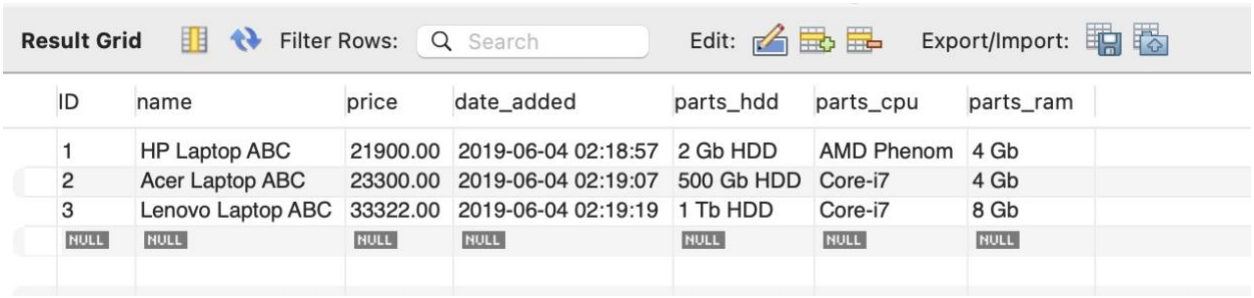
- Eclipse as the IDE
- Apache Tomcat as the web server
- MySQL Connector for JDBC functionality

Following requirements should be met:

- Create an HTML page to take in a product ID
- Set up JDBC to work with the application
- Create a servlet that will take the product ID and use JDBC to query the database for the product
- If the product is found, the servlet will display the product details, otherwise it will show an error message
- Document the step-by-step process involved in completing this task

1. Creating a database in MySQL and a table in it

- MySQL is already installed in your practice lab, (Refer FSD: Lab Guide - Phase 2)
- 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



ID	name	price	date_added	parts_hdd	parts_cpu	parts_ram
1	HP Laptop ABC	21900.00	2019-06-04 02:18:57	2 Gb HDD	AMD Phenom	4 Gb
2	Acer Laptop ABC	23300.00	2019-06-04 02:19:07	500 Gb HDD	Core-i7	4 Gb
3	Lenovo Laptop ABC	33322.00	2019-06-04 02:19:19	1 Tb HDD	Core-i7	8 Gb
NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Creating a dynamic web project

- Open Eclipse Environment
- Go the **File** menu. Choose **New->Dynamic Web Project**
- Enter the project name as **RetrievingProductDetailsUsingProductID**. 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

3. Adding the jar files for MySQL connection for Java

- **mysql-connector-java.jar** is already present in your lab. To learn about its directory path details you can refer the **lab guide for phase 2**
- Take **mysql-connector-java.jar** file from the folder mentioned in the lab guide for phase 2 and add it to the project's **WebContent/WEB-INF/lib** folder

