链滴

SpringBoot 入门教程 (十四) | 统一处理异常

作者：JavaFish
原文链接：https://ld246.com/article/1621476761576
来源网站：链滴

前言

如题，今天介绍 SpringBoot 是如何统一处理全局异常的。SpringBoot 中的全局异常处理主要起作的两个注解是 **@ControllerAdvice** 和 **@ExceptionHandler** 。

其中 @ControllerAdvice 是组件注解，添加了这个注解的类能够拦截 Controller 的请求，而 ExceptionHandler 注解可以设置全局处理控制里的异常类型来拦截要处理的异常。 比如：@ExceptionHandlr(value = NullPointException.class) 。

准备工作

- SpringBoot 2.1.3
- IDEA
- JDK 8

依赖配置

```
<dependencies>
        <!-- JPA 依赖 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <!-- web 依赖 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- mysql 连接类 -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <!-- lombok 依赖 -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <!-- 单元测试依赖 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

配置文件

```
spring:
  # 数据库相关
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf8&serverTi
ezone=UTC&useSSL=true
    username: root
    password: 123456

  jpa:
    hibernate:
      ddl-auto: update   #ddl-auto:设为 create 表示每次都重新建表
    show-sql: true
```

返回的消息类

```
public class Message<T> implements Serializable {

    /**
     * 状态码
     */
    private Integer code;

    /**
     * 返回信息
     */
    private String message;

    /**
     * 返回的数据类
     */
    private T data;

    /**
     * 时间
     */
    private Long time;

    // getter、setter 以及 构造方法略。。。
}
```

工具类

用于处理返回的数据以及信息类，代码注释很详细不说了。

```
public class MessageUtil {

    /**
     * 成功并返回数据实体类
     * @param o
     * @param <E>
     * @return
     */
```

```
    public static <E>Message<E> ok(E o){
        return new Message<>(200, "success", o, new Date().getTime());
    }

    /**
     * 成功，但无数据实体类返回
     * @return
     */
    public static <E>Message<E> ok(){
        return new Message<>(200, "success", null, new Date().getTime());
    }

    /**
     * 失败，有自定义异常返回
     * @param code
     * @param msg
     * @return
     */
    public static <E>Message<E> error(Integer code,String msg){
        return new Message<>(code, msg, null, new Date().getTime());
    }
}
```

自定义异常

通过继承 RuntimeException ， 声明 code 用于定义不同类型的自定义异常。主要是用于异常拦截出取 code 并将 code 设置到消息类中返回。

```
public class CustomException extends RuntimeException{

    /**
     * 状态码
     */
    private Integer code;

    public Integer getCode() {
        return code;
    }

    public void setCode(Integer code) {
        this.code = code;
    }

    public CustomException(Integer code, String message){

        super(message);
        this.code = code;

    }
}
```

异常拦截类

通过加入 @RestControllerAdvice 来声明该类可拦截 Controller 请求，同时在 handle方法加入 @ExceptionHandler 并在该注解中指定要拦截的异常类。

```
@RestControllerAdvice // 控制器增强处理(返回 JSON 格式数据)，添加了这个注解的类能被 classpth 扫描自动发现
public class ExceptionHandle {

    @ExceptionHandler(value = Exception.class) // 捕获 Controller 中抛出的指定类型的异常，也以指定其他异常
    public <E>Message<E> handler(Exception exception){

        if (exception instanceof CustomException){
            CustomException customException = (CustomException) exception;
            return MessageUtil.error(customException.getCode(), customException.getMessage());
        } else {
            return MessageUtil.error(120, "异常信息：" + exception.getMessage());
        }
    }
}
```

这里只对自定义异常以及未知异常进行处理，如果你在某方法中明确知道可能会抛出某个异常，可以多一个特定的处理。比如说你明确知道该方法可能抛出 NullPointException 可以追加 NullPointException 的处理：

