山东大学 计算机科学与技术 学院

现代软件开发技术 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：2020001300 | 姓名： | | 班级 |
| 实验题目：SpringMVC 前后端数据传递 | | | |
| 实验学时：2学时 | | 实验日期： 2023-9-26 | |
| 实验目的：  1.了解SpringMVC子框架的主要配置文件、注解，执行流程。  2. 使用SpringMVC框架，建立简单的网站实例；熟悉模块间信息传递。  3. 会使用表单标签进行数据绑定 | | | |
| 硬件环境： | | | |
| 软件环境： | | | |
| **实验内容与问题回答：**  **1.项目ch2\_2, SpringMVC的java配置**  **Q:（1）项目ch2\_2的java配置文件与ch2\_1中的配置文件的对应关系？**  SpringMVCConfig.java 对应 springmvc-servlet.xml   * + @Configuration 对应 <beans>   + @EnableWebMvc 对应 <mvc:annotation-driven>   + @ComponentScan 对应 <context:component-scan>   + @Bean 对应 <bean>   + InternalResourceViewResolver 对应 <org.springframework.web.servlet.view.InternalResourceViewResolver>   + setPrefix() 对应 <property name="prefix">   + setSuffix() 对应 <property name="suffix">   + addResourceHandlers() 对应 <mvc:resources>   **（2）SpringMVCConfig中，实现的接口是？配置的三项关键性配置内容?**  在SpringMVCConfig中，实现的接口是WebMvcConfigurer。WebMvcConfigurer是一个接口，用于配置Spring MVC的各种选项和特性。它提供了一组回调方法，允许我们自定义Spring MVC的行为。关于SpringMVCConfig中的三项关键性配置内容如下视图解析器的配置：使用@Bean注解定义了一个名为getViewResolver()的方法，返回一个InternalResourceViewResolver对象。  InternalResourceViewResolver用于解析JSP视图。  setPrefix()方法设置了视图的前缀，指定了JSP文件的存放路径。  setSuffix()方法设置了视图的后缀，指定了JSP文件的扩展名。  静态资源的配置  实现了WebMvcConfigurer接口，并重写了addResourceHandlers()方法。  addResourceHandlers()方法用于配置静态资源的处理器。  使用ResourceHandlerRegistry对象的addResourceHandler()方法指定了静态资源的URL路径模式。  使用addResourceLocations()方法指定了静态资源的存放路径。  组件扫描的配置：  使用@ComponentScan注解指定了控制器(Controller)类所在的包路径。  @ComponentScan用于告诉Spring在哪些包下扫描注解，并自动注册相应的Bean。  **（3）WebConfig中，实现的哪个接口？其中的自动启动方法名是？其中设置了哪几项关键性信息？**  在WebConfig中，实现的接口是WebApplicationInitializer。WebApplicationInitializer是一个Spring提供的接口，用于在Servlet容器启动时进行初始化配置。WebConfig中的自动启动方法名是onStartup。在WebConfig中，设置了以下几项关键性信息：注册Spring MVC的Java配置类：使用AnnotationConfigWebApplicationContext创建一个上下文对象ctx。使用ctx.register()方法注册了Spring MVC的Java配置类SpringMVCConfig。关联当前ServletContext：使ctx.setServletContext()方法将当前ServletContext与上下文对象关联起来。注册Spring MVC的DispatcherServlet：使用arg0.addServlet()方法注册了一个名为"dispatcher"的DispatcherServlet。使用new DispatcherServlet(ctx)创建了一个DispatcherServlet实例，并将上下文对象ctx作为参数传入。