



JavaEE 实验四

实验报告

姓　　名　　 Mcrivers

班　　级　　 软件工程2102班

学　　号

提交日期　　 2023.11.16

目录

[一、基础实验——Spring MVC 框架搭建 1](#_Toc184215782)

[1. 运行结果截图，修改后的关键代码，相应的运行结果或报错信息 1](#_Toc184215783)

[2. 整理 Spring MVC 应用从请求到响应的完整流程 1](#_Toc184215784)

[3. 总结三个配置文件的作用 1](#_Toc184215785)

[4. 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 2](#_Toc184215786)

[5. 碰到的问题及解决方案或思考 2](#_Toc184215787)

[6. 实验收获及总结 3](#_Toc184215788)

[二、提高实验——Spring MVC 的 Controller 3](#_Toc184215789)

[1. 运行结果截图 3](#_Toc184215790)

[2. 总结基于注解的开发方式的特点 7](#_Toc184215791)

[3. 总结@Qualifier 注解的作用 7](#_Toc184215792)

[4. 总结 Controller 接收请求参数的各种方法及其适用情况 8](#_Toc184215793)

[5. 碰到的问题及解决方案或思考 8](#_Toc184215794)

[6. 实验收获及总结 8](#_Toc184215795)

# 一、基础实验——Spring MVC 框架搭建

## 1. 运行结果截图，修改后的关键代码，相应的运行结果或报错信息

**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**

观察到，login.jsp页面可以正常访问。



图1.1 login.jsp登录界面

**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**

观察到，login.jsp页面可以正常访问。

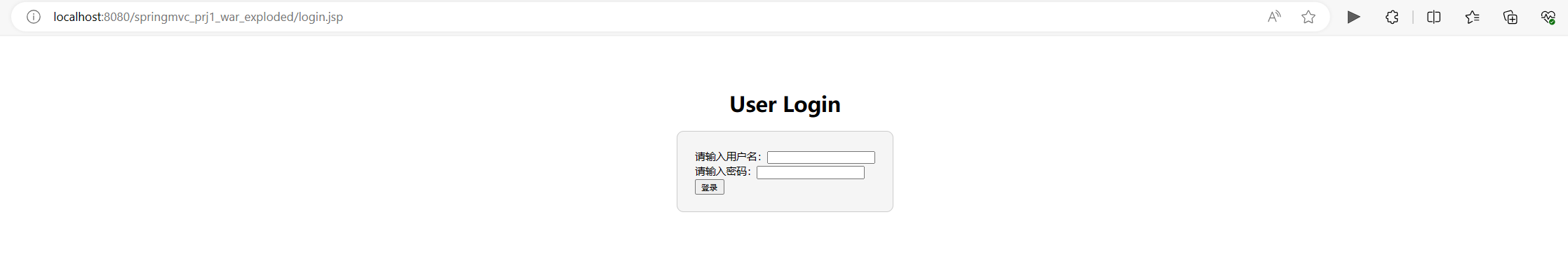


图1.2 login.jsp登录界面

## 2. 整理 Spring MVC 应用从请求到响应的完整流程

①DispatcherServlet 接收请求

②HandlerMapping 解析请求

③Controller 处理请求

④ModelAndView 返回视图信息

⑤ViewResolver 解析视图

⑥View 渲染视图

⑦Response 返回给客户端

## 3. 总结三个配置文件的作用

**applicationContext.xml：**

用来定义和配置应用程序中的所有 bean、依赖关系和其他配置，配置业务层、数据访问层等各种组件，定义它们的依赖关系、作用域等。

**springmvc-servlet.xml：**

用于配置 Spring MVC 的各种组件，例如 HandlerMapping、HandlerAdapter、ViewResolver。<mvc:annotation-driven/> 配置表示启用注解驱动的 Spring MVC，在使用注解开发时可以自动注册相关的处理器和适配器。同时，还定义了一个 UserController 的 Bean，并设置了它的依赖关系。另外，还配置了一个 InternalResourceViewResolver，用于解析视图名并转换为真正的视图资源。

**web.xml：**

配置 Servlet 定义、Servlet 映射、监听器等。在这个文件中，定义了名为 springmvc 的 DispatcherServlet，它负责拦截所有请求并将其分发给 Spring MVC 应用程序。同时，配置了 ContextLoaderListener 监听器，用于加载应用程序的根上下文。

