**Create a basic web application that handles user registration.**

Step 1: Set Up the Project

1. Create a new Maven or Gradle project in your favorite IDE.

2. Add the necessary Spring dependencies to your `pom.xml` (if using Maven) or `build.gradle` (if using Gradle):

```xml

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.10.RELEASE</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.10.RELEASE</version>

</dependency>

<!-- JSTL for JSP -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- Servlet API -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

```

3. Create a `web.xml` file in the `src/main/webapp/WEB-INF` directory:

```xml

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

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1">

<display-name>Spring MVC Example</display-name>

<servlet>

<servlet-name>dispatcher</servlet-name>

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

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring-config.xml</param-value>

</init-param>

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

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

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

</servlet-mapping>

</web-app>

```

4. Create a `spring-config.xml` file in the `src/main/webapp/WEB-INF` directory:

```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">

<!-- Enable component scanning -->

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

<!-- Configure the ViewResolver -->

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

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

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

</bean>

<!-- Enable MVC annotations -->

<mvc:annotation-driven/>

</beans>

```

5. Create a `src/main/webapp/WEB-INF/views` directory for your JSP views.

Step 2: Create a Controller

Create a simple controller class, for example, `UserController.java`:

```java

package com.example.controller;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

@Controller

public class UserController {

@GetMapping("/")

public String showRegistrationForm() {

return "registration";

}

@PostMapping("/register")

public String registerUser(@RequestParam String username, @RequestParam String password, Model model) {

// Handle user registration logic here

model.addAttribute("message", "Registration successful for " + username);

return "registration";

}

}

```

Step 3: Create JSP Views

Create a `registration.jsp` file in the `src/main/webapp/WEB-INF/views` directory to display the registration form:

```jsp

<!DOCTYPE html>

<html>

<head>

<title>User Registration</title>

</head>

<body>

<h1>User Registration</h1>

<form method="post" action="/register">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required/><br/><br/>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required/><br/><br/>

<input type="submit" value="Register"/>

</form>

<p>${message}</p>

</body>

</html>

```

Step 4: Run the Application

Now, you can run your Spring MVC application using your preferred servlet container (e.g., Tomcat). Access the application at `http://localhost:8080/` in your web browser, and you should see the user registration form. Submitting the form will trigger the `registerUser` method in the controller.

This example demonstrates a basic Spring MVC application for user registration without using Spring Boot. You can further enhance it by adding a database, authentication, and more features as needed.