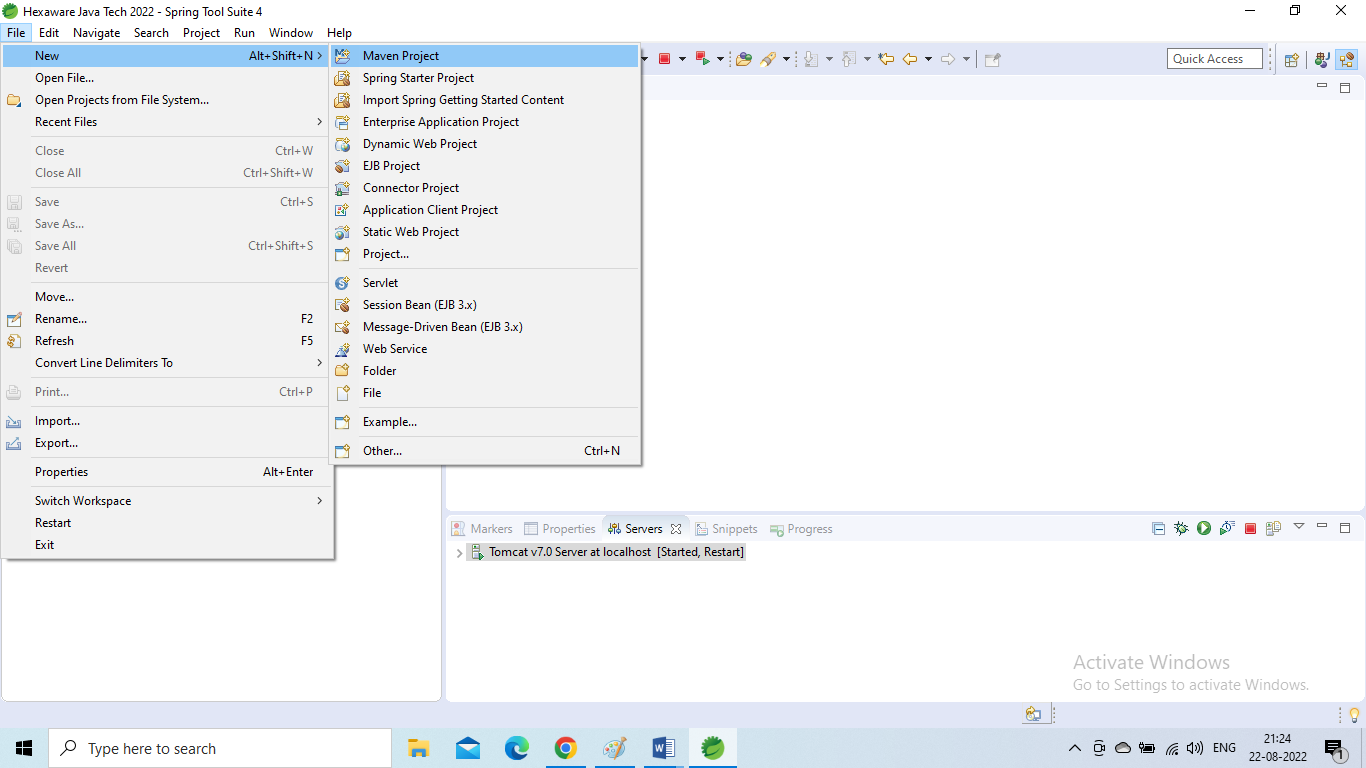
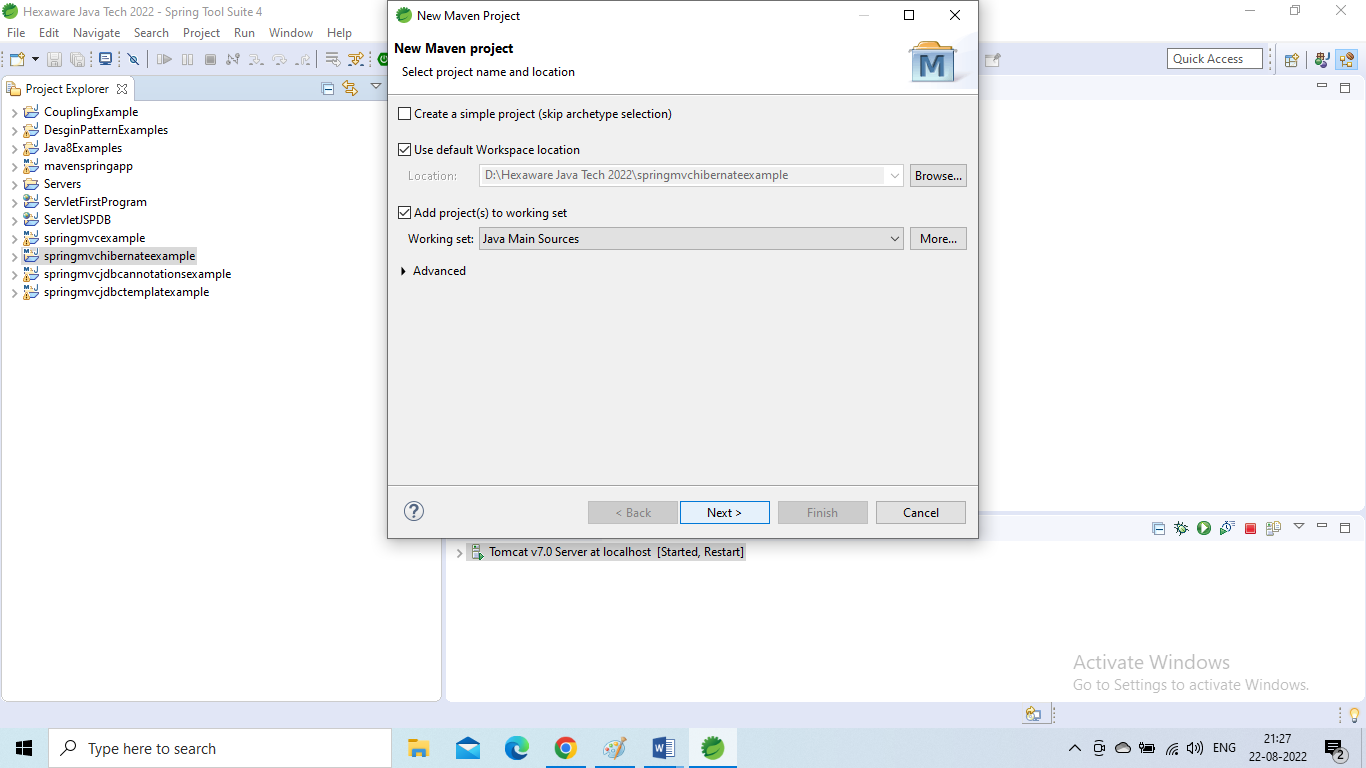