## 4. 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

本实验中的applicationContext.xml没有定义Spring Bean，而实验三中的application.xml定义了两个 Spring Bean。其中userAction 是一个 ID 为 userAction 的 Bean，它的类为 cn.edu.zjut.action.UserAction，并设置了其作用域为 prototype。userAction Bean 中设置了一个属性 userService，它引用了另一个 Bean，即 ID 为 userService 的 Bean。

## 5. 碰到的问题及解决方案或思考

问题：在UserController.java中，出现报错类 "UserController" 必须声明为抽象，或为实现 "Controller" 中的抽象方法 "handleRequest(HttpServletRequest, HttpServletResponse)"

解决方法：报错出现的原因是和jakarta不匹配，将servlet-api.jar包导入后使用javax，问题成功解决。

## 6. 实验收获及总结

通过这次实验，我深入了解了Spring MVC框架的核心组件，包括DispatcherServlet、HandlerMapping、Controller、ViewResolver等的作用和工作流程，掌握了如何配置Controller、HandlerMapping、ViewResolver等Spring MVC的关键元素，这让我对MVC设计模式和Spring框架有了更深入的认识。

# 二、提高实验——Spring MVC 的 Controller

## 1. 运行结果截图

**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**

login.jsp页面可以正常访问，且可以根据用户名和密码的匹配情况正常跳转

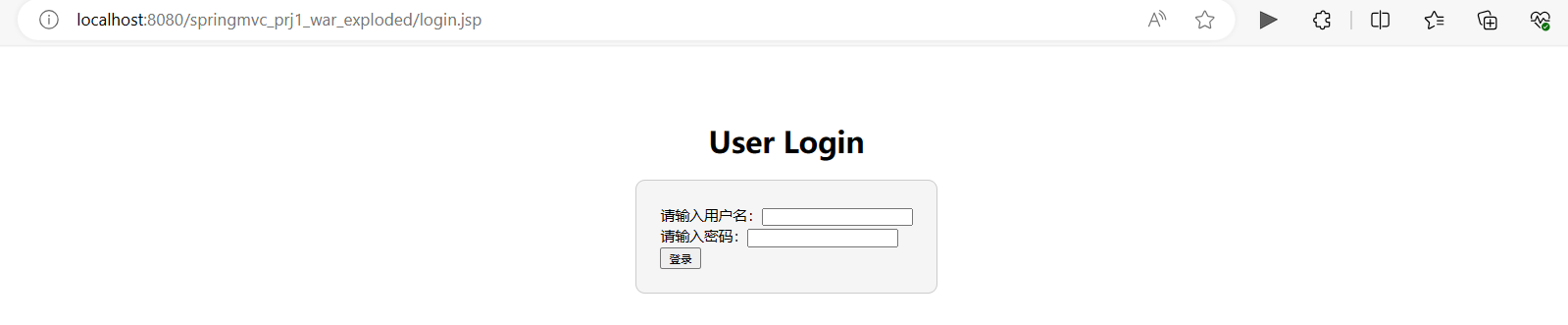


图2.1 login.jsp登录界面



图2.2 loginSucuess.jsp登录界面

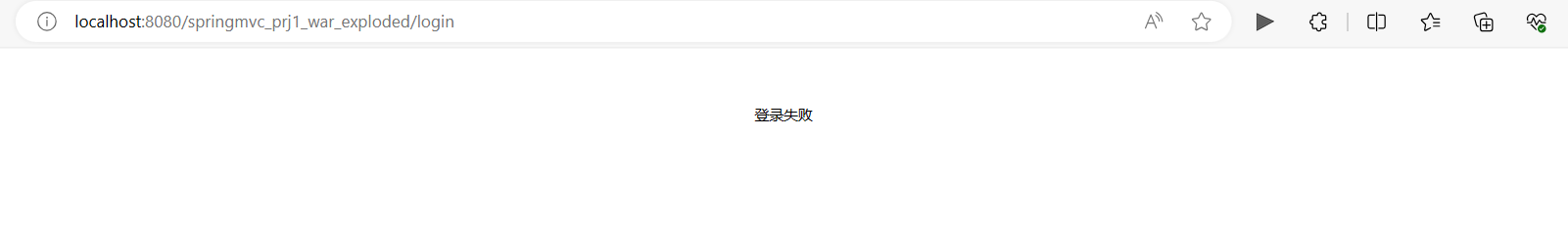


