**Assisted Practice: 3.7 Component Mapping**

This section will guide you to:

* Set up Eclipse to work with Hibernate
* Set up database tables to do collection mapping using XML
* Create an HTML page to call a servlet
* Create a servlet that will display data from the tables using Component Mapping

**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
* Hibernate for Java 5.2.1
* MySQL Connector for Java 8.0.16
* JTA v 1.1
* Java XML Bind (no version)
* JAXB OSGI v.2.4.0
* Java Activation (no version)

This lab has sixteen subsections, namely:

* + 1. Creating a dynamic web project
    2. Adding the jar files for Hibernate and its dependencies
    3. Creating a table eproduct in the database and filling it with sample data
    4. Creating a class Eproduct
    5. Creating a class ProductParts
    6. Creating a HibernateUtil class to initiate Hibernate in code
    7. Creating a hibernate table configuration file Eproduct.hbm.xml
    8. Configuring Hibernate with hibernate.cfg.xml
    9. Creating an HTML page index.html
    10. Creating a ProductDetails servlet
    11. Configuring web.xml
    12. Checking for servlet-api.jar
    13. Building the project
    14. Publishing and starting the project
    15. Running the project
    16. Pushing the code to your GitHub repositories

**Step 3.7.1:** Creating a dynamic web project

* Open Eclipse
* Go the **File** menu. Choose **New->Dynamic Web Project**
* Enter the project name as **HibernateComponentMapping**. 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 3.7.2:** Adding the jar files for Hibernate and its dependencies

* **Hibernate.jar** file is already present in your practice lab. (Refer FSD: Lab Guide - Phase 2)
* Take **hibernate.jar** from folder mentioned in the lab guide for phase 2 and add it to your project’s **WebContent/WEB-INF/lib** folder
* **mysql-connector-java.jar** file is present in your practice lab. (Refer FSD: Lab Guide - Phase 2)
* Take **mysql-connector-java.jar file** from the folder mentioned in the lab guide for phase 2 and add it to your 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

**Step 3.7.3:** 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 \*/;

-- Dump completed on 2019-06-07 10:05:19

**Step 3.7.4:** Creating a class EProduct

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

**package** com.ecommerce;

**import** java.math.BigDecimal;

**import** java.util.Collection;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Set;

**import** java.util.Map;

**public** **class** EProduct {

**private** long ID;

**private** **String** name;

**private** **BigDecimal** price;

**private** **Date** dateAdded;

**private** ProductParts parts;

**public** EProduct() {

}

**public** long getID() {**return** **this**.ID; }

**public** **String** getName() { **return** **this**.name;}

**public** **BigDecimal** getPrice() { **return** **this**.price;}

**public** **Date** getDateAdded() { **return** **this**.dateAdded;}

**public** ProductParts getParts() { **return** **this**.parts;}

**public** void setID(long id) { **this**.ID = id;}

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

**public** void setPrice(**BigDecimal** price) { **this**.price = price;}

**public** void setDateAdded(**Date** date) { **this**.dateAdded = date;}

**public** void setParts(ProductParts parts) { **this**.parts = parts;}

}

**Step 3.7.5:** Creating a class ProductParts

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

**package** com.ecommerce;

**public** **class** ProductParts {

**private** **String** hdd;

**private** **String** cpu;

**private** **String** ram;

**public** **String** getHdd() { **return** **this**.hdd;}

**public** **String** getCpu() { **return** **this**.cpu;}

**public** **String** getRam() { **return** **this**.ram;}

**public** void setHdd(**String** value) { **this**.hdd= value;}

**public** void setCpu(**String** value) { **this**.cpu= value;}

**public** void setRam(**String** value) { **this**.ram= value;}

}

**Step 3.7.6:** Creating a HibernateUtil class to initiate Hibernate in code

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

**package** com.ecommerce;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**public** **class** HibernateUtil {

**private** **static** **final** SessionFactory sessionFactory;

**static** {

**try** {

StandardServiceRegistry standardRegistry = **new** StandardServiceRegistryBuilder()

.configure("hibernate.cfg.xml").build();

Metadata metaData = **new** MetadataSources(standardRegistry).getMetadataBuilder().build();

sessionFactory = metaData.getSessionFactoryBuilder().build();