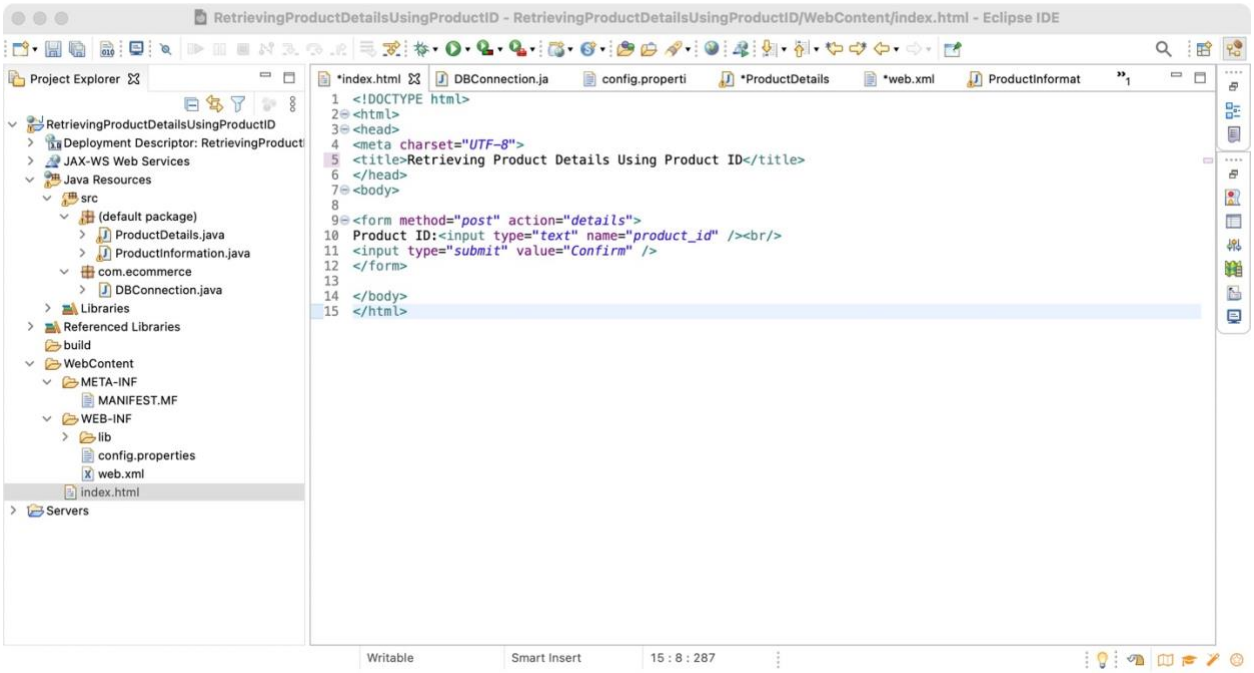
4. Creating an HTML page index.html

- In the Project Explorer, expand the project **RetrievingProductDetailsUsingProductID**
- 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="UTF-8">
<title>Retrieving Product Details Using Product ID</title>
</head>
<body>

<form method="post" action="details">
Product ID:<input type="text" name="product_id" /><br/>
<input type="submit" value="Confirm" />
</form>

