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： Spring Boot异常处理及JSP支持，Spring Data JPA（1） | | | |
| 实验学时：2学时 | | 实验日期： 2023-11-7 | |
| 实验目的：  1.能使用Spring Boot异常处理的多种技术途径  2.能设置Spring Boot平台支持JSP技术  3.会使用Spring Boot JPA实现数据库简单查询 | | | |
| 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.练习使用Spring Boot的多种异常处理技术手段  用STS4打开课本项目ch5\_3。  1.1 练习统一显示异常、错误显示页面error.html  （1）分析src/main/resources/templates目录下的index.html和error.html页面；  Q:error.html中使用了哪几种属性？  xmlns:th="http://www.thymeleaf.org"：该属性定义了Thymeleaf命名空间，允许在HTML标签中使用Thymeleaf的属性。  th:href：这是Thymeleaf中用于处理超链接的属性，用于动态设置href属性的值。在这里，它被用于引用CSS文件。  th:text：该属性用于设置标签的文本内容，可以通过Thymeleaf表达式动态地插入数据。在这里，它被用于设置span和div标签的文本内容。  Thymeleaf表达式 ${status}、${#dates.format(timestamp,'yyyy-MM-dd HH:mm:ss')}、${message}、${error}：这些表达式用于在页面中插入动态生成的内容，具体内容将在运行时由服务器生成。  （2）分析控制器中的代码  (a)打开src/main/java下com.ch.ch5\_3.controller包中GlobalExceptionHandler Controller.java，将其中所有代码行注释掉(选中，按ctrl/)，存盘。  (b)打开com.ch.ch5\_3.controller包中TestHandleExceptionController.java，  Q:其中有几种对应index.html的请求处理？抛出了哪几种异常？  Q:index.html中的nofound请求在controller中有响应请求处理吗？  /db：处理数据库异常，抛出了 SQLException。  /my：处理自定义异常，抛出了 MyException。  /no：处理未知异常，抛出了通用的 Exception。  （3）运行  运行src/main/java下com.ch.ch5\_3包中Ch53Application.java(run as Java Application)，  在浏览器地址栏输入：http://localhost:8080/ch5\_3/，点击网页中的不同链接，分析跳转网页中显示的各种信息。  输入<http://localhost:8080/ch5_3/>显示的网页如下：    之后我们分别点击该网页中的四个不同的链接，得到的结果如下面的四个页面所示：          Q:将path属性加入error.html页面，存盘，再次运行,显示什么？  1.2 练习用@ExceptionHandler注解处理异常  （1）在控制器中加入异常处理代码  打开src/main/java下com.ch.ch5\_3.controller包中BaseController.java，将其中@ExceptionHandler(value=Exception.class)注解及其下的整个方法代码复制粘贴至TestHandleExceptionController类中，存盘。  Q：代码中几个return语句返回的页面是？    @ExceptionHandkler(value=Exception.class)注解下的整个方法如上图所示。  其中return “sqlError”;语句返回的页面是sqlError.html页面；  return “myError”;语句返回的是myError.html页面；  return “noError”;语句返回的是noError.html页面；  打开src/main/resources/templates目录下的sqlError.html, myError.html, noError.html，分析代码。  **sqlError.html页面：**  这是一个HTML页面，用于显示 SQL 异常。在这个页面中，没有使用任何后端框架或模板引擎（如Thymeleaf）。