34_代码实战：基于 Spring AOP 机制，实现全局异常处理

1、开篇

上节课在 Controller 中的 health 方法里 throw 了一个 new exception，通过这种方式模拟系统内部抛出异常的情况，按照目前代码而言如果没有处理这个异常将会直接展示给用户 500 错误，显然是不友好的。因此，我们需要对系统内部异常采取统一的处理，从而才能发挥给前端页面有价值的信息。本节课我们会讲到基于 Spring AOP 机制，实现全局异常处理。今天的内容：

- ControllerAdvice、ResponseBody 与 ExceptionHandler 注释
- 定义自己的 GlobalExceptionHandler

2、ControllerAdvice 与 ExceptionHandler 注解

如果说需要对系统内部异常采取统一的处理，其思路还是通过 Spring AOP 去截获异常信息，进行统一处理然后再返回给客户端。在使用 Spring AOP 编写异常处理类之前先介绍几个注释。

ControllerAdvice，顾名思义是一个增强的 Controller，增强的概念在 SpringAOP 中就是我们需要织入方法的具体内容，回到本节内容就是需要进行异常处理的地方。使用 ControllerAdvice 可以实现三个方面的功能：全局异常处理、全局数据绑定和全局数据预处理。在我们的项目中，主要关注全局异常处理的功能。

如图 1 所示，在建立的 MyGlobalExceptionHandler 类上标记了 ControllerAdvice 注解，表示对 Controller 的请求进行增强。在类中的 customException 方法上面通过一个 ExceptionHandler 注解，参数使用 Exception.class 表示对 Exception 类型的异常进行处理。这里可以根据需要定义其他异常，甚至是自定义的异常类。然后通过 new ModelAndView 初始化出 ModelAndView，并且对其进行修改通过 addObject 方法和 setViewName 加入一些错误信息，从而通过 return 语句将这个 ModelAndView 返回给 Controller 的调用者。

```java
@ControllerAdvice
public class MyGlobalExceptionHandler {
    @ExceptionHandler(Exception.class)
    public ModelAndView customException(Exception e) {
        ModelAndView mv = new ModelAndView();
        mv.addObject("message", e.getMessage());
        mv.setViewName("myerror");
        return mv;
    }
}
```

图 1 ControllerAdvice 用法

3、定义自己的 GlobalExceptionHandler

有了上面对 ControllerAdvice 和 ExceptionHandler 注解的讲解，下面就按照我们之前的思路来建立全局异常处理类。我们在 spring/handler 目录下面建立 GlobalExceptionHandler，不出意料地使用了 ControllerAdvice 作为注解。下面 Responsebody 注解表示 Controller 方法返回结果直接写入 HTTP 响应正文（ResponseBody）中。该注解用于将 Controller 的方法返回的对象，通过适当的 HttpMessageConverter 转换为指定格式后，写入到 Response 对象的 body 数据区。

接下来使用 LoggerFactory 的 getLogger 方法产生 logger 用来记录日志。在自定义异常类中定义了几个方法。先看第一个 handleException 方法，在方法上面有一个 ExceptionHandler 注解，注解的参数标注的是一个 Exception.class 表示是用来处理一个 Exception 异常的。handleException 方法的输入参数是 Exception ，之后会通过 LOGGER 记录异常信息。然后将异常信息以 CommonResponse 的方式返回给客户端。