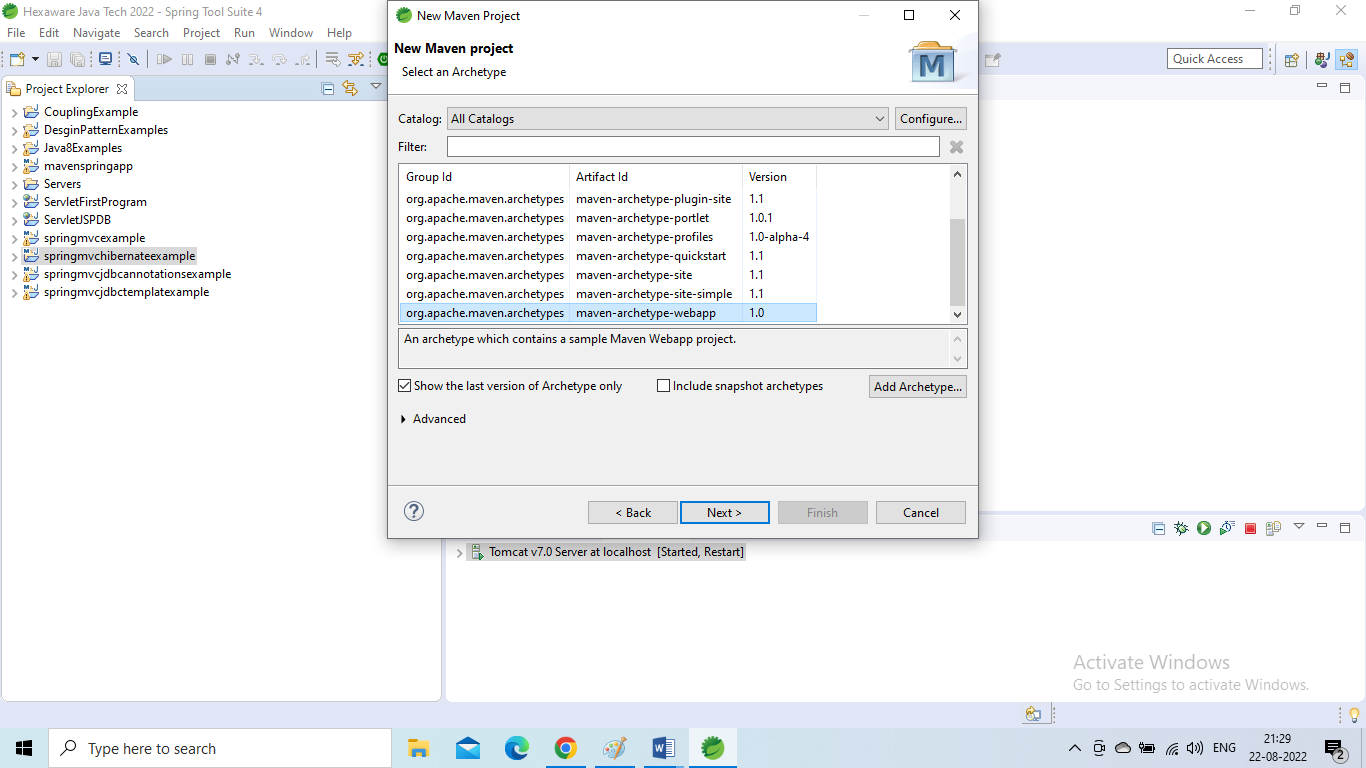
Step 1 :



