|  |
| --- |
| **Spring MVC CRUD Example** CRUD (Create, Read, Update and Delete) application is the most important application  for creating any project. It provides an idea to develop a large project. In spring MVC,  we can develop a simple CRUD application.  Here, we are using **JdbcTemplate** for database interaction. Create a table Here, we are using emp99 table present in the MySQL database. It has 4 fields: id, name, salary,  and designation. Here, id is auto incremented which is generated by the sequence.  Spring MVC CRUD Example Spring MVC CRUD Example1. Add dependencies to pom.xml file. **pom.xml**   1. <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc --> 2. **<dependency>** 3. **<groupId>**org.springframework**</groupId>** 4. **<artifactId>**spring-webmvc**</artifactId>** 5. **<version>**5.1.1.RELEASE**</version>** 6. **</dependency>** 8. <!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-jasper --> 9. **<dependency>** 10. **<groupId>**org.apache.tomcat**</groupId>** 11. **<artifactId>**tomcat-jasper**</artifactId>** 12. **<version>**9.0.12**</version>** 13. **</dependency>** 14. <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api --> 15. **<dependency>** 16. **<groupId>**javax.servlet**</groupId>** 17. **<artifactId>**servlet-api**</artifactId>** 18. **<version>**3.0-alpha-1**</version>** 19. **</dependency>** 20. <!-- https://mvnrepository.com/artifact/javax.servlet/jstl --> 21. **<dependency>** 22. **<groupId>**javax.servlet**</groupId>** 23. **<artifactId>**jstl**</artifactId>** 24. **<version>**1.2**</version>** 25. **</dependency>** 26. <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java --> 27. **<dependency>** 28. **<groupId>**mysql**</groupId>** 29. **<artifactId>**mysql-connector-java**</artifactId>** 30. **<version>**8.0.11**</version>** 31. **</dependency>** 32. <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc --> 33. **<dependency>** 34. **<groupId>**org.springframework**</groupId>** 35. **<artifactId>**spring-jdbc**</artifactId>** 36. **<version>**5.1.1.RELEASE**</version>** 37. **</dependency>**  2. Create the bean class Here, the bean class contains the variables (along setter and getter methods) corresponding  to the fields exist in the database.  **Emp.java**   1. **package** com.javatpoint.beans; 3. **public** **class** Emp { 4. **private** **int** id; 5. **private** String name; 6. **private** **float** salary; 7. **private** String designation; 9. **public** **int** getId() { 10. **return** id; 11. } 12. **public** **void** setId(**int** id) { 13. **this**.id = id; 14. } 15. **public** String getName() { 16. **return** name; 17. } 18. **public** **void** setName(String name) { 19. **this**.name = name; 20. } 21. **public** **float** getSalary() { 22. **return** salary; 23. } 24. **public** **void** setSalary(**float** salary) { 25. **this**.salary = salary; 26. } 27. **public** String getDesignation() { 28. **return** designation; 29. } 30. **public** **void** setDesignation(String designation) { 31. **this**.designation = designation; 32. } 34. }  3. Create the controller class **EmpController.java**   1. **package** com.javatpoint.controllers; 2. **import** java.util.List; 3. **import** org.springframework.beans.factory.annotation.Autowired; 4. **import** org.springframework.stereotype.Controller; 5. **import** org.springframework.ui.Model; 6. **import** org.springframework.web.bind.annotation.ModelAttribute; 7. **import** org.springframework.web.bind.annotation.PathVariable; 8. **import** org.springframework.web.bind.annotation.RequestMapping; 9. **import** org.springframework.web.bind.annotation.RequestMethod; 10. **import** com.javatpoint.beans.Emp; 11. **import** com.javatpoint.dao.EmpDao; 12. @Controller 13. **public** **class** EmpController { 14. @Autowired 15. EmpDao dao;//will inject dao from XML file 17. /\*It displays a form to input data, here "command" is a reserved request attribute 18. \*which is used to display object data into form 19. \*/ 20. @RequestMapping("/empform") 21. **public** String showform(Model m){ 22. m.addAttribute("command", **new** Emp()); 23. **return** "empform"; 24. } 25. /\*It saves object into database. The @ModelAttribute puts request data 26. \*  into model object. You need to mention RequestMethod.POST method 27. \*  because default request is GET\*/ 28. @RequestMapping(value="/save",method = RequestMethod.POST) 29. **public** String save(@ModelAttribute("emp") Emp emp){ 30. dao.save(emp); 31. **return** "redirect:/viewemp";//will redirect to viewemp request mapping 32. } 33. /\* It provides list of employees in model object \*/ 34. @RequestMapping("/viewemp") 35. **public** String viewemp(Model m){ 36. List<Emp> list=dao.getEmployees(); 37. m.addAttribute("list",list); 38. **return** "viewemp"; 39. } 40. /\* It displays object data into form for the given id. 41. \* The @PathVariable puts URL data into variable.