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：安全控制(二），热部署 | | | |
| 实验学时：2学时 | | 实验日期： 2023-12-5 | |
| 实验目的：  1.会使用基于MyBatis的Spring Boot Security操作  2.会使用Spring Boot的热部署。 | | | |
| 硬件环境： | | | |
| 软件环境：  操作系统：windows 10  开发平台与工具：IDEA开发平台、jdk-11.0.1、apache-tomcat-9.0.37 | | | |
| 实验内容与问题回答：  1.打开项目ch7\_2,  与ch7\_1相比,分析如下项目中，哪些有不同？  （1）pom.xml中依赖有什么不同？  ch7\_1项目使用Spring Data JPA和REST，而ch7\_2项目使用MyBatis。此外，ch7\_2项目还包含了Spring Security用于安全性配置。  （2）src/main/resources下的application.properties配置文件有不同吗？  这两个application.properties配置文件存在一些不同之处，主要体现在以下几点：  1.server.servlet.context-path的值：  ch7\_1项目的server.servlet.context-path为/ch7\_1，而ch7\_2项目的为/ch7\_2，这决定了项目的基础URL路径。  2.JPA和MyBatis配置：  ch7\_1项目使用了Spring Data JPA，配置了相关JPA属性，而ch7\_2项目使用了MyBatis，配置了MyBatis的相关属性，如包别名、映射文件路径等。  3.其他：  部分项目特定的配置，如logging.level.com.ch.ch7\_2.repository在ch7\_2中出现，用于设置日志级别。  这些配置的不同反映了两个项目在数据访问、持久化和基础URL路径等方面的差异。  （3）实体类中有什么不同？  ch7\_1项目使用了 JPA 进行持久化，而 ch7\_2项目没有使用 JPA，可能选择了其他数据持久化的方式。这导致了两者在实体类的定义上存在较大的差异。  （4）数据访问层有什么不同？  ch7\_1项目使用了 Spring Data JPA 简化了数据访问层的开发，而ch7\_2项目则使用了 MyBatis，需要手动编写 SQL 查询语句。这反映了两种不同的持久化技术选择，分别是基于 JPA 的 Spring Data 和基于 MyBatis 的手动 SQL。  （5）resources\mappers文件夹中的映射文件的内容？  MyUserMapper.xml中,参数与数据访问层是如何映射传递的？  在XML映射文件和相应的MyUserRepository接口中，参数和数据访问的映射关系如下：  MyBatis XML映射文件（MyUserRepository.xml）：  ResultMap（myResult）：  myResult是用于映射查询结果集到MyUser实体的resultMap。  Select查询（findByUsername）：  findByUsername查询带有id findByUsername 接受string类型的参数（用户名），并使用myResult resultMap将结果集映射到MyUser实体。该查询用于按用户名查找用户信息。  Select查询（findRoleByUser）：  findRoleByUser带有id findRoleByUser 接受integer类型的参数（用户id），并使用Authority resultType将结果集映射到Authority实体。该查询用于查找与用户关联的所有角色（权限）。  Insert查询（save）：  save带有id save 接受MyUser类型的参数，用于注册用户。keyProperty="id"和useGeneratedKeys="true"属性表示应将生成的主键（id）填充回MyUser对象的id属性中。  Insert查询（saveUserAuthority）：  saveUserAuthority带有id saveUserAuthority 接受两个参数（user\_id和authority\_id），用于保存用户与其权限之间的关联。  MyUserRepository接口：  方法签名：  MyUserRepository接口声明了几个方法，对应于XML文件中定义的SQL查询。每个方法表示一个特定的数据访问操作。  注解：  @Mapper：指示这个接口是一个MyBatis Mapper接口。  方法参数：  接口中的方法参数对应于XML查询中使用的参数。例如，findByUsername方法接受一个String参数用于用户名。  