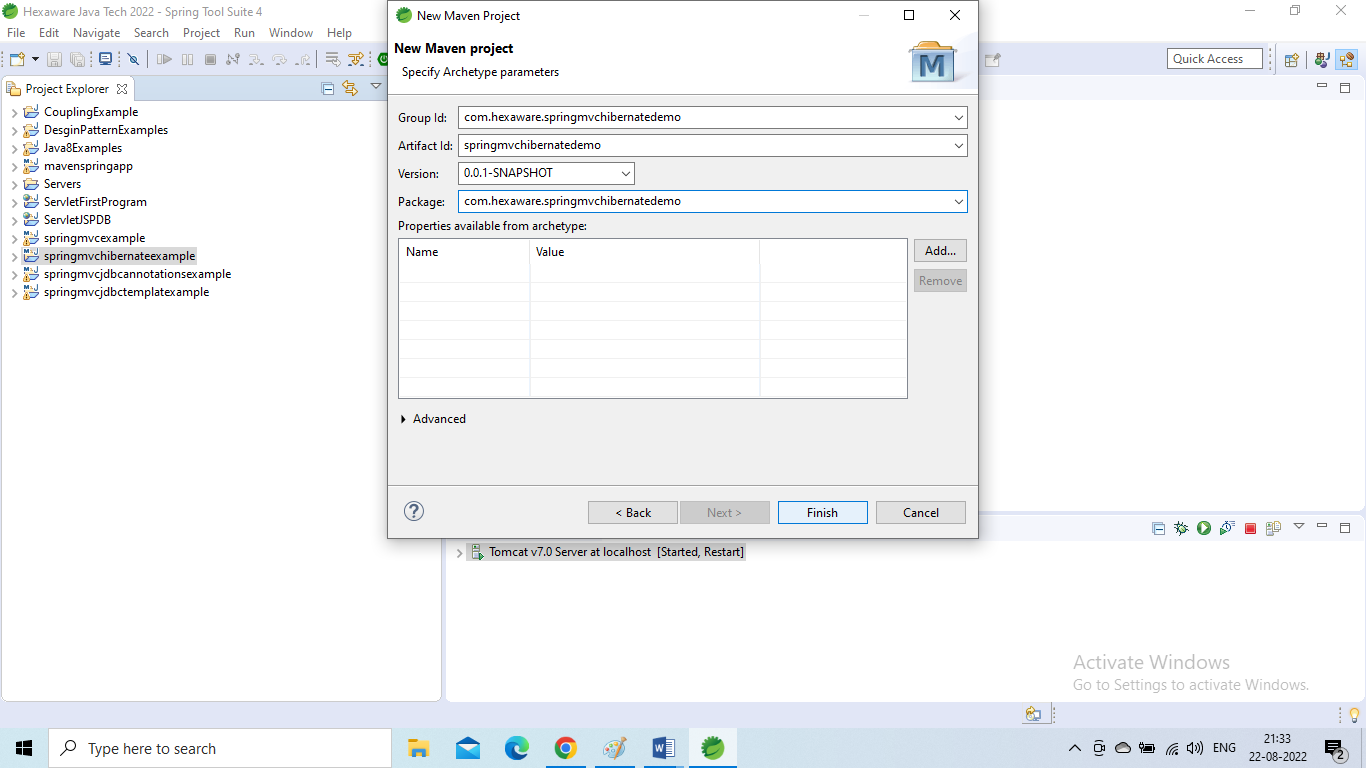
Step 2:



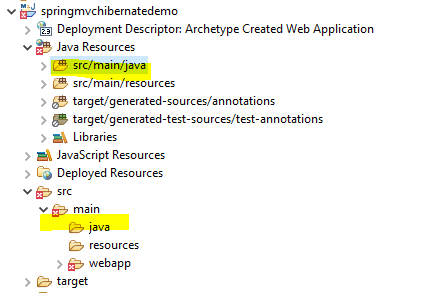
Step 3:



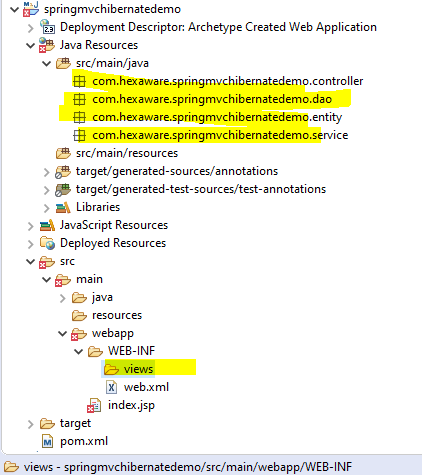
Step 4:



Step 5:

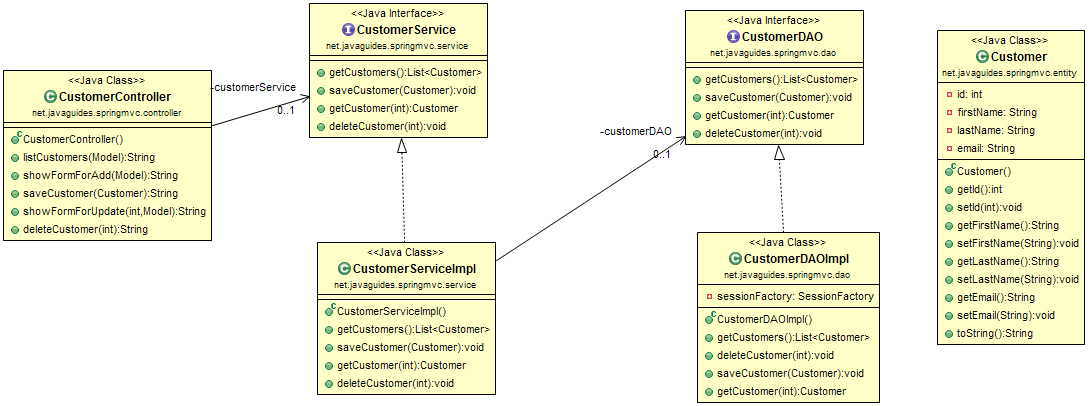


Step 6:



Step 7:

**Class Diagram**



Step 8:

# **Register a DispatcherServlet using XML-based Spring configuration**

In Spring MVC, The *DispatcherServlet* needs to be declared and mapped for processing all requests either using web.xml configuration.

Let's use the below configuration in the *web.xml* file:

<?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\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>spring-mvc-crud-demo</display-name>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

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

<param-value>/WEB-INF/spring-mvc-crud-demo-servlet.xml</param-value>

</init-param>

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

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

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

</servlet-mapping>

</web-app>

Step 9:

**Spring and Hibernate Integration using XML-based Spring configuration**

Hibernate configuration used in the example is based on hibernate XML based configuration.

Let's create *spring-mvc-crud-demo-servlet.xml* file and add the following configuration to it:

<?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:context="http://www.springframework.org/schema/context"

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

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

xsi:schemaLocation="

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

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

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

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

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

http://www.springframework.org/schema/mvc/spring-mvc.xsd

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

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

<!-- Add support for component scanning -->

<context:component-scan base-package=" com.hexaware.springmvchibernatedemo" />

<!-- Add support for conversion, formatting and validation support -->

<mvc:annotation-driven/>

<!-- Define Spring MVC view resolver -->

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

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

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

</bean>

<!-- Step 1: Define Database DataSource / connection pool -->

<bean id="myDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"

destroy-method="close">

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

<property name="jdbcUrl" value="jdbc:mysql://localhost:3306/demo?useSSL=false&amp;serverTimezone=UTC" />

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

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

<!-- these are connection pool properties for C3P0 -->

<property name="minPoolSize" value="5" />

<property name="maxPoolSize" value="20" />

<property name="maxIdleTime" value="30000" />

</bean>

<!-- Step 2: Setup Hibernate session factory -->

<bean id="sessionFactory"

class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

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

<property name="packagesToScan" value=" com.hexaware.springmvchibernatedemo.entity" />

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>

<prop key="hibernate.show\_sql">true</prop>

</props>

</property>

</bean>

<!-- Step 3: Setup Hibernate transaction manager -->

<bean id="myTransactionManager"

class="org.springframework.orm.hibernate5.HibernateTransactionManager">

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

</bean>

<!-- Step 4: Enable configuration of transactional behavior based on annotations -->

<tx:annotation-driven transaction-manager="myTransactionManager" />

<!-- Add support for reading web resources: css, images, js, etc ... -->

<mvc:resources location="/resources/" mapping="/resources/\*\*"></mvc:resources>

</beans>

**Configuring Spring MVC View Resolvers**

**Add support for conversion, formatting and validation support:**

<!-- Add support for conversion, formatting and validation support -->

<mvc:annotation-driven/>

**Add support for component scanning :**

<!-- Add support for component scanning -->

<context:component-scan base-package=" com.hexaware.springmvchibernatedemo" />

**Enable configuration of transactional behavior based on annotations:**

<!-- Step 4: Enable configuration of transactional behavior based on annotations -->

<tx:annotation-driven transaction-manager="myTransactionManager" />

**Add support for reading web resources: css, images, js, etc:**

<!-- Add support for reading web resources: css, images, js, etc ... -->

<mvc:resources location="/resources/" mapping="/resources/\*\*"></mvc:resources>

**Define Spring MVC view resolver:**

<!-- Define Spring MVC view resolver -->

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

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

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

</bean>

Step 10:

**Add Dependencies - pom.xml File**

<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>com.hexaware.springmvchibernatedemo</groupId>

<artifactId>springmvchibernatedemo</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>springmvchibernatedemo Maven Webapp</name>

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

<properties>

<failOnMissingWebXml>false</failOnMissingWebXml>

<spring.version>5.1.0.RELEASE</spring.version>

<hibernate.version>5.3.5.Final</hibernate.version>

<hibernate.validator>5.4.1.Final</hibernate.validator>

<c3p0.version>0.9.5.2</c3p0.version>

<jstl.version>1.2.1</jstl.version>

<tld.version>1.1.2</tld.version>

<servlets.version>3.1.0</servlets.version>

<jsp.version>2.3.1</jsp.version>

<hsqldb.version>1.8.0.10</hsqldb.version>

</properties>

<dependencies>

<!-- Spring MVC Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Spring ORM -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Hibernate Core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.17.Final</version>

</dependency>

<!-- Hibernate Validator -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>${hibernate.validator}</version>

</dependency>

<!-- JSTL Dependency -->

<dependency>

<groupId>javax.servlet.jsp.jstl</groupId>

<artifactId>javax.servlet.jsp.jstl-api</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>${tld.version}</version>

</dependency>

<!-- Servlet Dependency -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>${servlets.version}</version>

<scope>provided</scope>

</dependency>

<!-- JSP Dependency -->

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>javax.servlet.jsp-api</artifactId>

<version>${jsp.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.47</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

# **JPA Entity - Customer.java**

Create a *@Entity* class, whose field names are annotated JPA annotations. We will use this entity class for mapping the database table with *Customer*. It will also be used for binding from data to the model using *@ModelAttribute* annotation in the controller's handler method.

**package** com.hexaware.springmvchibernatedemo.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name = "customer")

**public** **class** Customer {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** **int** id;

@Column(name = "first\_name")

**private** String firstName;

@Column(name = "last\_name")

**private** String lastName;

@Column(name = "email")

**private** String email;

**public** Customer() {

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

@Override

**public** String toString() {

**return** "Customer [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", email=" + email + "]";

}

}

# **Spring MVC Controller Class - CustomerController.java**

Create a controller class to handle the *Customer* form data as follows

**package** com.hexaware.springmvchibernatedemo.controller;

**import** java.util.List;

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

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

**import** org.springframework.ui.Model;

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

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

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

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

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

**import** com.hexaware.springmvchibernatedemo.entity.Customer;

**import** com.hexaware.springmvchibernatedemo.service.CustomerService;

@Controller

@RequestMapping("/customer")

**public** **class** CustomerController {

@Autowired

**private** CustomerService customerService;

@GetMapping("/list")

**public** String listCustomers(Model theModel) {

List < Customer > theCustomers = customerService.getCustomers();

theModel.addAttribute("customers", theCustomers);

**return** "list-customers";

}

@GetMapping("/showForm")

**public** String showFormForAdd(Model theModel) {

Customer theCustomer = **new** Customer();

theModel.addAttribute("customer", theCustomer);

**return** "customer-form";

}

@PostMapping("/saveCustomer")

**public** String saveCustomer(@ModelAttribute("customer") Customer theCustomer) {

customerService.saveCustomer(theCustomer);

**return** "redirect:/customer/list";

}

@GetMapping("/updateForm")

**public** String showFormForUpdate(@RequestParam("customerId") **int** theId,

Model theModel) {

Customer theCustomer = customerService.getCustomer(theId);

theModel.addAttribute("customer", theCustomer);

**return** "customer-form";

}

@GetMapping("/delete")

**public** String deleteCustomer(@RequestParam("customerId") **int** theId) {

customerService.deleteCustomer(theId);

**return** "redirect:/customer/list";

}}

**Service Layer - CustomerService.java and CustomerServiceImpl.java**

Let's create the *CustomerService* interface and its implementation under *net.javaguides.springmvc.service* package as follows.

