Spring MVC - Sample Project For Finding Doctors Online with MySQL

Last Updated : 23 Jul, 2025

Spring MVC Framework follows the Model-View-Controller design pattern. It is used to develop web applications. It works around DispatcherServlet. DispatcherServlet handles all the HTTP requests and responses. With MySQL as the backend, we can store all doctor details and by using Spring MVC functionality we can get the details of doctors as the online pattern. Let us see it as a maven project here.

MySQL Queries:

As we are getting the details via MySQL, let us have some data for it

```
DROP DATABASE IF EXISTS geeksforgeeks;
CREATE DATABASE geeksforgeeks;
USE geeksforgeeks;

DROP TABLE geeksforgeeks.DoctorsDetails;

CREATE TABLE DoctorsDetails (
  id int(6) unsigned NOT NULL,
  doctorName varchar(50) NOT NULL,
  doctorRegistrationNumber varchar(10) NOT NULL,
  qualification varchar(30) NOT NULL,
  gender varchar(10) DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

INSERT INTO geeksforgeeks.DoctorsDetails
(id,doctorName, doctorRegistrationNumber, qualification,gender)
VALUES
```

```
(1,'doctorA','123-456','MDDCH','Female');

INSERT INTO geeksforgeeks.DoctorsDetails
(id,doctorName, doctorRegistrationNumber, qualification,gender)
VALUES
(1,'doctorB','111-222','MSNeuro','Male');

INSERT INTO geeksforgeeks.DoctorsDetails
(id,doctorName, doctorRegistrationNumber, qualification,gender)
VALUES
(1,'doctorC','222-444','MDGynae','Female');

INSERT INTO geeksforgeeks.DoctorsDetails
(id,doctorName, doctorRegistrationNumber, qualification,gender)
VALUES
(1,'doctorD','199-998','MSNephro','Male');

INSERT INTO geeksforgeeks.DoctorsDetails
(id,doctorName, doctorRegistrationNumber, qualification,gender)
VALUES
(1,'doctorE','444-666','MDCardio','Female');

SELECT * FROM geeksforgeeks.DoctorsDetails;

--If required, at last point of time we can truncate table
truncate table geeksforgeeks.DoctorsDetails;
```

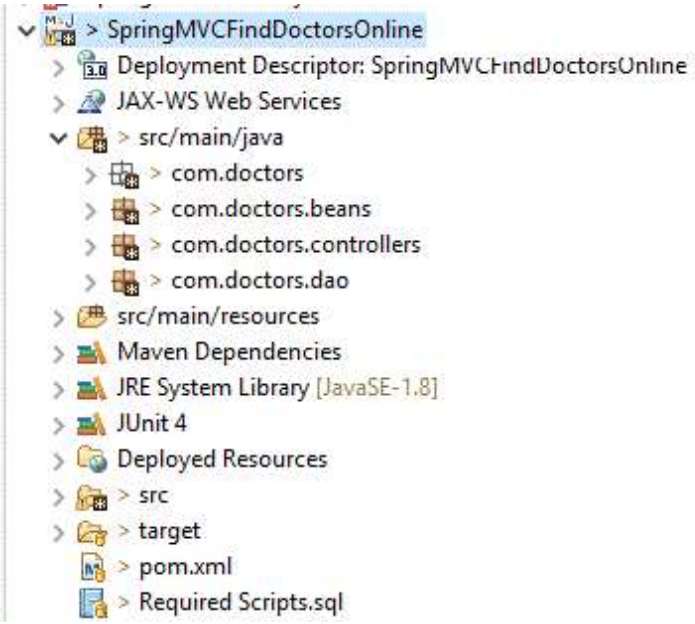
DBData Output:

```
36 •  SELECT * FROM geeksforgeeks.DoctorsDetails;
37
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	doctorName	doctorRegistrationNumbe	qualification	gender
1	doctorA	123-456	MDDCH	Female
2	doctorB	111-222	MSNeuro	Male
3	doctorC	222-444	MDGynae	Female
4	doctorD	199-998	MSNephro	Male
5	doctorE	444-666	MDCardio	Female
NULL	NULL	NULL	NULL	NULL

Project Structure:



This is going to get executed as a maven project

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
                             https://maven.apache.org/maven-v4_0_0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.doctors</groupId>
   <artifactId>SpringMVCFindDoctorsOnline</artifactId>
   <packaging>war</packaging>
   <properties>
      <maven.compiler.source>1.8</maven.compiler.source>
      <maven.compiler.target>1.8</maven.compiler.target>
   </properties>
```

```
    <version>0.0.1-SNAPSHOT</version>
    <name>SpringMVCFindDoctorsOnline Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-
webmvc -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-all</artifactId>
            <version>1.9.5</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.1.1.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.1.1.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>5.1.1.RELEASE</version>
            <scope>test</scope>
        </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>
   </dependencies>
   <build>
      <finalName>SpringMVCFindDoctorsOnline</finalName>
      <sourceDirectory>src/main/java</sourceDirectory>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.0.0-M3</version>
            <configuration>
               <testFailureIgnore>true</testFailureIgnore>
               <shutdown>kill</shutdown>
               <!-- Use it if required-->
            </configuration>
         </plugin>
         <!-- This should be added to overcome Could not initialize
              class
org.apache.maven.plugin.war.util.WebappStructureSerializer -->
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>3.3.2</version>
         </plugin>
      </plugins>
   </build>
</project>
```

