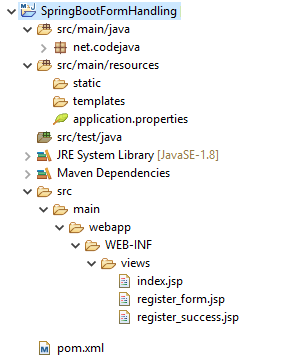
**Basic Project Structure**



**Step 1:**

To use JSP (JavaServer Pages) with Spring Boot, you must add the following dependency in the pom.xml file:

<dependency>

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

    <artifactId>tomcat-embed-jasper</artifactId>

    <scope>provided</scope>

</dependency>

And configure Spring MVC resolver in the application.properties file as follows:

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

This means you must put JSP files under this directory in the project:

src/main/webapp/WEB-INF/views

**Step 2:**

package net.codejava;

import org.springframework.stereotype.Controller;

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

@Controller

public class MvcController {

    @RequestMapping("/")

    public String home() {

        System.out.println("Going home...");

        return "index";

    }

}

**@RestController**

**public** **class** MVCController {

@GetMapping ("/")

**public** **ModelAndView** home() {

ModelAndView mav = **new** ModelAndView("index");

**return** mav;

}

**Step 3: Create Model**

package net.codejava;

import java.sql.Date;

public class User {

    private String name;

    private String email;

    private String gender;

    private String password;

    private String **profession;//**

    private String note;

    private Date birthday;

    private boolean married;

    // getters...

    // setters...

    // override toString()

}

**Step 4:**

index.jsp

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Spring Boot Form Handling Example</title>

</head>

<body>

    <h1>Spring Boot Form Handling Example</h1>

    <h2>Spring Form Tags with JSP</h2>

**<a href="/register">Click here</a>**

</body>

</html>

**Step 5: Handle the request when user click on link**

The showForm() handler method will be called when the user clicks the hyperlink /register in the homepage. It creates and adds two objects to the model:

public String **showForm(Model model)** {//when user click on register

    User user = new User();

    List<String> professionList = **Arrays.asList("Developer", "Designer", "Tester");**

    model.addAttribute("user", user);

    model.addAttribute("**professionList**", **professionList**);

    return "register\_form";

}

The professionList to display values for the dropdown list (select box) on the form.

**Step 6: register\_form.jsp**

We use Spring form tags declare the following directive at the beginning of the file:

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

<head>

<meta charset="ISO-8859-1">

<title>User Registration Form</title>

</head>

<body>

    <div align="center">

        <h2>User Registration</h2>

        <form:form action="**register**" **method="post"** **modelAttribute="user"**>

            <form:label path="name">Full name:</form:label>

            <form:input path="name"/><br/>

            <form:label path="email">E-mail:</form:label>

            <form:input path="email"/><br/>

            <form:label path="password">Password:</form:label>

            <form:password path="password"/><br/>

            <form:label path="birthday">Birthday (yyyy-mm-dd):</form:label>

            <form:input path="birthday"/><br/>

            <form:label path="gender">Gender:</form:label>

            <form:radiobutton path="gender" value="Male"/>Male

            <form:radiobutton path="gender" value="Female"/>Female<br/>

**<form:label path="profession">Profession:</form:label>**

**<form:select path="profession" items="${professionList}"** /><br/>

            <form:label path="married">Married?</form:label>

            <form:checkbox path="married"/><br/>

            <form:label path="note">Note:</form:label>

            <form:textarea path="note" cols="25" rows="5"/><br/>

            <form:button>Register</form:button>

        </form:form>

    </div>

</body>

Note:

Note that the value of the path attribute must refer to a field in the model object.

**<form:form action="register" method="post" modelAttribute="user">**

 the modelAttribute attribute that specifies name of an object in the model that maps to this form.

**Step 7: Handle form Submission**

@PostMapping("/register")

public String submitForm(@ModelAttribute("user") User user) {

    System.out.println(user);

    return "register\_success";

}

**Step 8:** register\_success.jsp

<body>

    <div align="center">

        <h2>Registration Succeeded!</h2>

        <span>Full name:</span><span>${user.name}</span><br/>

        <span>E-mail:</span><span>${user.email}</span><br/>

        <span>Password:</span><span>${user.password}</span><br/>

        <span>Birthday:</span><span>${user.birthday}</span><br/>

        <span>Gender:</span><span>${user.gender}</span><br/>

        <span>Profession:</span><span>${user.profession}</span><br/>

        <span>Married?:</span><span>${user.married}</span><br/>

        <span>Note:</span><span>${user.note}</span><br/>

    </div>

</body>

**To handle Server Side Form Validation in Spring Boot:**

<https://www.codejava.net/frameworks/spring-boot/spring-boot-form-validation-tutorial>