Spring MVC中异常处理

1 全局异常处理

1.1 什么是全局异常处理器

全局异常处理器是Spring MVC框架中的一种异常处理机制，用于统一 **处理由控制器抛出的异常。**

全局异常处理器可以帮助我们捕获和处理控制器中的异常，并且可以根据不同的异常类型进行不同的处理操作，从而保障应用的健壮性和稳定性。

当然，Spring MVC中有内置的异常处理对象，但是呈现的结果对于用户端不友好，所以实际项目我们一般会使用全局异常处理器处理异常。

1.2 全局异常处理器的配置

Spring MVC中的全局异常处理器可以通过以下方式进行配置：

1. 创建 `exception.GlobalExceptionHandler` 类，并添加异常处理方法；

   使用 `@ControllerAdvice` 注解 或者 `@RestControllerAdvice` 注解标注该类；

2. 在异常处理方法上添加 `@ExceptionHandler` 注解，用于指定控制器中需要处理的异常类型。

1.3 使用流程

1）创建全局异常处理器类

工程目录下创建 `exception.GlobalHandlerException`

- `@ControllerAdvice` 注解

  定义全局异常处理器，处理Controller中抛出的异常。

- `@RestControllerAdvice` 注解

复合注解，是`@ControllerAdvice`注解和`@ResponseBody`注解的组合；

用于捕获Controller中抛出的异常并对异常进行统一的处理，还可以对返回的数据进行处理。

2）创建异常处理方法

在异常处理方法上添加 `@ExceptionHandler` 注解

- `@ExceptionHandler` 注解

  用于捕获Controller处理请求时抛出的异常，并进行统一的处理。

- 示例

```
/**
    ex.getMessage()方法：用于捕获异常信息
*/
@ExceptionHandler
public JsonResult doHandleRuntimeException(RuntimeException ex){
    log.error("error is " + ex.getMessage());
    return new JsonResult(0,ex.getMessage());
}
```

1.4 全局异常处理器示例

1）微博详情页异常抛出

```
public JsonResult selectById(int id){
    if(id < 0) {
        throw new IllegalArgumentException("id值无效");
    }
    ... ...
}
```

2）全局异常处理

`exception.GlobalHandlerException` 类

```

package cn.tedu.weibo.exception;

import cn.tedu.weibo.common.response.JsonResult;
import cn.tedu.weibo.common.response.StatusCode;
import lombok.Setter;
import lombok.extern.slf4j.Slf4j;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.*;

/**
 * RestControllerAdvice 描述的类型为一个全局异常处理对象类型，
 * 当某个Controller方法中出现了异常，系统底层就会查找有没有定义全局异常处理对象。
 * 这个全局异常处理对象中有没有定义对应的异常处理方法，假如有就调用此方法处理异常。
 */
```

```
@Slf4j
@RestControllerAdvice //=@ControllerAdvice+@ResponseBody
public class GlobalExceptionHandler {
    /**
     *  @ExceptionHandler 描述的方法为一个异常处理方法，在此注解内部可以定义具体的异常处理
     *  类型(例如RuntimeException),此注解描述的方法需要定义一个异常类型的形式参数，
     *  通过这个参数接收具体的异常对象(也可以接收其异常类型对应的子类类型的异常)。
     */
    @ExceptionHandler
    public JsonResult doHandleRuntimeException(RuntimeException ex){
        log.error("error is " + ex.getMessage());
        return new JsonResult(0,ex.getMessage());
    }

    /**
     * 假如用全局异常处理对象处理Controller类中出现的异常，全局异常处理对象会优先查找与
Controller
     * 中相匹配的异常处理方法，假如没有，会查找对应异常的父类异常处理方法。
     */
    @ExceptionHandler
    public JsonResult doHandleRuntimeException(IllegalArgumentException ex){
        log.error("IllegalArgumentException is " + ex.getMessage());
        return new JsonResult(0,ex.getMessage());
    }
}
```

3）重启工程测试

http://localhost:8080/v1/weibo/selectById?id=-1

2 关于Throwable

在开发实践中，通常会添加一个处理`Throwable`的方法，它将可以处理所有类型的异常，则不会再出现`500`错误!

`GlobalExceptionHandler`中添加处理 Throwable 的方法

```
@ExceptionHandler
public JsonResult handleThrowable(Throwable e) {
    return new JsonResult(8888, "程序运行过程中出现了Throwable");
}
```