方法返回类型：  方法的返回类型对应于查询的期望结果类型。例如，findByUsername方法返回MyUser，findRoleByUser方法返回List<Authority>。  （6）service层中的不同？  register()方法中为什么要分别调用save()和saveUserAuthority()方法来实现用户注册？  在两个项目的service层中，主要的不同点在于使用了不同的持久化框架（Spring Data JPA 和 MyBatis），因此对数据的操作方式有所差异。以下是两个项目service层的不同之处：  ch7\_1项目（Spring Data JPA）： Spring Data JPA 提供了方便的 save 方法用于保存实体对象，但是对于关联表的操作需要手动创建相关实体对象。在这里，权限信息是通过手动创建权限对象，然后添加到用户的权限列表中的方式实现的。  ch7\_2项目（MyBatis）： MyBatis 需要手动编写 SQL 语句，因此在这里通过自定义的 MyBatis 方法 saveUserAuthority 直接将用户的权限信息插入关联表中。  （7）controller层中有不同吗？  ch7\_1项目的包名是 com.ch.ch7\_1.controller。  ch7\_2项目的包名是 com.ch.ch7\_2.controller。  ch7\_1项目导入的实体类和服务类路径是 com.ch.ch7\_1.entity 和 com.ch.ch7\_1.service。  ch7\_2项目导入的实体类和服务类路径是 com.ch.ch7\_2.entity 和 com.ch.ch7\_2.service。  （8）前端页面有不同吗？  Thymeleaf注释方式：  在两份代码中都使用了Thymeleaf的注释方式，即 th:inline="javascript"，但是在第一份代码中使用的是 th:inline="javascript"，而在第二份代码中使用的是 th:inline="text"。  这两种方式都可以用于将JavaScript代码嵌入Thymeleaf模板中，不同的是 th:inline="javascript" 会将JavaScript代码中的特殊字符进行转义，而 th:inline="text" 不会进行转义。  一般来说，推荐使用 th:inline="javascript"，因为它可以防止一些潜在的安全问题。  其他方面的代码结构和Thymeleaf语法使用都是一致的。  （9）安全控制包中有不同吗？  在提供的代码中，两个项目的安全控制包（security包）基本上是相似的，但存在一些细微的差异。以下是这些差异的摘要：  1.角色名称差异:  在第一个项目中，MySecurityConfigurerAdapter 中使用了 .hasRole("USER") 来授权用户访问 /user/\*\* 的路径，而在第二个项目中使用了 .hasAnyRole("USER","ADMIN")。  同样，对于 /admin/\*\* 的路径，第一个项目使用了 .hasAnyRole("ADMIN", "DBA")，而第二个项目使用了 .hasAnyRole("ADMIN","USER")。  2.角色匹配差异:  在第一个项目中，MyAuthenticationSuccessHandler 的 getTargetURL 方法检查 ROLE\_USER 和 ROLE\_ADMIN 角色，而在第二个项目中检查 ROLE\_USER 和 ROLE\_ADMIN 角色。  这些差异可能是有意为之，具体取决于你的项目需求。总体而言，两个项目的安全控制方案都使用了Spring Security，但在角色授权的具体实现上有一些不同。  （10）启动类中有什么不同？  在提供的两个项目的启动类中，主要区别在于@MapperScan注解的配置。以下是两个项目启动类的不同之处：  Ch71Application (第一个项目):  在第一个项目中，并没有使用@MapperScan注解。这可能意味着该项目使用了默认的MyBatis配置，MyBatis会自动扫描com.ch.ch7\_1包及其子包下的Mapper接口。  Ch72Application (第二个项目):  在第二个项目中，使用了@MapperScan("com.ch.ch7\_2.repository")注解，显式地指定了MyBatis扫描Mapper接口的包路径为com.ch.ch7\_2.repository。  总体而言，这两个启动类的主要不同之处在于第二个项目显式配置了MyBatis的Mapper扫描路径，而第一个项目则可能依赖于默认的MyBatis配置。  测试运行：http://localhost:8080/ch7\_2/  Q: 运行效果与ch7\_1是否相同？  