山东大学 计算机科学与技术 学院

现代软件开发 课程实验报告

|  |  |  |
| --- | --- | --- |
| 学号：2020001300 | 姓名： | 邮箱： |
| 实验题目：实验二 基于注解的依赖注入 | | |
| 实验过程中遇到和解决的问题：  （记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）  1. 项目ch1\_8，Spring的数据库编程  （1）安装MySQL。  首先在课程网站下载MySql,安装。记下用户名root和密码，服务端口。  （2）创建数据库，spingtest，并在其中创建表user.  可利用命令行：   1. 登录：mysql -u root -p     (b)创建数据库: create database springtest default charset utf8 collate utf8\_general\_ci;  (c)打开数据库: use springtest;  (d)创建表：create table user(uid int(10) primary key auto\_increment,uname varchar(20),usex varchar(10));    （3）打开项目ch1\_8,练习jdbcTemplate对数据库的访问。  （a）该项目WEB-INF/lib下新增了哪些支持包？  答：myaql-connector-java-5.1.45-bin.jar;spring-jbdc-5.1.4.RELEASE.jar;spring-tx-5.1.4.RELEASE.jar;  （b）  (c) 分析entity包中的MyUser.java，对照其中的属性与MySQL中的user表中的属性：  答：MyUser.java中定义了MyUser类，其中private Integer uid对应user表中的uid int(10) primary key auto\_increment，private String uname对应表中的uname varchar(20)，private String usex对应表中的usex varchar(10)。  （d）分析dao包中的接口及实现类文件。  在dao包中，可能包含接口和实现类文件，用于定义和实现与数据库交互的操作。Dao接口文件定义了与数据库交互相关的方法，使用JdbcTemplate来执行SQL语句和处理结果。Dao实现类文件实现了Dao接口定义的方法，具体实现了与数据库的交互逻辑，例如执行SQL语句、处理查询结果集等。它们依赖于JdbcTemplate来执行数据库操作。  （e）分析service包中的接口及实现类文件。  Service接口文件定义了业务逻辑相关的方法，例如处理数据、业务计算等。它们调用Dao层提供的方法来实现具体的业务逻辑。  Service实现类文件实现了Service接口定义的方法，具体实现了业务逻辑的处理，例如调用Dao层的方法执行数据库操作、处理数据等。  （f）分析config包中的配置文件  config包中SpringJDBCConfig.java文件创建配置类，通过@Value注解注入属性文件中的4个参数内容而实现与mysql的连接（利用驱动实现），并且在该文件中配置了jdbcTemplate实例实现相关方法，在最后为数据源添加事务管理器来处理mysql中的数据。    如果更换其他的数据库，此项目应该如何改？  1.替换数据库驱动：根据新数据库的类型，将原有数据库驱动的依赖替换为新数据库的驱动依赖。例如，如果从MySQL切换到PostgreSQL，需要将原来的mysql-connector-java.jar替换为相应的postgresql.jar。  2.修改连接信息：在属性文件（如jdbc.properties）中修改数据库连接信息，包括驱动类名、数据库URL、用户名和密码等。根据新数据库的要求，进行相应的修改。  3.调整SQL语句：不同的数据库可能有不同的SQL语法和函数支持。因此，需要检查项目中的SQL语句，确保其与新数据库兼容。如果存在不兼容的语句，需要进行相应的调整。  4.数据库特性和限制：不同的数据库具有不同的特性和限制，例如事务处理、并发控制、存储过程等。在更换数据库后，需要重新评估项目中对这些特性的使用和依赖，确保其在新数据库中的支持和兼容性。  5.数据库迁移和数据转移：如果项目中已经存在数据，需要考虑如何进行数据库迁移和数据转移。可以使用数据库迁移工具（如Flyway或Liquibase）来管理数据库结构的变更，并根据需要将现有的数据迁移到新数据库中。  针对项目ch1\_8，通过实践解答如下问题：  （1） 在SpringJDBCConfig.java配置文件中，  （a）怎样配置PropertySource 对应jdbc.property属性文件；该属性文件独立出来的好处？  使用注解@Value进行相关属性的注入，并指定属性文件的路径，为项目配置数据源  好处：将属性文件独立出来的好处是可以将不同环境下的配置信息（如数据库连接信息）与代码分离，方便在不同环境中进行配置的切换和管理。通过将配置信息独立出来，可以避免在代码中硬编码配置信息，提高了代码的可维护性和灵活性。  （b）注解声明了哪两个Bean，它们是怎样注入的？  注解声明了两个Bean：DataSource和JdbcTemplate。它们是通过在方法上使用@Bean注解来声明的，并通过依赖注入的方式进行注入。  （2）jdbcTemplate实例的注入及应用  （a）导入的包与之前项目有什么不同？对不同版本的数据库，需要换什么？  导入的包与之前项目有所不同，但主要的差别在于数据库驱动的依赖。  