### Demo搭建出现的小问题

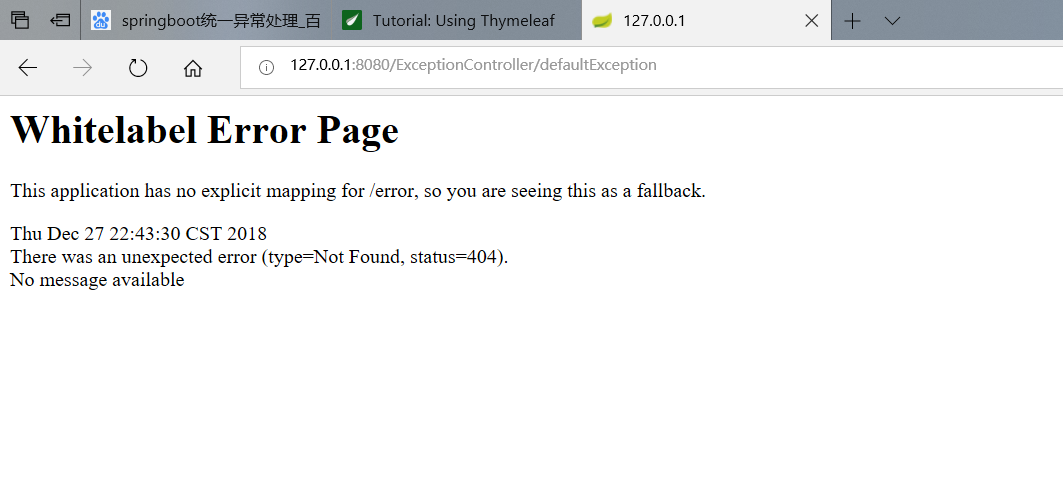
1. 在使用thymeleaf时，对象的set,get方法必须按规则写。属性id对应的方法必须为getId和setId。
2. spring的数据库用户名设置是spring.datasource.username，而不是spring.datasource.username。
3. mysql存入数据出现乱码，首先查看mysql中字段是否为utf8，如果是则将配置的数据库路径修改为如下字样jdbc:mysql://47.74.238.43:3306/hapi\_mysql?useUnicode=true&characterEncoding=utf8。
4. mybatis由于选择了启动依赖所以使用mybatis很方便，只需配置数据源即可，如果没有选择mybatis启动依赖，则可以按spring+mybatis整合方式配置。

### Spring异常处理

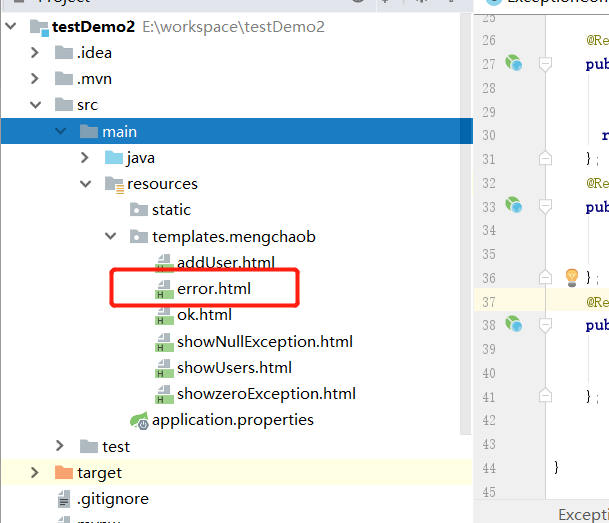
springboot提供了五种异常处理方案

#### Springboot自带方案，自定义错误页面

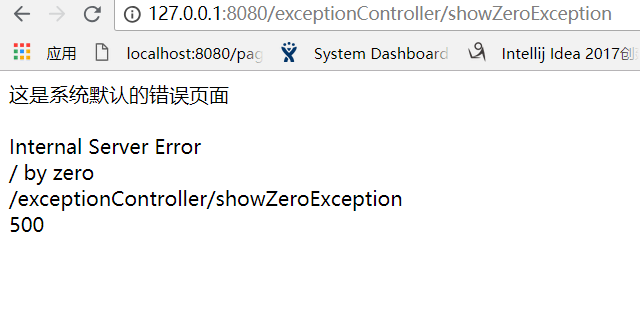
SpringBoot 默认的处理异常的机制：SpringBoot 默认的已经提供了一套处理异常的机制。一旦程序中出现了异常 SpringBoot 会像/error 的 url 发送请求。在 springBoot 中提供了一个叫 BasicExceptionController 来处理/error 请求，然后跳转到默认显示异常的页面来展示异常信息。



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，需要再src/main/resources/templates 目录下创建 error.html 页面。注意：名称必须叫 error。







#### @ExceptionHandler

此ExceptionHandler注解,捕获了ArithmeticException，这个ExceptionController所抛出的捕获了ArithmeticException异常都会被这个方法捕捉，跳转到对应页面如果想要捕捉别的异常，那么可以在注解中可以放多个异常class@ExceptionHandler({ArithmeticException.class,java.lang.NullPointerException.class}) 如果需要对不同的异常做不同的处理，可以在捕捉异常的方法中做异常类型判断；也可以在创建一个@ExceptionHandler。