图2.3 loginFail.jsp登录界面

**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**

login.jsp页面可以正常访问，且可以根据用户名和密码的匹配情况正常跳转

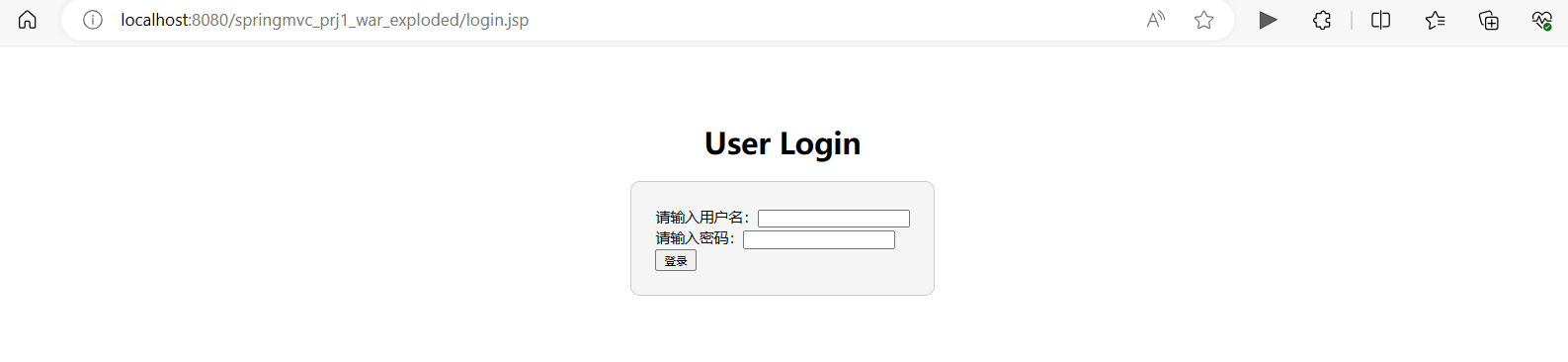


图2.4 login.jsp登录界面

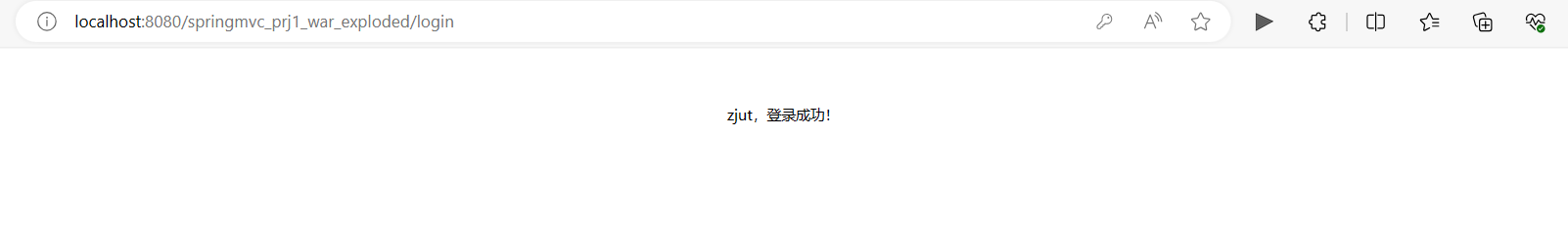


图2.5 loginSuccess.jsp登录成功界面

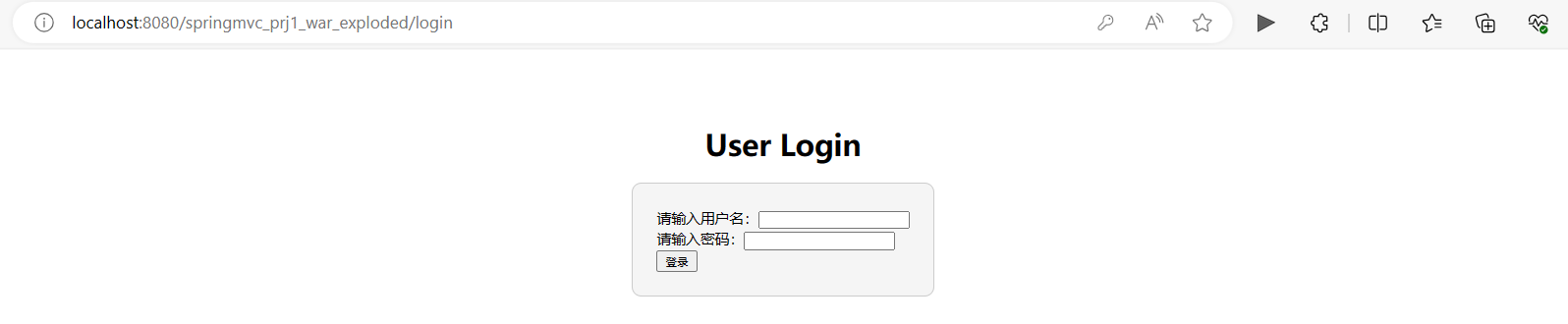


图2.6 登录失败时，回到login.jsp页面

**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**



图2.7 login.jsp登录界面

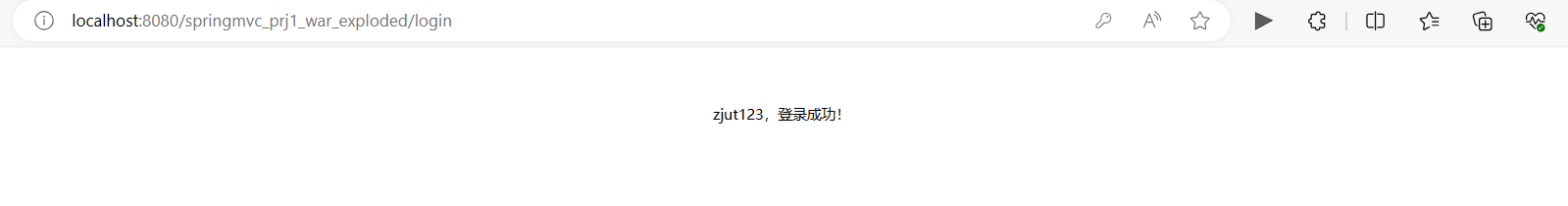


图2.8 loginSuccess.jsp登录成功界面

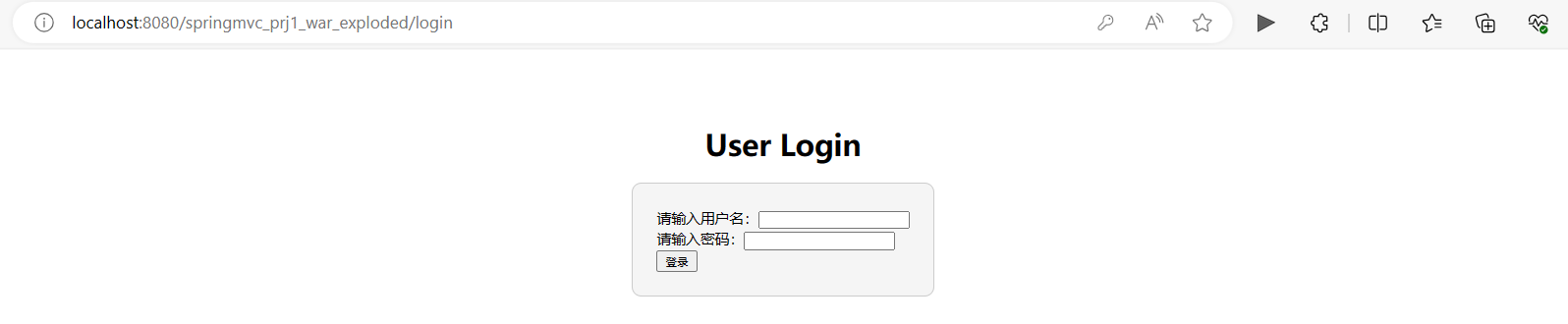


