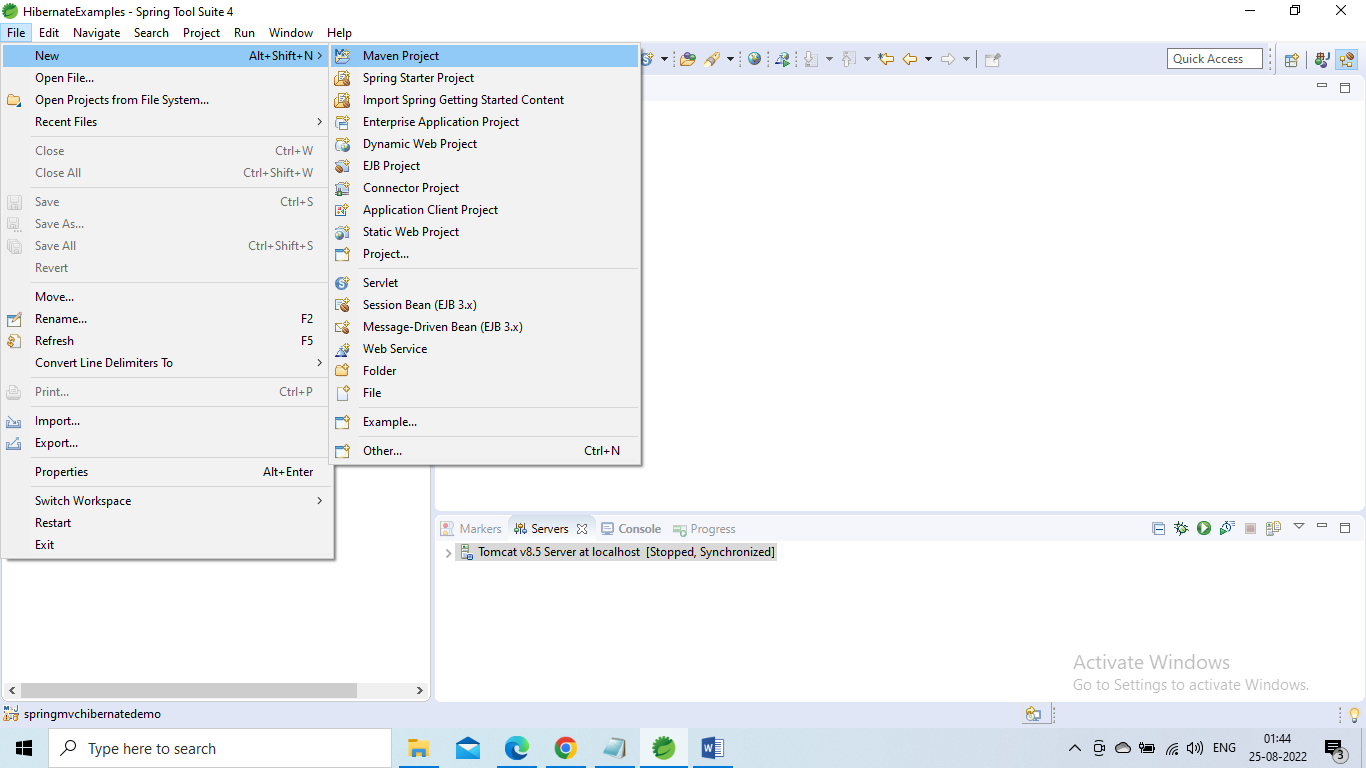
