**Product Details Portal**

DESCRIPTION

Create a servlet-based web application that shows a form to enter product details. Capture the details in a servlet and then display the data that was entered.

**Background of the problem statement:**  
As a part of developing an ecommerce web application, you have to prototype a form for adding products into the system entered by the users. There is no database involved here, so  
the product is just captured and displayed without storing it anywhere.

**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
* Go to <http://www.java2s.com/Code/Jar/j/Downloadjta11jar.htm>
* Click on **jta-1\_1.jar.zip** link to download it
* Extract **jta-1\_1.jar** from it and add it to your project’s **WebContent/WEB-INF/lib** folder
* Go to <http://www.java2s.com/Code/Jar/j/Downloadjavaxxmlbindjar.htm>
* Click on **javax.xml/javax.xml.bind.jar.zip** link to download it
* Extract **javax.xml.bind.jar** from it and add it to your project’s **WebContent/WEB-INF/lib** folder
* Go to <https://jar-download.com/artifacts/com.sun.xml.bind>
* Click on the button **Download jaxb-osgi.jar** to download it
* Extract **jaxb-osgi-2.4.0-b180830.0438.jar** from it and add it to your project’s **WebContent/WEB-INF/lib** folder

Creating a table eproduct in the database and filling it with sample data

* 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**
* Enter the following script and execute it:

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `eproduct`

DROP TABLE IF EXISTS `eproduct`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `eproduct` (

`ID` bigint(20) NOT NULL AUTO\_INCREMENT,

`name` varchar(100) DEFAULT NULL,

`price` decimal(10,2) DEFAULT NULL,

`date\_added` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`parts\_hdd` varchar(10) DEFAULT NULL,

`parts\_cpu` varchar(10) DEFAULT NULL,

`parts\_ram` varchar(10) DEFAULT NULL,

PRIMARY KEY (`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `eproduct`

--

LOCK TABLES `eproduct` WRITE;

/\*!40000 ALTER TABLE `eproduct` DISABLE KEYS \*/;

INSERT INTO `eproduct` VALUES (1,'HP Laptop ABC',21900.00,'2019-06-04 07:18:57','2 Gb HDD','AMD Phenom','4 Gb'),(2,'Acer Laptop ABC',23300.00,'2019-06-04 07:19:07','500 Gb HDD','Core-i7','4 Gb'),(3,'Lenovo Laptop ABC',33322.00,'2019-06-04 07:19:19','1 Tb HDD','Core-i7','8 Gb');

/\*!40000 ALTER TABLE `eproduct` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

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=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<h1>Welcome to My Products</h1>

<a href=*"read-product"*>Read Product</a>

<br />

<a href=*"add-product"*>Add Product</a>

</body>

</html>

Creating a class EProduct

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

package com.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="eproduct")

public class EProduct {

@Id

@GeneratedValue

@Column(name="eproduct\_id")

private int id;

@Column(name="eproduct\_name")

private String name;

@Column(name="eproduct\_price")

private Double price;

//ALt + S + R

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

Creating a ReadProductservlet

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

package com.servlets;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

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 org.hibernate.Session;

import org.hibernate.SessionFactory;

import com.entity.EProduct;

import com.util.Util;

/\*\*

\* Servlet implementation class ReadProductServlet

\*/

@WebServlet("/read-product")

public class ReadProductServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public ReadProductServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

PrintWriter out = response.getWriter();

SessionFactory sf = Util.buildSessionFactory();

Session session = sf.openSession();

List<EProduct> products = session.createQuery(" from EProduct").list();

//Show data on HTML

out.print("<h1> Product List :- </h1>");

out.print("<style> table,td,th { border:2px solid red; padding: 10px; }</style>" );

out.print("<table>");

out.print("<tr>");

out.print("<th> Product Id </th>");

out.print("<th> Product Name </th>");

out.print("<th> Product Price </th>");

out.print("</tr>");

for(EProduct p : products) {

out.print("<tr>");

out.print("<td>"+p.getId()+"</td>");

out.print("<td>"+p.getName()+"</td>");

out.print("<td>"+p.getPrice()+"</td>");

out.print("</tr>");

}

out.print("</table>");

session.close();

}

/\*\*

\* @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);

}

}

Creating a hibernate table util configuration file

* In the Project Explorer, expand **Product\_Details->Java Resources**
* Right click on **src** and choose **New->Other**
* In **Package,** enter com.util and in **Name** enter util and click on **Finish**

Enter the following code:

package com.simplilearn.util;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import com.simplilearn.entity.Courses;

import com.simplilearn.entity.EProduct;

import com.simplilearn.entity.PhoneNumber;

import com.simplilearn.entity.Student;

public class HibernateUtil {

public static SessionFactory buildSessionFactory() {

// Step 1

Configuration cfg = new Configuration();

cfg.configure("local.cfg.xml ");

cfg.addAnnotatedClass(EProduct.class);

cfg.addAnnotatedClass(Student.class);

cfg.addAnnotatedClass(PhoneNumber.class);

cfg.addAnnotatedClass(Courses.class);

//Step 2

SessionFactory sf = cfg.buildSessionFactory();

return sf;

}

}

Creating a hibernate table util configuration file

* In the Project Explorer, expand **Product\_Details->Java Resources**
* Right click on **src** and choose **New->Other->General->File**
* In **Name** enter local.cfg.xml and click on **Finish**

Enter the following code:

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

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/ecommerce</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">password</property>

<!-- Optional properties -->

<property name="hibernate.show\_sql">true</property>

<property name="hibernate.format\_sql">true</property>

<property name="hbm2ddl.auto">update</property>

</session-factory>

</hibernate-configuration>

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 **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 the **Add** button to move **Product\_Details** 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

Running the project

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

**Step 3.7.16:** 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**