# 第六章：SpringBoot异常处理

## 概述

Springboot对异常处理提供了5种方法，分别是：

* 自定义错误页面
* @ExceptionHandler注解处理异常
* @ControllerAdvice+@ExceptionHandler注解
* 配置SimpleMappingExceptionResolver处理异常
* 自定义HandlerExceptionResovler类处理异常

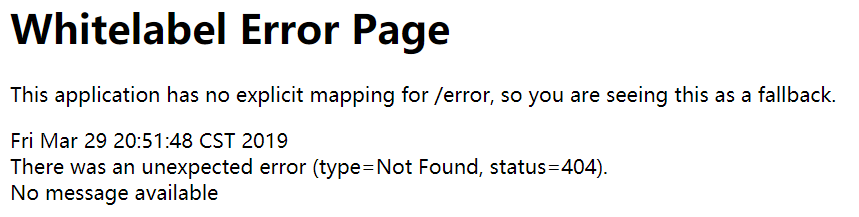
## 自定义错误页面

SpringBoot 默认的处理异常的机制：SpringBoot 默认的提供了一套处理异常的机制。

一旦程序中出现了异常 SpringBoot 会像/error 的 url 发送请求。在 springBoot 中提供了一个

叫 BasicExceptionController 来处理/error 请求，然后跳转到默认显示异常的页面来展示异常

信息。



如果我们需要将所有的异常同一跳转到自定义的错误页面，需要在src/main/resources/templates 目录下创建 error.html 页面。注意：名称必须叫 error

|  |
| --- |
| <!DOCTYPE html>  <html xmlns:th=*"http://www.thymeleaf.org"*>  <head>  <meta charset=*"UTF-8"*>  <title>错误页面</title>  </head>  <body>  <h2>请求错误，请联系管理员！</h2>  <h3>错误信息：<font color=*"red"* th:text=*"${exception}"*></font></h3>  </body>  </html> |

## @ExceptionHandler注解处理异常

### Controller

|  |
| --- |
| @Controller  **public** **class** IndexController {  @RequestMapping("/show")  **public** String show() {  String str = **null**;  str.length();  **return** "index";  }    @RequestMapping("/show2")  **public** String show2() {  System.***out***.println(1/0);  **return** "index";  }  @ExceptionHandler(value= {ArithmeticException.**class**})  **public** ModelAndView arithmeticExceptionHandler(Exception e) {  ModelAndView view = **new** ModelAndView();  view.addObject("error", e.toString());//保存错误信息  view.setViewName("error1");  **return** view;  }    @ExceptionHandler(value= {NullPointerException.**class**})  **public** ModelAndView nullPointerExceptionionHandler(Exception e) {  ModelAndView view = **new** ModelAndView();  view.addObject("error", e.toString());//保存错误信息  view.setViewName("error2");  **return** view;  }  } |

### 页面

#### error1.html

|  |
| --- |
| <h1>error1页面</h1>  <h2>请求错误，请联系管理员！</h2>  <h3>错误信息：<font color=*"red"* th:text=*"${error}"*></font></h3> |

#### error2.html

|  |
| --- |
| <h1>error2页面</h1>  <h2>请求错误，请联系管理员！</h2>  <h3>错误信息：<font color=*"red"* th:text=*"${error}"*></font></h3> |

## @ControllerAdvice+@ExceptionHandler注解

### 方法

需要创建一个能够处理异常的全局异常类,在该类上需要添加@ControllerAdvice 注解

### 案例

|  |
| --- |
| @ControllerAdvice  **public** **class** GlobalException {  @ExceptionHandler(value= {ArithmeticException.**class**})  **public** ModelAndView arithmeticExceptionHandler(Exception e) {  ModelAndView view = **new** ModelAndView();  view.addObject("error", e.toString());//保存错误信息  view.setViewName("error1");  **return** view;  }    @ExceptionHandler(value= {NullPointerException.**class**})  **public** ModelAndView nullPointerExceptionionHandler(Exception e) {  ModelAndView view = **new** ModelAndView();  view.addObject("error", e.toString());//保存错误信息  view.setViewName("error2");  **return** view;  }  } |

## 配置SimpleMappingExceptionResolver处理异常

### 方法

在全局异常类中添加一个方法完成异常的同一处理

### 案例

|  |
| --- |
| @Configuration  **public** **class** GlobalException {  @Bean  **public** SimpleMappingExceptionResolver getSimpleMappingExceptionResolver() {  SimpleMappingExceptionResolver resolver = **new** SimpleMappingExceptionResolver();  Properties mappings = **new** Properties();  //设置对应异常映射的页面  //参数1：异常类型 参数2：关联映射页面  mappings.put("java.lang.ArithmeticException", "error1");  mappings.put("java.lang.NullPointerException", "error2");  //设置异常与映射的关联信息  resolver.setExceptionMappings(mappings);  **return** resolver;  }  } |

## 自定义HandlerExceptionResovler类处理异常

### 方法

需要在全局异常处理类中实现HandlerExceptionResolver接口

### 案例

|  |
| --- |
| @Configuration  **public** **class** GlobalException **implements** HandlerExceptionResolver{  @Override  **public** ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) {  // **TODO** Auto-generated method stub  ModelAndView mv = **new** ModelAndView();  // 判断不同异常类型，做不同视图跳转  **if** (ex **instanceof** ArithmeticException) {  mv.setViewName("error1");  }  **if** (ex **instanceof** NullPointerException) {  mv.setViewName("error2");  }  mv.addObject("error", ex.toString());  **return** mv;  }  } |