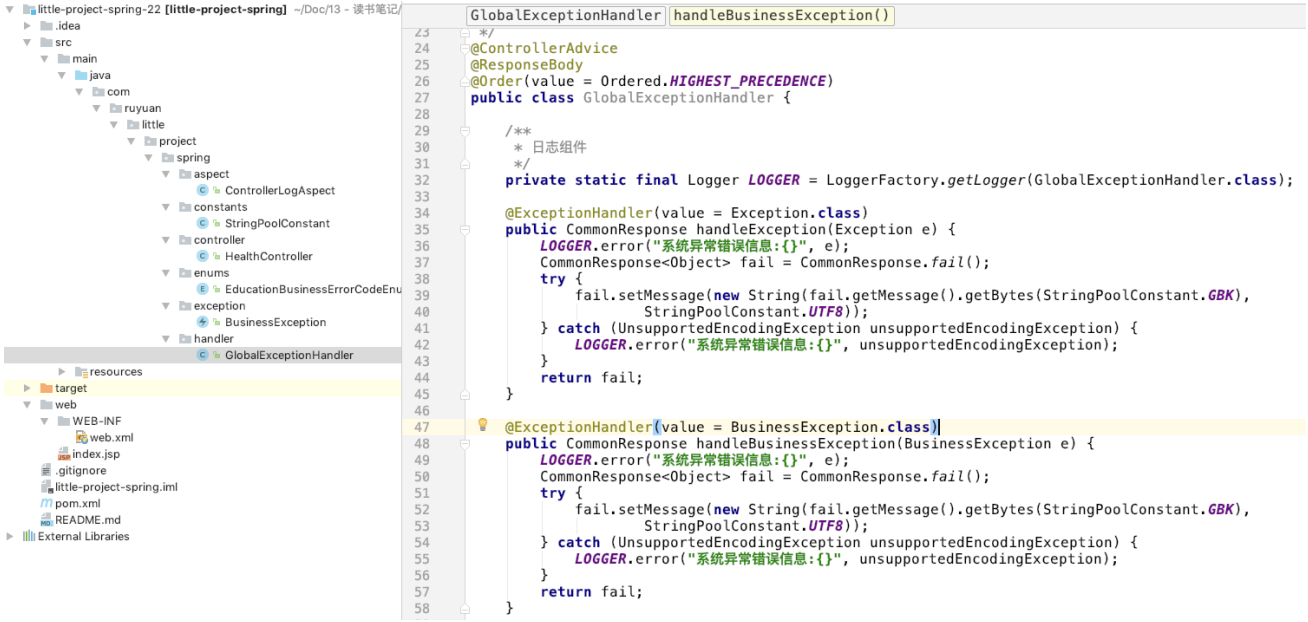

图 2 添加 GlobalExceptionHandler 类

可以从图 2 中下面一个方法 handleBusinessException 唯一的不同是在 ExceptionHandler 注解的参数上。这里使用的参数类型是 BusinessException.class，是我们自己定义的业务异常类。

如图 3 所示，spring/exception 目录下面建立 BusinessException 类，其继承与 RuntimeException 类。这个 BusinessException 可以在全局异常处理类中被处理，如果在业务代码中也接抛出这个异常，也是可以被 GlobalExceptionHandler 类中的 handleBusinessException 方法捕获到并处理的。

图 3 自定义的 BusinessException 类

正如前面定义了 BusinessException 类，我们将其应用到 HealthController 中。如图 4 所示，返回到 HealthController 的 health 方法中，注释掉 return CommonResponse.success()语句，加入 throw new BusinessException(”系统异常，请联系管理员”);。由于使用了 GlobalExceptionHandler 类对 Controller 类中的异常信息进行截获，当该异常抛出的时候会被包装然后，通过友好的方式返回给客户端。

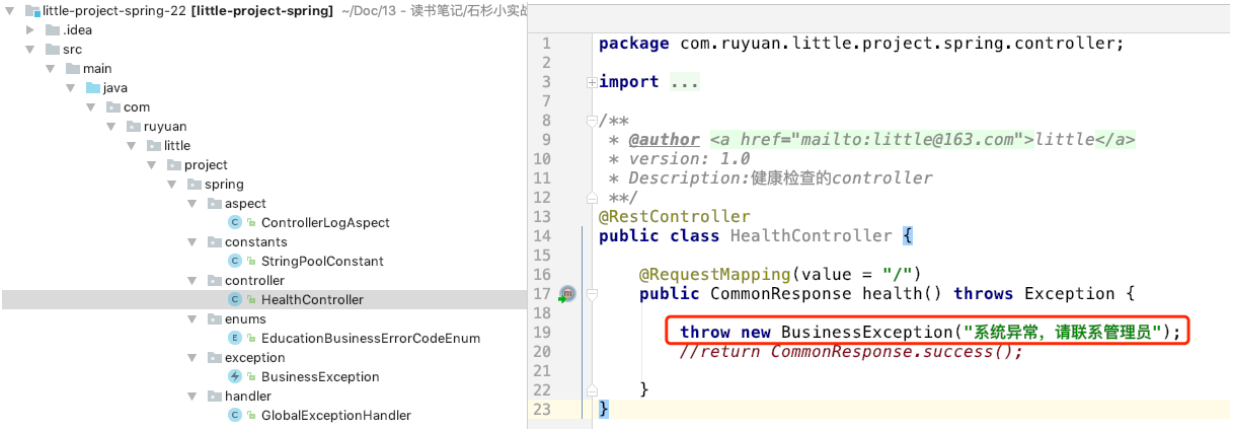

图 4 在 HealthController 中直接抛出异常

4、总结

本节课在介绍了利用 Spring AOP 实现全局异常处理的思路，从 ControllerAdvice 与 ExceptionHandler 注解入手，我们创建了自己的全局异常处理类 GlobalExceptionHandler。到 Controller 中抛出异常的时候 GlobalExceptionHandler 中有对应的方法就可以捕获到。针对不同的异常类，可以编写不同的异常处理方法。

下节课我们一起来验证 GlobalExceptionHandler 类能否处理 Controller 中抛出的异常信息，从而解决页面报错的问题。这里将代码给大家，下期见，拜拜。

友情提示：本章讲述代码只是部分核心代码，完整代码请查阅文末链接中代码，谢谢。