We have to create a bean file and its structure should match with MySQL geeksforgeeks.DoctorsDetails

Doctor.java

```
public class Doctor {

    private int id;
    private String doctorName;
    private String doctorRegistrationNumber;
    private String gender;
    private String qualification;

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
```

```
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getDoctorName() {
        return doctorName;
    }

    public void setDoctorName(String doctorName) {
        this.doctorName = doctorName;
    }

    public String getDoctorRegistrationNumber() {
        return doctorRegistrationNumber;
    }

    public void setDoctorRegistrationNumber(String doctorRegistrationNumber) {
        this.doctorRegistrationNumber = doctorRegistrationNumber;
    }

    public String getQualification() {
        return qualification;
    }

    public void setQualification(String qualification) {
        this.qualification = qualification;
    }

}
```

Let us come to controller java

DoctorController.java

```
import java.sql.SQLException;

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.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.ModelAndView;

import com.doctors.beans.Doctor;
import com.doctors.dao.DoctorDao;

@Controller
@SessionAttributes("doctor")
```

```
public class DoctorController {
    // @Autowired
    // will inject dao from xml file
    DoctorDao dao;

    @Autowired
    public DoctorController(DoctorDao dao) {
        this.dao = dao;
    }

    @ModelAttribute("doctor")
    public Doctor getDoctor() {
        return new Doctor();
    }

    // for searchform
    @RequestMapping("/doctorsearchform")
    public String searchform(Model m) {
        m.addAttribute("command", new Doctor());
        return "doctorsearchform";
    }

    // It provides a facility to check doctors online
    @RequestMapping(value = "/checkDoctorsOnline", method = RequestMethod.POST)
    public ModelAndView calculateAmountForConsumedUnits(@ModelAttribute("doctor") Doctor
doctor) {

        ModelAndView mav = null;
        Doctor doctor1 = null;
        try {
            if (doctor.getDoctorName() != null && doctor.getDoctorName() != "") {
                doctor1 = dao.getDoctorsByName(doctor.getDoctorName());
            }
            if (doctor.getDoctorRegistrationNumber() != null &&
doctor.getDoctorRegistrationNumber() != "") {
                doctor1 =
dao.getDoctorsByRegistrationNumber(doctor.getDoctorRegistrationNumber());
            }
            mav = new ModelAndView("welcome");
            if (null != doctor1) {
                System.out.println(doctor1.getId() + "..." + doctor1.getDoctorName() + ".." +
doctor1.getDoctorRegistrationNumber()
                + doctor1.getGender());
                boolean isAvailable = false;

                mav.addObject("DoctorName", doctor1.getDoctorName());
                mav.addObject("RegistrationNumber", doctor1.getDoctorRegistrationNumber());
                mav.addObject("Gender", doctor1.getGender());
                mav.addObject("Qualification", doctor1.getQualification());
            }
        else {
            mav.addObject("DoctorName", doctor1.getDoctorName());
            mav.addObject("RegistrationNumber", "Not available Online");
        }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }

        return mav;
    }

}
```

DAO class

DoctorDao.java

```
import java.sql.SQLException;

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

import com.doctors.beans.Doctor;

public class DoctorDao {
    JdbcTemplate template;

    public void setTemplate(JdbcTemplate template) {
        this.template = template;
    }

    public Doctor getDoctorsByName(String doctorName) throws SQLException {
        String sql = "select * from doctorsdetails where doctorname=?";
        return template.queryForObject(sql, new Object[] {doctorName},
                new BeanPropertyRowMapper<Doctor>(Doctor.class));
    }

    public Doctor getDoctorsByRegistrationNumber(String registrationNumber) throws
SQLException {
        String sql = "select * from doctorsdetails where doctorRegistrationNumber=?";
        return template.queryForObject(sql, new Object[] {registrationNumber},
                new BeanPropertyRowMapper<Doctor>(Doctor.class));
    }

    public Doctor getDoctorsById(int id) throws SQLException {
        String sql = "select * from doctorsdetails where id =?";
        return template.queryForObject(sql, new Object[] { id },
                new BeanPropertyRowMapper<Doctor>(Doctor.class));
    }

    public Doctor getDoctorsByGender(String gender) throws SQLException {
        String sql = "select * from doctorsdetails where gender=?";
        return template.queryForObject(sql, new Object[] {gender},
                new BeanPropertyRowMapper<Doctor>(Doctor.class));
    }

}
```

spring-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans/"
```

```
    xmlns:xsi="https://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.SpringMVCFindDoctorsOnline.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/geeksforgeeks?
serverTimezone=UTC"></property>
        <property name="username" value="root"></property>
        <property name="password" value="admin"></property>
    </bean>