*/\*  
 \* @Author mengchao  
 \* @Description //****TODO*** *\* @Date 23:42 2018-12-27  
 \* @Param  
 \* @return  
 \*\*/*@ExceptionHandler({ArithmeticException.**class**})  
**public** String exceptionNameHanddler(Exception exception,Model model){  
 *//方法里面处理异常信息，并返回到视图解析器* model.addAttribute(**"message"**,exception.getMessage());  
 **return "/errorinfo/exceptionHandler"**;  
}

#### @ControllerAdvice+@ExceptionHandler 针对spring环境下所有的controller抛出的异常

@ControllerAdvice  
public class ExceptionAdvice {//java.lang.NullPointerException.class  
 //同controller中的ExceptionHandler注解一样，设定方法捕获的异常  
 @ExceptionHandler(java.lang.NullPointerException.class)  
 public String nullPointerException(Exception e,Model model){//这里面也可以返回视图 ModelAndView  
 model.addAttribute("message",e.getMessage()+"空指针异常！");  
 return "/errorinfo/exceptionAdvice";  
 }  
  
}

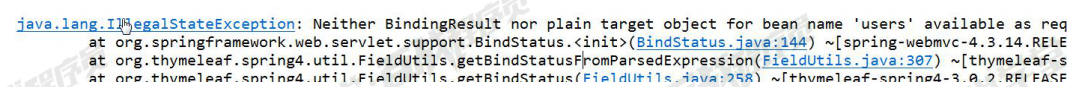
1. 在java中配置异常解析器SimpleMappingExceptionResolver同意处理异常  
   @Configuration  
   **public class** ExceptionConfig {  
    @Bean  
    **public** SimpleMappingExceptionResolver getSimpleMappingExceptionResolver(){  
    SimpleMappingExceptionResolver smer = **new** SimpleMappingExceptionResolver();  
    Properties p = **new** Properties();  
    *//可以配置多个异常捕捉，但是需要注意，p中的key必须是异常的全路径* p.put(**"java.lang.ArrayIndexOutOfBoundsException"**,**"/errorinfo/exceptionConfig"**);  
    smer.setExceptionMappings(p);  
    *//页面可以通过${exception}获取信息* ModelAndView mav = **new** ModelAndView();  
    **return** smer;  
    }  
   }
2. 自定义 HandlerExceptionResolver 类处理异常

使用此种方法设定全局异常处理一定要注意，必须有@bean注解那一段，没有这一段代码，系统默认会查找默认路径下的error.html,当error.html不存在才会找到我们在类中配置的试图路径。当存在@bean注解那一段代码后，系统才会一直取我们在类中配置的路径

@Configuration  
**public class** MyHandlerExceptionResolver **implements** HandlerExceptionResolver {  
 @Bean  
 **public** MyHandlerExceptionResolver getexceptionHandler(){  
 **return new** MyHandlerExceptionResolver();  
 }  
 @Override  
 **public** ModelAndView resolveException(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o, Exception e) {  
 ModelAndView mav = **new** ModelAndView();  
 *//通过判断不同的异常类型，跳转不同页面* **if**(e **instanceof** java.io.IOException){  
 mav.setViewName(**"/errorinfo/MyHandlerExceptionResolver"**);  
 mav.addObject(**"info"**,**"找不到文件！"**);  
 }  
 mav.addObject(**"exceptioninfo"**,e.fillInStackTrace());  
*/\* Map<String, Object> model = new HashMap<String, Object>();  
 model.put("appName", appName);  
 return new ModelAndView("/error/jsonError", model);\*/* **return** mav;  
 }  
}

表单校验  
1.表单校验，用的是hibnet的一个工具包。在controller中需要的路径方法的参数前增加@Validated注解，在此参数对象对应的属性中增加@NotBlank,@Min(15),Max(30)注解标识需要校验的字段及校验内容和返回信息。

1. 注意,被@Validated注解的参数，其注解名称最好将参数对应类的类名首字母变为小写作为形参的名字，这样在使用校验结果集BindingResult(封装了校验结果)时不需要在做特殊处理，否则需要利用注解@@ModelAttribute重定义属性名称，这样在页面就可以使用重定义的属性名称访问对应属性值了。
2. 当模板文件会有两个路径不同的路径跳转的话，必须保证两个路径都存在model，否则页面会报如下所示的错误。



4.[javax.validation.UnexpectedTypeException: HV000030: No validator could be found for constraint](https://www.cnblogs.com/softidea/p/6044123.html)

使用hibernate validator出现上面的错误， 需要 注意

@NotNull 和 @NotEmpty  和@NotBlank 区别

@NotEmpty 用在集合类上面  
@NotBlank 用在String上面  
@NotNull    用在基本类型上

5.在使用表单校验时，@Validated注解的参数后面必须紧跟着 BindingResult对象参数，否则springboot会抛出异常