# Spring JdbcTemplate Tutorial

We are assuming that you have created the following table

create table employee(

id number(10),

name varchar2(100),

salary number(10)

);

**Employee.java**

This class contains 3 properties with constructors and setter and getters.

**package** com.javatpoint;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** **float** salary;

//no-arg and parameterized constructors

//getters and setters

}

**EmployeeDao.java**

It contains one property jdbcTemplate and three methods saveEmployee(), updateEmployee and deleteEmployee().

**package** com.javatpoint;

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

**public** **class** EmployeeDao {

**private** JdbcTemplate jdbcTemplate;

**public** **void** setJdbcTemplate(JdbcTemplate jdbcTemplate) {

**this**.jdbcTemplate = jdbcTemplate;

}

**public** **int** saveEmployee(Employee e){

    String query="insert into employee values(

    '"+e.getId()+"','"+e.getName()+"','"+e.getSalary()+"')";

**return** jdbcTemplate.update(query);

}

**public** **int** updateEmployee(Employee e){

    String query="update employee set

    name='"+e.getName()+"',salary='"+e.getSalary()+"' where id='"+e.getId()+"' ";

**return** jdbcTemplate.update(query);

}

**public** **int** deleteEmployee(Employee e){

    String query="delete from employee where id='"+e.getId()+"' ";

**return** jdbcTemplate.update(query);

}

}

**applicationContext.xml**

The **DriverManagerDataSource** is used to contain the information about the database such as driver class name, connnection URL, username and password.

Here, we are using the JdbcTemplate object in the EmployeeDao class, so we are passing it by the setter method but you can use constructor also.

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

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

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

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

<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />

<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />

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

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

</bean>

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

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

</bean>

<bean id="edao" **class**="com.javatpoint.EmployeeDao">

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

</bean>

</beans>

**Test.java**

This class gets the bean from the applicationContext.xml file and calls the saveEmployee() method. You can also call updateEmployee() and deleteEmployee() method by uncommenting the code as well.

**package** com.javatpoint;

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

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

**public** **class** Test {

**public** **static** **void** main(String[] args) {

    ApplicationContext ctx=**new** ClassPathXmlApplicationContext("applicationContext.xml");

    EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");

**int** status=dao.saveEmployee(**new** Employee(102,"Amit",35000));

    System.out.println(status);

    /\*int status=dao.updateEmployee(new Employee(102,"Sonoo",15000));

    System.out.println(status);

    \*/

    /\*Employee e=new Employee();

    e.setId(102);

    int status=dao.deleteEmployee(e);

    System.out.println(status);\*/

}

}

# Hibernate and Spring Integration

**Create the table in the database**

CREATE TABLE  "EMP558"

   (    "ID" NUMBER(10,0) NOT NULL ENABLE,

    "NAME" VARCHAR2(255 CHAR),

    "SALARY" FLOAT(126),

     PRIMARY KEY ("ID") ENABLE

   )

**Employee.java**

It is a simple POJO class. Here it works as the persistent class for hibernate.

**package** com.javatpoint;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** **float** salary;

//getters and setters

}

**employee.hbm.xml**

This mapping file contains all the information of the persistent class.

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

<!DOCTYPE hibernate-mapping PUBLIC

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

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<**class** name="com.javatpoint.Employee" table="emp558">

          <id name="id">

          <generator **class**="assigned"></generator>

          </id>

          <property name="name"></property>

          <property name="salary"></property>

</**class**>

</hibernate-mapping>

**EmployeeDao.java**

It is a java class that uses the **HibernateTemplate** class method to persist the object of Employee class.

**package** com.javatpoint;

**import** org.springframework.orm.hibernate3.HibernateTemplate;

**import** java.util.\*;

**public** **class** EmployeeDao {

HibernateTemplate template;

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

**this**.template = template;

}

//method to save employee

**public** **void** saveEmployee(Employee e){

    template.save(e);

}

//method to update employee

**public** **void** updateEmployee(Employee e){

    template.update(e);

}

//method to delete employee

**public** **void** deleteEmployee(Employee e){

    template.delete(e);

}

//method to return one employee of given id

**public** Employee getById(**int** id){

    Employee e=(Employee)template.get(Employee.**class**,id);

**return** e;

}

//method to return all employees

**public** List<Employee> getEmployees(){

    List<Employee> list=**new** ArrayList<Employee>();

    list=template.loadAll(Employee.**class**);