    <bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="ds"></property>
    </bean>

    <bean id="dao" class="com.doctors.dao.DoctorDao">
        <property name="template" ref="jt"></property>
    </bean>

</beans>
```

JSP Pages

indexPage.jsp

```
<center><B><a href="doctorsearchform">Find Doctors Online</a></B></center>
```

We would be getting a page like below

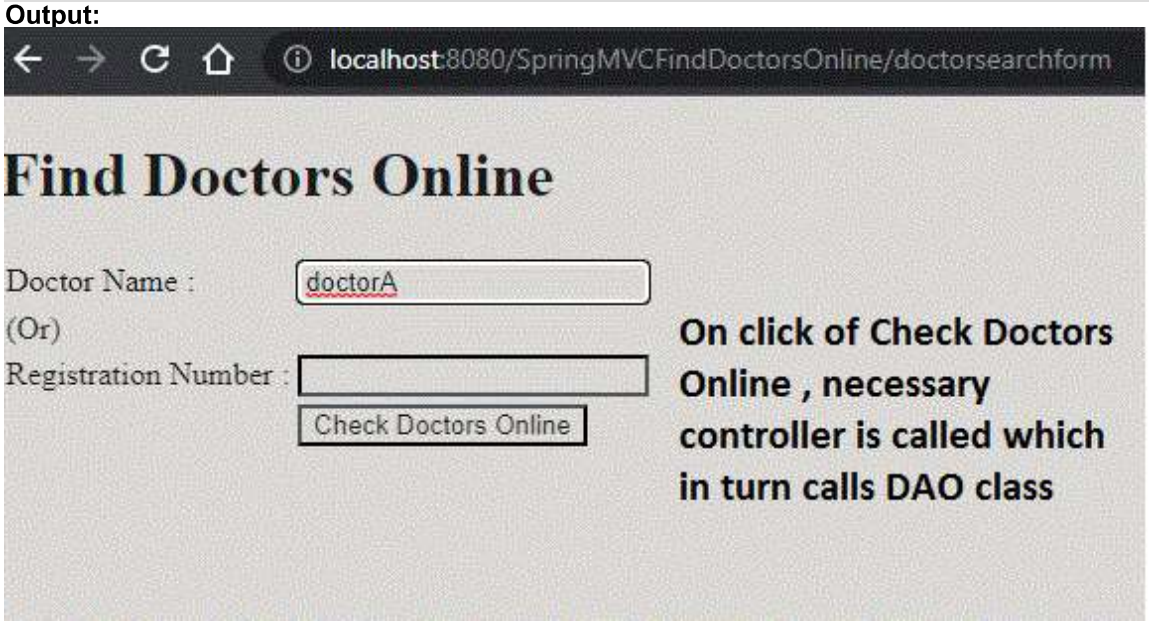
On click of the "Find Doctors Online" link, we can get the below page

doctorsearchform.jsp

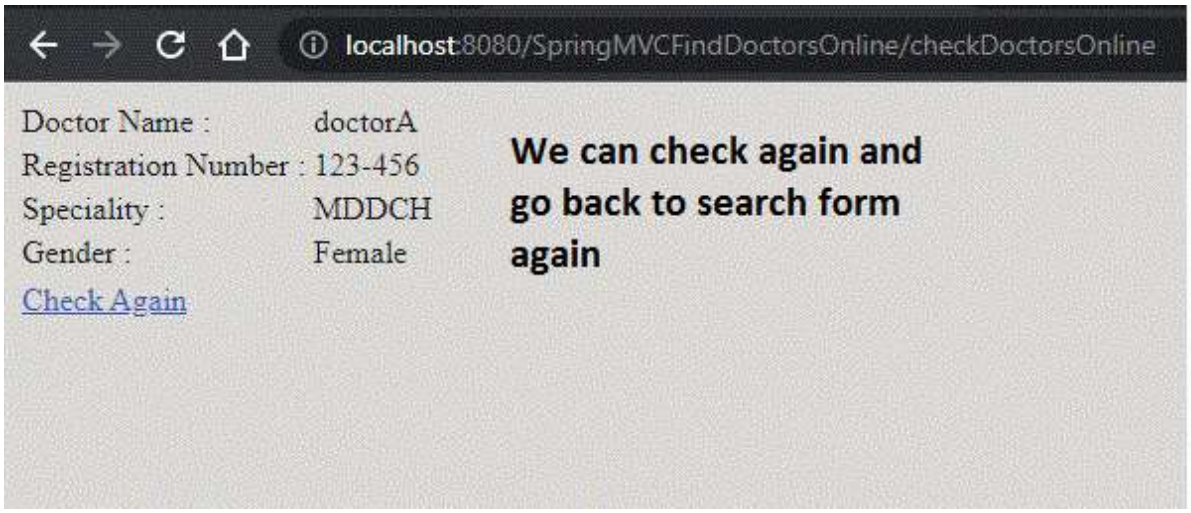
```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://www.oracle.com/technetwork/java/index.html prefix="c"%>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"https://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
      <title>Find Doctors Online</title>
   </head>
   <body>
      <h1>Find Doctors Online</h1>
      <form:form  method="post" action="/SpringMVCFindDoctorsOnline/checkDoctorsOnline"  >
         <table >