图2.9 登录失败时，回到login.jsp页面

**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**



图2.10 login.jsp登录界面



图2.11 loginSuccess.jsp登录成功界面

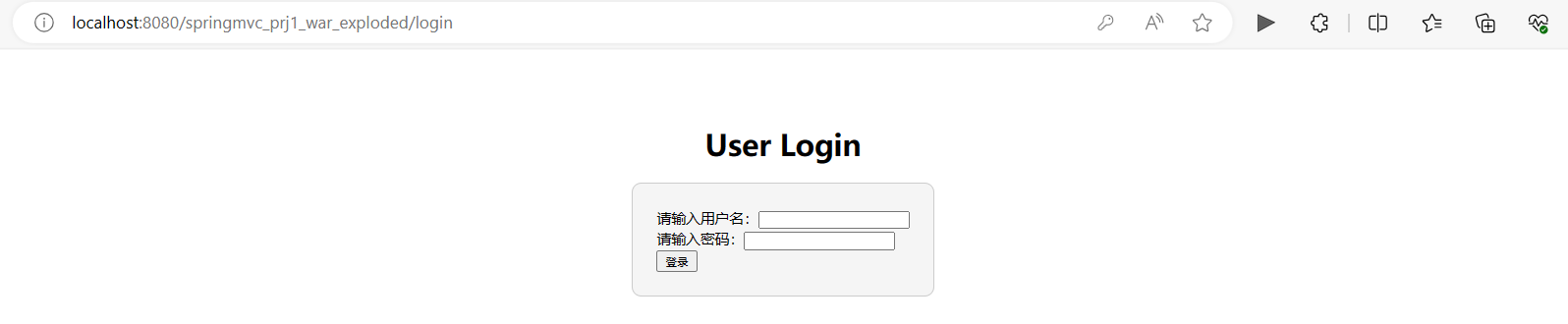


图2.12 登录失败时，回到login.jsp页面

**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**



图2.13 login.jsp登录界面

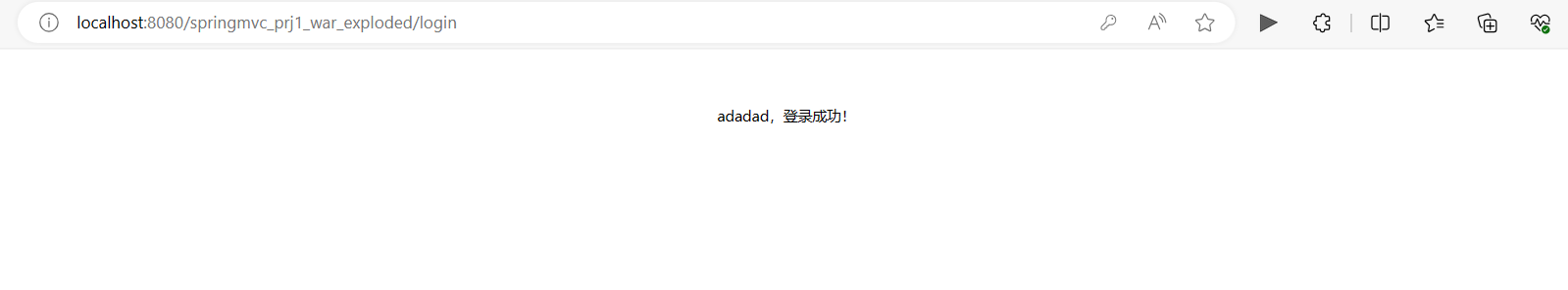


图2.14 loginSuccess.jsp登录成功界面

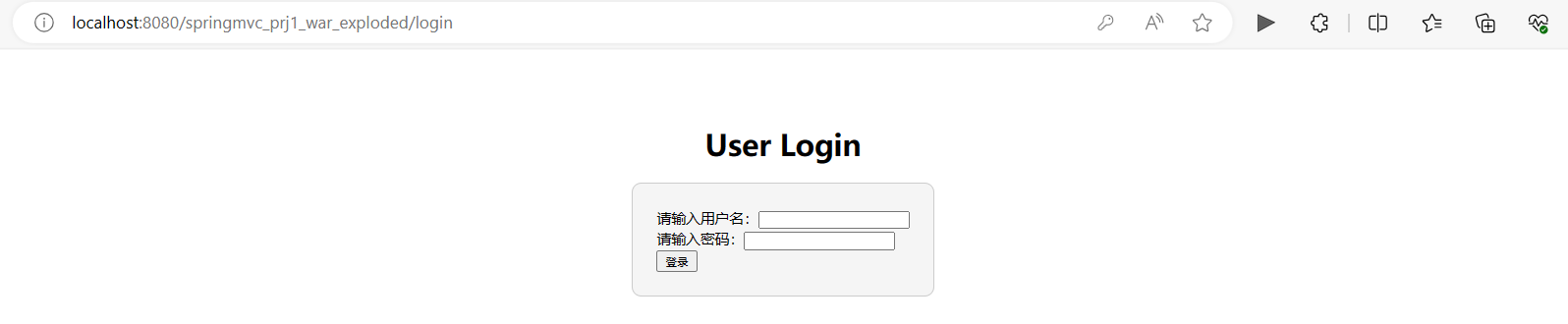


图2.15 登录失败时，回到login.jsp页面

**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**

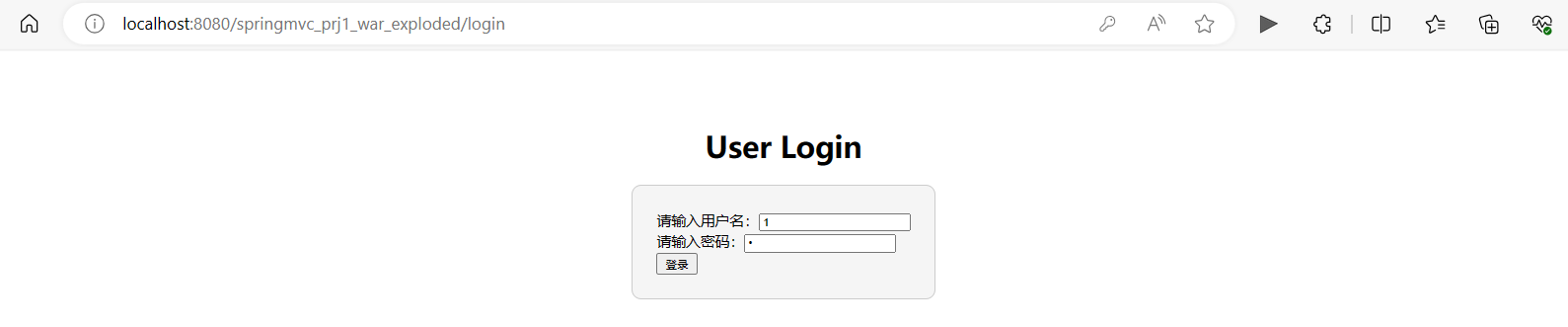