### CustomerService.java

**import** java.util.List;

**import** com.hexaware.springmvchibernatedemo.entity.Customer;

**public** **interface** CustomerService {

**public** List < Customer > getCustomers();

**public** **void** saveCustomer(Customer theCustomer);

**public** Customer getCustomer(**int** theId);

**public** **void** deleteCustomer(**int** theId);

}

**CustomerServiceImpl.java**

We are using *@Transactional* annotation which is applied at service layer for transaction support. *@Service* annotation used to annotate service layer implementation classes as in below file.

**package** com.hexaware.springmvchibernatedemo.service;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.hexaware.springmvchibernatedemo.dao.CustomerDAO;

**import** com.hexaware.springmvchibernatedemo.entity.Customer;

@Service

**public** **class** CustomerServiceImpl **implements** CustomerService {

@Autowired

**private** CustomerDAO customerDAO;

@Transactional

**public** List < Customer > getCustomers() {

**return** customerDAO.getCustomers();

}

@Transactional

**public** **void** saveCustomer(Customer theCustomer) {

customerDAO.saveCustomer(theCustomer);

}

@Transactional

**public** Customer getCustomer(**int** theId) {

**return** customerDAO.getCustomer(theId);

}

@Transactional

**public** **void** deleteCustomer(**int** theId) {

customerDAO.deleteCustomer(theId);

}

}

**DAO Layer - CustomerDAO.java and CustomerDAOImpl.java**

### CustomerDAO.java

**package** com.hexaware.springmvchibernatedemo.dao;

**import** java.util.List;

**import** com.hexaware.springmvchibernatedemo.entity.Customer;

**public** **interface** CustomerDAO {

**public** List < Customer > getCustomers();

**public** **void** saveCustomer(Customer theCustomer);

**public** Customer getCustomer(**int** theId);

**public** **void** deleteCustomer(**int** theId); }

### CustomerDAOImpl.java

**package** com.hexaware.springmvchibernatedemo.dao;

**import** java.util.List;

**import** javax.persistence.Query;

**import** javax.persistence.criteria.CriteriaBuilder;

**import** javax.persistence.criteria.CriteriaQuery;

**import** javax.persistence.criteria.Root;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

**import** org.springframework.stereotype.Repository;

**import** com.hexaware.springmvchibernatedemo.entity.Customer;

@Repository

**public** **class** CustomerDAOImpl **implements** CustomerDAO {

@Autowired

**private** SessionFactory sessionFactory;

**public** List < Customer > getCustomers() {

Session session = sessionFactory.getCurrentSession();

CriteriaBuilder cb = session.getCriteriaBuilder();

CriteriaQuery < Customer > cq = cb.createQuery(Customer.**class**);

Root < Customer > root = cq.from(Customer.**class**);

cq.select(root);

Query query = session.createQuery(cq);

**return** query.getResultList();

}

**public** **void** deleteCustomer(**int** id) {

Session session = sessionFactory.getCurrentSession();

Customer book = session.byId(Customer.**class**).load(id);

session.delete(book);

}

**public** **void** saveCustomer(Customer theCustomer) {

Session currentSession = sessionFactory.getCurrentSession();

currentSession.saveOrUpdate(theCustomer);

}

**public** Customer getCustomer(**int** theId) {

Session currentSession = sessionFactory.getCurrentSession();

Customer theCustomer = currentSession.get(Customer.**class**, theId);

**return** theCustomer;

}

}

**JSP Views - customer-form.jsp and list-customers.jsp**

**customer-form.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

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

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

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Spring MVC 5 - form handling </title>

<link href=*"*<c:url value=*"/resources/css/bootstrap.min.css"* />*"*

rel=*"stylesheet"*>

<script src=*"*<c:url value=*"/resources/js/jquery-1.11.1.min.js"* />*"*></script>

