实验四 Spring MVC 基础应用——基于 Spring MVC 框架的用户登录模块

一、基础实验——Spring MVC 框架搭建

（一）实验目的

1、掌握 Spring MVC 应用的基本开发步骤和常规配置；
2、掌握 Spring 环境搭建的基本方法，能在 WEB 应用中使用 Spring MVC，并能在 Eclipse 等 IDE 环境中开发 Spring MVC 应用；
3、观察配置文件 springmvc-servlet.xml 中的主要元素及其作用，并能够正确应用；
4、理解 Struts2 框架中 MVC 设计模式的体现，理解 Controller，DispatcherServlet，springmvc-servlet.xml 的主要作用，并能够正确应用；
5、观察配置文件 applicationContext.xml 中对 Spring MVC 业务控制器 Controller 的配置，理解 Spring 容器对 Controller 的管理；
6、能够分析对比 Spring MVC+Spring 与 Struts+Spring 两种模式的异同。

（二）基本知识与原理

1、Spring MVC 框架主要由 DispatcherServlet、处理器映射、控制器、视图解析器和视图组成；
2、Spring MVC 的工作流程如图 4-1 所示：
（1）客户端请求提交到 DispatcherServlet；
（2）DispatcherServlet 借助于 Spring MVC 提供的 HandlerMapping 定位到具体的 Controller；
（3）DispatcherServlet 将请求提交到 Controller；
（4）Controller 调用业务逻辑处理用户请求后返回 ModelAndView 对象；
（5）DispatcherServlet 借助于 ViewResolver（视图解析器）找到 ModelAndView 指定的视图；
（6）视图负责将结果显示到客户端。

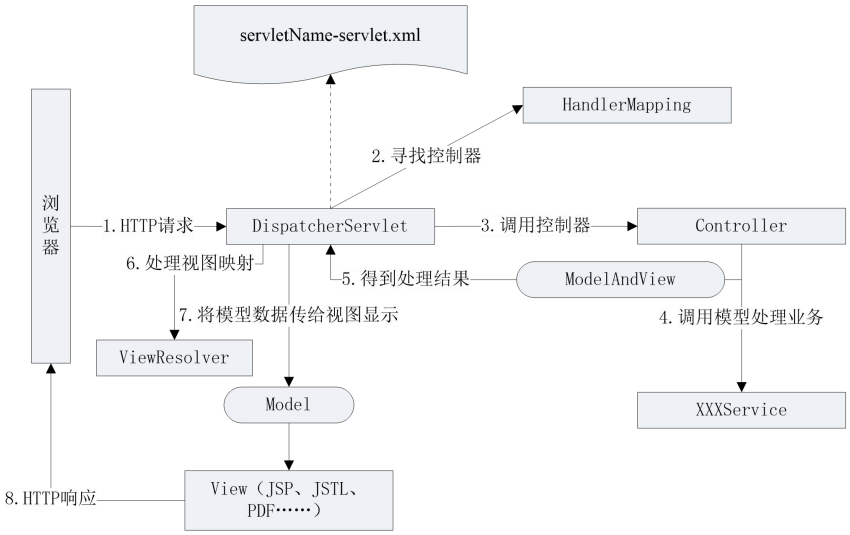

图 4-1 Spring MVC 工作流程

3、DispatcherServlet 是 Spring MVC 中的核心控制器，客户端对服务器端的所有请求都经过 DispatcherServlet 来统一分发；

4、HandlerMapping 接口负责完成客户请求到 Controller 映射；

5、Controller 是单个 Http 请求处理过程中的控制器；

6、ModelAndView 是 Http 请求过程中返回的模型（Model）和视图（View）；

7、ViewResolver（视图解析器）在 Web 应用中负责查找 View 对象，从而将相应结果渲染给客户；

8、编译运行基于 Spring MVC 框架的 Web 工程，需要导入 Spring 核心 jar 包的基础上，再增加 Spring web 和 Spring webmvc 的 jar 包。

（三）实验内容及步骤

1、登录 http://struts.apache.org/download.cgi 站点，下载 Struts2 的最新版（Full Distribution）；

2、在 Eclipse 中新建 Web 工程 springmvc-prj1；

3、将 Spring MVC 相关的中的 jar 包增加到 Web 应用中，即复制到“%workspace% springmvc-prj1\WebContent\WEB-INF\lib”路径下，如下图所示；

