

Spring Assignment

Prajwal_77

Q1. Implement Spring mvc

STEPS TO IMPLEMENT SPRING MVC

Create a *Dynamic Web Project* with a name **SPR-MVC-HELLO-APP** and create a package **com.prajwal** under the src folder in the created project.

POM.XML-

```
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.prajwal</groupId>
    <artifactId>spr-mvc-hello-app</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>SPR-MVC-CRUD-APP</name>
    <description>Spring project</description>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <release>1.8</release>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.3</version>
            </plugin>
        </plugins>
    </build>

    <properties>
        <spring.version>4.3.20.RELEASE</spring.version>
        <aspectj.version>1.8.13</aspectj.version>
        <mysql.connector.version>8.0.13</mysql.connector.version>
        <hibernate.version>4.3.0.Final</hibernate.version>
    </properties>
</project>
```

```
</properties>
<dependencies>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.version}</version>
```

```

</dependency>
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>${aspectj.version}</version>
</dependency>

<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>${aspectj.version}</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>${mysql.connector.version}</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>${hibernate.version}</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>${hibernate.version}</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-ehcache</artifactId>
    <version>${hibernate.version}</version>
</dependency>
</dependencies>

</project>

```

2.CREATE A CLASS CONTROLLER inside the cntr pacakage

package com.prajwal.cntr;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

```
@Controller
```

```
public class HelloController {
```

```
    @RequestMapping(value = { "/hello.htm" })
```

```
    public String sayHello(ModelMap model) {
```

```
        model.put("msg", "Hello Spring MVC !!!!!!!"); //
```

```
model data
```

```
        return "info"; // view name
```

```
    }
```

```
}
```

// http://localhost:8080/HibernateApp2/hello.htm.....in the url type this last hello.htm

Step3

Create below mentioned `info.jsp` into the folder `Webapp`

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Success Page</title>
</head>
<body>
<%=request.getAttribute("msg")%>
</body>
```

</html>

Step4

Create below mentioned **dispatcher-servlet.xml** into the folder Webapp/WEB-INF

And mention the details

```
<?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:jdbc="http://www.springframework.org/schema/jdbc"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd">

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

    <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewReso
lver" >
        <property name="prefix" value="/" ></property>
        <property name="suffix" value=".jsp" ></property>
    </bean>

</beans>
```

5.Web.xml-

```
<?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"
```

```

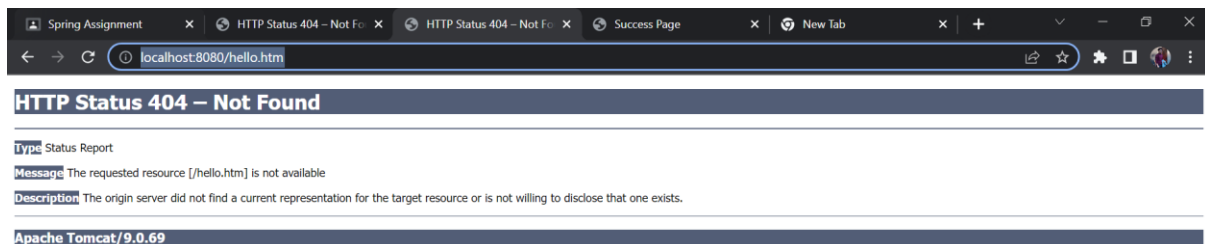
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
id="WebApp_ID" version="4.0">
<display-name>spr-mvc-hello-app</display-name>
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
</welcome-file-list>

<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
</servlet>
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>*.htm</url-pattern>
</servlet-mapping>

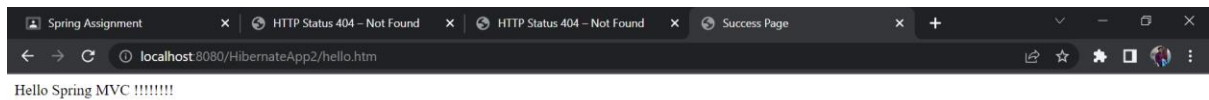
</web-app>

```

Final output-



**http://localhost:8080/hello.htm.....in the url type this last
hello.htm**



Q2. Implement CRUD through Spring boot application

Steps to make crud app in spring boot

1. [include dependency while making project] Dependencies-web, MySQL driver, jpa

POM.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.0</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.prajwal</groupId>
  <artifactId>SPRING-BOOT-CRUD-APP</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>SPRING-BOOT-CRUD-APP</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
```

```

                <artifactId>spring-boot-starter-data-
jpa</artifactId>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
            </dependency>

            <dependency>
                <groupId>com.mysql</groupId>
                <artifactId>mysql-connector-j</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
            </dependency>
        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-
plugin</artifactId>
                </plugin>
            </plugins>
        </build>
    </project>

```

2. enter jdbc/hibernate properties in application. properties file in resource folder

JDBC PROPERTIES

```

spring.datasource.url=jdbc:mysql://localhost:3306/java?useSSL=false&
allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=Prajwal@123

```

HIBERNATE PROPERTIES

```

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5
Dialect

```


spring.jpa.hibernate.ddl-auto=update

3. make an entity to manage its record

```
package com.prajwal.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int no;
    private String name;

    public int getNo() {
        return no;
    }

    public void setNo(int no) {
        this.no = no;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

4. create interface that extends CrudRepository or JpaRepository has predefined methods to perform CRUD operations on given entity.

