## @ResponseStatus注解处理异常

@ResponseStatus注解，可以修饰一个异常类，**在发生异常的时候返回指定的错误码和消息**，在返回的 reason中包含中文的时候，就会出现中文乱码的问题

### 1.1 我们创建一个异常类，加上注解:

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

**@ResponseStatus(value=HttpStatus.FORBIDDEN,reason="用户不匹配")**

public class UserNotMatchException extends RuntimeException{}

1）ResponseStatus注解是修饰类的

2）它有两个属性，value属性是http状态码，比如404，500等。reason是错误信息

### 1.2 写一个目标方法抛出该异常

@RequestMapping("/testResponseStatus")

public String testResponseStatus(int i){

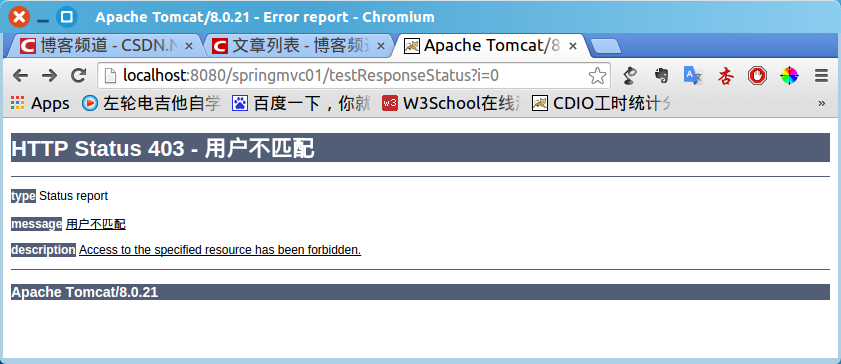
if(i==0)

throw new UserNotMatchException();

return "hello";

}

### 1.3 结果



## @ExceptionHandler的使用

### 2.1 创建一个baseController类，用于捕获全局异常，并处理

|  |
| --- |
| **public** **class** baseController {  /\*\* 基于@ExceptionHandler捕获全局的异常，并处理 \*/  @ExceptionHandler  **public** String exp(HttpServletRequest request, Exception ex) {    request.setAttribute("ex", ex);  // 根据不同错误转向不同页面  **if**(ex **instanceof** BusinessException) {  **return** "businessError";  }**else** **if**(ex **instanceof** ParameterException) {  **return** "parameterError";  }**else** **if**(ex **instanceof** MyException) {  **return** "myError";  }**else** **if**(ex **instanceof** SQLException) {  **return** "sqlError";  }**else** {  **return** "404";  }  }  } |

### 2.2 在具体的controller类中，继承baseController，当出现异常时，抛给baseController处理

|  |
| --- |
| @Controller  **public** **class** ExceptionController **extends** baseController{  @Autowired  ExceptionServiceInterface exceptionService;  @RequestMapping("/showIndex")  **public** String showIndex(){  **return** "index";  }  @RequestMapping("/daoSqlEx")  **public** **void** daoSqlEx() **throws** Exception{  exceptionService.daoSqlEx();  }    @RequestMapping("/daoParaEx")  **public** **void** daoParaEx() **throws** Exception{  exceptionService.daoParaEx();  }  ........  } |

### 2.3 在对应的错误页面中，打印出错误信息

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*  pageEncoding=*"utf-8"* isErrorPage=*"true"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>Insert title here</title>  </head>  <body>  <h1>sql数据库异常：</h1><%=exception %>  <h2>错误内容是：</h2>  <%  exception.printStackTrace(response.getWriter());  %>  </body>  </html> |

## @ControllerAdvice和@ExceptionHander配合使用

### 3.1 构建一个全局异常处理的类ExceptionAdvice

**即当controller类抛出异常后，直接交由ExceptionAdvice进行处理异常**

|  |
| --- |
| @ControllerAdvice  **public** **class** ExceptionAdvice {  /\*\* 基于@ExceptionHandler捕获全局的异常，并处理 \*/  @ExceptionHandler  **public** String exp(HttpServletRequest request, Exception ex) {    request.setAttribute("ex", ex);    // 根据不同错误转向不同页面  **if**(ex **instanceof** BusinessException) {  **return** "businessError";  }**else** **if**(ex **instanceof** ParameterException) {  **return** "parameterError";  }**else** **if**(ex **instanceof** MyException) {  **return** "myError";  }**else** **if**(ex **instanceof** SQLException) {  **return** "sqlError";  }**else** {  **return** "404";  }  }  } |

注意：此时controller类不用在extends该ExceptionAdvice ，类似Spring的AOP面向切面编程