图2.16 login.jsp登录界面

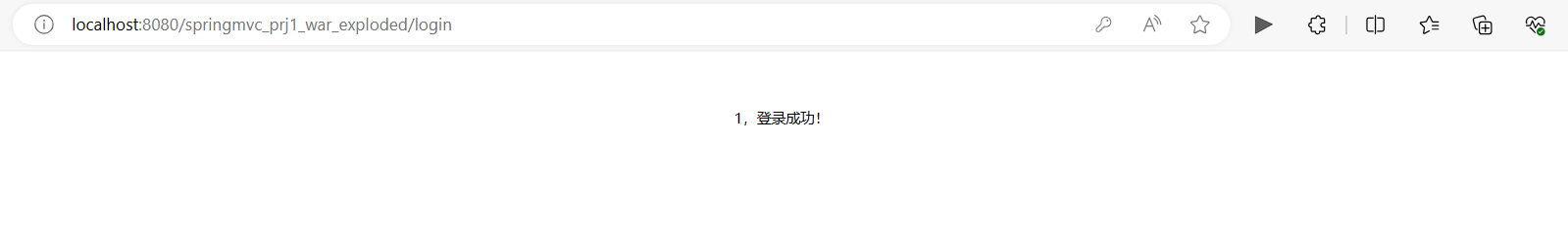


图2.17 loginSuccess.jsp登录界面

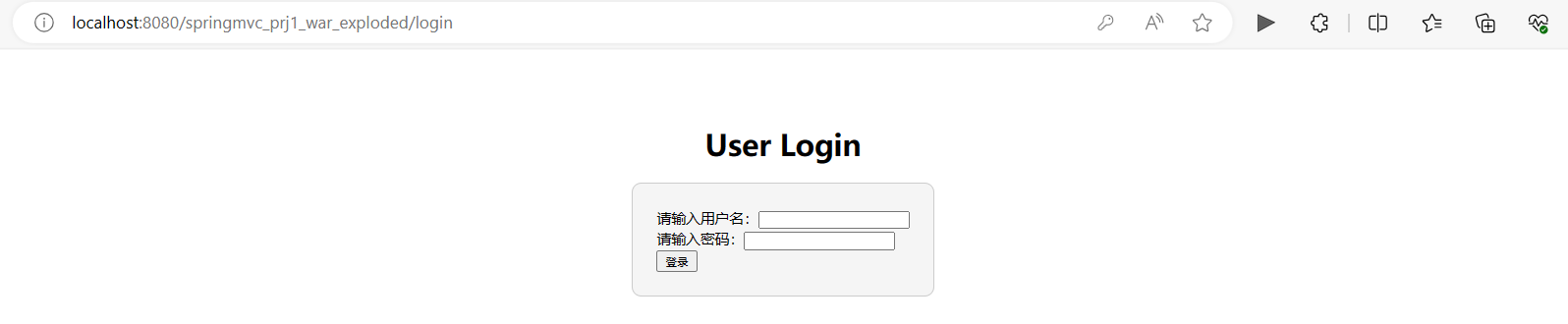


图2.18 登录失败时，回到login.jsp页面

**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**



图2.19 login.jsp登录界面

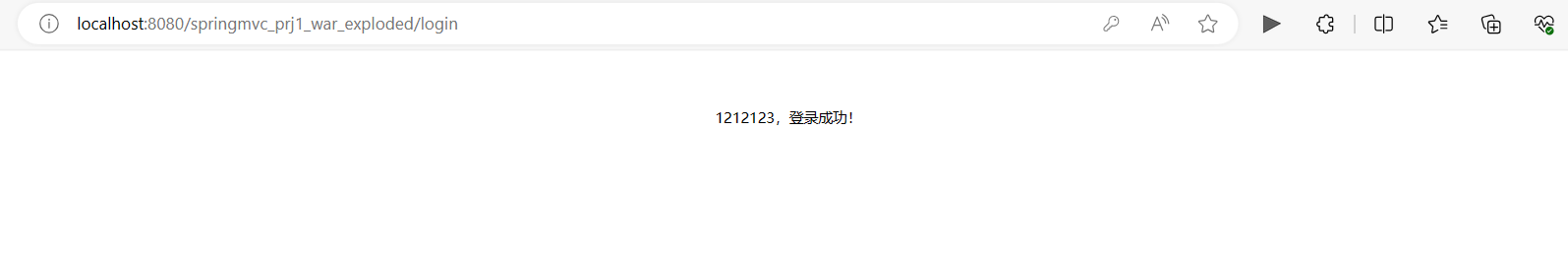


图2.20 loginSuccess.jsp登录界面

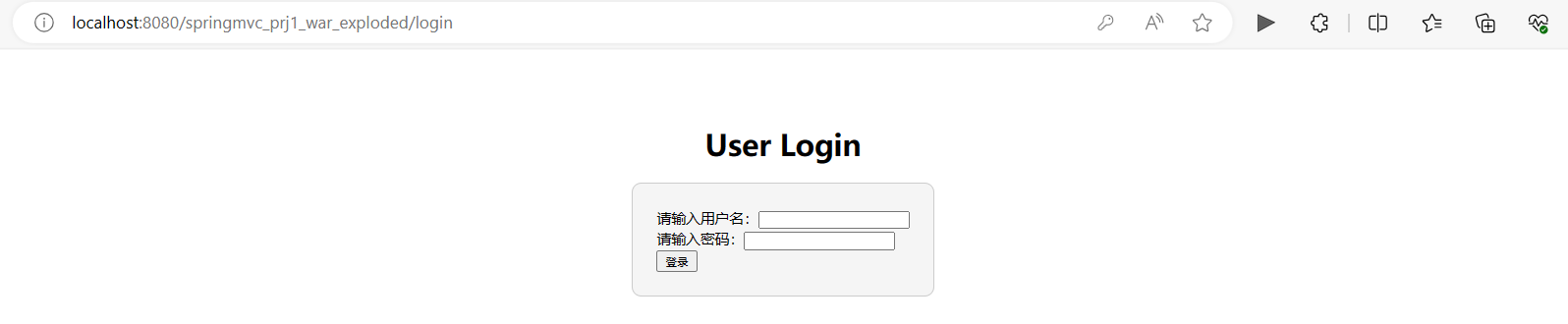


图2.21 登录失败时，回到login.jsp页面

**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\_war\_exploded/login/zjut/zjut，并记录运行结果**

观察到直接进入了loginSuccess.jsp页面，且用户名是zjut