该页面包含一个标题（<title>）和一个显示文本（"SQL异常"）的主体部分（<body>）。  **myError.html页面：**  这是一个HTML页面，用于显示自定义异常信息。与sqlError.html文件类似，这个页面同样没有使用任何后端框架或模板引擎（如Thymeleaf）。该页面包含一个标题（<title>）和一个显示文本（"自定义异常"）的主体部分（<body>）。  **noError.html页面：**  这是一个HTML页面，用于显示未知异常信息。同样，与前两个页面类似，这个页面也没有使用任何后端框架或模板引擎（如Thymeleaf）。该页面包含一个标题（<title>）和一个显示文本（"未知异常"）的主体部分（<body>）。  （2）再次运行Ch53Application,在浏览器地址栏输入：http://localhost:8080/ch5\_3/，点击网页中的不同链接，  Q:结果网页中显示与1.1中有什么不同？  Q:为什么点网页中的“404错误”还是和1.1中显示相同？            1.3 练习用继承方式统一处理异常  （1）将刚才粘贴至TestHandleExceptionController类中@ExceptionHandler及其下方法代码删除。  （2）在类头TestHandleExceptionController后添加代码 extends BaseController，继承父类BaseController，如下：  public class TestHandleExceptionController extends BaseController{  （3）查看父类BaseController中的代码  （4）再运行Ch53Application,在浏览器地址栏输入：http://localhost:8080/ch5\_3/，点击网页中的不同链接，  Q: 跳转网页中显示的与1.2是否相同？why?  相同  1.4 练习用@ControllerAdvice注解统一处理全局异常  （1）将刚才添加到TestHandleExceptionController类后的“extends BaseController”代码删除，保存。  （2）将GlobalExceptionHandlerController.java中注释掉的所有代码行恢复(选中，按ctrl/)，保存。  （3）再运行Ch53Application,在浏览器地址栏输入：http://localhost:8080/ch5\_3/，点击网页中的不同链接，  Q: 看跳转网页中显示的与1.2是否相同？          跳转到的页面结果如上图所示，可以看到四个链接的结果还是和1.2中的结果一致。  Q: error.html网页的作用。  Q: 本节异常统一处理 与 1.3的继承 相比，两种技术有不同？本例是否了实现模块之间的解耦？  Error.html网页是一个使用Thymeleaf模板引擎的HTML页面，主要用于处理404错误（页面未找到）。让我们逐步解释它的结构和作用：  <!DOCTYPE html>：指定HTML5文档类型。  <html xmlns:th="http://www.thymeleaf.org">：定义HTML元素，并引入Thymeleaf命名空间，允许在HTML中使用Thymeleaf的模板语法。  在异常处理方面，使用两种不同的技术：使用注解 @ExceptionHandler 处理异常和使用继承的方法，涉及到一些设计和架构的考虑。以下是两种方法的比较：  **使用 @ExceptionHandler 处理异常：**  通过在单独的方法上使用 @ExceptionHandler 注解，可以将异常处理逻辑从正常的业务逻辑中分离出来，实现了一种分层的设计。可以根据异常的类型有选择性地处理不同类型的异常，提高了灵活性。适用于局部异常处理，比如在特定的控制器或控制器类中处理异常。  解耦性：相对较高的解耦性，因为异常处理逻辑被封装在单独的方法中，不干扰正常的业务逻辑。  **使用继承的方法处理异常：**  通过创建一个基础的控制器类，让其他控制器继承该类，实现了对异常的统一处理。所有控制器共享相同的异常处理逻辑，确保了一致性。适用于全局的异常处理，尤其是对于整个应用程序级别的异常。  解耦性：相对较低的解耦性，因为异常处理逻辑被集中在一个基础类中，可能会影响到所有继承了该类的控制器。  解耦性比较：  使用 @ExceptionHandler 处理异常的方式更具解耦性，因为异常处理逻辑与业务逻辑分开，可以更容易地修改或替换异常处理逻辑而不影响正常的业务逻辑。  使用继承的方法，虽然实现了一致的异常处理，但在一些情况下可能导致较低的解耦性，因为修改基础控制器的异常处理逻辑可能会影响到所有继承了该基础类的控制器。  