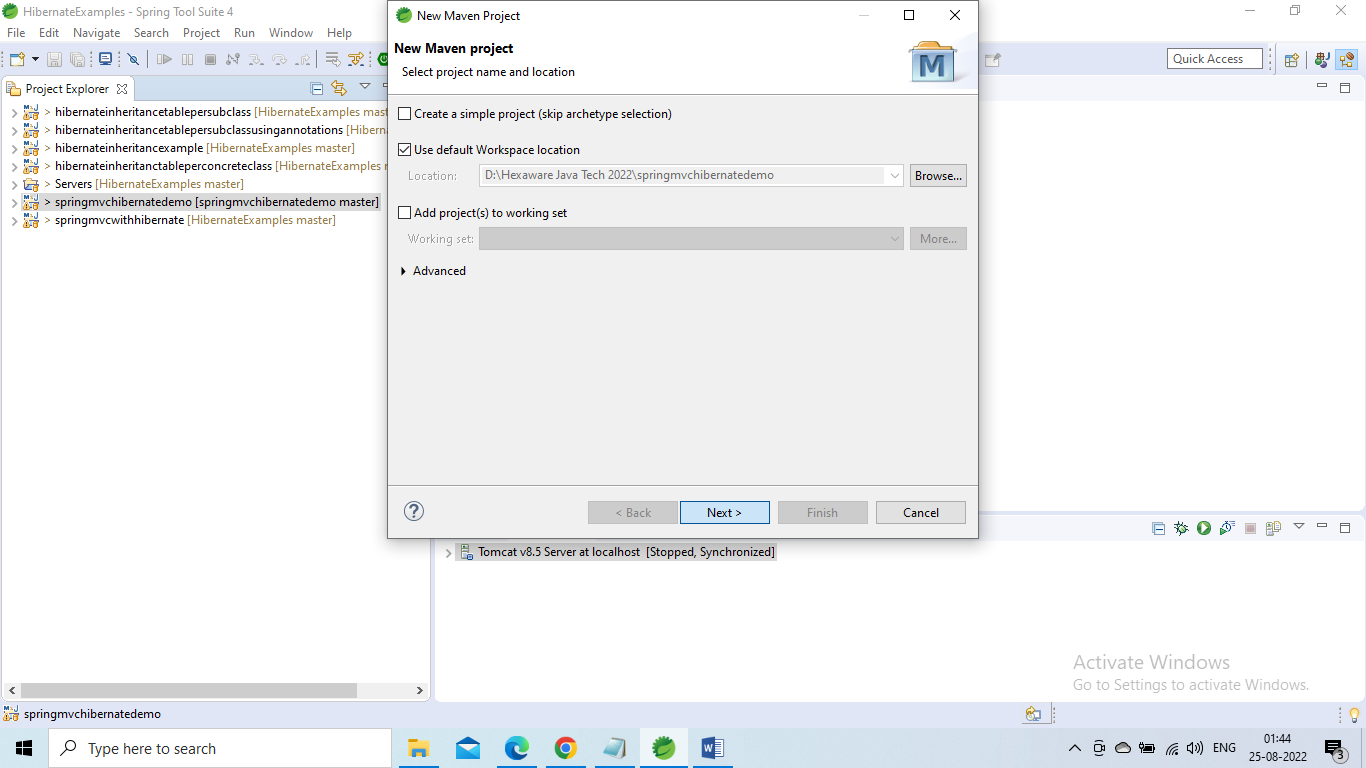
