**Assisted Practice: 1.3 JDBC Framework**

This section will guide you to:

* Set up Eclipse to work with Spring and JDBC
* Create a pom.xml configuration file to add all the required components
* Set up database tables to use in JDBC
* Create an entity class EProduct
* Create a DAO class EProductDAO
* Create a JSP file to call a Controller
* Create a Controller to work with the database using JDBC
* Create a JSP view to display data from the Controller

**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
* All other software components are defined in the Project Object Model pom.xml

This lab has fourteen subsections, namely:

* + 1. Creating a Maven project which is web enabled
    2. Creating pom.xml for including the required components
    3. Creating a table eproduct in the database and filling it with sample data
    4. Creating an Entity class EProduct
    5. Creating a DAO class EProductDAO
    6. Creating a Controller class MainController
    7. Creating the dispatcher servlet main-servlet.xml
    8. Creating a view index.jsp
    9. Creating a view listProducts.jsp
    10. Configuring web.xml
    11. Building the project
    12. Publishing and starting the project
    13. Running the project
    14. Pushing the code to your GitHub repositories

**Step 1.3.1:** Creating a Maven Project which is web enabled

* Open Eclipse
* Go to the **File** menu. Choose **New->Maven Project**
* Uncheck **Create a Simple Project** and check **Use Default Workspace Location** and click on **Next**
* From the **archetype** list, choose the row that has **Artifact Id** as **maven-archetype-webapp** and click on **Next**
* Enter **Group Id** as **com** and **Artifact Id** as **SpringSetup** and click on **Finish**
* This will create the project files in the Project Explorer
* Before building the project, we need to confirm that the **servlet.jar** has been added to the project
* Servlet is already installed in your practice labs (Refer FSD: Lab Guide - Phase 2)
* To add it to the project, follow the below mentioned steps:
  + In the Project Explorer, right click and choose **Properties**
  + Select **Java Build Path** from the options on the left
  + Click on **Libraries** tab on the right
  + If there is an existing entry for the **servlet.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.jar** file and click on **Ok**
  + Click on **Apply and Close**

**Step 1.3.2:** Creating pom.xml for including the required components

* In the Project Explorer, expand **SpringSetup** and double click **pom.xml**
* Add the following entries:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>SpringSetup</groupId>

<artifactId>SpringSetup</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>SpringSetup Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<org.springframework.version>5.1.6.RELEASE</org.springframework.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.16</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>javassist</groupId>

<artifactId>javassist</artifactId>

<version>3.12.1.GA</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.9</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>javax.persistence</groupId>

<artifactId>persistence-api</artifactId>

<version>1.0.2</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<finalName>SpringSetup</finalName>

</build>

</project>

**Step 1.3.3:** Creating a table eproduct in the database and filling it with sample data

* MySQL is already installed in your practice lab. You can refer to the lab guide for phase 2 to learn about its directory path details
* 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:

--

-- 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**,

**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,'Acer Laptop ABC',23300.00,'2019-06-04 07:19:07'),(3,'Lenovo Laptop ABC',33322.00,'2019-06-04 07:19:19');

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

**UNLOCK** **TABLES**;

**Step 1.3.4:** Creating an Entity class EProduct

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce.entity and in **Name** enter Eproduct and click on **Finish**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**public** **class** EProductEntity {

**private** long ID;

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

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

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

**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** 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;}

}

**Step 1.3.5:** Creating a DAO class EProductDAO

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce.dao and in **Name** enter EProductDAO and click on **Finish**
* Enter the following code:

**package** com.ecommerce.dao;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.List;

**import** org.springframework.jdbc.core.BeanPropertyRowMapper;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.jdbc.core.RowMapper;

**import** com.ecommerce.entity.EProduct;

**public** **class** EProductDAO {

JdbcTemplate template;

**public** void setTemplate(JdbcTemplate template) {

**this**.template = template;

}

**public** **List**<EProduct> getProducts(){

**return** template.query("select \* from eproduct",**new** RowMapper<EProduct>(){

**public** EProduct mapRow(**ResultSet** rs, int row) **throws** **SQLException** {

EProduct e=**new** EProduct();

e.setId(rs.getInt(1));

e.setName(rs.getString(2));

e.setPrice(rs.getBigDecimal(3));

e.setDateAdded(rs.getDate(4));