相同  Q: 认证通过前、认证通过后，访问 http://localhost:8080/ch7\_2/login ，会转向哪个页面？为什么？  认证通过前转向login界面    认证通过后转向登陆成功界面  Q：登录普通用户后，点“安全退出”，为何地址栏显示 http://localhost:8080/ch7\_2/login?logout ，页面显示红字“用户已注销成功！”？  查看 login.html 中的代码可以发现使用的是red颜色显示：  Q:将service层 UserServiceImpl.java中的logout()方法，最后一行 return "redirect:/login?logout"; 改为 return "/login?logout"; 可以吗？为什么？  不可以，会转到错误页面，因为去掉了重定向关键字 redirect    Q:登录管理员账号admin 后，不能访问普通用户user的页面 http://localhost:8080/ch7\_2/user/loginSuccess,会转哪个网页？为什么？  会转到拒绝访问页面  2.练习Spring Boot的热部署。  打开课本项目ch9\_1。  （1）分析pom.xml文件，删除其中的mysql和mybatis依赖（用不到）。  Q:其中支持热部署的依赖是哪一条？  Q:热部署实现原理是什么？    热部署的原理：  热部署是指在不停止应用程序的情况下，将新的代码、资源文件等直接应用到运行中的应用程序。Spring Boot的spring-boot-devtools模块提供了开发时的热部署支持。  实现原理主要包括以下几个关键点：  1.Classloader的隔离： Spring Boot DevTools 使用两个不同的类加载器来实现热部署。一个类加载器用于加载应用程序的类，另一个类加载器用于加载不会改变的第三方库。  2.文件系统监听： DevTools会监视项目的classpath，当classpath下的文件发生变化时，会触发应用程序的重新加载。  3.应用程序重启： 当检测到文件发生变化时，DevTools触发应用程序的重新启动。在重新启动时，Spring Boot尽力保留应用程序的状态，以减少不必要的开发周期中的停顿。  4.触发方式： 可以通过多种方式触发热部署，包括IDE的保存操作、文件系统的更改等。  总的来说，Spring Boot DevTools通过使用类加载器的隔离和文件系统监听，实现了在开发过程中的热部署，提高了开发效率。在开发阶段，这允许开发者在修改代码后快速看到变化，而无需手动重启整个应用程序。  （2）打开src/main/resources下的application.properties配置文件。  Q:为什么其中没有关闭模板引擎缓存的设置\*.cache=false？  使用模板引擎的页面默认是开启缓存的，如果修改了页面内容则刷新页面是得不到修改后的页面的效果的，因此关闭。  （3）分析控制器类  打开src/main/java下com.ch.ch9\_1包中的TestDevToolsController.java，分析其中的代码，  Q: @RequestMapping注解的请求路径对应的方法？  对应的方法是：testDevTools  Q: @RestController注解怎样影响 return 的执行结果？与@Controller的不同？  将返回的执行结果直接指定为 JSON 格式，使用@Controller 还需要添加@RespondBody 注解 。  （4）测试运行项目  （在STS4中选择Project菜单中的Build Automatically选项）  运行src/main/java下com.ch.ch9\_1包中Ch91Application.java(run as Java Application)，  观察控制台窗口中显示的的信息，看项目是否正常运行。  项目正常运行。  （a）用浏览器访问“http://localhost:8080/testDevTools”观察返回的字符串信息。    （b）修改src/main/java下com.ch.ch9\_1包中的TestDevToolsController.java中的return语句，如：  return "test DevTools 888";  修改存盘后，观察观察控制台窗口中显示的的信息，Q:与上次的信息有什么不同？    （c）用浏览器刷新访问“http://localhost:8080/testDevTools”观察返回的字符串信息。  Q:为何不需要手动Relaunch项目，返回的字符串是否已变化？    字符串变化，因为使用了热部署创建一个Restart Class Loader加载更新的类，只加载了部分更新的类，实现了较快重启。 | | | |
| 思考题解答：  无 | | | |