```
if (exception instanceof CustomException){
     CustomException customException = (CustomException) exception;
     return MessageUtil.error(customException.getCode(), customException.getMessage());
} else if (exception instanceof NullPointException ){
     return MessageUtil.error(500, "空指针异常信！");
} else {
     return MessageUtil.error(120, "异常信息：" + exception.getMessage());
}
```

controller 层

```
@RestController
@RequestMapping("/student")
public class StudentController {

    @Autowired
    private StudentService studentService;

    @GetMapping("/{id}")
    public Message<Student> findStudentById(@PathVariable("id") Integer id){

        if (id < 0){
            //测试自定义错误
            throw new CustomException(110, "参数不能是负数！");

        } else if (id == 0){
            //硬编码，为了测试
            Integer i = 1/id;
            return null;
        } else {
```

```
            Student student = studentService.findStudentById(id);
            return MessageUtil.ok(student);
        }
    }
}
```

完整代码

https://github.com/turoDog/Demo/tree/master/springboot_exception_demo

如果觉得对你有帮助，请给个 Star 再走呗，非常感谢。

Postman 测试

访问 http://localhost:8080/student/5 测试正常返回数据结果。


```
{
    "code": 200,
    "message": "success",
    "data": {
        "id": 5,
        "name": "一个优秀的废人",
        "age": 24
    },
    "time": 1551540624460
}
```


访问 http://localhost:8080/student/0 测试未知异常的结果。


```
{
    "code": 120,
    "message": "异常信息：/ by zero",
    "data": null,
    "time": 1551541604717
}
```


访问 http://localhost:8080/student/-11 测试自定义异常的结果。

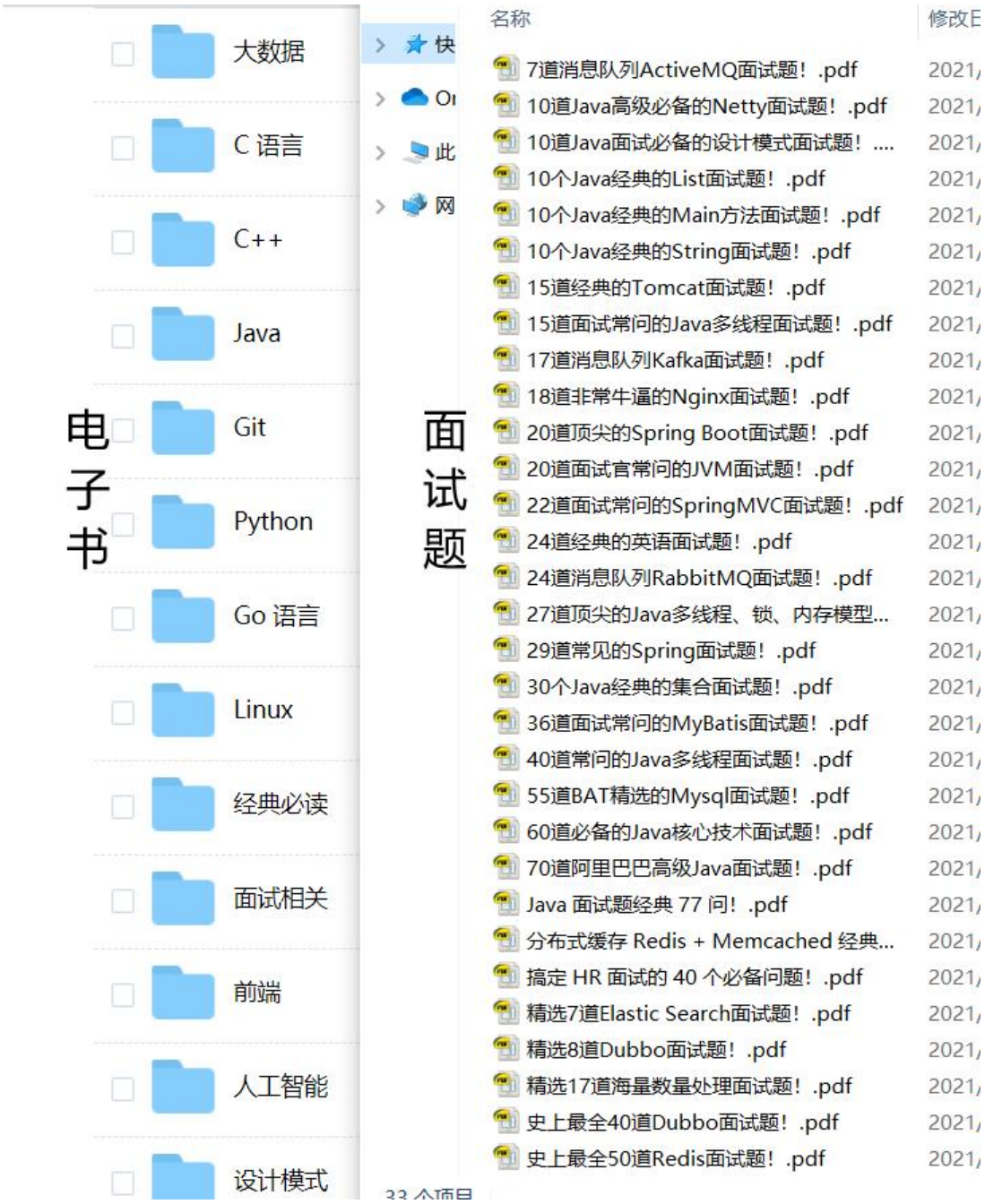