</body>
```

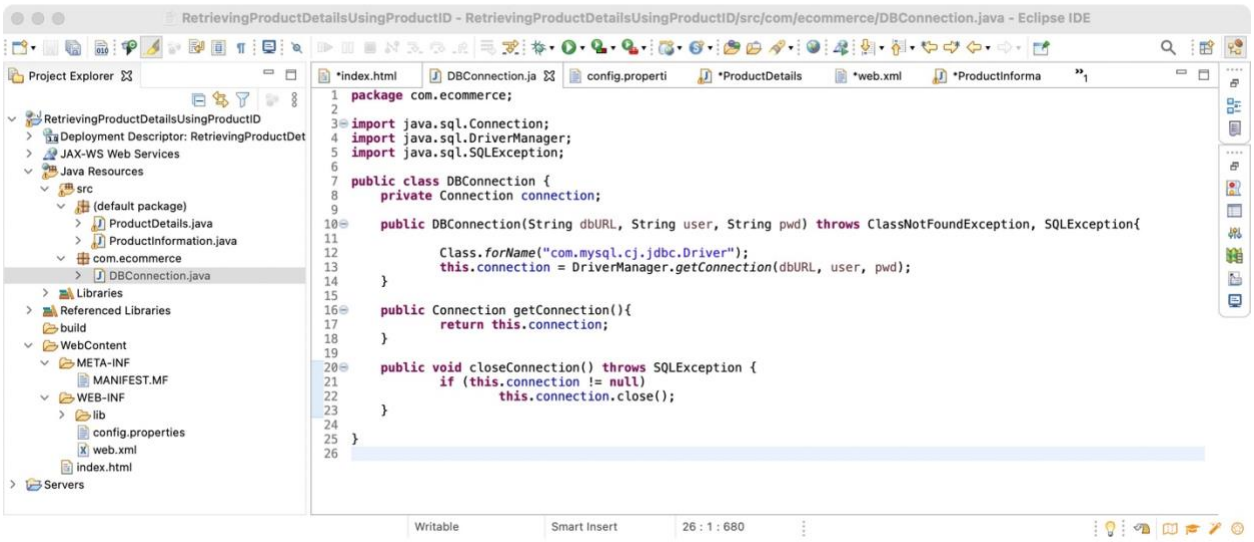
- </html>

- Click on the **Save** icon



5. Creating a DBConnection class to initiate a JDBC connection in code

- In the Project Explorer, expand **RetrievingProductDetailsUsingProductID ->Java Resources**
- Right click on **src** and choose **New->Class**
- In **Package,** enter **com.ecommerce** and in **Name** enter **DBConnection** and click on **Finish**
- Enter the following code:



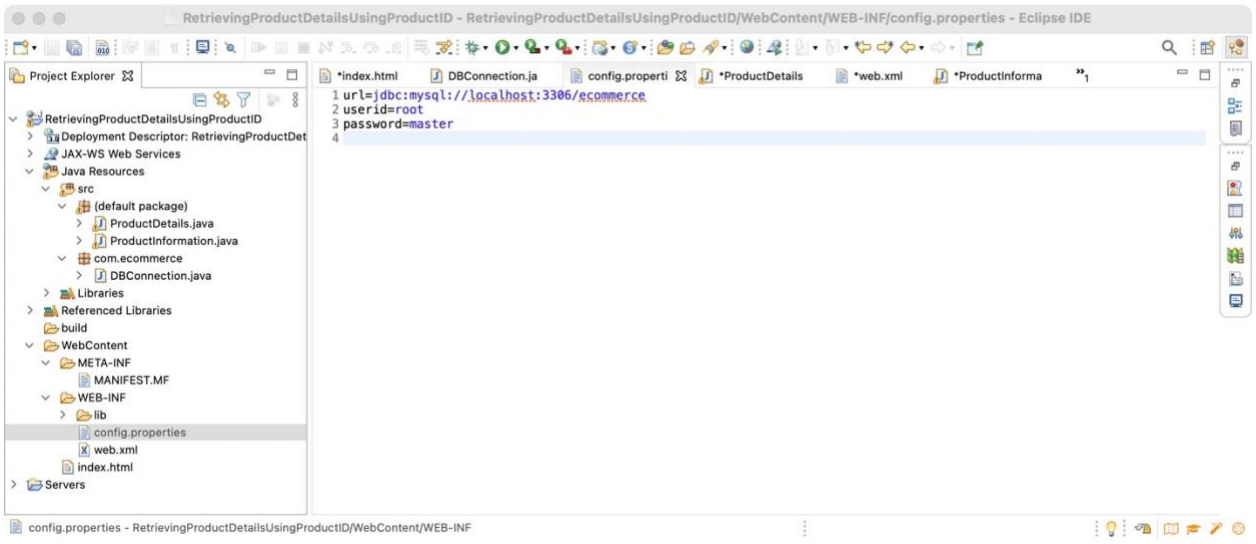

```java
package com.ecommerce;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
	private Connection connection;

   public DBConnection(String dbURL, String user, String pwd) throws
ClassNotFoundException, SQLException{

		Class.forName("com.mysql.cj.jdbc.Driver");
		this.connection = DriverManager.getConnection(dbURL, user, pwd);
   }

   public Connection getConnection(){
		return this.connection;
   }
```

```
    public void closeConnection() throws SQLException {
        if (this.connection != null)
            this.connection.close();
    }

}
```

6. Creating a config.properties file to store JDBC credentials

- In the Project Explorer, expand the project **RetrievingProductDetailsUsingProductID**
- Expand **WebContent**. Right click on **WebContent**. Choose **New->File**
- Enter the filename as **config.properties** and click on **Finish**
- Enter the following data:

```
url=jdbc:mysql://localhost:3306/ecommerce
userid=root
password=master
```



7. Creating a ProductDetails servlet

- In the Project Explorer, expand **RetrievingProductDetailsUsingProductID ->Java Resources**
- Right click on **src** and choose **New->Servlet**
- In **Class Name,** enter **ProductDetails** and click on **Finish**
- Enter the following code:

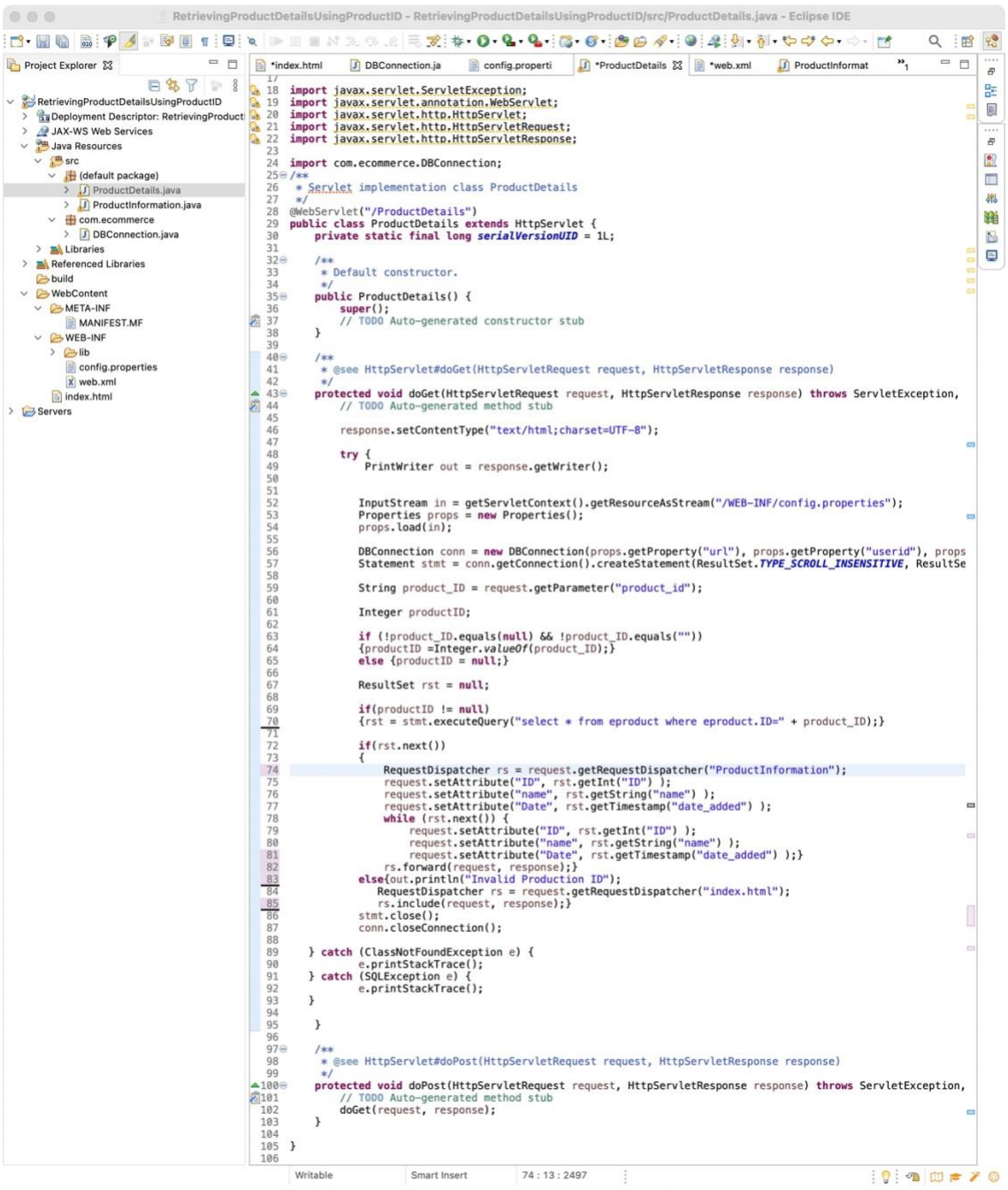
```
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.math.BigDecimal;
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

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.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.ecommerce.DBConnection;
/**
 * Servlet implementation class ProductDetails
 */
@WebServlet("/ProductDetails")
public class ProductDetails extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * Default constructor.
     */
    public ProductDetails() {
        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.setContentType("text/html;charset=UTF-8");

	try {
	 PrintWriter out = response.getWriter();



	InputStream in = getServletContext().getResourceAsStream("/WEB-
INF/config.properties");
	Properties props = new Properties();
	props.load(in);

	DBConnection conn = new DBConnection(props.getProperty("url"),
props.getProperty("userid"), props.getProperty("password"));
	Statement stmt =
conn.getConnection().createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