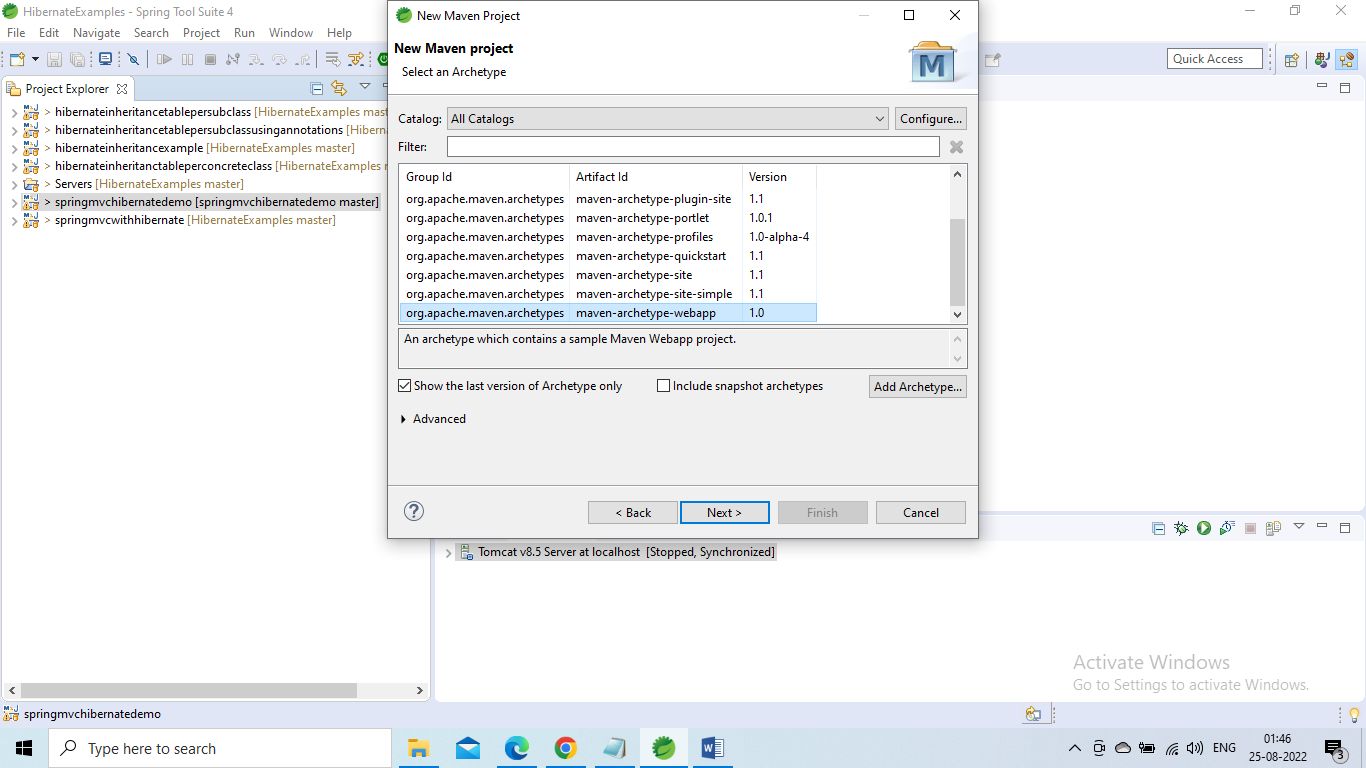
**Step 1:**