2.练习设置Spring Boot平台支持JSP。  停止之前运行的项目ch5\_3。用STS4打开课本项目ch5\_4，分析如下文件：  （1）pom.xml，与ch5\_3项目中的pom.xml文件相比，  Q:增加了什么依赖？  添加了Servlet依赖；  添加了Tomcat依赖；  添加了Jasper依赖，它是Tomcat的使用引擎，可以将Web应用在内嵌的Tomcat下运行；  （2）打开src/main/resources/中的application,perpertie，与之前的项目相比，  Q:增加了哪些内容？  spring.mvc.view.prefix=/WEB-INF/jsp/ 和 spring.mvc.view.suffix=.jsp：  设置Spring MVC视图解析器的前缀和后缀。  spring.mvc.view.prefix定义了视图文件的前缀，这里设置为/WEB-INF/jsp/，表示视图文件在/WEB-INF/jsp/目录下。  spring.mvc.view.suffix定义了视图文件的后缀，这里设置为.jsp，表示视图文件的后缀为.jsp。  这种配置结构通常适用于传统的Java Web项目，其中JSP（JavaServer Pages）作为视图层的一部分，而/WEB-INF/目录下的内容不直接对外暴露，以增强项目的安全性。  （3）查看src/main/resources/templates中,  Q:还有视图页面吗？  没有视图页面了  （4）打开src下的main/webapp/WEB-INF/jsp中的index.jsp文件，分析其中的代码。  Q:对比项目ch5\_2中index.html文件,两者现实的效果一样吗？各自使用的技术有什么不同？  **对index.jsp的分析如下：**  1.页面头部信息：  设置页面字符编码为UTF-8。引入JSTL标签库，使用<%@ taglib prefix="c" uri= "http://java.sun.com /jsp/jstl/core" %>。获取应用的基本路径（base path）。  2.页面主体部分：  使用Bootstrap样式，引入了css/bootstrap.min.css和css/bootstrap-theme.min.css。页面包含一个具有样式的面板，展示一个标题。  3.图书列表展示：  使用JSTL的<c:forEach>标签循环遍历${books}集合，其中${books}是在控制器中设置的图书集合。展示每本书的封面、书名、作者、ISBN、价格、出版社等信息。使用EL（Expression Language）表达式${aBook.xxx}或${book.xxx}获取图书对象的属性。这个JSP页面主要用于展示图书列表，利用JSTL的标签循环遍历展示多本书的信息。EL表达式用于获取和展示图书对象的属性。  **两个文件的对比：**  上一个项目ch5\_2中的index.html文件中的代码是使用Thymeleaf模板引擎的HTML文件，与ch5\_4项目中的使用JSP的index.jsp文件有相似的结构，但存在一些区别。以下是对两者的主要区别的分析：  1.Thymeleaf语法：  Thymeleaf使用自定义的th命名空间，而不是JSP中的<%@ page %>和JSTL标签库。Thymeleaf使用th:xxx属性来执行表达式和操作，而JSP使用EL表达式${xxx}和JSTL标签<c:xxx>。  2.图片路径处理：  在Thymeleaf中，图片路径的拼接使用了Thymeleaf的表达式${'images/' + ${aBook.picture}}，而在JSP中使用了EL表达式${aBook.picture}。Thymeleaf的语法更加灵活，可以在表达式中执行更复杂的操作。  3.样式设置：  两者都使用了Bootstrap样式，但在Thymeleaf中使用了Thymeleaf的语法th:class来设置样式，而在JSP中直接使用了Bootstrap的类名。  （5）打开com.ch.ch5\_4.controller包中的文件：ThymeleafController.java，分析其中的代码，  Q:与项目ch5\_2中的相同文件名代码相比，有什么不同？  **ThymeleaController.java文件的分析如下:**  这段代码是一个Spring MVC的Controller类，用于处理请求并渲染Thymeleaf模板。  1. Controller注解：@Controller 注解标识这是一个处理请求的控制器类。  2. 请求映射：@RequestMapping("/") 注解表示处理根路径的请求。当用户访问根路径时，将会调用 index 方法。  