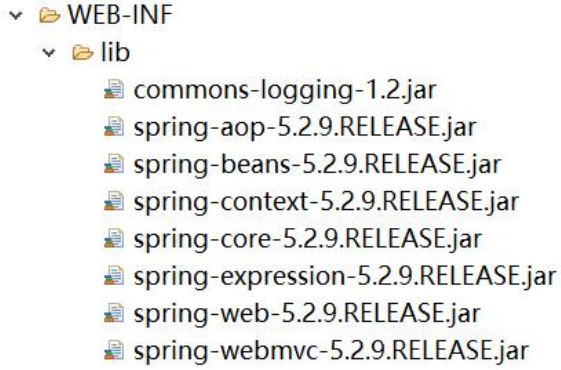

图 4-2 Spring MVC 相关 jar 包

4、在 springmvc-prj1 中新建 login.jsp 页面，作为用户登录的视图，代码片段如下：

```
<form action="login" method="post">
    请输入用户名：<input name="username" type="text"><BR>
    请输入密码：<input name="password" type="password">
    <input type="submit" value="登录">
</form>
```

5、在 springmvc-prj1 中新建 loginSuccess.jsp，作为登录成功的视图，代码片段如下：

```
<body>
    ${uname}，登录成功！
</body>
```

6、在 springmvc-prj1 中新建 loginFail.jsp 页面作为登录失败的视图，在页面中显示“登录失败”，代码略；

7、在 springmvc-prj1 中新建 cn.edu.zjut.bean 包，并在其中创建 UserBean.java，用于记录登录用户信息，代码略；

8、在 springmvc-prj1 中新建 cn.edu.zjut.service 包，并在其中创建 IUserService 接口及其实现类 UserService.java，用于实现登录逻辑，为简化登录逻辑，将登录成功的条件设置为：用户名和密码相同，代码略；

9、在 springmvc-prj1 中新建 cn.edu.zjut.controller 包，并在其中创建业务控制器 UserController.java，简化登录处理直接返回登录成功的视图，代码如下：

```
package cn.edu.zjut.controller;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;

public class UserController implements Controller{
       private IUserService userServ;
       public void setUserServ(IUserService userServ) {
           this.userServ=userServ;   }

       public ModelAndView handleRequest(HttpServletRequest arg0,
             HttpServletResponse arg1) throws Exception {
           return new ModelAndView("/loginSuccess.jsp");
       }
}
```

10、在工程 springmvc-prj1 的 WEB-INF 目录中创建 springmvc-servlet.xml 文件，在其中配置 UserController 和 UserService 实例等相关内容，代码如下：

```
<?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:p="http://www.springframework.org/schema/p"
    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">

    <!-- 简化配置，相当于配置了 HandlerMapping、HandlerAdapter、
HandlerExceptionResolver 三个特殊的 Bean，且在容器中注册了一系列支持 HTTP 消
息转换的 Bean -->
    <mvc:annotation-driven/>

    <bean name="/login"
class="cn.edu.zjut.controller.UserController">
        <property name="userServ" ref="userService"/>
    </bean>
    <bean id="userService" class="cn.edu.zjut.service.UserService"/>
</beans>
```

11、在工程 springmvc-prj1 的 WEB-INF 目录中创建 applicationContext.xml 文件，代码如下：

```
<?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:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>
```

12、在工程 springmvc-prj1 的 WEB-INF 目录中创建 web.xml 文件，增加 Spring MVC 核心控制器 DispatcherServlet 的配置，同时添加对 Spring 监听器的配置，代码片段如下：

```
<!-- 定义 Struts2 的核心 Filter -->
<servlet>
     <servlet-name>springmvc</servlet-name>
     <servlet-class>
          org.springframework.web.servlet.DispatcherServlet
```

```
     </servlet-class>
</servlet>
<servlet-mapping>
     <servlet-name>springmvc</servlet-name>
     <url-pattern>/</url-pattern>
</servlet-mapping>

<listener>
     <listener-class>
           org.springframework.web.context.ContextLoaderListener
     </listener-class>
</listener>
```

13、将 springmvc-prj1 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

14、修改业务控制器 UserController.java 中处理方法的 return 语句，使用逻辑视图名，代码片段如下：

```
public ModelAndView handleRequest(HttpServletRequest arg0,
            HttpServletResponse arg1) throws Exception {
       return new ModelAndView("loginSuccess");
}
```