	String product_ID = request.getParameter("product_id");

	Integer productID;

	if (!product_ID.equals(null) && !product_ID.equals(""))
	{productID =Integer.valueOf(product_ID);}
	else {productID = null;}
```

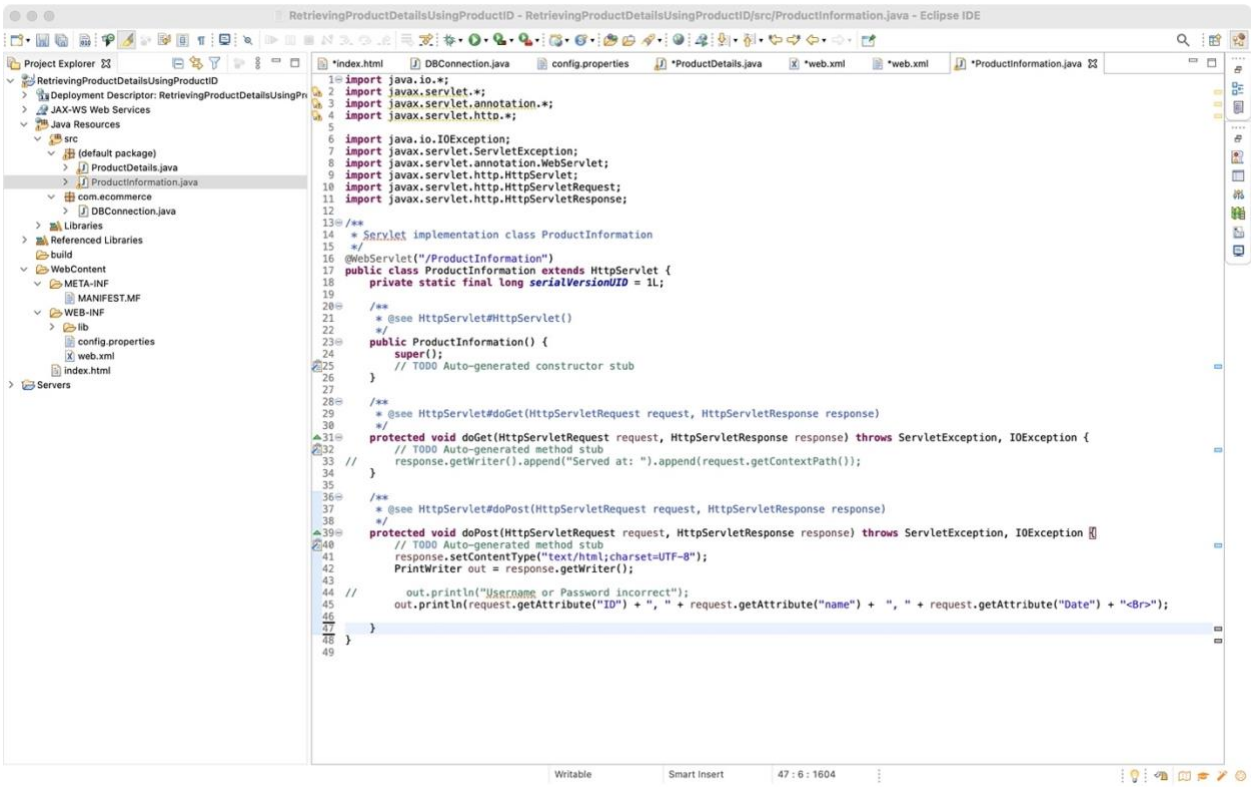
```
ResultSet rst = null;

if(productID != null)
{rst = stmt.executeQuery("select * from eproduct where eproduct.ID=" + product_ID);}



if(rst.next())
{
    RequestDispatcher rs = request.getRequestDispatcher("ProductInformation");

    request.setAttribute("ID", rst.getInt("ID") );
    request.setAttribute("name", rst.getString("name") );
    request.setAttribute("Date", rst.getTimestamp("date_added") );

    while (rst.next()) {
            request.setAttribute("ID", rst.getInt("ID") );
        request.setAttribute("name", rst.getString("name") );
        request.setAttribute("Date", rst.getTimestamp("date_added") );
}

    rs.forward(request, response);
}
else
{
   out.println("Invalid Production ID");
   RequestDispatcher rs = request.getRequestDispatcher("index.html");
   rs.include(request, response);