使用servlet.addMapping()方法设置了DispatcherServlet的URL映射为"/"，即根路径。使用servlet.setLoadOnStartup()方法设置了DispatcherServlet的启动顺序为1  **（4）@RequestMapping("/register") 注解的方法是？返回的String由谁处理？**    返回的String会由Spring MVC的视图解析器处理。  **（5）以下几种形式请求，哪种可以访问？为什么？**  **A.http://localhost:8080/ch2\_2/**  **B.http://localhost:8080/index.jsp**  **C.http://localhost:8080/ch2\_2/index/login**  **D.http://localhost:8080/ch2\_2/jsp/login.jsp**  C只有C符合路径要求  **2. 项目ch2\_3，Controller接收请求参数的常见方式**  **（1）分析UserController.java中，请求处理方法中的参数类型有常见有哪几种？**  HttpServletRequest/HttpServletResponse：  这些参数用于获取和操作HTTP请求和响应的相关信息。  HttpSession：  HttpSession参数用于获取和操作当前会话的相关信息，如会话属性。  Model/ModelMap：  Model或ModelMap参数用于向视图传递数据，可以通过添加属性到Model对象中来传递数据给视图。  @ModelAttribute注解：  通过@ModelAttribute注解修饰的参数，可以从请求中获取特定名称的属性，并将其绑定到方法参数中。  常见的POJO实体类参数：  通过在方法参数中添加POJO实体类参数，可以从请求中获取相应的参数值，并将其绑定到实体类的属性中。  基本数据类型参数：  方法参数也可以是基本数据类型，如int、String等，用于获取请求中特定名称的参数值。  **（2）尝试修改以下接收参数形式，体会请求参数信息的不同接收方式。**  **a.通过实体bean接受参数(UserForm user)**  **UserController类中的register方法，形参为第1种方式。**  **b.通过方法的形参接收请求参数(String uname)**  **修改UserController类中的register方法，形参为第2种方式。**  **c.通过@RequestParam注解接收请求参数(@RequestParam String uname)**  **修改UserController类中的register方法，形参为第3种方式。为什么在主页点“注册”会出现400错误？**  **d.通过@ModelAttribute注解接收请求参数(@ModelAttribute("user") UserForm user)，将请求参数封装成Model数据。**  **修改UserController类中的register方法，形参为第4种方式。**  **（3）在登陆验证成功后的“主页面”main.jsp中 加 ${u.uname }，接收login在session对象中设置的对象属性。**  **Q: u对象是通过什么方式传到该页面的?**  user对象存储在session中，并在main.jsp页面中使用${u.uname}来接收该对象的uname属性值，实现了将用户信息传递到主页面的功能  **（4） Controller中 Log日志对象的作用？**  记录日志信息：通过日志对象，可以将关键的操作、错误信息、异常堆栈等记录到日志文件中，方便后续的日志分析和问题排查。  调试代码：通过在代码中插入日志输出语句，可以在开发和调试过程中输出变量值、方法执行情况等关键信息，帮助定位问题和理解代码的执行流程。  监控系统状态：通过记录关键的系统状态信息，如请求处理时间、请求参数、请求结果等，可以对系统性能进行监控和分析，从而优化系统的性能和稳定性。  日志级别控制：日志对象通常支持不同的日志级别，如DEBUG、INFO、WARN、ERROR等。通过设置适当的日志级别，可以灵活控制日志的输出，避免过度记录或者遗漏重要的日志信息。  **（5）Run on server, 该程序的运行逻辑是？如果注册其他用户名，能登录吗？为什么？**   1. 用户访问登录页面（login.jsp）进行登录操作。 2. 在UserController的login方法中，接收用户提交的登录表单数据(UserForm对象)。 3. 调用UserService的login方法进行登录验证，该方法会检查用户名和密码是否匹配。 4. 如果登录验证成功，将UserForm对象存储到HttpSession中，并跳转到主页面（main.jsp）。 5. 如果登录验证失败，将错误信息添加到Model中，并返回登录页面（login.jsp）重新登录。   