3. Model对象：方法参数中的 Model model 用于向视图传递数据。  4. 图书数据：创建了一个 Book 对象 teacherGeng，并设置了相关属性。创建了一个 List<Book> 对象  **两个项目代码的不同点如下：**  1.图书数据：数据方面有一些差异，如ISBN、价格、出版社、作者、图片等属性的取值不同。  2.视图模板：这两个控制器返回的 "index" 视图可能对应不同的Thymeleaf模板文件，文件中的内容和渲染逻辑可能不同。  （6）运行com.ch.ch5\_4包中的 Ch54Application.java(run as Java Application）。  在浏览器地址栏用 http://localhost:8080/ch5\_4/,测试。    思考对比：Thymeleaf和JSP两种视图技术特点。  -----------------------------  3. 实验指导书，实验八 基于JSP引擎的Spring Boot Web开发  对应实验项目shiyan8  -----------------------------  4. 练习使用Spring Boot JPA实现数据库简单查询  （0）外部数据库准备  使用MySql，按ch6-1项目的要求创建名为springbootjpa的数据库。  （1）用STS4打开课本项目ch6\_1，分析pom.xml文件，  Q:增加了什么依赖？    可以看到该项目增加的依赖主要是添加了MYSQL依赖。  （2）分析application.preperties中的数据库配置参数  Q:数据源(spring.datasource.\*)配置包括哪些？  数据源的配置主要包括以下这些，包括数据库的地址、用户名、密码和数据库的驱动。  Q；JPA持久化(spring.jpa.\*)配置有哪些？  JPA持久化的配置主要有以下这些：包括数据库的类型、是否在日志中显示SQL语句、指定自动创建、更新数据库表等配置。    Q；还有什么配置？  除上述配置外，主要还有以下两个配置：      （3）被持久化的实体类  打开src/main/java下的com.ch.ch6\_1.entity包中的MyUser.java,分析其中的代码，  Q:以下标签的作用是什么？  @Entity  @Table  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  **@Entity 注解：**  表示这是一个 JPA 实体类，将会被持久化到数据库中。每个被 JPA 管理的类都需要使用 @Entity 进行注解。注意： 如果没有提供 name 属性，则默认使用类名的小写形式作为数据库表名。在这里，通过 @Table(name = "user\_table") 指定了数据库表的名称为 "user\_table"。  **@Table 注解：**  指定数据库表的信息，例如表名。在这个例子中，通过 @Table(name = "user\_table") 明确指定了数据库表的名称为 "user\_table"。  **@Id 注解：**  标识该属性为实体的主键。注意： 如果属性名与数据库表的主键列名一致，可以省略 @Column 注解。  **@GeneratedValue 注解：**  指定主键的生成策略，即如何生成唯一的主键值。GenerationType.IDENTITY： 表示主键由数据库自动生成（例如，使用自增长的方式）。在这里，通过 @GeneratedValue(strategy = GenerationType.IDENTITY) 指定了采用自增长的方式生成主键。  （4）数据访问层  打开src/main/java下的com.ch.ch6\_1.repository包中的UserRepository.java，分析其代码。  Q:其中哪些地方用了查询关键字？  在上述代码中，查询关键字主要出现在方法的命名规则上，这是 Spring Data JPA 的一项特性，通过方法的命名规则，可以生成相应的查询语句，而不必编写实际的 SQL 语句。以下是使用了查询关键字的两个方法：  **public MyUser findByUname(String uname);**  作用： 通过用户名 (uname) 查询用户对象。  规则解释： 方法名以 findBy 开头，后面跟着实体类属性的名称，这里是 Uname（根据 JavaBean 的规范，首字母大写）。Spring Data JPA 会解析方法名并生成相应的查询语句，这里是根据用户名进行查询。  **public List<MyUser> findByUnameLike(String uname);**  作用： 通过用户名 (uname) 模糊查询用户对象列表。  