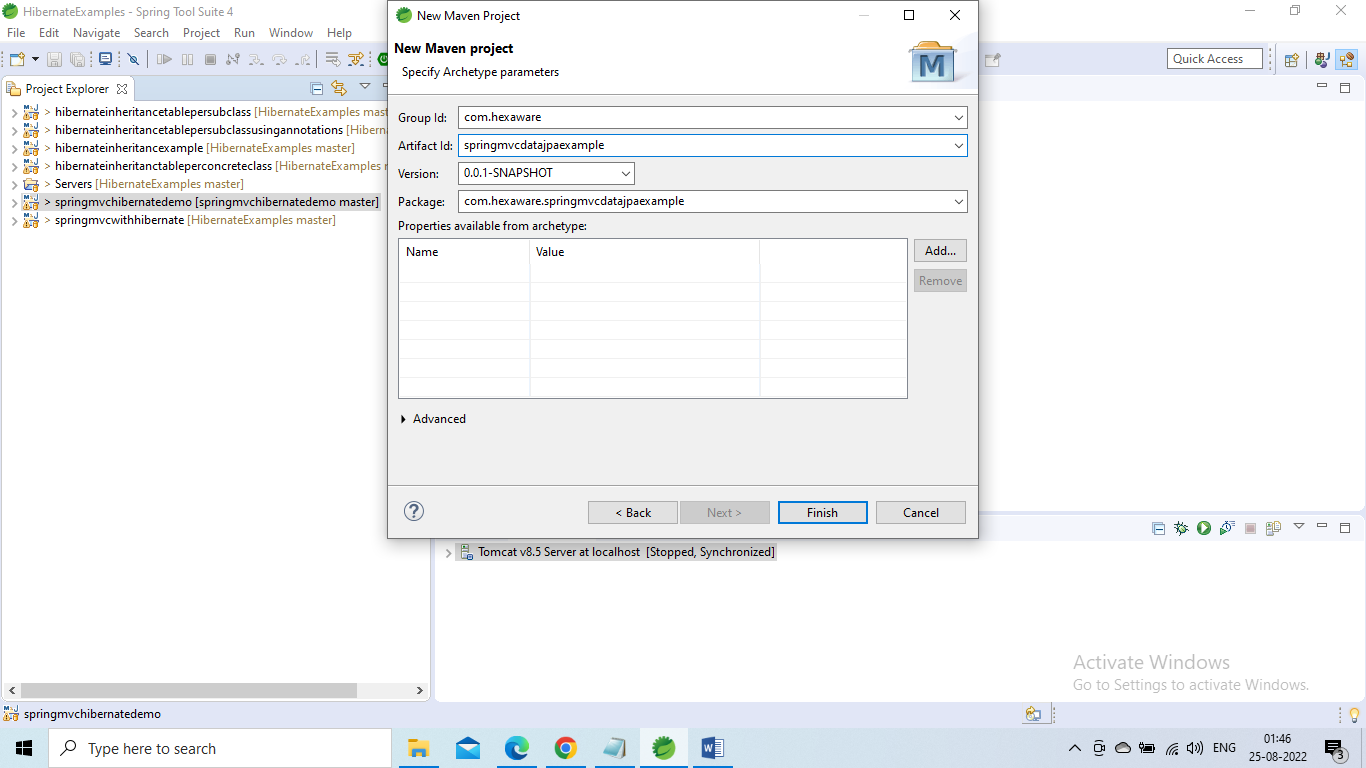
**Step 2:**



**Step 3:**



**Step 4:**



**Step 5:**

**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</groupId>

<artifactId>springmvcdatajpaexample</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>springmvcdatajpaexample Maven Webapp</name>

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

<properties>

<failOnMissingWebXml>false</failOnMissingWebXml>

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

<hibernate.version>5.2.17.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>

<log4j.version>1.2.17</log4j.version>

</properties>

<dependencies>

<!-- Spring MVC Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

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

<exclusions>

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

<!-- Spring ORM -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

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

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

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

</dependency>

<!-- Hibernate Validator -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

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

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.data/spring-data-jpa -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-jpa</artifactId>

<version>2.1.0.RELEASE</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>

<!-- logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>1.7.20</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.1.7</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>

