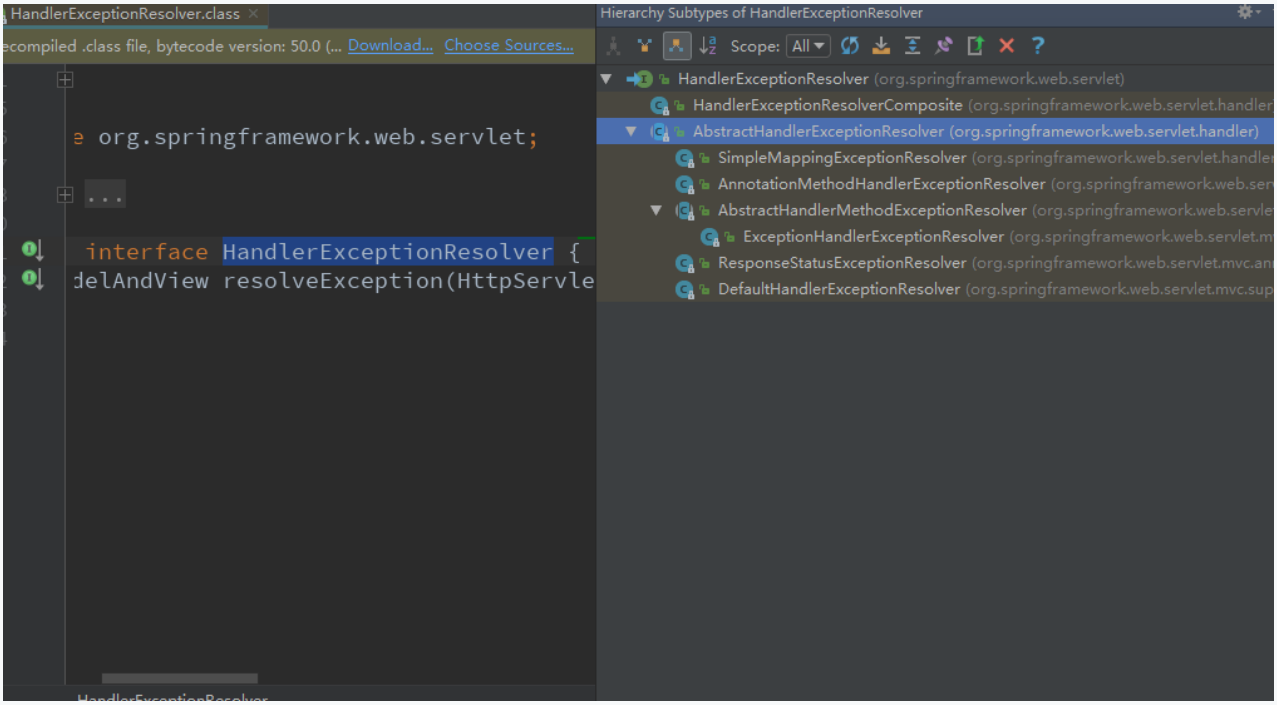
13 SpringMVC-异常处理

`SpringMVC`

一：异常处理

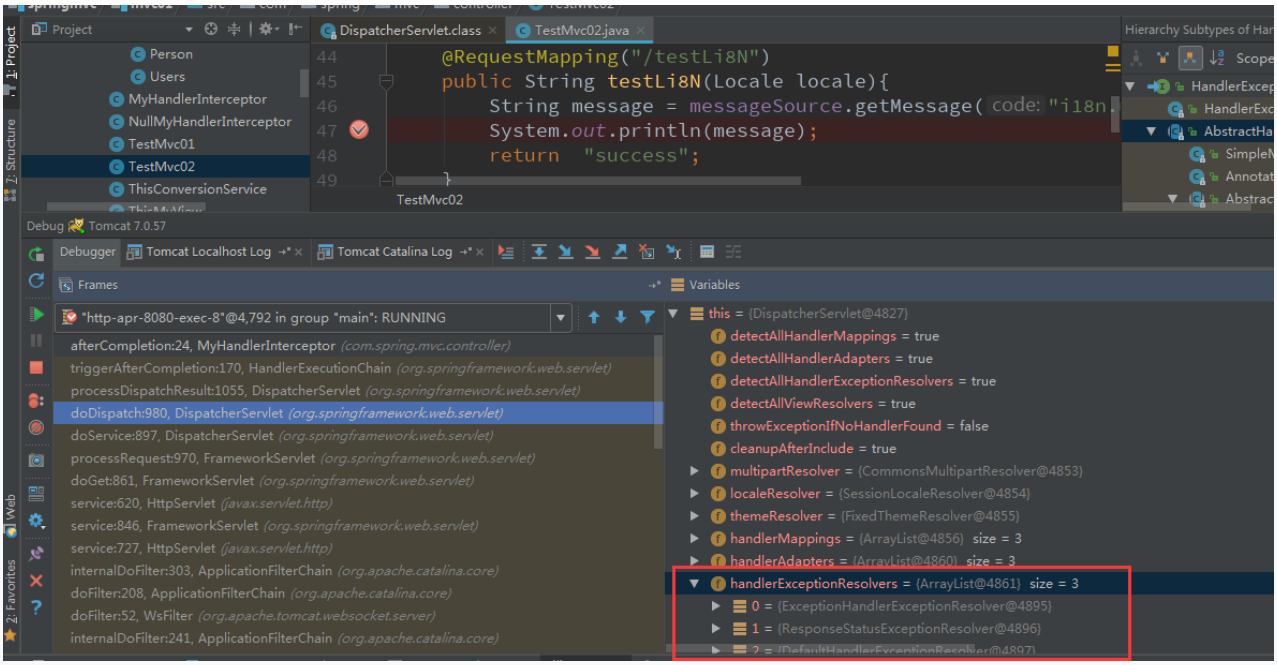
1、Spring MVC 通过 HandlerExceptionResolver 处理程序的异常，包括 Handler 映射、数据绑定以及目标方法执行时发生的异常

2、HandlerExceptionResolver的实现类



3、DispatcherServlet 默认装配的 HandlerExceptionResolver，在开发中使用`<mvc:annotation-driven />`会加载3个异常处理类，

4、随便在一个Controller请求方法中加上Dug断点测试，查询



5、那么这里讲解就讲解这3个实现类
ExceptionHandlerExceptionResolver