规则解释： 方法名以 findBy 开头，后面跟着实体类属性的名称，这里是 Uname，再跟着 Like 表示模糊查询。Spring Data JPA 会解析方法名并生成相应的查询语句，这里是根据用户名进行模糊查询。  这种方法命名规则是 Spring Data JPA 提供的一种便捷方式，通过简单的命名规则就能生成相应的查询语句，而无需显式地编写 SQL 语句。这大大简化了数据访问层的开发工作。  （5）业务层  打开src/main/java下的com.ch.ch6\_1.service包中的UserService.java和UserServiceImpl.java，分析其代码。  Q:其中怎样使用了实体类MyUser和数据访问层中的方法？  **@Service 和 @Transactional 注解：**  @Service 注解标注这是一个服务层的类，用于标识这个类提供业务逻辑。  @Transactional 注解用于开启事务管理，表示这个类中的方法应该被事务管理。  **@Autowired 注解：**  用于进行依赖注入，将 UserRepository 注入到 UserServiceImpl 中，使得 UserServiceImpl 可以使用 UserRepository 的方法进行数据访问。  **saveAll 方法：**  创建了三个 MyUser 对象，并设置了它们的属性。将这三个对象添加到一个 List 中。调用 userRepository.saveAll(users) 方法，将整个列表保存到数据库中。  **findAll 方法：**  调用 userRepository.findAll() 方法，返回数据库中所有的 MyUser 对象列表。  （6）控制器类  打开src/main/java com.ch.ch6\_1.controller包中的UserTestController.java，分析其代码。  Q:其中使用了哪几个注解及其的作用？  Q:@RequestMapping对应的方法  **@Controller 注解：**  作用： 标识这是一个 Spring MVC 控制器（Controller）类，用于处理 HTTP 请求和响应。  **@Autowired 注解：**  作用： 进行依赖注入，将 UserService 注入到 UserTestController 中，使得 UserTestController 可以调用 UserService 中的方法。  （7）视图  打开src/main/resources/templates目录下的showAll.html和showAuser.html,  Q:其中的使用了哪几个属性？  **对于showaAll.html页面：**  这段代码使用了 Thymeleaf 模板引擎中的一些属性，这些属性主要用于动态渲染 HTML 页面，将后端数据显示在前端页面上。以下是使用到的属性：  xmlns:th="http://www.thymeleaf.org"：  作用： 声明 Thymeleaf 的命名空间，使得 Thymeleaf 标签能够在 HTML 中生效。  th:href="@{css/bootstrap.min.css}" 和 th:href="@{css/bootstrap-theme.min.css}"：  作用： 设置链接的地址，这里用于引入 Bootstrap 的样式文件。th:href 是 Thymeleaf 提供的属性，用于动态设置链接地址。  th:text="${title}"：  作用： 设置元素的文本内容，这里用于动态显示页面标题。${title} 表示从后端传递的 title 属性中获取数据，并显示在页面上。  **对于showAuser.html页面：**  这段代码同样使用了 Thymeleaf 模板引擎中的一些属性，主要用于动态渲染 HTML 页面，将后端数据显示在前端页面上。以下是使用到的属性：  xmlns:th="http://www.thymeleaf.org"：  作用： 声明 Thymeleaf 的命名空间，使得 Thymeleaf 标签能够在 HTML 中生效。  th:href="@{css/bootstrap.min.css}" 和 th:href="@{css/bootstrap-theme.min.css}"：  作用： 设置链接的地址，这里用于引入 Bootstrap 的样式文件。th:href 是 Thymeleaf 提供的属性，用于动态设置链接地址。  （8）运行  运行src/main/java下com.ch.ch6\_1包中Ch61Application.java(run as Java Application)，  （a）在浏览器地址栏输入：http://localhost:8080/ch6\_1/save/，保存数据（创建表）。  （b）在浏览器地址栏输入：http://localhost:8080/ch6\_1/findAll，查询所有用户。  （c）在浏览器地址栏输入：http://localhost:8080/ch6\_1/findByUnameLike?uname=陈 ，模糊查询所有姓陈的用户。  （d）在浏览器地址栏输入：http://localhost:8080/ch6\_1/findByUname?uname=陈恒2 ，查一个名为陈恒2的用户。  （e）在浏览器地址栏输入：http://localhost:8080/ch6\_1/getOne?id=1 ，查一个id为1的用户。  上述测试的结果如下图所示：  (a)    (b)    (c)    (d)    (e)    Q:数据访问层中有具体查询实现代码吗？为什么JPA的接口方法可以不写实现就能实现功能？  数据访问层中没有具体查询的实现代码。  JPA（Java Persistence API）是一种规范，它提供了一组用于在Java应用程序中进行对象-关系映射（ORM）的接口。这些接口定义了一系列规范，包括实体的定义、持久化、查询等方面。JPA规范的实现由不同的厂商提供，例如Hibernate、EclipseLink等。  在Spring中，Spring Data JPA 是对JPA规范的进一步简化和封装，提供了一种基于接口的编程模型，通过继承JPA接口，无需写具体的实现代码，就可以实现一些基本的CRUD（Create, Read, Update, Delete）操作。  Q:JPQL与具体的数据库产品有关吗？SQL呢?  JPQL（Java Persistence Query Language）是 JPA（Java Persistence API）中定义的查询语言，它是一种面向对象的查询语言，与具体的数据库产品无关。JPQL查询的对象是实体类和其属性，而不是数据库表和列。因此，JPQL提供了一种抽象层，使得开发者可以更加灵活地进行查询，而不用关心底层数据库的具体实现。  与之不同，SQL（Structured Query Language）是一种用于关系数据库的标准查询语言，它是与具体数据库产品相关的。每种数据库产品都有自己的SQL方言，有些SQL语句在一个数据库中可能有效，但在另一个数据库中可能需要进行调整。  JPQL和SQL之间的主要区别在于：  对象模型 vs 关系模型： JPQL是面向对象的查询语言，查询的是实体对象及其属性，而SQL是关系模型的查询语言，查询的是数据库表和列。  数据库无关性： JPQL提供了一种与数据库无关的查询方式，使得应用程序更容易移植到不同的数据库系统。相反，SQL语句可能需要根据具体数据库的不同而调整。  灵活性和表达力： JPQL提供了更高级的查询功能，可以方便地进行对象导航、继承关系查询等。SQL语句通常更底层，可能需要更多的手动优化和调整。  Q:本项目与之前的访问数据库项目(如实验2,ch1\_8，TestServiceImpl.java)比，有SQL代码吗？  无SQL代码。 | | | |
| 思考题解答：  1.Thymeleaf和JSP两种视图技术特点：  Thymeleaf的特点：   1. 自然的模板语法：Thymeleaf使用类似HTML的语法，使得模板文件更加易读和易理解。它可以与HTML标签无缝结合，使开发者能够更直观地编写模板。 2. 客户端和服务器端渲染：Thymeleaf可以在服务器端和客户端两个环境中进行渲染。这意味着可以在服务器端生成完整的HTML文件并将其发送给客户端，也可以使用JavaScript在客户端动态渲染。 3. 强大的表达式语言：Thymeleaf提供了强大的表达式语言，可以在模板中进行条件判断、循环和变量赋值等操作。这使得模板更加灵活且具有动态性。 4. 支持国际化：Thymeleaf内置了对国际化的支持，可以根据用户的语言环境来显示不同的内容，使得开发多语言应用更加方便。   JSP的特点：   1. 基于Java的标签语法：JSP使用基于Java的标签语法，可以在HTML文件中插入Java代码。这使得开发者可以直接在视图文件中编写逻辑处理代码，实现动态页面生成。 2. 轻量级和易学习：JSP是JavaEE标准的一部分，可以与JavaEE的其他技术无缝集成。它的学习曲线相对较低，适合初学者入门。 3. 支持Java的所有功能：JSP可以使用Java的所有功能和类库，包括数据库访问、网络通信等。这使得JSP在处理复杂业务逻辑时更加灵活和强大。 4. 完整的生命周期：JSP有完整的生命周期，可以在编译、初始化、执行和销毁等阶段进行相关操作。这使得开发者可以在不同阶段对JSP进行处理和优化 | | | |