```
{
    "code": 110,
    "message": "参数不能是负数！",
    "data": null,
    "time": 1551540768856
}
```


后语

如果本文对你哪怕有一丁点帮助，请帮忙点好看。你的好看是我坚持写作的动力。

初次见面，也不知道送你们啥。干脆就送**几百本电子书**和**2021最新面试资料**吧。微信搜索**JavaFish**复**电子书**送你 1000+ 本编程电子书；回复**面试**获取 50 套大厂面试题；回复**1024**送你一套完整的 jav 视频教程。

面试题都是有答案的，如下所示：有需要的就来拿吧，**绝对免费，无套路获取。**


电子书
大数据
C 语言
C++
Java
Git
Python
Go 语言
Linux
经典必读
面试相关
前端
人工智能
设计模式
面试题
名称
修改日
7道消息队列ActiveMQ面试题！.pdf
10道Java高级必备的Netty面试题！.pdf
10道Java面试必备的设计模式面试题！....
10个Java经典的List面试题！.pdf
10个Java经典的Main方法面试题！.pdf
10个Java经典的String面试题！.pdf
15道经典的Tomcat面试题！.pdf
15道面试常问的Java多线程面试题！.pdf
17道消息队列Kafka面试题！.pdf
18道非常牛逼的Nginx面试题！.pdf
20道顶尖的Spring Boot面试题！.pdf
20道面试官常问的JVM面试题！.pdf
22道面试常问的SpringMVC面试题！.pdf
24道经典的英语面试题！.pdf
24道消息队列RabbitMQ面试题！.pdf
27道顶尖的Java多线程、锁、内存模型...
29道常见的Spring面试题！.pdf
30个Java经典的集合面试题！.pdf
36道面试常问的MyBatis面试题！.pdf
40道常问的Java多线程面试题！.pdf
55道BAT精选的Mysql面试题！.pdf
60道必备的Java核心技术面试题！.pdf
70道阿里巴巴高级Java面试题！.pdf
Java 面试题经典 77 问！.pdf
分布式缓存 Redis + Memcached 经典...
搞定 HR 面试的 40 个必备问题！.pdf
精选7道Elastic Search面试题！.pdf
精选8道Dubbo面试题！.pdf
精选17道海量数量处理面试题！.pdf
史上最全40道Dubbo面试题！.pdf
史上最全50道Redis面试题！.pdf
2021/

JavaFish

微信扫描二维码，关注我的公众号