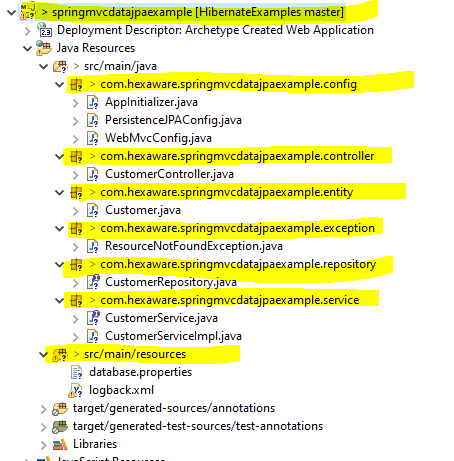
</plugins>

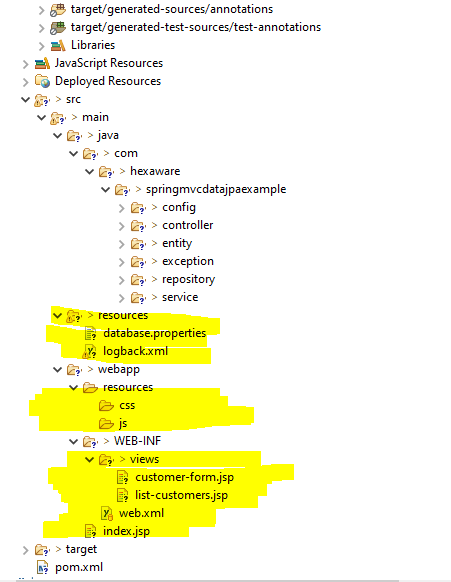
</build>

</project>

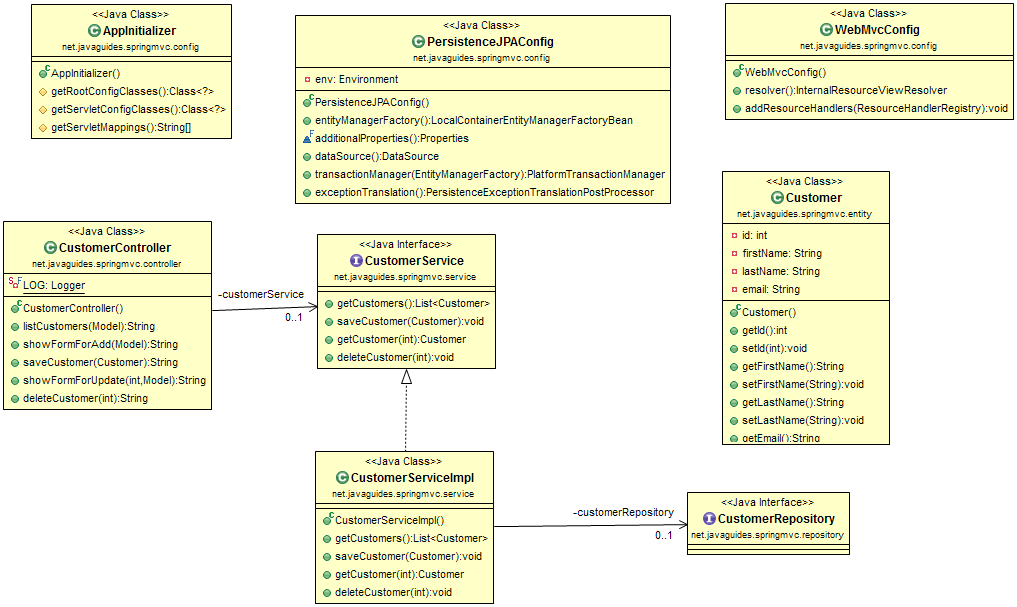
**Step 6:**

## Project Structure





## Class Diagram



## AppInitializer - Register a DispatcherServlet using Java-based Spring configuration

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

In a Servlet 3.0+ environment, you can use *AbstractAnnotationConfigDispatcherServletInitializer* class to register and initialize the *DispatcherServlet* programmatically as follows.

**package** com.hexaware.springmvcdatajpaexample.config;

**import** org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

**public** **class** AppInitializer **extends** AbstractAnnotationConfigDispatcherServletInitializer {

@Override

**protected** Class <?> [] getRootConfigClasses() {

**return** **new** Class[] {

PersistenceJPAConfig.**class**

};

//return null;

}

@Override

**protected** Class <?> [] getServletConfigClasses() {

**return** **new** Class[] {

WebMvcConfig.**class**

};

}

@Override

**protected** String[] getServletMappings() {

**return** **new** String[] {

"/"

};

}

}

**PersistenceJPAConfig - Spring Data JPA and Hibernate Configuration using Java-based Spring configuration**

**Database Configuration - database.properties**

# jdbc.X

jdbc.driverClassName=com.mysql.cj.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/sample?useSSL=false

jdbc.user=root

jdbc.pass=root

# hibernate.X

hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

hibernate.show\_sql=true

hibernate.hbm2ddl.auto=create-drop

hibernate.cache.use\_second\_level\_cache=false

hibernate.cache.use\_query\_cache=false

### **PersistenceJPAConfig.java**

Please note that the *@EnableJpaRepositories* annotation which enables usage of JPA repositories. The *net.javaguides.springmvc.repository* package will be scanned to detect repositories. In the *entityManagerFactory* bean, I determined that *Hibernate* will be used as JPA implementation.