\*/ 42. @RequestMapping(value="/editemp/{id}") 43. **public** String edit(@PathVariable **int** id, Model m){ 44. Emp emp=dao.getEmpById(id); 45. m.addAttribute("command",emp); 46. **return** "empeditform"; 47. } 48. /\* It updates model object. \*/ 49. @RequestMapping(value="/editsave",method = RequestMethod.POST) 50. **public** String editsave(@ModelAttribute("emp") Emp emp){ 51. dao.update(emp); 52. **return** "redirect:/viewemp"; 53. } 54. /\* It deletes record for the given id in URL and redirects to /viewemp \*/ 55. @RequestMapping(value="/deleteemp/{id}",method = RequestMethod.GET) 56. **public** String delete(@PathVariable **int** id){ 57. dao.delete(id); 58. **return** "redirect:/viewemp"; 59. } 60. }  4. Create the DAO class Let's create a DAO class to access the required data from the database.  **EmpDao.java**   1. **package** com.javatpoint.dao; 2. **import** java.sql.ResultSet; 3. **import** java.sql.SQLException; 4. **import** java.util.List; 5. **import** org.springframework.jdbc.core.BeanPropertyRowMapper; 6. **import** org.springframework.jdbc.core.JdbcTemplate; 7. **import** org.springframework.jdbc.core.RowMapper; 8. **import** com.javatpoint.beans.Emp; 10. **public** **class** EmpDao { 11. JdbcTemplate template; 13. **public** **void** setTemplate(JdbcTemplate template) { 14. **this**.template = template; 15. } 16. **public** **int** save(Emp p){ 17. String sql="insert into Emp99(name,salary,designation) values('"+p.getName()+"'," 18. +p.getSalary()+",'"+p.getDesignation()+"')"; 19. **return** template.update(sql); 20. } 21. **public** **int** update(Emp p){ 22. String sql="update Emp99 set name='"+p.getName()+"', salary="+p.getSalary()+",designation='" 23. +p.getDesignation()+"' where id="+p.getId()+""; 24. **return** template.update(sql); 25. } 26. **public** **int** delete(**int** id){ 27. String sql="delete from Emp99 where id="+id+""; 28. **return** template.update(sql); 29. } 30. **public** Emp getEmpById(**int** id){ 31. String sql="select \* from Emp99 where id=?"; 32. **return** template.queryForObject(sql, **new** Object[]{id},**new** 33. BeanPropertyRowMapper<Emp>(Emp.**class**)); 34. } 35. **public** List<Emp> getEmployees(){ 36. **return** template.query("select \* from Emp99",**new** RowMapper<Emp>(){ 37. **public** Emp mapRow(ResultSet rs, **int** row) **throws** SQLException { 38. Emp e=**new** Emp(); 39. e.setId(rs.getInt(1)); 40. e.setName(rs.getString(2)); 41. e.setSalary(rs.getFloat(3)); 42. e.setDesignation(rs.getString(4)); 43. **return** e; 44. } 45. }); 46. } 47. }  5. Provide the entry of controller in the web.xml file **web.xml**   1. **<?xml** version="1.0" encoding="UTF-8"**?>** 2. **<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/ 3. xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" id="WebApp\_ID" version="3.0"**>** 4. **<display-name>**SpringMVC**</display-name>** 5. **<servlet>** 6. **<servlet-name>**spring**</servlet-name>** 7. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>** 8. **<load-on-startup>**1**</load-on-startup>** 9. **</servlet>** 10. **<servlet-mapping>** 11. **<servlet-name>**spring**</servlet-name>** 12. **<url-pattern>**/**</url-pattern>** 13. **</servlet-mapping>** 14. **</web-app>**  6. Define the bean in the xml file **spring-servlet.xml**   1. **<?xml** version="1.0" encoding="UTF-8"**?