对于注册其他用户名的情况，不能登录。原因是在登录验证中，会通过UserService的login方法来验证用户名和密码是否匹配。如果注册了其他用户名，但在登录时使用了不匹配的用户名，登录验证会失败，无法通过验证，因此无法登录。只有使用正确注册的用户名和密码才能成功登录。  **在浏览器地址栏，输入http://localhost:8080/ch2\_3/html/NewFile.html 能访问吗？**  不能访问。原因是在Spring MVC中，默认情况下，WEB-INF目录下的资源是受保护的，无法直接通过浏览器地址栏访问。WEB-INF目录用于存放Web应用程序的配置文件和私有的资源，它们不应该被直接访问  **输入http://localhost:8080/ch2\_3/WEB-INF/jsp/main.jsp能访问吗？为什么？**  也不能访问。同样的原因，WEB-INF目录下的资源是受保护的，无法直接通过浏览器地址栏访问  **Q: register.jsp页面中的user.uname 是通过什么方式传到该页面的?**  在register.jsp页面中，user.uname 是通过ModelAttribute注解的方式传递到该页面的。  **（6）分析项目中拦截器代码实现，（用户可定义http请求处理前后的处理）**  **(a)定义自己的拦截器类，实现HandlerInterceptor接口。**  **实现方法:**  **boolean preHandle(HttpServletRequest request,HttpServletResponse response,Object handler)**  **void postHandle(HttpServletRequest request,HttpServletResponse response,Object handler, ModelAndView modelAndView)**  **（b）配置拦截器**  **在配置文件中，配置拦截器Bean;**  **重写addInterceptors方法注册拦截器**  **查看每次请求，在控制台输出的信息。**  **Q:拦截器对应了第一章中的什么编程技术？**  拦截器对应了AOP（面向切面编程）技术  **3. 项目ch2\_4, 表单标签库与数据绑定，完成以下内容：**  **（1）项目新引入支持JSTL的附加jar包是？**    **（2）WebConfig中的配置：编码过滤器及设置**  **SpringMVCConfig中的配置**    **（3）视图层：**  **userAdd.jsp中使用的JSTL标签及数据绑定**  **userList.jsp中使用的JSTL标签及数据绑定, <%@tablib ...%> ，两者的不同**  两者的不同：  标签库声明：<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>用于引入Spring的表单标签库，而<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>用于引入JSTL的核心标签库。  使用的标签不同：form:form、form:input、form:checkboxes、form:checkbox、form:select、form:options是Spring表单标签库的标签，而<c:forEach>、<c:url>、EL表达式${}是JSTL标签库的标签。  数据绑定方式不同：在userAdd.jsp中使用了Spring的数据绑定方式，通过modelAttribute属性将表单绑定到模型属性上；而在userList.jsp中使用了JSTL的数据绑定方式，通过EL表达式${}获取并输出模型中的属性值。  **(4)pojo包中的User类中的属性，与表单中对象属性的绑定。**    userName：对应表单中的用户名字段，通过表单标签库的form:input标签的path属性进行绑定。  hobby：对应表单中的兴趣爱好字段，通过表单标签库的form:checkboxes标签的path属性进行绑定。在表单中，多个复选框会绑定到数组hobby上。  friends：对应表单中的朋友字段，通过表单标签库的form:checkbox标签的path属性进行绑定。在表单中，多个复选框会绑定到数组friends上。  carrer：对应表单中的职业字段，通过表单标签库的form:select标签的path属性进行绑定。  houseRegister：对应表单中的户籍字段，通过表单标签库的form:select标签的path属性进行绑定。  remark：对应表单中的个人描述字段，通过表单标签库的form:textarea标签的path属性进行绑定。  **(5)sevice包中UserService及其实现类UserServiceImpl，addUser(User u)与getUsers()方法的实现。