*@EnableTransactionManagement* annotation enables Spring's annotation-driven transaction management capability, similar to the support found in Spring's **[tx:\*](https://www.blogger.com/null)** XML namespace.

**package** com.hexaware.springmvcdatajpaexample.config;

**import** java.util.Properties;

**import** javax.persistence.EntityManagerFactory;

**import** javax.sql.DataSource;

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

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.annotation.PropertySource;

**import** org.springframework.core.env.Environment;

**import** org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor;

**import** org.springframework.data.jpa.repository.config.EnableJpaRepositories;

**import** org.springframework.jdbc.datasource.DriverManagerDataSource;

**import** org.springframework.orm.jpa.JpaTransactionManager;

**import** org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

**import** org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;

**import** org.springframework.transaction.PlatformTransactionManager;

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

@Configuration

@EnableTransactionManagement

@PropertySource({

"classpath:database.properties"

})

@ComponentScan({

"com.hexaware.springmvcdatajpaexample"

})

@EnableJpaRepositories(basePackages = "com.hexaware.springmvcdatajpaexample.repository")

**public** **class** PersistenceJPAConfig {

@Autowired

**private** Environment env;

**public** PersistenceJPAConfig() {

**super**();

}

@Bean

**public** LocalContainerEntityManagerFactoryBean entityManagerFactory() {

**final** LocalContainerEntityManagerFactoryBean entityManagerFactoryBean = **new** LocalContainerEntityManagerFactoryBean();

entityManagerFactoryBean.setDataSource(dataSource());

entityManagerFactoryBean.setPackagesToScan(**new** String[] {

"com.hexaware.springmvcdatajpaexample.entity"

});

**final** HibernateJpaVendorAdapter vendorAdapter = **new** HibernateJpaVendorAdapter();

entityManagerFactoryBean.setJpaVendorAdapter(vendorAdapter);

entityManagerFactoryBean.setJpaProperties(additionalProperties());

**return** entityManagerFactoryBean;

}

**final** Properties additionalProperties() {

**final** Properties hibernateProperties = **new** Properties();

hibernateProperties.setProperty("hibernate.hbm2ddl.auto", env.getProperty("hibernate.hbm2ddl.auto"));

hibernateProperties.setProperty("hibernate.dialect", env.getProperty("hibernate.dialect"));

hibernateProperties.setProperty("hibernate.cache.use\_second\_level\_cache", env.getProperty("hibernate.cache.use\_second\_level\_cache"));

hibernateProperties.setProperty("hibernate.cache.use\_query\_cache", env.getProperty("hibernate.cache.use\_query\_cache"));

// hibernateProperties.setProperty("hibernate.globally\_quoted\_identifiers", "true");

**return** hibernateProperties;

}

@Bean

**public** DataSource dataSource() {

**final** DriverManagerDataSource dataSource = **new** DriverManagerDataSource();

dataSource.setDriverClassName(env.getProperty("jdbc.driverClassName"));

dataSource.setUrl(env.getProperty("jdbc.url"));

dataSource.setUsername(env.getProperty("jdbc.user"));

dataSource.setPassword(env.getProperty("jdbc.pass"));

**return** dataSource;

}

@Bean

**public** PlatformTransactionManager transactionManager(**final** EntityManagerFactory emf) {

**final** JpaTransactionManager transactionManager = **new** JpaTransactionManager();

transactionManager.setEntityManagerFactory(emf);

**return** transactionManager;

}

@Bean

**public** PersistenceExceptionTranslationPostProcessor exceptionTranslation() {

**return** **new** PersistenceExceptionTranslationPostProcessor();

}

}

**WebMvcConfig - Spring MVC Bean Configuration using Java-based Spring configuration**

Create an MVCConfig class and annotated with *@Configuration*, *@EnableWebMvc*, and *@ComponentScan* annotations.