15、修改配置文件 springmvc-servlet.xml，在其中配置 Spring MVC 的视图解析器（ViewResolver），代码片段如下：

```
<!-- ViewResolver -->
<bean
class="org.springframework.web.servlet.view.InternalResourceViewRes
olver"  id="internalResourceViewResolver">
   <!-- 前缀 -->
   <property name="prefix" value="/"/>
   <!-- 后缀 -->
   <property name="suffix" value=".jsp"/>
</bean>
```

16、将 springmvc-prj1 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；修改后的关键代码，及相应的运行结果或报错信息；

（2）根据实验过程，整理 Spring MVC 应用从请求到响应的完整流程，并记录下来；

（3）根据实验过程，总结三个配置文件的作用，并记录下来；
（4）根据实验过程，观察配置文件 applicationContext.xml 中对 Spring MVC 业务控制器 Controller 的配置，将其与配置文件 applicationContext.xml 中对 Struts 业务控制器 Action 的配置（实验三）进行比较，分析主要区别并记录下来；
（5）碰到的问题及解决方案或思考；
（6）实验收获及总结。
2、上交程序源代码，代码中应有相关注释。

二、提高实验——Spring MVC 的 Controller

（一）实验目的

1、进一步理解 Spring MVC 进行 Web 应用开发时，Controller 在该应用中的核心作用；
2、掌握基于注解的 Spring MVC 框架中 Controller 控制器的实现方法；
3、掌握 Spring MVC 框架中 Controller 接收请求参数的不同方法，并能根据实际情况选择合适的接收方式；
4、掌握 Spring MVC 框架中 Controller 实现重定向和转发的方法。

（二）基本知识与原理

1、传统风格的控制器不仅需要在配置文件中部署映射，而且只能编写一个处理方法，不够灵活；而基于注解的控制器具有以下两个优点：
（1）在基于注解的控制器类中可以编写多个处理方法，进而可以处理多个请求，因此允许将相关的操作编写在同一个控制器中，方便维护；
（2）基于注解的控制器无需在配置文件中部署映射，仅需使用 RequestMapping 注释类型注解一个方法进行请求处理。
2、Spring MVC 框架基于 Spring 框架，因此具有一个依赖注入的优点，可以通过 org.springframework.beans.factory.annotation.Autowired 注解类型将依赖注入到一个属性（成员变量）或方法：
（1）为了能被作为依赖注入，类必须使用 org.springframework.stereotype.Service 注解类型注明为@Service（一个服务）；
（2）另外，还需要在配置文件中使用<context:component-scan base-package="基本包"/>元素来扫描依赖基本包。
3、Controller 接收请求参数的方法有很多种：

（1）通过实体 Bean 接收请求参数；
（2）通过处理方法的形参接收请求参数；
（3）通过 HttpServletRequest 接收请求参数；
（4）通过@PathVariable 接收 URL 中的请求参数；
（5）通过@RequestParam 接收请求参数；
（6）通过@ModelAttribute 接收请求参数。

4、在 Spring MVC 框架中，控制器类中处理方法的 return 语句默认是转发到视图，同时 return 语句同样可以实现重定向（客户端行为）或转发给另一个处理请求（服务器行为）。

（三）实验内容及步骤

1、在 cn.edu.zjut.controller 包中，修改 UserController.java，使用@Controller 注解来声明该类的实例是一个控制器 Controller，并使用@RequestMapping 注解将请求 URI 映射到方法上，代码片段如下：

```
package cn.edu.zjut.controller;
import java.lang.annotation.Annotation;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.mvc.Controller;
……

@Controller
public class UserController {
……
    @RequestMapping("/login")
    public String login(UserBean user, Model model) {
       if (userServ.login(user))  {
              model.addAttribute("uname", user.getUsername());
              return "loginSuccess";
         }
         else return "login";
      }
}
```

2、继续修改 UserController.java，使用@Autowired 注解来进行依赖注入，完成 userService 属性的注入，并删除 setUserService 方法，代码片段如下：

```
……
import org.springframework.stereotype.Controller;
import org.springframework.beans.factory.annotation.Autowired;

@Controller
```

```
public class UserController {
     //@Autowired
     private IUserService userServ;
     /*
     public void setUserServ(IUserService userServ) {
          this.userServ=userServ;
     }
     */
}
```