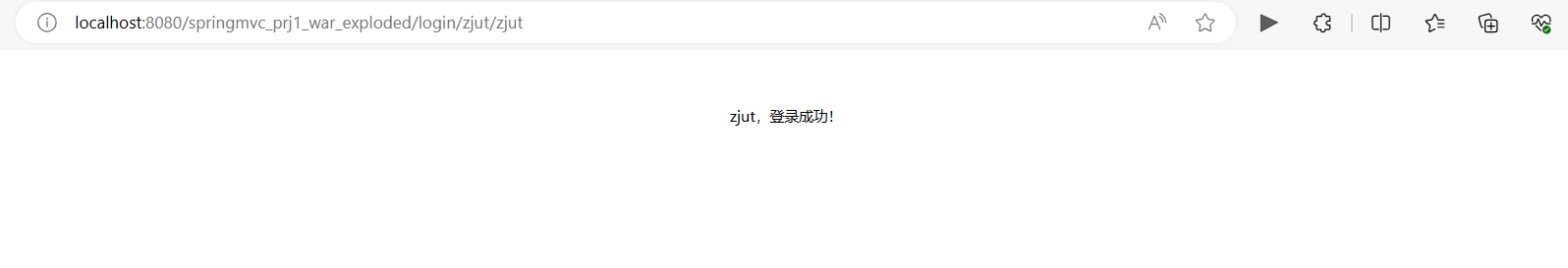


图2.20 loginSuccess.jsp登录成功界面

## 2. 总结基于注解的开发方式的特点

①简化配置: 注解减少了配置文件的需求。

②提高可读性和可维护性: 注解直接附加在代码上，使得代码更加直观和易读。

③更加灵活: 注解通常具有更多的选项和参数，可以为特定场景提供更灵活的配置选项。

④更好的类型安全性: 注解能够提供更好的类型检查和编译期检查。

## 3. 总结@Qualifier 注解的作用

@Qualifier的作用是指定特定 bean 的名称，当多个实现类存在时，通过 @Qualifier 可以指定要注入的具体实现类。

## 4. 总结 Controller 接收请求参数的各种方法及其适用情况

①@RequestParam 注解

适用情况： 用于获取单个请求参数的值。

②使用 @PathVariable 注解

适用情况： 从 URL 中获取参数

③使用 HttpServletRequest 对象

适用情况： 获取所有请求参数

④使用 @RequestBody 注解

适用情况： 用于获取请求体中的参数

⑤使用 @ModelAttribute 注解

适用情况： 用于将表单数据绑定到对象上

## 5. 碰到的问题及解决方案或思考

问题：@Controller注解出现报错“应为注解类型”

解决方案：不导入org.springframework.web.servlet.mvc.Controller，而导入org.springframework.stereotype.Controller

## 6. 实验收获及总结

在这个实验中，我深入了解了Spring MVC框架中Controller的核心作用和不同的请求参数接收方法，通过注解配置@Controller和@Autowired实现了依赖注入，体验了@RequestMapping的URI映射功能，了解了多种参数接收方式，包括@ModelAttribute、@RequestParam、HttpServletRequest等。这次实验拓展了对Spring MVC控制器的认识，为以后的Web应用开发提供了更多的选择和灵活性。