**return** list;

}

}

**applicationContext.xml**

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

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

        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="dataSource" **class**="org.apache.commons.dbcp.BasicDataSource">

        <property name="driverClassName"  value="oracle.jdbc.driver.OracleDriver"></property>

        <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"></property>

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

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

    </bean>

    <bean id="mysessionFactory"  **class**="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

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

        <property name="mappingResources">

        <list>

        <value>employee.hbm.xml</value>

        </list>

        </property>

        <property name="hibernateProperties">

            <props>

                <prop key="hibernate.dialect">org.hibernate.dialect.Oracle9Dialect</prop>

                <prop key="hibernate.hbm2ddl.auto">update</prop>

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

            </props>

        </property>

    </bean>

    <bean id="template" **class**="org.springframework.orm.hibernate3.HibernateTemplate">

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

    </bean>

    <bean id="d" **class**="com.javatpoint.EmployeeDao">

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

    </bean>

    </beans>

**InsertTest.java**

**package** com.javatpoint;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.ClassPathResource;

**import** org.springframework.core.io.Resource;

**public** **class** InsertTest {

**public** **static** **void** main(String[] args) {

    Resource r=**new** ClassPathResource("applicationContext.xml");

    BeanFactory factory=**new** XmlBeanFactory(r);

    EmployeeDao dao=(EmployeeDao)factory.getBean("d");

    Employee e=**new** Employee();

    e.setId(114);

    e.setName("varun");

    e.setSalary(50000);

    dao.saveEmployee(e);

}

}

Spring MVC CRUD Example

### **Add dependencies to pom.xml file**

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

**<dependency>**

**<groupId>**org.springframework**</groupId>**

**<artifactId>**spring-webmvc**</artifactId>**

**<version>**5.1.1.RELEASE**</version>**

**</dependency>**

<!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-jasper -->

**<dependency>**

**<groupId>**org.apache.tomcat**</groupId>**

**<artifactId>**tomcat-jasper**</artifactId>**

**<version>**9.0.12**</version>**

**</dependency>**

    <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

**<dependency>**

**<groupId>**javax.servlet**</groupId>**

**<artifactId>**servlet-api**</artifactId>**

**<version>**3.0-alpha-1**</version>**

**</dependency>**

<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->

**<dependency>**

**<groupId>**javax.servlet**</groupId>**

**<artifactId>**jstl**</artifactId>**

**<version>**1.2**</version>**

**</dependency>**

    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->

**<dependency>**

**<groupId>**mysql**</groupId>**

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

**<version>**8.0.11**</version>**

**</dependency>**

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

**<dependency>**

**<groupId>**org.springframework**</groupId>**

**<artifactId>**spring-jdbc**</artifactId>**

**<version>**5.1.1.RELEASE**</version>**

**</dependency>**

### **Create the bean class**

**Emp.java**

**package** com.javatpoint.beans;

**public** **class** Emp {

**private** **int** id;

**private** String name;

**private** **float** salary;

**private** String designation;

**public** **int** getId() {

**return** id;

}

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

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

**public** String getDesignation() {

**return** designation;

}

**public** **void** setDesignation(String designation) {

**this**.designation = designation;

}

}

### **Create the controller class**

**EmpController.java**

**package** com.javatpoint.controllers;

**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.ModelAttribute;

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

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

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

**import** com.javatpoint.beans.Emp;

**import** com.javatpoint.dao.EmpDao;

@Controller

**public** **class** EmpController {

    @Autowired

    EmpDao dao;//will inject dao from XML file

    /\*It displays a form to input data, here "command" is a reserved request attribute

     \*which is used to display object data into form

     \*/

    @RequestMapping("/empform")

**public** String showform(Model m){

        m.addAttribute("command", **new** Emp());

**return** "empform";

    }

    /\*It saves object into database. The @ModelAttribute puts request data

     \*  into model object. You need to mention RequestMethod.POST method

     \*  because default request is GET\*/

    @RequestMapping(value="/save",method = RequestMethod.POST)

**public** String save(@ModelAttribute("emp") Emp emp){

        dao.save(emp);

**return** "redirect:/viewemp";//will redirect to viewemp request mapping

    }

    /\* It provides list of employees in model object \*/

    @RequestMapping("/viewemp")

**public** String viewemp(Model m){

        List<Emp> list=dao.getEmployees();

        m.addAttribute("list",list);

**return** "viewemp";

    }

    /\* It displays object data into form for the given id.