3、修改 Spring MVC 配置文件 springmvc-servlet.xml，增加配置使得 Spring 自动扫描 controller 包及其子包中的所有 Bean，并删除对 UserController 实例的配置，代码片段如下：

```
<!-- 配置 Spring 自动扫描指定包及其子包中的所有 Bean -->
<context:component-scan base-package="cn.edu.zjut.controller"/>
```

4、将 springmvc-prj1 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

5、修改 UserService.java，使用@Service 注解来声明该类的实例是一个业务逻辑（服务）Service，代码片段如下：

```
import org.springframework.stereotype.Service;
import cn.edu.zjut.bean.UserBean;

@Service("userServ")
public class UserService implements IUserService {
    ……
}
```

6、修改 UserController.java，使用@Qualifier 注解来辅助@Autowired 注解对依赖注入的实例对象进行匹配，代码片段如下：

```
@Controller
public class UserController {
     //@Autowired
     //@Qualifier("userServ")
     private IUserService userServ;
     /*
     public void setUserServ(IUserService userServ) {
          this.userServ=userServ;
     }
     */
}
```

7、修改 Spring MVC 配置文件 springmvc-servlet.xml，增加配置使得 Spring 自动 service 包及其子包中的所有 Bean，并删除对 UserService 实例的配置，代码片段如下：

```
<!-- 配置 Spring 自动扫描指定包及其子包中的所有 Bean -->
<context:component-scan base-package="cn.edu.zjut.controller"/>
<context:component-scan base-package="cn.edu.zjut.service"/>
```

8、将 springmvc-prj1 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

9、将 Spring MVC 配置文件 springmvc-servlet.xml 更名为 springmvc.xml；

10、修改 Web 应用的 web.xml 文件，添加对 Spring MVC 配置文件 springmvc.xml 的配置，代码片段如下：

```
<web-app>
  <servlet>
     <servlet-name>springmvc</servlet-name>
     <servlet-class>
          org.springframework.web.servlet.DispatcherServlet
     </servlet-class>
     <init-param>
          <param-name>contextConfigLocation</param-name>
          <param-value>/WEB-INF/springmvc.xml</param-value>
     </init-param>
     <load-on-startup>1</load-on-startup>
  </servlet>

……
<web-app>
```

11、将 springmvc-prj1 重新布署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

12、修改 UserController.java，通过@ModelAttribute 接收请求参数，代码片段如下：

```
@RequestMapping("/login")
//暴露为模型数据
public String login( @ModelAttribute("user") UserBean user) {
     if (userServ.login(loginUser))  {
          //model.addAttribute("user", user);
          return "loginSuccess";
     }
     else return "login";
}
</html>
```

13、将 springmvc-prj1 重新布署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

14、修改 UserController.java，通过形参接收请求参数，代码片段如下：

```
@RequestMapping("/login")
```

```
public String login( String username, String password, Model model) {
     if (username.equals(password))  {
          model.addAttribute("uname", username);
          return "loginSuccess";
     }
     else return "login";
}
</html>
```

15、将 springmvc-prj1 重新布署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

16、修改 UserController.java，通过@RequestRaram 接收请求参数，代码片段如下：

```
@RequestMapping("/login")
public String login( @RequestRaram String username,
              @RequestRaram String password, Model model) {
     if (username.equals(password))  {
          model.addAttribute("uname", username);
          return "loginSuccess";
     }
     else return "login";
}
</html>
```

17、将 springmvc-prj1 重新布署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

18、修改 UserController.java，通过 HttpServletRequest 接收请求参数，代码片段如下：

```
@RequestMapping("/login")
public String login( HttpServletRequest req,  Model model) {
     String username=req.getParameter("username");
     String password=req.getParameter("password");
     if (username.equals(password))  {
          model.addAttribute("uname", username);
          return "loginSuccess";
     }
     else return "login";
}
</html>
```

19、将 springmvc-prj1 重新布署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并记录运行结果；

20、修改 UserController.java，通过@PathVariable 接收请求参数，代码片段如下：

```
@RequestMapping(value="/login/{uname}/{upass}",
```

```
                    method=Request Method.GET)
public String login( @PathVariable String uname,
                        @PathVariable String upass, Model model) {
     if (username.equals(password))  {
          model.addAttribute("uname", username);
          return "loginSuccess";
     }
     else return "login";
}
</html>
```

21、将 springmvc-prj1 重新布署在 Tomcat 服务器上，通过地址栏输入 URL 来进行访问：http://localhost:8080/springmvc-prj1/login/zjut/zjut，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）结合实验过程，总结基于注解的开发方式的特点，并记录下来；