>** 2. **<beans** xmlns="http://www.springframework.org/schema/beans" 3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 4. xmlns:context="http://www.springframework.org/schema/context" 5. xmlns:mvc="http://www.springframework.org/schema/mvc" 6. xsi:schemaLocation=" 7. http://www.springframework.org/schema/beans 8. http://www.springframework.org/schema/beans/spring-beans.xsd 9. http://www.springframework.org/schema/context 10. http://www.springframework.org/schema/context/spring-context.xsd 11. http://www.springframework.org/schema/mvc 12. http://www.springframework.org/schema/mvc/spring-mvc.xsd"**>** 13. **<context:component-scan** base-package="com.javatpoint.controllers"**></context:component-scan>** 15. **<bean** class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>** 16. **<property** name="prefix" value="/WEB-INF/jsp/"**></property>** 17. **<property** name="suffix" value=".jsp"**></property>** 18. **</bean>** 20. **<bean** id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource"**>** 21. **<property** name="driverClassName" value="com.mysql.jdbc.Driver"**></property>** 22. **<property** name="url" value="jdbc:mysql://localhost:3306/test"**></property>** 23. **<property** name="username" value=""**></property>** 24. **<property** name="password" value=""**></property>** 25. **</bean>** 27. **<bean** id="jt" class="org.springframework.jdbc.core.JdbcTemplate"**>** 28. **<property** name="dataSource" ref="ds"**></property>** 29. **</bean>** 31. **<bean** id="dao" class="com.javatpoint.dao.EmpDao"**>** 32. **<property** name="template" ref="jt"**></property>** 33. **</bean>** 34. **</beans>**  7. Create the requested page **index.jsp**   1. **<a** href="empform"**>**Add Employee**</a>** 2. **<a** href="viewemp"**>**View Employees**</a>**  8. Create the other view components **empform.jsp**   1. **<**%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%**>** 2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%**>** 4. **<h1>**Add New Employee**</h1>** 5. **<form:form** method="post" action="save"**>** 6. **<table** **>** 7. **<tr>** 8. **<td>**Name : **</td>** 9. **<td><form:input** path="name"  **/></td>** 10. **</tr>** 11. **<tr>** 12. **<td>**Salary :**</td>** 13. **<td><form:input** path="salary" **/></td>** 14. **</tr>** 15. **<tr>** 16. **<td>**Designation :**</td>** 17. **<td><form:input** path="designation" **/></td>** 18. **</tr>** 19. **<tr>** 20. **<td>** **</td>** 21. **<td><input** type="submit" value="Save" **/></td>** 22. **</tr>** 23. **</table>** 24. **</form:form>**   **empeditform.jsp**  Here "/SpringMVCCRUDSimple" is the project name, change this if you have different project name. For live application, you can provide full URL.   1. **<**%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%**>** 2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%**>** 4. **<h1>**Edit Employee**</h1>** 5. **<form:form** method="POST" action="/SpringMVCCRUDSimple/editsave"**>** 6. **<table** **>** 7. **<tr>** 8. **<td></td>** 9. **<td><form:hidden**  path="id" **/></td>** 10. **</tr>** 11. **<tr>** 12. **<td>**Name : **</td>** 13. **<td><form:input** path="name"  **/></td>** 14. **</tr>** 15. **<tr>** 16. **<td>**Salary :**</td>** 17. **<td><form:input** path="salary" **/></td>** 18. **</tr>** 19. **<tr>** 20. **<td>**Designation :**</td>** 21. **<td><form:input** path="designation" **/></td>** 22. **</tr>** 24. **<tr>** 25. **<td>** **</td>** 26. **<td><input** type="submit" value="Edit Save" **/></td>** 27. **</tr>** 28. **</table>** 29. **</form:form>**   **viewemp.jsp**   1. **<**%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%**>** 2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%**>** 4. **<h1>**Employees List**</h1>** 5. **<table** border="2" width="70%" cellpadding="2"**>** 6. **<tr><th>**Id**</th><th>**Name**</th><th>**Salary**</th><th>**Designation**</th>** 7. **<th>**Edit**</th><th>**Delete**</th></tr>** 8. **<c:forEach** var="emp" items="${list}"**>** 9. **<tr>** 10. **<td>**${emp.id}**</td>** 11. **<td>**${emp.name}**</td>** 12. **<td>**${emp.salary}**</td>** 13. **<td>**${emp.designation}**</td>** 14. **<td><a** href="editemp/${emp.id}"**>**Edit**</a></td>** 15. **<td><a** href="deleteemp/${emp.id}"**>**Delete**</a></td>** 16. **</tr>** 17. **</c:forEach>** 18. **</table>** 19. **<br/>** 20. **<a** href="empform"**>**Add New Employee**</a>**   **Output:**  Spring MVC CRUD Example  On clicking **Add Employee**, you will see the following form.  Spring MVC CRUD Example  Fill the form and **click Save** to add the entry into the database.  Spring MVC CRUD Example  Now, click **Edit** to make some changes in the provided data.  Spring MVC CRUD Example  Now, click **Edit Save** to add the entry with changes into the database.  Spring MVC CRUD Example  Now, click **Delete** to delete the entry from the database.  Spring MVC CRUD Example  [Download this example (developed using Eclipse)](https://static.javatpoint.com/sppages/download/SpringMVCCRUDSimple.zip) |