            <tr>
               <td>Doctor Name : </td>
               <td>
                  <form:input path="doctorName"/>
               </td>
            </tr>
            <tr>
               <td>  (Or) </td>
            </tr>
            <tr>
               <td>Registration Number : </td>
               <td>
                  <form:input path="doctorRegistrationNumber"/>
               </td>
            </tr>
            <tr>
               <td> </td>
               <td><input type="submit" value="Check Doctors Online" /></td>
            </tr>
         </table>
      </form:form>
   </body>
</html>
```

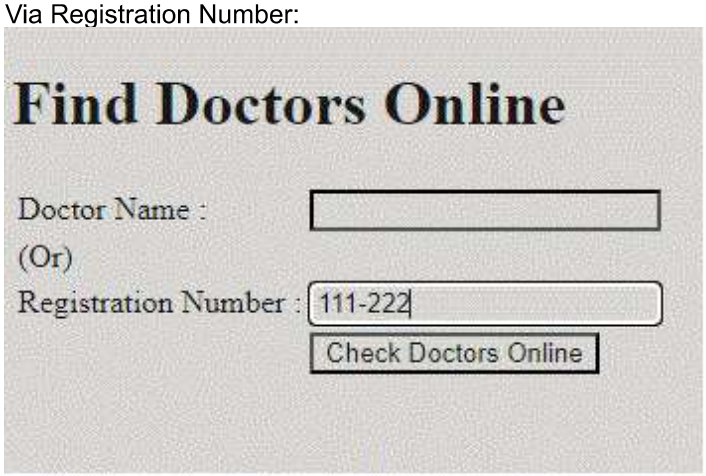
Output:

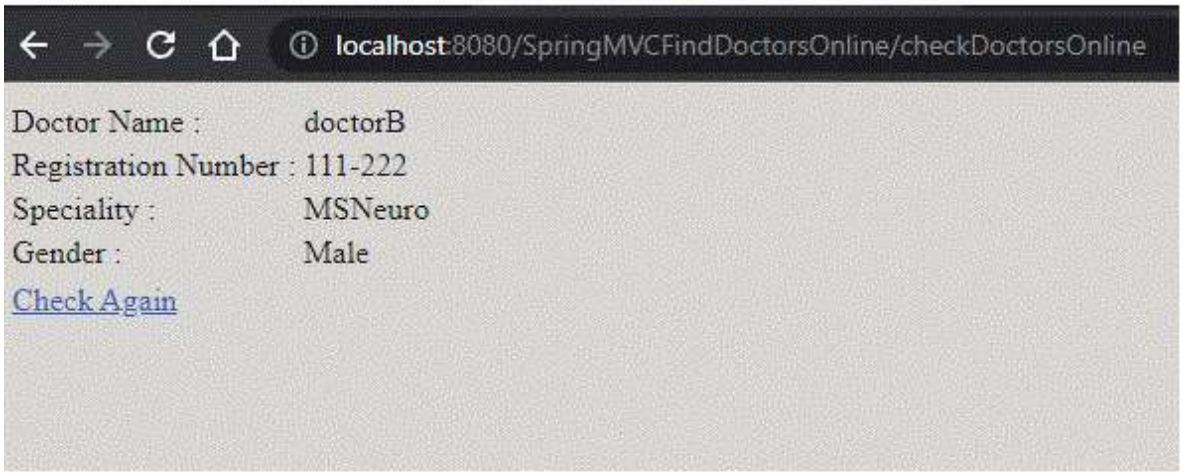


Via Doctor Name/Registration Number, we can check. As "doctorA" is given , it will use "getDoctorsByName" and returns back the doctor details



Via Registration Number:





Similarly, we can check doctor details easily via the above ways.

Comment

More info