ResponseStatusExceptionResolver
DefaultHandlerExceptionResolver(对一些特殊的异常进行处理如请求方式不支持,在这里不在过多讲解)
SimpleMappingExceptionResolver

二、ExceptionHandlerExceptionResolver

1、主要处理 Handler 中用 @ExceptionHandler 注解定义的方法

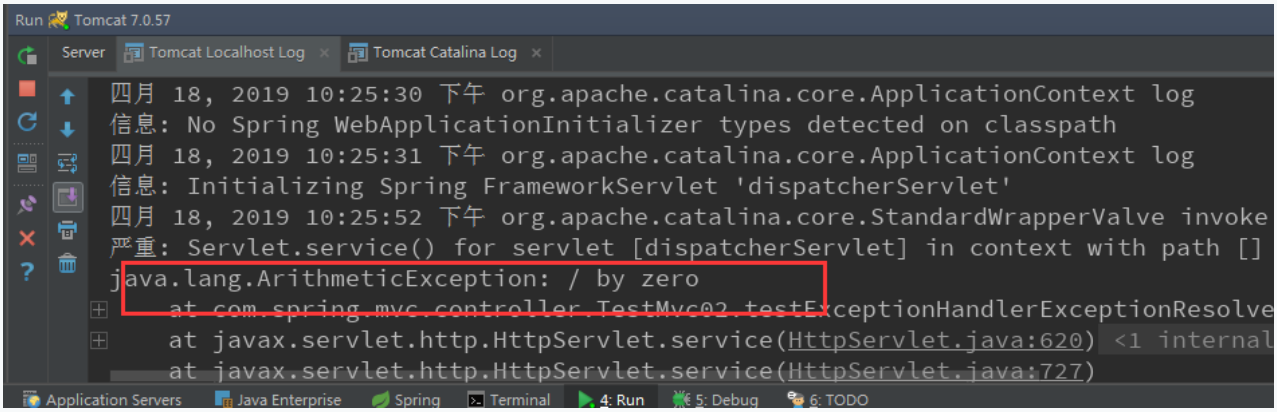
2、@ExceptionHandler 注解定义的方法优先级问题：例如发生的是 NullPointerException，但是声明的异常有RuntimeException 和 Exception，此候会根据异常的最近继承关系找到继承深度最浅的那个 @ExceptionHandler注解方法，即标记了 RuntimeException 的方法