     \* The @PathVariable puts URL data into variable.\*/

    @RequestMapping(value="/editemp/{id}")

**public** String edit(@PathVariable **int** id, Model m){

        Emp emp=dao.getEmpById(id);

        m.addAttribute("command",emp);

**return** "empeditform";

    }

    /\* It updates model object. \*/

    @RequestMapping(value="/editsave",method = RequestMethod.POST)

**public** String editsave(@ModelAttribute("emp") Emp emp){

        dao.update(emp);

**return** "redirect:/viewemp";

    }

    /\* It deletes record for the given id in URL and redirects to /viewemp \*/

    @RequestMapping(value="/deleteemp/{id}",method = RequestMethod.GET)

**public** String delete(@PathVariable **int** id){

        dao.delete(id);

**return** "redirect:/viewemp";

    }

}

### **Create the DAO class**

Let's create a DAO class to access the required data from the database.

**EmpDao.java**

**package** com.javatpoint.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.javatpoint.beans.Emp;

**public** **class** EmpDao {

JdbcTemplate template;

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

**this**.template = template;

}

**public** **int** save(Emp p){

    String sql="insert into Emp99(name,salary,designation) values('"+p.getName()+"',"+p.getSalary()+",'"+p.getDesignation()+"')";

**return** template.update(sql);

}

**public** **int** update(Emp p){

    String sql="update Emp99 set name='"+p.getName()+"', salary="+p.getSalary()+",designation='"+p.getDesignation()+"' where id="+p.getId()+"";

**return** template.update(sql);

}

**public** **int** delete(**int** id){

    String sql="delete from Emp99 where id="+id+"";

**return** template.update(sql);

}

**public** Emp getEmpById(**int** id){

    String sql="select \* from Emp99 where id=?";

**return** template.queryForObject(sql, **new** Object[]{id},**new** BeanPropertyRowMapper<Emp>(Emp.**class**));

}

**public** List<Emp> getEmployees(){

**return** template.query("select \* from Emp99",**new** RowMapper<Emp>(){

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

            Emp e=**new** Emp();

            e.setId(rs.getInt(1));

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

            e.setSalary(rs.getFloat(3));

            e.setDesignation(rs.getString(4));

**return** e;

        }

    });

}

}

### **Provide the entry of controller in the web.xml file**

**web.xml**

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

**<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/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" id="WebApp\_ID" version="3.0"**>**

**<display-name>**SpringMVC**</display-name>**

**<servlet>**

**<servlet-name>**spring**</servlet-name>**

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

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

**</servlet>**

**<servlet-mapping>**

**<servlet-name>**spring**</servlet-name>**

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

**</servlet-mapping>**

**</web-app>**

### **Define the bean in the xml file**

**spring-servlet.xml**

**<?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: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"**>**

**<context:component-scan** base-package="com.javatpoint.controllers"**></context:component-scan>**

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

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

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

**</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/test"**></property>**

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

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

**</bean>**

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

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

**</bean>**

**<bean** id="dao" class="com.javatpoint.dao.EmpDao"**>**

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

**</bean>**

**</beans>**

### **Create the requested page**

**index.jsp**

1. **<a** href="empform"**>**Add Employee**</a>**
2. **<a** href="viewemp"**>**View Employees**</a>**

### **Create the other view components**

**empform.jsp**

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

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

**<h1>**Add New Employee**</h1>**

**<form:form** method="post" action="save"**>**

**<table** **>**

**<tr>**

**<td>**Name : **</td>**

**<td><form:input** path="name"  **/></td>**

**</tr>**

**<tr>**

**<td>**Salary :**</td>**

**<td><form:input** path="salary" **/></td>**

**</tr>**

**<tr>**

**<td>**Designation :**</td>**

**<td><form:input** path="designation" **/></td>**

**</tr>**

**<tr>**

**<td>** **</td>**

**<td><input** type="submit" value="Save" **/></td>**

**</tr>**

**</table>**

**</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.

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

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

**<h1>**Edit Employee**</h1>**

**<form:form** method="POST" action="/SpringMVCCRUDSimple/editsave"**>**

**<table** **>**

**<tr>**

**<td></td>**

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

**</tr>**

**<tr>**

**<td>**Name : **</td>**

**<td><form:input** path="name"  **/></td>**

**</tr>**

**<tr>**

**<td>**Salary :**</td>**

**<td><form:input** path="salary" **/></td>**

**</tr>**

**<tr>**

**<td>**Designation :**</td>**

**<td><form:input** path="designation" **/></td>**

**</tr>**

**<tr>**

**<td>** **</td>**

**<td><input** type="submit" value="Edit Save" **/></td>**

**</tr>**

**</table>**

**</form:form>**

**viewemp.jsp**

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

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

**<h1>**Employees List**</h1>**

**<table** border="2" width="70%" cellpadding="2"**>**

**<tr><th>**Id**</th><th>**Name**</th><th>**Salary**</th><th>**Designation**</th><th>**Edit**</th><th>**Delete**</th></tr>**