**      **(6) Controller包中的UserController控制器类，**  **@Autowired 注入**  **以下请求响应方法的实现，及返回、重定向。**  **public String inputUser(Model model)**  **public String addUser(@ModelAttribute User user,Model model)**  **public String listUser(Model model)**    **(7)修改前端页面中的表单，加入使用radiobotton,radiobottons等新标签表示如性别等属性。**  **Q:对比传统的HTML标签< name= id= >,taglib中的标签在使用形式、功能上有什么不同点？**  使用形式：  传统的HTML标签直接在HTML代码中使用，通过设置name和id属性进行标识和访问。  Taglib中的标签通过在JSP页面中声明taglib，并使用特定的标签前缀来引入标签库。然后使用标签库中定义的标签来生成HTML代码。  功能：  传统的HTML标签提供了一些基本的表单元素和控件，如文本输入框、复选框、下拉列表等。它们主要用于展示和接收用户输入的数据。  Taglib中的标签提供了更高级和丰富的功能，如数据绑定、表单验证、生成URL等。它们通过标签属性和表达式来与后台的数据模型进行绑定和操作，可以更方便地处理表单数据。  数据绑定：  传统的HTML标签需要通过name属性来与后台的数据模型进行绑定，然后在后台通过获取请求参数的方式来获取用户输入的数据。  Taglib中的标签通过特定的属性和表达式来与后台的数据模型进行绑定，表单数据会直接传递到后台的数据模型中，无需手动获取请求参数。  表单验证：  传统的HTML标签需要通过JavaScript或后台代码来进行表单验证，包括数据格式、数据完整性等。  Taglib中的标签可以通过属性来指定表单验证规则，可以在前端进行基本的表单验证，也可以在后台进行更复杂的验证逻辑。  **Q:例ch2\_4对比之前的例ch2\_3有什么技术不同？**  例ch2\_3中使用的是JSTL（JavaServer Pages Standard Tag Library）标签库，通过JSTL的核心标签库来进行循环遍历和输出。例ch2\_4中使用的是Spring的表单标签库，通过Spring的表单标签库来生成表单元素、进行数据绑定和表单验证。  **Q:例ch2\_4使用的user存取数据库了吗？设计service层的意义。**  在例ch2\_4中并没有涉及到将User存取数据库的操作。该示例主要展示了如何使用Spring的表单标签库来处理表单数据绑定和验证的功能。  设计Service层的意义在于实现业务逻辑的封装和解耦。Service层作为业务逻辑的中间层，负责处理具体的业务操作，包括数据的处理、转换、组装等。Service层的设计有以下几个意义：  业务逻辑的封装：将业务逻辑封装在Service层中，可以使业务操作更加清晰、可读性更高。通过将不同的业务操作拆分到不同的Service方法中，可以提高代码的可维护性和可扩展性。  业务逻辑的解耦：通过将业务逻辑独立于其他层（如控制层和持久层），实现了不同层之间的解耦。这样可以使每一层的职责更加明确，方便单独修改和测试。  事务管理：Service层通常是事务的切入点，通过在Service层进行事务管理，可以确保业务操作的原子性和一致性。在Service方法上使用事务注解，可以方便地控制事务的提交和回滚。  可复用性：将业务逻辑封装在Service层中，可以实现业务逻辑的复用。不同的控制层可以调用相同的Service方法来完成相同的业务操作，避免重复编写相同的代码  **Q:例ch2\_4,如果职业是"IT民工"，能加入吗？**    **不可以** | | | |
| 思考题解答：  **1.模块间的数据传递常有几种形式？具体的技术方法和相关注释有哪些？**  方法参数传递：数据可以通过方法的参数进行传递。调用方将数据作为参数传递给被调用方的方法，被调用方可以直接使用传递的参数进行操作。这是一种常见的数据传递方式，可以通过方法调用来传递数据。  返回值传递：数据可以通过方法的返回值进行传递。调用方调用被调用方的方法，并接收返回的数据。被调用方在方法内部处理数据，并将处理结果作为返回值返回给调用方。这种方式适用于需要返回结果的场景  共享数据：模块之间可以共享一些公共的数据，这些数据可以被多个模块访问和修改。共享数据可以放在全局变量、缓存、数据库等共享的存储介质中，各个模块可以直接读取和修改这些数据。这种方式适用于需要多个模块共享数据的场景   1. **前端技术与后端分离独立发展以来，出现了哪些新的技术？**   **前端框架React，VUE,** **Angular，后端框架springboot** | | | |