（3）根据实验步骤 7，总结@Qualifier 注解的作用，并记录下来；

（4）结合实验过程，查找相关资料，总结 Controller 接收请求参数的各种方法及其适用情况，并记录下来；

（5）碰到的问题及解决方案或思考；

（6）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

三、扩展实验——Spring MVC 框架中的数据绑定

（一）实验目的

1、掌握 Spring MVC 框架中数据绑定的方法和基本步骤；

2、理解 Spring MVC 框架中数据绑定的作用；

3、掌握 Spring MVC 表单标签库中常用标签的基本使用方法；

4、能参考 Spring MVC 表单标签库的使用说明文档，对各类标签进行灵活应用；

5、进一步理解 Spring MVC 框架中 Model 的作用；

6、掌握基本的 JSON 数据格式，并能在 Spring MVC 框架中实现浏览器与控制器类之间的 JSON 数据交互。

（二）基本知识与原理

1、在 Spring MVC 框架中，数据绑定有这样几层含义：

（1）绑定请求参数输入值到领域模型、

（2）模型数据到视图的绑定（输入验证失败时）、

（3）模型数据到表单元素的绑定；

2、表单标签库中包含了可以用在 JSP 页面中渲染 HTML 元素的标签。JSP 页面使用 Spring 表单标签库时，必须在 JSP 页面开头处声明 taglib 指令，指令代码如下：<%@ taglib prefix="form"
uri="http://www.springframework.org/tags/form" %>；

3、表单标签库中有 form、input、password、hidden、textarea、checkbox、checkboxes、radiobutton、radiobuttons、select、option、options、errors；

4、表单中的 modelAttribute 属性用于暴露 form backing object 的模型属性名称，属性值绑定一个 JavaBean 对象；

5、表单中 input 标签的 path 属性将文本框输入值绑定到 form backing object 的一个属性；

6、Spring MVC 框架中数据绑定应用的需要的相关配置如图 4-3 所示：

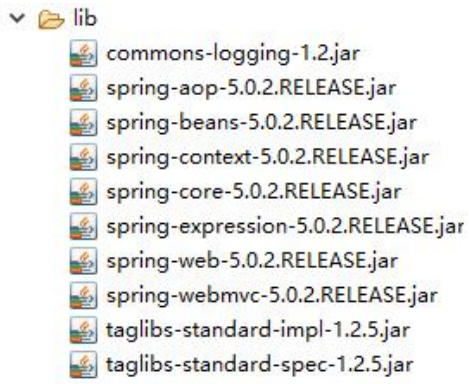

图 4-3 数据绑定应用的需要的相关配置

7、Spring MVC 在数据绑定的过程中，需要对传递数据的格式和类型进行转换，它既可以转换 String 等类型的数据，也可以转换 JSON 等其他类型的数据；

8、JSON（JavaScript Object Notation，JS 对象标记）是一种轻量级的数据交换格式；

9、与 XML 一样，JSON 也是基于纯文本的数据格式，它有两种数据结构：

（1）对象结构以“{”开始，以“}”结束。中间部分由 0 个或多个以英文“,”分隔的 key/value 对构成，key 和 value 之间以英文“:”分隔；

（2）数组结构以“[”开始，以“]”结束。中间部分由 0 个或多个以英文“,”分隔的值的列表组成；

10、为实现浏览器与控制器类之间的 JSON 数据交互，Spring MVC 提供了 MappingJackson2HttpMessageConverter 实现类默认处理 JSON 格式请求响应；

11、Jackson 开源包及其描述如下所示：

（1）jackson-annotations-2.9.4.jar：JSON 转换注解包；

（2）jackson-core-2.9.4.jar：JSON 转换核心包；

（3）jackson-databind-2.9.4.jar：JSON 转换的数据绑定包；

12、在 Spring MVC 框架使用注解开发时，需要用到两个重要的 JSON 格式转换注解，分别是@RequestBody 和@ResponseBody。

（三）实验内容及步骤

1、根据教材第 12 章，实现用户信息添加功能，步骤略。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验过程，查找相关资料，分析说明数据绑定的优点，并记录下来；

（3）根据实验过程，查找相关资料，总结 Spring MVC 有那些表单标签，其中可以绑定集合数据的标签有哪些，并记录下来；

（4）碰到的问题及解决方案或思考；

（5）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。