 }
```

```
            stmt.close();
            conn.closeConnection();

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }

            }

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

}
```



8. Creating a ProductInformation servlet

- In the Project Explorer, expand **RetrievingProductDetailsUsingProductID ->Java Resources**
- Right click on **src** and choose **New->Servlet**
- In **Class Name,** enter **ProductInformation** and click on **Finish**
- Enter the following code:



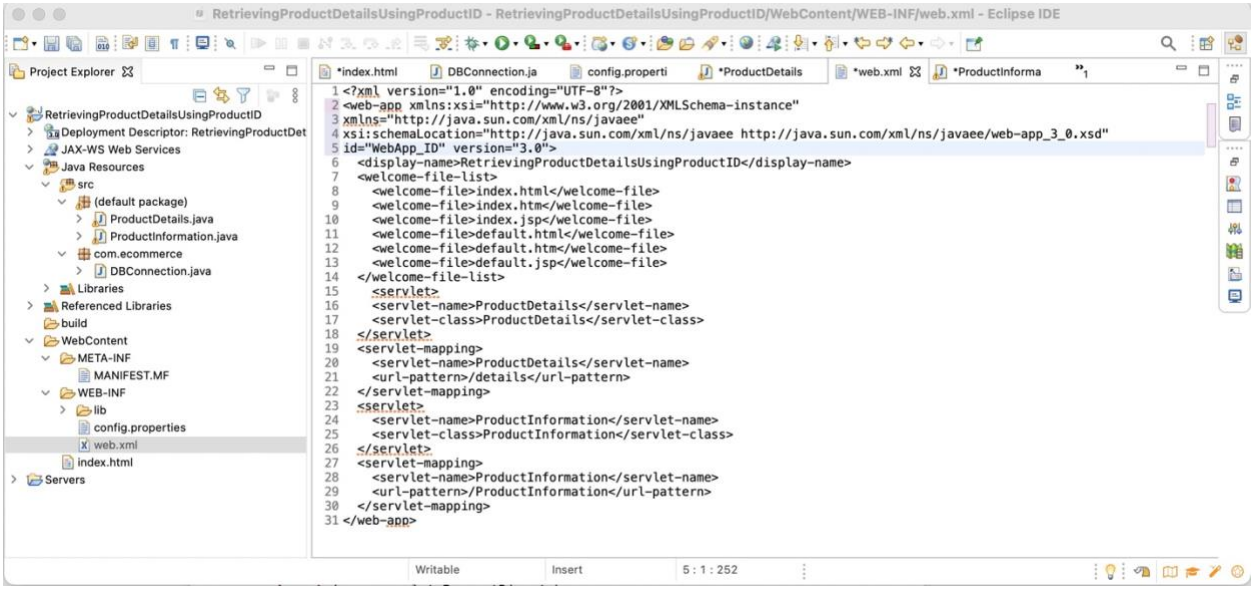

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

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;

/**
 * Servlet implementation class ProductInformation
 */
@WebServlet("/ProductInformation")
public class ProductInformation extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ProductInformation() {
        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 {
        // TODO Auto-generated method stub
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

//        out.println("Username or Password incorrect");
        out.println(request.getAttribute("ID") + ", " + request.getAttribute("name") +  ", " + request.getAttribute("Date") + "<Br>");

    }
}
```

```
@WebServlet("/ProductInformation")
public class ProductInformation extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ProductInformation() {
        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 {
		// TODO Auto-generated method stub
		response.setContentType("text/html;charset=UTF-8");
```

```
        PrintWriter out = response.getWriter();

//        out.println("Username or Password incorrect");
        out.println(request.getAttribute("ID") + ", " + request.getAttribute("name") +  ", " +
request.getAttribute("Date") + "<Br>");

            }

}
```