<script src=*"*<c:url value=*"/resources/js/bootstrap.min.js"* />*"*></script>

</head>

<body>

<div class=*"container"*>

<div class=*"col-md-offset-2 col-md-7"*>

<h2 class=*"text-center"*>Spring MVC 5 + Hibernate 5 + JSP + MySQL

Example</h2>

<div class=*"panel panel-info"*>

<div class=*"panel-heading"*>

<div class=*"panel-title"*>Add Customer</div>

</div>

<div class=*"panel-body"*>

<form:form action=*"saveCustomer"* cssClass=*"form-horizontal"*

method=*"post"* modelAttribute=*"customer"*>

<!-- need to associate this data with customer id -->

<form:hidden path=*"id"* />

<div class=*"form-group"*>

<label for=*"firstname"* class=*"col-md-3 control-label"*>First

Name</label>

<div class=*"col-md-9"*>

<form:input path=*"firstName"* cssClass=*"form-control"* />

</div>

</div>

<div class=*"form-group"*>

<label for=*"lastname"* class=*"col-md-3 control-label"*>Last

Name</label>

<div class=*"col-md-9"*>

<form:input path=*"lastName"* cssClass=*"form-control"* />

</div>

</div>

<div class=*"form-group"*>

<label for=*"email"* class=*"col-md-3 control-label"*>Email</label>

<div class=*"col-md-9"*>

<form:input path=*"email"* cssClass=*"form-control"* />

</div>

</div>

<div class=*"form-group"*>

<!-- Button -->

<div class=*"col-md-offset-3 col-md-9"*>

<form:button cssClass=*"btn btn-primary"*>Submit</form:button>

</div>

</div>

</form:form>

</div>

</div>

</div>

</div>

</body>

</html>

## **list-customers.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

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

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>List of Customers</title>

<link href=*"*<c:url value=*"/resources/css/bootstrap.min.css"* />*"*

rel=*"stylesheet"*>

<script src=*"*<c:url value=*"/resources/js/jquery-1.11.1.min.js"* />*"*></script>

<script src=*"*<c:url value=*"/resources/js/bootstrap.min.js"* />*"*></script>

</head>

<body>

<div class=*"container"*>

<div class=*"col-md-offset-1 col-md-10"*>

<h2>CRM - Customer Relationship Manager</h2>

<hr />

<input type=*"button"* value=*"Add Customer"*

onclick="window.location.href='showForm'; return false;"

class=*"btn btn-primary"* />

<br/><br/>

<div class=*"panel panel-info"*>

<div class=*"panel-heading"*>

<div class=*"panel-title"*>Customer List</div>

</div>

<div class=*"panel-body"*>

<table class=*"table table-striped table-bordered"*>

<tr>

<th>First Name</th>

<th>Last Name</th>

<th>Email</th>

<th>Action</th>

</tr>

<!-- loop over and print our customers -->

<c:forEach var=*"tempCustomer"* items=*"*${customers}*"*>

<!-- construct an "update" link with customer id -->

<c:url var=*"updateLink"* value=*"/customer/updateForm"*>

<c:param name=*"customerId"* value=*"*${tempCustomer.id}*"* />

</c:url>

<!-- construct an "delete" link with customer id -->

<c:url var=*"deleteLink"* value=*"/customer/delete"*>

<c:param name=*"customerId"* value=*"*${tempCustomer.id}*"* />

</c:url>

<tr>

<td>${tempCustomer.firstName}</td>

<td>${tempCustomer.lastName}</td>

<td>${tempCustomer.email}</td>

<td>

<!-- display the update link --> <a href=*"*${updateLink}*"*>Update</a>

| <a href=*"*${deleteLink}*"*

onclick="if (!(confirm('Are you sure you want to delete this customer?'))) return false">Delete</a>

</td>

</tr>

</c:forEach>

</table>

</div>

</div>

</div>

</div>

</body>

</html>

## **Serve Static Resources - CSS and JS**

1. Create a *resource* folder under webapp directory.
2. Create *css* and *js* folders under the *resource* directory.
3. Download and keep **bootstrap.min.css** file under *css* folder
4. download and keep **bootstrap.min.js** and **jquery-1.11.1.min.js** files under the resource directory. Note that bootstrap min js is depended on jquery min js.

## **Build and Run an application**