需要将mysql-connector-java.jar替换为适用于适合版本的驱动包。  （b）配置文件中，4个注解@Value的作用？  @Value("${jdbc.driverClassName}")：将属性文件中jdbc.driverClassName对应的值注入到driverClassName变量中。  @Value("${jdbc.url}")：将属性文件中jdbc.url对应的值注入到url变量中。  @Value("${jdbc.username}")：将属性文件中jdbc.username对应的值注入到username变量中。  @Value("${jdbc.password}")：将属性文件中jdbc.password对应的值注入到password变量中。  （c）实体类MyUser与MySQL中的user表是怎样对应的？query方法返回的List<MyUser>对象是怎样遍历的？  MyUser类中的private Integer uid对应user表中的uid int(10) primary key auto\_increment，private String uname对应表中的uname varchar(20)，private String usex对应表中的usex varchar(10)。  query方法所返回的List<MyUser>类利用循环将List中的各节点进行遍历：    （d）DAO层对应的Bean声明注解是？  DAO层对应的Bean声明注解是@Repository。  （e）"insert into user values(null,?,?)", 对应主码处为何可以用null?  为了保证主码唯一性，设置为null则mysql会自动设置未使用过的主码数值，如果不为null而是具体的数据，则很可能会使用到相同的主码而不符合主码特性。  2. 项目ch1\_9，基于@Transactional注解的声明式事务管理  熟悉类、方法中基于@Transactional注解声明式事务处理。(基于项目ch1\_8 jdbcTemplate访问MySQL数据的应用修改)   1. 打开项目ch1\_9, 支持声明式事务处理的包是？   支持声明式事务处理的包为service包   1. 在配置文件中怎样开启事务管理支持和为数据源添加事务管理的Bean？   使用注解@EnableTransactionManagement来开启声明式事务管理支持，而为数据源添加事务管理的实例则是通过构造数据员事务管理对象来实现：     1. 在业务Service中，插入的哪两条数据导致异常？   在user表中uid为主键，param5与param6对应插入uid的数据正好相等，则在进行update操作时会出现插入两个相同uid数据的情况，因此出现异常    （4）试验如果用try...catch捕捉异常，事务还回滚吗？  试验在try...catch中的仍然可以引发事务回滚的两种方式：  方式1：在catch语句中添加 throw new RuntiomException();  方式2：添加 TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();  Q:使用try...catch 对比前面不用try...catch，有什么区别？  注：MySQL中，表要使用支持事务的引擎，如innoDB（默认），如果是myisam，事务不起作用。  答：如果使用try...catch捕捉异常，事务仍然会回滚。事务的回滚是通过Spring的事务管理机制实现的，与异常的捕捉方式无关。  方式1：在catch语句中添加 throw new RuntimeException()。通过抛出RuntimeException或其子类的异常，可以触发事务回滚。  java  Copy  try {  // 执行业务逻辑  } catch (Exception e) {  // 处理异常  throw new RuntimeException("发生异常，触发事务回滚");  }  方式2：添加TransactionAspectSupport.currentTransactionStatus().setRollbackOnly()。通过调用TransactionAspectSupport类的currentTransactionStatus()方法获取当前事务的状态，并调用setRollbackOnly()方法将事务标记为回滚状态。  Copy  try {  // 执行业务逻辑  } catch (Exception e) {  // 处理异常  TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();  }  使用try...catch对比前面不使用try...catch的区别如下：  不使用try...catch：异常抛出后，由Spring的事务管理机制捕获异常并触发事务回滚。  使用try...catch：异常被捕获后，可以在catch块中进行自定义的异常处理操作。如果在catch块中抛出RuntimeException或调用setRollbackOnly()方法，仍然可以触发事务回滚。否则，如果没有再次抛出异常或调用回滚方法，事务将继续提交。  3.项目ch2\_1，开发一个SpringMVC简单应用  （1）SpringMVC项目新增依赖的JAR包是?  该项目的配置文件位于？  该项目的配置文件通常位于src/main/webapp/WEB-INF目录下，命名为web.xml。    （2）基于xml配置；了解Spring MVC项目的执行流程。  （a）web.xml中，DispatcherServlet的配置中，以下参数的含义？  <servlet> ...  <load-on-startup>1</load-on-startup>....  <servlet-mapping>...  <url-pattern>/</url-pattern>...    Q:配置文件命名规则？  <servlet>标签用于配置Servlet，其中<servlet-name>指定Servlet的名称，<servlet-class>指定Servlet的类名。  <load-on-startup>指定Servlet的启动顺序。当值为1时，表示在应用启动时就加载该Servlet。  <servlet-mapping>标签用于指定Servlet的映射规则，其中<url-pattern>指定了该Servlet所处理的URL模式。在本例中<url-pattern>/</url-pattern>`表示所有的请求都交给DispatcherServlet处理。  配置文件的命名规则是根据Servlet规范的要求，命名为web.xml。  （b）springmvc-servlet.xml中，  <context:component-scan base-package="包名"/>  ...  <bean ...id="internalResourceViewResolver">  <property name="prefix" value="..." />  <property name="suffix" value="..." />  springmvc-servlet.xml中的相关配置如下：  <context:component-scan>用于指定需要扫描的基础包路径，可以自动扫描并注册带有特定注解的类（例如@Controller、@Service等）。  <bean>用于定义Spring的Bean，其中id属性指定Bean的名称，<property>标签用于设置Bean的属性。  （3）controller层  （a）IndexController类中的关键注解有哪些？  两个：@Controller：将该类标识为一个控制器。  @RequestMapping：用于映射请求URL和方法的关系。  （b）IndexController类中的方法是怎样与客户端主页index.jsp中的请求相对应的？  IndexController类中的方法与客户端主页index.jsp中的请求相对应是通过@RequestMapping注解来实现的。方法上的@RequestMapping注解指定了方法处理的URL路径，当客户端请求与该路径匹配时，对应的方法将被调用。  （c）web访问请求的分级映射：  @RequestMapping("/index")  public class IndexController {  @RequestMapping("/login")  public String login() { …  Q:方法返回的是什么？通过哪个第三方转换为视图展现？视图除了JSP，用其他技术可以吗？  方法返回的是视图名称，是一个逻辑视图名称（如"login"/“register”）。Spring MVC通过视图解析器（如InternalResourceViewResolver）来将视图名称解析为具体的视图资源。视图解析器将逻辑视图名称与实际的视图资源（如JSP文件）进行映射，然后将其渲染为最终的HTML响应。  除了JSP，Spring MVC还支持其他的视图技术，如Thymeleaf、Freemarker、Velocity等。可以根据项目需求选择适合的视图技术，并在配置文件中配置相应的视图解析器。例如，使用Thymeleaf作为视图技术，可以添加Thymeleaf的相关依赖，并在配置文件中配置Thymeleaf的视图解析器。  （4）测试  （a）在右边栏项目名称ch2\_1上右击，选择 Run as/ Run on Server  当控制台显示 信息: Server startup in [XXX] milliseconds (启动成功。Tomcat 需按上次实验步骤先配置好）  （b）打开浏览器，在地址栏输入 <http://localhost:8080/ch2_1/>    参照《Spring Boot从入门到实战\_实验指导书》，完成其中：  实验三 Spring的事务管理(基于注解的事务管理）  项目代码 shiyan3  调试运行。 | | |
| 结果分析与体会：  通过完成基于注解的依赖注入的实验，我不仅掌握了注解的使用方法和依赖注入的原理，还加深了对Spring框架的理解。对我今后的Java开发工作有不小的帮助，使我能够更加高效地利用Spring框架进行开发，并且提升了我的实践能力。针对本次实验，我对注解的体悟主要有一下几点：   1. 注解简化了依赖注入的配置：相比传统的XML配置方式，使用注解进行依赖注入能够大大简化配置的过程。通过在需要注入的类和属性上使用相应的注解，可以直观地表达出依赖关系，使代码更加简洁和可读。同时，注解还提供了更灵活的注入方式，例如根据类型、名称等进行注入，满足了不同场景下的依赖需求。 2. 注解提高了代码的可维护性和可扩展性：使用注解可以将组件的角色和职责清晰地体现出来，使代码的结构更加清晰。不同的注解用于标记不同类型的组件，如@Component用于通用组件、@Repository用于数据访问组件、@Service用于服务组件、@Controller用于控制器组件等。这样的划分使得代码更易于理解和维护，并且有助于后续的扩展和修改。   （3）Spring框架定义了一系列常用注解，如@Component、@Repository、@Service、@Controller和@Autowired等。这些注解覆盖了常见的开发场景，能够满足不同组件的需求。通过深入学习和实践这些注解的使用方法，我对Spring框架的功能和特性有了更深入的了解。 | | |