9. Configuring web.xml

- In the Project Explorer, expand **RetrievingProductDetailsUsingProductID ->WebContent->WEB-INF**
- Double click on **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_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>RetrievingProductDetailsUsingProductID</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
    <servlet>
    <servlet-name>ProductDetails</servlet-name>
    <servlet-class>ProductDetails</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProductDetails</servlet-name>
    <url-pattern>/details</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>ProductInformation</servlet-name>
    <servlet-class>ProductInformation</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProductInformation</servlet-name>
    <url-pattern>/ProductInformation</url-pattern>
  </servlet-mapping>
```

</web-app>

10. 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 **RetrievingProductDetailsUsingProductID** 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**

11. Building the project

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

12. Publishing and starting the project

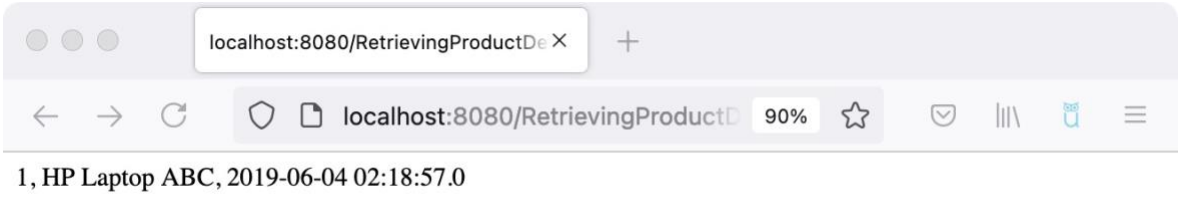
- If you do not see the **Servers** tab near the bottom of the IDE, go to **Window** menu and click on **Show View->Servers**
- Right click the **Server** entry and choose **Add and Remove**

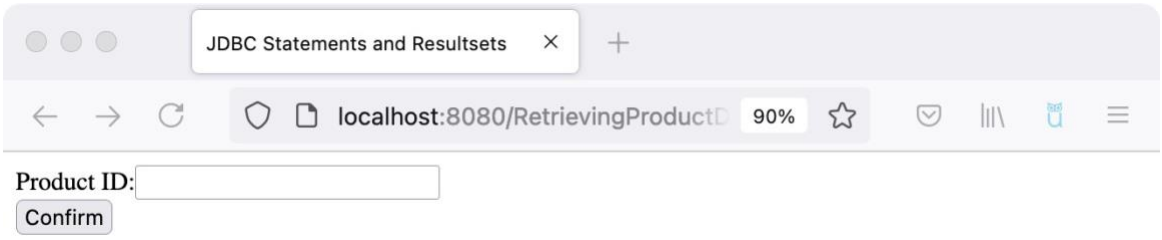
- Click the **Add** button to move **RetrievingProductDetailsUsingProductID** from the **Available** list to the **Configured** list
- Click on **Finish**
- Right click the **Server** entry and click on **Publish**
- Right click the **Server** entry and click on **Start**
- This will start the server

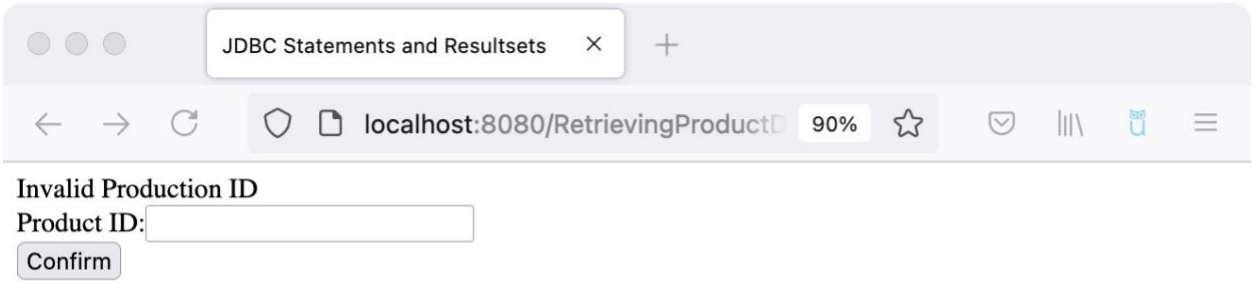
13. Running the project

- To run the project, open a web browser and type:

  http://localhost:8080/RetrievingProductDetailsUsingProductID







14. 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