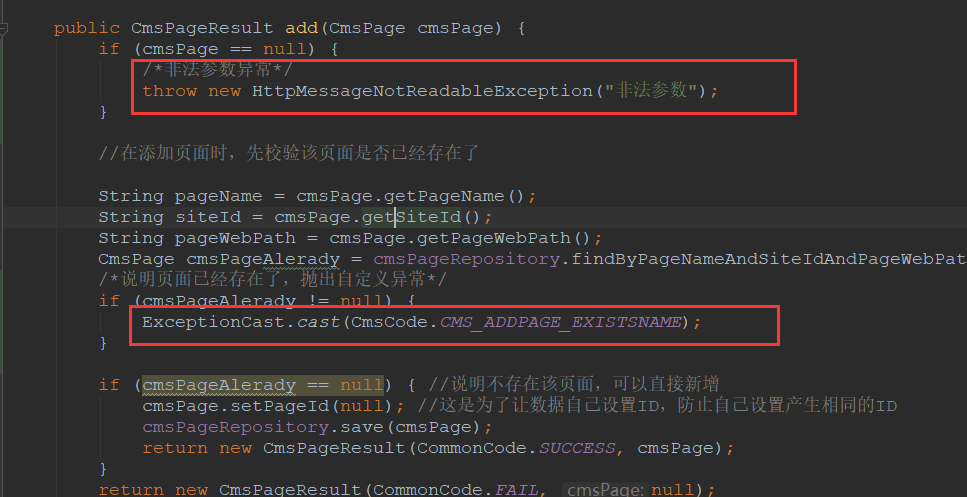
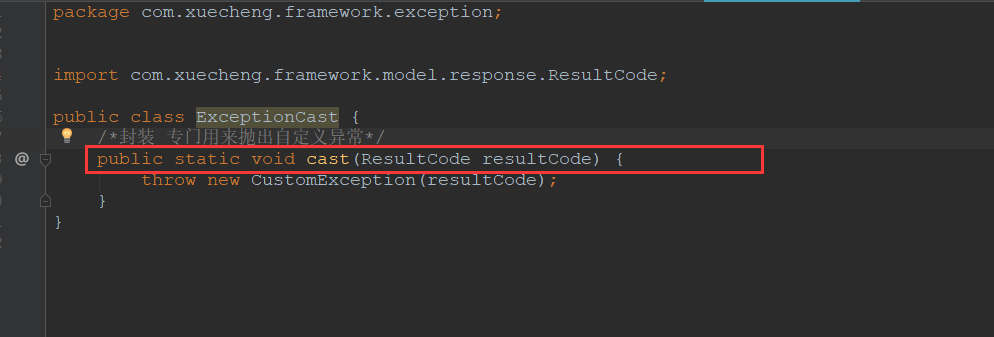
**全局异常统一处理类：建议结合代码和课程笔记看**

package com.xuecheng.framework.exception;  
  
/\*异常处理类 所有的异常全被都在这被捕获，统一做处理\*/  
  
  
import com.google.common.collect.ImmutableMap;  
import com.xuecheng.framework.model.response.CommonCode;  
import com.xuecheng.framework.model.response.ResponseResult;  
import com.xuecheng.framework.model.response.ResultCode;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.http.converter.HttpMessageNotReadableException;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.annotation.ResponseBody;  
  
  
@ControllerAdvice /\*使用springboot注解@ControllerAdvice，表示是一个增强的Controller，其中最常用就是用来做全局异常处理\*/  
public class ExceptionCatch {  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(ExceptionCatch.class);  
  
 /\*使用ImmutableMap存放非自定义异常，key是异常类型，value是ResultCode\*/  
 /\*使用ImmutableMap的特点是一旦创建，里面的值不可变，并且线程是安全的\*/  
 private static ImmutableMap<Class<? extends Throwable>, ResultCode> *EXCEPTIONS*;  
 /\*ImmutableMap 需要用Builder来构建\*/  
 protected static ImmutableMap.Builder<Class<? extends Throwable>, ResultCode> *builder* = ImmutableMap.*builder*();  
  
 static {  
 /\*使用builde向map中添加key，value，这里添加了一个key是请求参数不可用的异常，错误吗为CommonCode.INVALIDPARAM\*/  
 *builder*.put(HttpMessageNotReadableException.class, CommonCode.*INVALIDPARAM*);  
 /\*后面你还可以添加其他错误异常\*/  
 // builder.put(FileNotFoundException.class,CommonCode.INVALIDPARAM);  
 }  
  
  
 @ExceptionHandler(CustomException.class) /\*代表该方法只针对处理自定义CustomException异常的\*/  
 @ResponseBody/\*相应给前端以json格式，否则前端无法解析，这里返回的数据\*/  
 public ResponseResult customerException(CustomException customException) {  
 *LOGGER*.error("catch exception:{" + customException.getMessage() + "}");  
 ResultCode resultCode = customException.getResultCode();  
 return new ResponseResult(resultCode);  
 }  
  
 /\*专门处理不可预知的异常，即非自定义异常\*/  
 @ExceptionHandler(Exception.class)  
 @ResponseBody  
 public ResponseResult exception(Exception e) {  
 *LOGGER*.error("catch exception:{"+e.getMessage()+"}");  
 if(*EXCEPTIONS* == null) {  
 *EXCEPTIONS* = *builder*.build(); /\*开始构建，构建好了，map里面之前put进去才会真正2进入map中,否则EXCEPTIONS是空值  
 这不一定要先做\*/  
 }  
  
 /\*那么先从map中看是不是我们知道的异常，如果是的话，就返回对应对的value（错误信息）,否则一律返回9999错误状态码\*/  
 ResultCode resultCode = *EXCEPTIONS*.get(e.getClass());  
 if (resultCode != null) { /\*说明map中有这中错误类型对应的返回给客户端的错误信息\*/  
 return new ResponseResult(resultCode);  
 }  
 else {  
 return new ResponseResult(CommonCode.*SERVER\_ERROR*);  
 }  
 }  
}

**抛出异常让其捕获：**



**封装抛出自定义异常：**



**自定义异常类：**