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

**<tr>**

**<td>**${emp.id}**</td>**

**<td>**${emp.name}**</td>**

**<td>**${emp.salary}**</td>**

**<td>**${emp.designation}**</td>**

**<td><a** href="editemp/${emp.id}"**>**Edit**</a></td>**

**<td><a** href="deleteemp/${emp.id}"**>**Delete**</a></td>**

**</tr>**

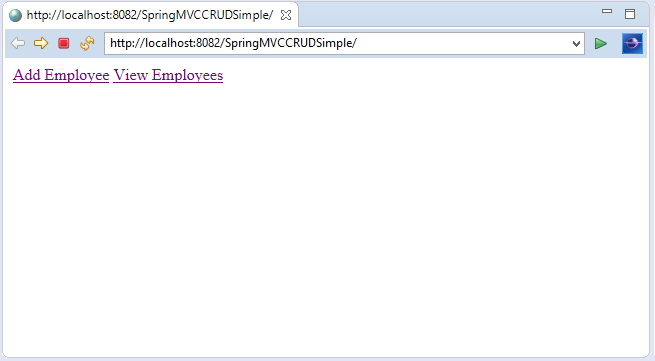
**</c:forEach>**

**</table>**

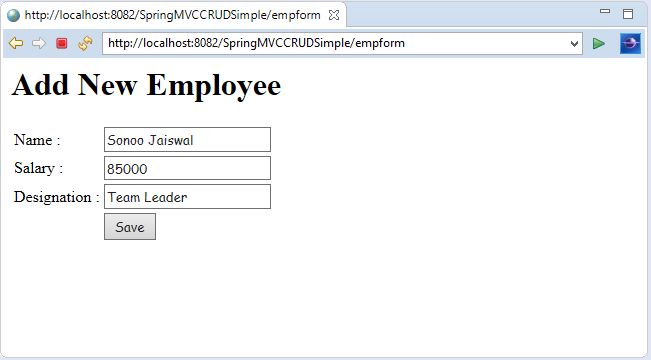
**<br/>**

**<a** href="empform"**>**Add New Employee**</a>**

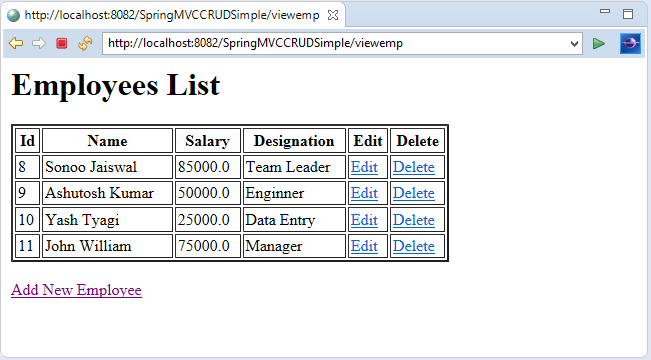
**Output:**



On clicking **Add Employee**, you will see the following form.



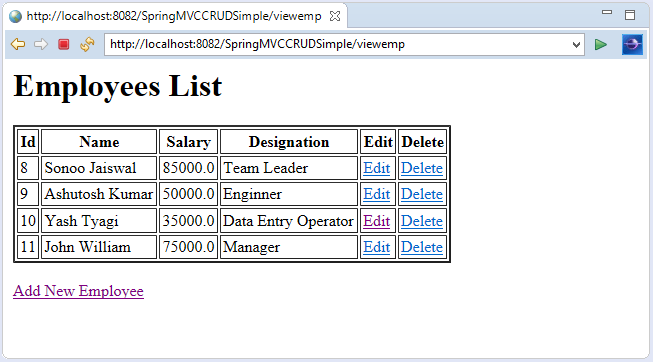
Fill the form and **click Save** to add the entry into the database.



Now, click **Edit** to make some changes in the provided data.



Now, click **Edit Save** to add the entry with changes into the database.



Now, click **Delete** to delete the entry from the database.