} **catch** (**Throwable** th) {

**throw** **new** **ExceptionInInitializerError**(th);

}

}

**public** **static** SessionFactory getSessionFactory() {

**return** sessionFactory;

}

}

**Step 3.7.7:** Creating a hibernate table configuration file EProduct.hbm.xml

* In the Project Explorer, expand **HibernateConfig->Java Resources**
* Right click on **src** and choose **New->Other**
* Select **General->File** and click on **Next**
* In filename, enter **EProduct.hbm.xml** and click on **Finish**
* Enter the following code:

<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

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

<hibernate-mapping package="com.ecommerce">

<class name="EProduct" table="eproduct">

<id name="ID" type="long" column="ID">

<generator class="identity"/>

</id>

<property name="name" type="string" column="NAME"/>

<property name="price" type="big\_decimal" column="PRICE"/>

<property name="dateAdded" type="timestamp" column="DATE\_ADDED"/>

<component name="parts" class="com.ecommerce.ProductParts">

<property name="hdd" column="parts\_hdd" type="string" />

<property name="cpu" column="parts\_cpu" type="string" />

<property name="ram" column="parts\_ram" type="string" />

</component>

</class>

</hibernate-mapping>

**Step 3.7.8:** Configuring Hibernate with hibernate.cfg.xml

* In the Project Explorer, expand **HibernateComponentMapping->Java Resources**
* Right click on **src** and choose **New->Other**
* Select **General->File** and click on **Next**
* In filename, enter **hibernate.cfg.xml** and click on **Finish**
* Enter the following code:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

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

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

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

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

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

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

<mapping resource="com/ecommerce/EProduct.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**Step 3.7.9:** Creating an HTML page index.html

* In the Project Explorer, expand the project **HibernateComponentMapping**
* 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**>Hibernate Component Mapping</**title**>

</**head**>

<**body**>

<**a** href="details">Product Details</**a**><**br**>

</**body**>

</**html**>

* Click on the **Save** icon

**Step 3.7.10:** Creating a ProductDetails servlet

* In the Project Explorer, expand **HibernateComponentMapping->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.PrintWriter;

**import** javax.servlet.ServletConfig;

**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.transaction.**\***;

**import** javax.xml.bind.**\***;

**import** java.io.Serializable;

**import** java.math.BigDecimal;

**import** java.util.ArrayList;

**import** java.util.Calendar;

**import** java.util.Collection;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Set;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**import** com.ecommerce.EProduct;

**import** com.ecommerce.HibernateUtil;

**import** com.ecommerce.ProductParts;

/\*\*

**\*** Servlet implementation class ProductDetails

\*/

**@WebServlet("/ProductDetails")**

**public** **class** ProductDetails **extends** HttpServlet {

**private** **static** **final** long serialVersionUID = 1L;

/\*\*

**\*** **@see** HttpServlet**#**HttpServlet()

\*/

**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

**try** {

SessionFactory factory = HibernateUtil.getSessionFactory();

Session session = factory.openSession();

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

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

out.println("<html><body>");

out.println("<b>Component Mapping</b><br>");

**for**(EProduct p: list) {

out.println("ID: " + **String**.valueOf(p.getID()) + ", Name: " + p.getName() +

", Price: " + **String**.valueOf(p.getPrice()) + ", Date Added: " + p.getDateAdded().toString());

ProductParts parts = p.getParts();

out.println("Parts =" + parts.getCpu() + ", " + parts.getHdd() + ", " + parts.getRam());

out.println("<hr>");

}

session.close();

out.println("</body></html>");

} **catch** (**Exception** ex) {

**throw** ex;

}

}

/\*\*

**\*** **@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);

}

}

**Step 3.7.11:** Configuring web.xml

* In the Project Explorer, expand **HibernateComponentMapping->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://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd" id="WebApp\_ID" version="4.0">

<display-name>HibernateComponentMapping</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>

</web-app>

**Step 3.7.12:** 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 **HibernateComponentMapping** 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**

**Step 3.7.13:** Building the project

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

**Step 3.7.14:** 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 **HibernateComponentMapping** 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

**Step 3.7.15:** Running the project

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

**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**