```
package com.prajwal.dao;

import org.springframework.data.repository.CrudRepository;
```

```
import org.springframework.stereotype.Repository;

import com.prajwal.entity.Department;

@Repository
public interface DepartmentDao extends CrudRepository<Department,
Integer> {
```

5.create service class for entity

```
package com.prajwal.service;

import java.util.List;

import com.prajwal.entity.Department;

public interface DepartmentService {
    void add(Department department);

    void modify(Department department);

    void remove(int no);

    Department get(int no);

    List<Department> getAll();
```

```
package com.prajwal.service;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.prajwal.entity.Department;
import com.prajwal.dao.DepartmentDao;

@Service
public class DepartmentServiceImpl implements DepartmentService {
```

```

@Autowired
private DepartmentDao departmentDao;

@Override
public void add(Department department) {
    departmentDao.save(department);
}

@Override
public void modify(Department department) {
    departmentDao.save(department);
}

@Override
public void remove(int no) {
    departmentDao.deleteById(no);
}

@Override
public Department get(int no) {
    Optional<Department> opt = departmentDao.findById(no);
    Department department = opt.get();
    return department;
}

@Override
public List<Department> getAll() {
    Iterable<Department> itr = departmentDao.findAll();
    List<Department> lst = new ArrayList<Department>();
    itr.forEach(ele -> lst.add(ele));
    return lst;
}
}

```

6.create rest controller

```
package com.prajwal.cntr;
```

```
import java.util.List;
```

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

```
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.prajwal.entity.Department;
import com.prajwal.service.DepartmentService;
```

```
@RestController
```

```
public class DepartmentController {
```

```
    @Autowired
```

```
    private DepartmentService departmentService;
```

```
    @PostMapping(value = { "/dept" })
```

```
    public String departmentAdd(@RequestBody Department department) {
```

```
        departmentService.add(department);
```

```
        return "success";
```

```
    }
```

```
    @PutMapping(value = { "/dept" })
```

```
    public String departmentUpdate(@RequestBody Department department) {
```

```
        departmentService.modify(department);
```

```
        return "success";
```

```
    }
```

```
    @DeleteMapping(value = { "/dept/{no}" })
```

```
    public String departmentDelete(@PathVariable int no) {
```

```

        departmentService.remove(no);

        return "success";
    }
}

```

```

@GetMapping(value = { "/dept/{no}" })

public Department departmentSelect(@PathVariable int no) {

    return departmentService.get(no);

}

```

```

@GetMapping(value = { "/dept" })

public List<Department> departmentSelectAll() {

    return departmentService.getAll();