**return** e;

}

});

}

}

**Step 1.3.6:** Creating a Controller class MainController

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce.controller and in **Name** enter MainControllerand click on **Finish**
* Enter the following code:

**package** com.ecommerce.controller;

**import** org.apache.log4j.Logger;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.ecommerce.entity.EProduct;

**import** com.ecommerce.dao.EProductDAO;

**@Controller**

**public** **class** MainController {

**@Autowired**

EProductDAO eproductDAO;

**@RequestMapping(value = "/listProducts", method = RequestMethod.GET)**

**public** **String** listProducts(ModelMap map)

{

**List**<EProduct> list= eproductDAO.getProducts();

model.addAttribute("list",list);

**return** "listProducts";

}

}

**Step 1.3.7:** Creating the dispatcher servlet main-servlet.xml

* In the Project Explorer, expand **SpringSetup->src->main->webapp**
* Right click on **WEB-INF** and choose **New->File**
* In **filename,** enter **main-servlet.xml** and click on **Finish**
* Enter the following script:

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:jee="http://www.springframework.org/schema/jee"

xmlns:lang="http://www.springframework.org/schema/lang"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:util="http://www.springframework.org/schema/util"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-2.5.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-2.5.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">

<context:annotation-config />

<context:component-scan base-package="com.ecommerce.controller" />

<bean id="jspViewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass"

value="org.springframework.web.servlet.view.JstlView"></property>

<property name="prefix" value="/WEB-INF/view/"></property>

<property name="suffix" value=".jsp"></property>

</bean>

<bean id="messageSource"

class="org.springframework.context.support.ReloadableResourceBundleMessageSource">

<property name="basename" value="classpath:messages"></property>

<property name="defaultEncoding" value="UTF-8"></property>

</bean>

<bean id="dataSource"

class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close"

p:driverClassName="${jdbc.driverClassName}"

p:url="${jdbc.databaseurl}" p:username="${jdbc.username}"

p:password="${jdbc.password}"></bean>

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver"></property>

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

<property name="username" value="root"></property>

<property name="password" value="master"></property>

</bean>

<bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">

<property name="dataSource" ref="ds"></property>

</bean>

<bean id="dao" class="com.ecommerce.dao.EProductDAO">

<property name="template" ref="jt"></property>

</bean>

<tx:annotation-driven />

<bean id="transactionManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory" ref="sessionFactory"></property>

</bean>

</beans>

**Step 1.3.8:** Creating a view index.jsp

* In the Project Explorer, expand **SpringSetup->src->main->webapp->WEB-INF**
* Right click on **view** and choose **New->File**
* In **filename,** enter **index.jsp** and click on **Finish**
* Enter the following code:

<**html**>

<**body**>

<**h2**>Spring Application</**h2**>

<**a** href="listProducts">List Products</**a**>

</**body**>

</**html**>

**Step 1.3.9:** Creating a view listProducts.jsp

* In the Project Explorer, expand **SpringSetup->src->main->webapp->WEB-INF->view**
* Right click on **view** and choose **New->File**
* In **filename,** enter **listProducts.jsp** and clickon **Finish**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>List Products</**title**>

</**head**>

<**body**>

<c:forEach var="product" items="${list}">

${product.name}, ${product.price}, ${product.dateAdded}<**br**>

</c:forEach>

</**body**>

</**html**>

**Step 1.3.10:** Configuring web.xml

* In the Project Explorer, expand **SpringSetup->src->main->webapp->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="http://java.sun.com/xml/ns/javaee"

version="2.5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<display-name>Archetype Created Web Application</display-name>

<welcome-file-list>

<welcome-file>/WEB-INF/view/index.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>main</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>main</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/main-servlet.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

</web-app>

**Step 1.3.11:** Building the project

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

**Step 1.3.12:** 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 **SpringSetup** 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 1.3.13:** Running the project

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

**Step 1.3.14:** 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**