实验 13 Spring MVC 基础与进阶

一、 实验目的

1. 熟练掌握 Spring MVC 核心配置（前端控制器、视图解析器、注解驱动等）的原理与扩展配置
2. 深入理解 Spring MVC 控制器的多种用法（请求参数绑定、模型数据传递、会话管理）

二、 实验要求

注：本实验要求使用 Tomcat 9。 Tomcat 10 不支持 Spring MVC。

基础功能：

在登录界面输入用户名和密码，点击登录按钮后，控制台输出用户名和密码；若输入为空，页面给出友好提示。

进阶功能：

用 Postman 等接口测试工具，向控制器发送包含用户名和密码的 JSON 数据，查看结果是否正确，并在控制台输出 JSON 数据。

三、 实验步骤

1. 创建项目

在 Eclipse 的 Package Explorer 空白区右键，创建”maven Project”

2. 加入 Spring MVC 相关 Jar 包。基础功能的 pom.xml 文件内容如下：

```
    <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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example.springmvc</groupId>
    <artifactId>exp15</artifactId>
    <packaging>war</packaging>
    <version>0.0.1-SNAPSHOT</version>
    <name>exp15 Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <properties>
        <!-- Spring 版本号  -->
        <spring.version>5.3.18</spring.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <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-expression</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-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-test</artifactId>
            <version>${spring.version}</version>
        </dependency>
    </dependencies>
    <build>
        <finalName>exp15</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.1</version>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.0</version>
                <configuration>
                    <source>11</source>
                    <target>11</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

进阶功能要增加的依赖：

```
        <!-- JSON 解析（Jackson） -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
```

```
            <version>2.17.1</version>
        </dependency>
```

3. 配置前端控制器（web.xml）

```
<servlet>
   <servlet-name>springmvc</servlet-name>
   <servlet-class>
                org.springframework.web.servlet.DispatcherServlet
         </servlet-class>
   <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc-config.xml</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
   <servlet-name>springmvc</servlet-name>
   <url-pattern>/</url-pattern>
</servlet-mapping>
```

4. 编写 JSP 代码

代码存放位置”WEB-INF/jsp/”

本实验最少编写一个 JSP 文件，实现如图 1 所示的登录页面。

图 1 登录页面

以下是基础功能向导：

5. 编写实体类代码

(1)在项目中创建 cn.edu.sdju.soft.bean 包，并创建一个类

(2)class 的私有属性名可以与页面表单的组件名一致，并确定相应的数据类型。

(3)生成属性的 get/set 方法。

6. 编写控制器类

(1) 在项目中创建 cn.edu.sdju.soft.controller 包

(2) 所有的页面跳转都通过控制器进行，因此，控制器应编写两个方法。一个方法用于接收请求时跳转到登录页，一个用于向控制台输出相关信息。

7. 创建 SpringMVC 的配置文件 springmvc-config.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans

https://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
    https://www.springframework.org/schema/mvc/spring-mvc.xsd">


    <!-- 定义视图解析器 -->
    <bean id="viewResolver" class=

"org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!-- 设置前缀 -->
        <property name="prefix" value="/WEB-INF/jsp/" />
        <!-- 设置后缀 -->
        <property name="suffix" value=".jsp" />
```

```
        </bean>
    </beans>
```

8. **配置控制器**

修改 Spring MVC 的配置文件，加入如下内容：

```
    <!- - 指定需要扫描的包 - ->
    <context:component-scan
base-package="com.example.springmvc.controller" />

    <! - -增加注解扫描配置 - ->
    <mvc:annotation-driven />
```

用注解扫描方式配置控制器，使用@Controller 注解

9. **使用@RequestMapping 注解 配置控制器的请求路径**

10. **执行测试**。给控制器发请求，页面跳转到登录页面，输入用户名和密码后，查看控制台，是否输出了用户名和密码。

截图要求：JSP 登录页面图，控制台输出内容图

以下是进阶功能向导：

在步骤 6 中的控制器上增加一个新方法：

```
@PostMapping("/login/json")
@ResponseBody //  表示返回 JSON 数据，而非视图
public Map<String, Object> loginByJson(@RequestBody User user) { … }
```

可从参数 user 中获取用户名和密码，并输出到控制台。执行测试时，可使用 Postman 等工具，以 POST 方式访问 http://localhost:8080/项目名/login/json，

请求体为 JSON 格式（如{"username":"test123","password":"Test1234"}），查看返回结果是否正确；

附：java 语言命名规范

- 选择有意义并且具有描述性的名字（见名知意）.
- 变量和方法:

  小写.如果包含多个单词，第一个单词第一个字母小写，其余单词第一个字母大写。如：radius，area，computeArea.
- 类名、接口名:

  每个单词第一个字母大写。如 ComputeArea.
- 常量名:

所有字母都大写，用下划线连接每个单词。如 PI， MAX_VALUE

- 包名: 所有字母小写，用.连接每个单词。如 cn.edu.sdju.soft

Naming Conventions of Java Language

- Choose meaningful and descriptive names.
- Variables and method names:

  Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name. For example, the variables radius and area, and the method computeArea.
- Class names:

  Capitalize the first letter of each word in the name. For example, the class name ComputeArea.
- Constants:

  Capitalize all letters in constants, and use underscores to connect words. For example, the constant PI and MAX_VALUE