}

```

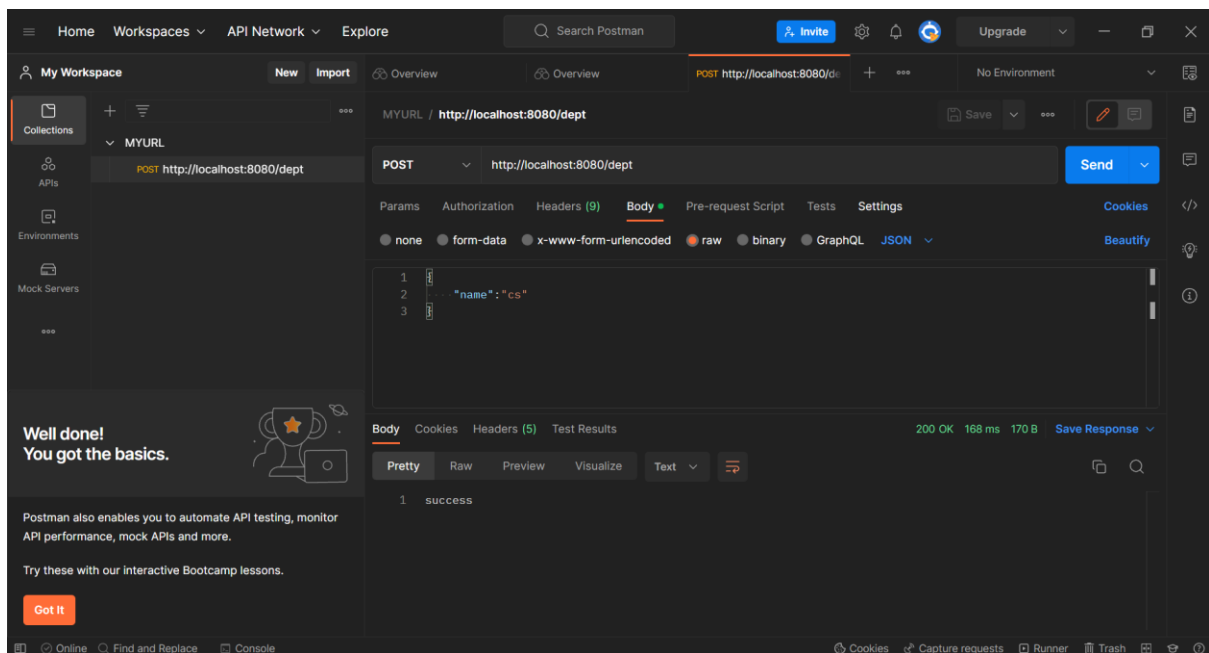
```

}

```

CRUD through Spring boot application -

INSERT RECORD -



```
mysql> select * from department;
+-----+-----+
| no | name |
+-----+-----+
| 1 | cs |
| 2 | it |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

UPDATE RECORD -

The screenshot displays the Postman API client interface. The left sidebar shows the 'My Workspace' section with a collection named 'MYURL' containing a 'POST http://localhost:8080/dept' request. The main panel shows a 'PUT http://localhost:8080/dept' request. The 'Body' tab is selected, showing a JSON payload:

```
{  "no": 2,  "name": "sport"}
```

. The 'Send' button is visible. Below the request, the response is shown in the 'Body' tab, indicating a '200 OK' status with a response time of 143 ms and a body size of 170 B. The response body is 'success'.

Well done!
You got the basics.

Postman also enables you to automate API testing, monitor API performance, mock APIs and more.

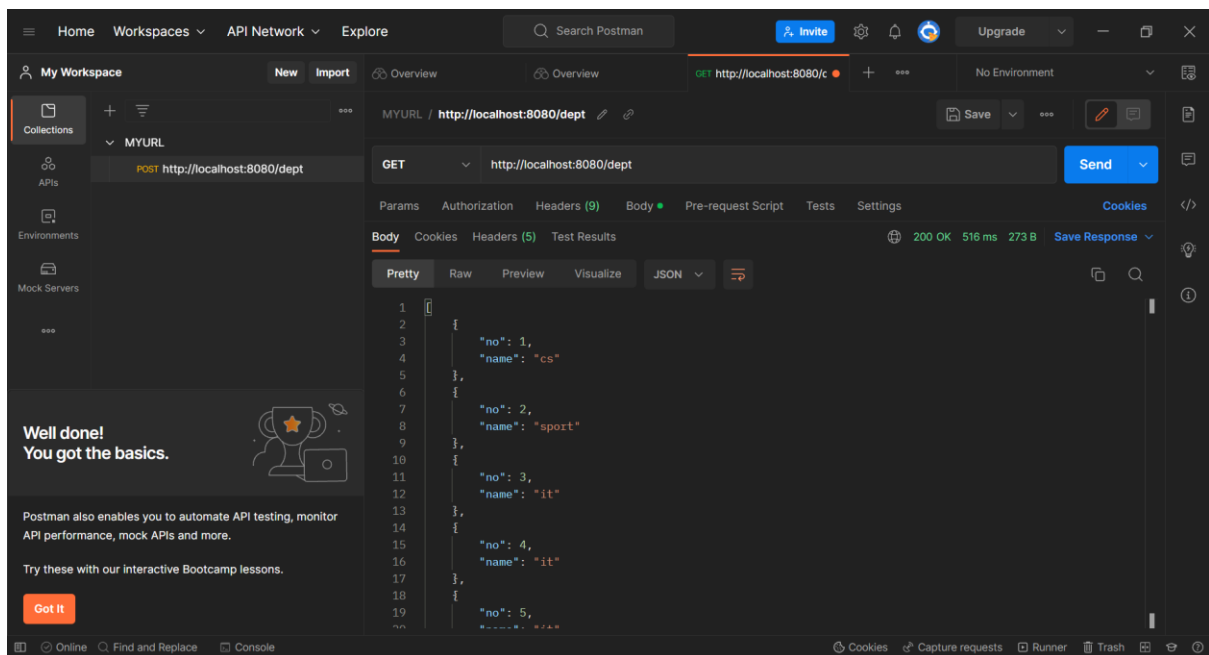
Try these with our interactive Bootcamp lessons.

[Got It](#)

```
mysql> select * from department;
+-----+-----+
| no | name |
+-----+-----+
| 1 | cs |
| 2 | sport |
| 3 | it |
| 4 | it |
| 5 | it |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

GET THE RECORD –



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/dept`. The response is a 200 OK status with a response time of 516 ms and a body size of 273 B. The response body is displayed in JSON format, showing an array of five department records.

```
[{"no": 1, "name": "cs"}, {"no": 2, "name": "sport"}, {"no": 3, "name": "it"}, {"no": 4, "name": "it"}, {"no": 5, "name": "it"}]
```

DELETE THE RECORD-