1、代码展示

1、创建一个controller层请求

```
    @RequestMapping("/testExceptionHandler")
    public String testExceptionHandler(@RequestParam("id") int id)  {
        System.out.println("id = " + 10/id);
        return  "success";
    }
==========页面请求==============
<a href="testExceptionHandler?id=0" >testExceptionHandler</a>
```

2、执行的结果会进行报异常，那么这个异常怎么处理



3、处理异常，并将异常返回到页面

4、创建一个异常处理方法

```
/**
     * @param ex   异常类型的参数
     *          如这里要返回异常信息到页面,不能使用Map对象,
     *              直接返回ModelAndView即可
     *       异常优秀级？
     *              会进行找离抛出异常最近的一个异常
     */
  @ExceptionHandler({ArithmeticException.class})
    public ModelAndView testArithmeticException(Exception ex)  {
        System.out.println("ex = " + ex);
      ModelAndView modelAndView = new ModelAndView();
      modelAndView.setViewName("error");
      modelAndView.addObject("exception",ex.getMessage());
      //返回到异常页面
        return  modelAndView;
    }
==========页面回去异常信息==============
<body>
异常的页面
```

```
${exception}
</body>
```

> 5、这里的异常处理只是针对当前Controller层的异常，那么怎么设置全局异常处理？

2、全局异常处理

> 1、创建一全局处理异常的类
> 2、ExceptionHandlerMethodResolver 内部若找不到@ExceptionHandler 注解的话，会找@ControllerAdvice 中的@ExceptionHandler 注解方法

```
@ControllerAdvice
public class ThisArithmeticException {
    @ExceptionHandler({ArithmeticException.class})
    public ModelAndView testArithmeticException(Exception ex)  {
        System.out.println("------报异常了 = " + ex);
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("error");
        modelAndView.addObject("exception",ex.getMessage());
        //返回到异常页面
        return  modelAndView;
    }
}
```

三、ResponseStatusExceptionResolver(设置状态码)

> 1、创建一个自定义的异常类

```
==================================
        自定义的异常类
==================================
/**
 *
 * value状态码
 * reason信息
 */
```

```
@ResponseStatus(value = HttpStatus.FORBIDDEN,reason = "数字不能为0")
public class ThisMyRuntimException  extends  RuntimeException{
    //代码省略.......
}
==================================
        请求地址
==================================
    @RequestMapping("/testThisMyRuntimException")
    public String testThisMyRuntimException(@RequestParam("id") int id)
{
        if(id==0){
          throw new ThisMyRuntimException();
        }
        return  "success";
    }
```

2、结果

四、SimpleMappingExceptionResolver(配置文件指定异常页面)

1、创建一个Controller请求

```
    @RequestMapping("/testSimpleMappingException")
    public String testSimpleMappingException(@RequestParam("id") int id
)  {
        String[] strings = new String[5];
```

```
        /*当id为20时候，出现数组下标越界异常*/
        System.out.println(strings[id]);
        return  "success";
    }
```

2、spring.xml文件的配置

```
<!--使用SimpleMappingExceptionResolver配置异常处理-->
    <bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
            <!--配置异常的属性名,默认的不会生效了-->
        <property name="exceptionAttribute" value="exe"></property>
        <!--异常映射-->
        <property name="exceptionMappings">
                <props>
                    <!--配置异常转向的页面
                        注意，编写异常时全类名
                    -->
                     <prop key="java.lang.ArrayIndexOutOfBoundsException">error</prop>
                </props>
            </property>
    </bean>
```

3、页面信息获取页面

```
<br/>
默认的：${exception}
<br/>
配置的属性名：${exe}
```