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

**import** org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

**import** org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

**import** org.springframework.web.servlet.view.InternalResourceViewResolver;

**import** org.springframework.web.servlet.view.JstlView;

@Configuration

@EnableWebMvc

@ComponentScan(basePackages = {

"com.hexaware.springmvcdatajpaexample.controller"

})

**public** **class** WebMvcConfig **implements** WebMvcConfigurer {

@Bean

**public** InternalResourceViewResolver resolver() {

InternalResourceViewResolver resolver = **new** InternalResourceViewResolver();

resolver.setViewClass(JstlView.**class**);

resolver.setPrefix("/WEB-INF/views/");

resolver.setSuffix(".jsp");

**return** resolver;

}

@Override

**public** **void** addResourceHandlers(ResourceHandlerRegistry registry) {

registry

.addResourceHandler("/resources/\*\*")

.addResourceLocations("/resources/");

}

}

## JPA Entity - Customer.java

**package** com.hexaware.springmvcdatajpaexample.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\_list")

**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

**package** com.hexaware.springmvcdatajpaexample.controller;

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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.springmvcdatajpaexample.entity.Customer;

**import** com.hexaware.springmvcdatajpaexample.exception.ResourceNotFoundException;

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

@Controller

@RequestMapping("/customer")

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

**private** **static** **final** Logger ***LOG*** = LoggerFactory.*getLogger*(CustomerController.**class**);

@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) {

***LOG***.debug("inside show customer-form handler method");

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) **throws** ResourceNotFoundException {

Customer theCustomer = customerService.getCustomer(theId);

theModel.addAttribute("customer", theCustomer);

**return** "customer-form";

}

@GetMapping("/delete")

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

customerService.deleteCustomer(theId);

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

}

}

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

## CustomerService.java

Let's create CustomerService class with Customer service operations.

**package** com.hexaware.springmvcdatajpaexample.service;

**import** java.util.List;

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

**import** com.hexaware.springmvcdatajpaexample.exception.ResourceNotFoundException;

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

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

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

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

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

}

**CustomerServiceImpl.java**

**package** com.hexaware.springmvcdatajpaexample.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.springmvcdatajpaexample.entity.Customer;

**import** com.hexaware.springmvcdatajpaexample.exception.ResourceNotFoundException;

**import** com.hexaware.springmvcdatajpaexample.repository.CustomerRepository;

@Service

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

@Autowired

**private** CustomerRepository customerRepository;

@Override

@Transactional

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

**return** customerRepository.findAll();

}

@Override

@Transactional

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

customerRepository.save(theCustomer);

}

@Override

@Transactional

**public** Customer getCustomer(**int** id) **throws** ResourceNotFoundException {

**return** customerRepository.findById(id).orElseThrow(

() - > **new** ResourceNotFoundException(id));

}

@Override

@Transactional

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

customerRepository.deleteById(theId);

}

}

## Spring Data JPA Repository - CustomerRepository.java

**package** com.hexaware.springmvcdatajpaexample.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

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

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

@Repository("customerRepository")

**public** **interface** CustomerRepository **extends** JpaRepository<Customer, Integer> {

}

## Custom Exception - ResourceNotFoundException.java

**package** com.hexaware.springmvcdatajpaexample.exception;

**public** **class** ResourceNotFoundException **extends** Exception {

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

**public** ResourceNotFoundException(Object resourId) {

**super**(resourId != **null** ? resourId.toString() : **null**);

} }

## 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>Customer Management System</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"*>

<h3 class=*"text-center"*>Customer Management System</h3>

<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 http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Customer Management System</title>

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

rel=*"stylesheet"*>

<%@ page isELIgnored=*"false"*%>

<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"*>

<h3 class=*"text-center"*>Customer Management System</h3>

<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>

<div class=*"footer"*>

<p>Footer</p>

</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 js is depends on jquery js.

## Build and Run an application

1. As we are using maven build tool so first, we will need to build this application using following maven command:
2. clean install
3. Once build success, then we will run this application on tomcat server 8.5 in IDE or we can also deploy war file on external tomcat webapps folder and run the application.