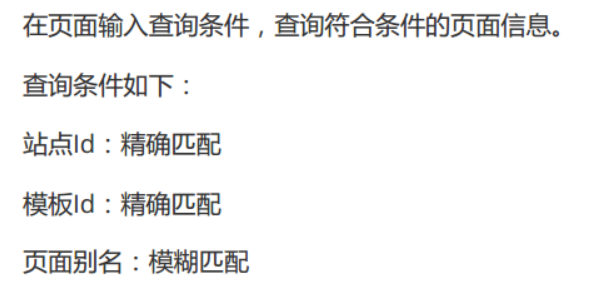
1. **使用MongoRepository的findAll方法实行自定义条件查询(查询现有页面)**





**实现通过站点名称+页面名称来查找页面**

findAll(Example<S> var1, Pageable var2

//条件匹配器

ExampleMatcher exampleMatcher = ExampleMatcher.matching();

exampleMatcher = exampleMatcher.withMatcher("pageAliase",

ExampleMatcher.GenericPropertyMatchers.contains());

//页面别名模糊查询，需要自定义字符串的匹配器实现模糊查询

//ExampleMatcher.GenericPropertyMatchers.contains() 包含

//ExampleMatcher.GenericPropertyMatchers.startsWith()//开头匹配

**2.新增页面、修改页面、删除页面**

新增页面

在cms\_page集合中创建页面名称、站点Id、页面webpath(其实对应的是物理路径，利用nginx进行了代理)为唯一索引，用这三个条件确定唯一页面。添加之前要确定页面唯一性，利用这三个参数去查询页面，如果查到则不添加。

@ApiOperation("添加页面")  
public CmsPageResult add(CmsPage cmsPage);

修改页面操作流程

1、用户进入修改页面，在页面上显示了修改页面的信息

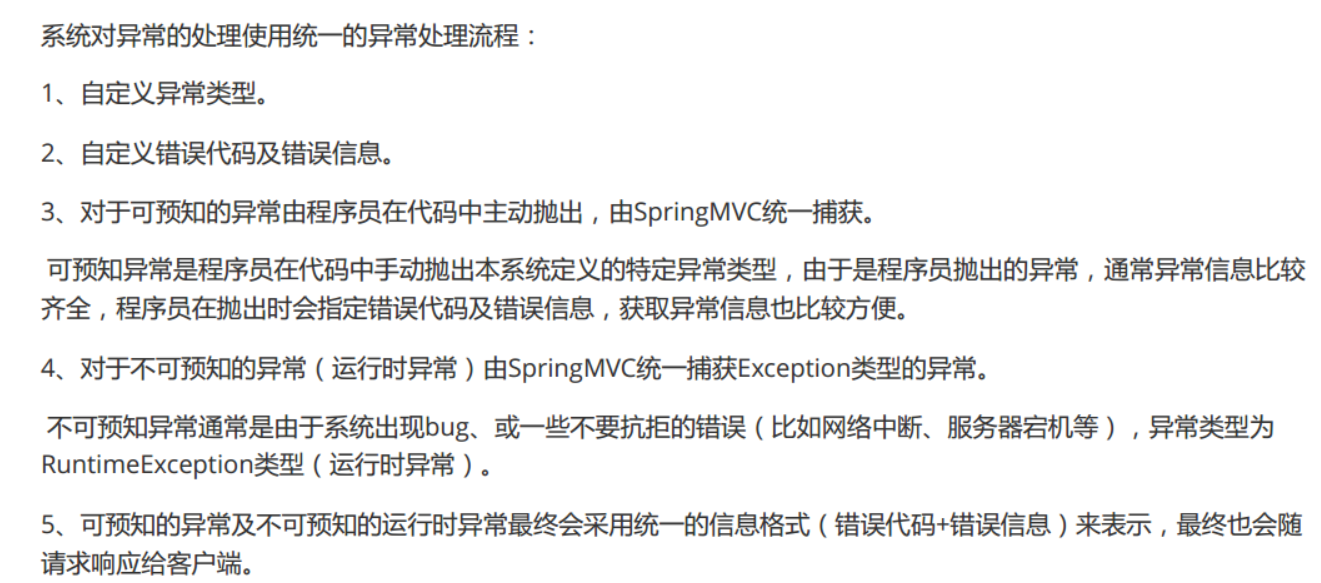
2、用户修改页面的内容，点击“提交”，提示“修改成功”或“修改失败”

@ApiOperation("通过ID查询页面")  
public CmsPage findById(String id);  
@ApiOperation("修改页面")  
public CmsPageResult edit(String id,CmsPage cmsPage);

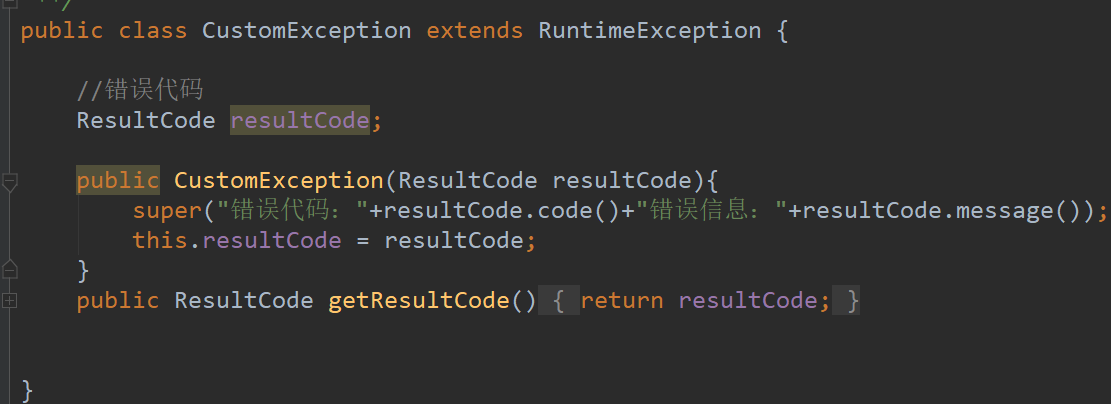
**3.统一的异常处理**

在统一异常处理类中去捕获异常，无需controller捕获异常，向用户返回统一规范的响应信息。

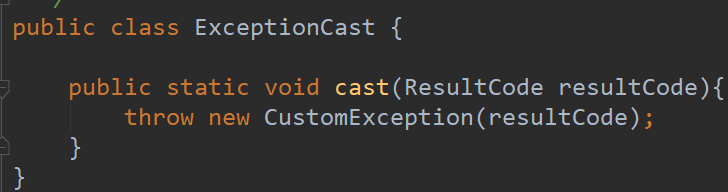
异常处理流程



自定义异常类型



异常抛出类

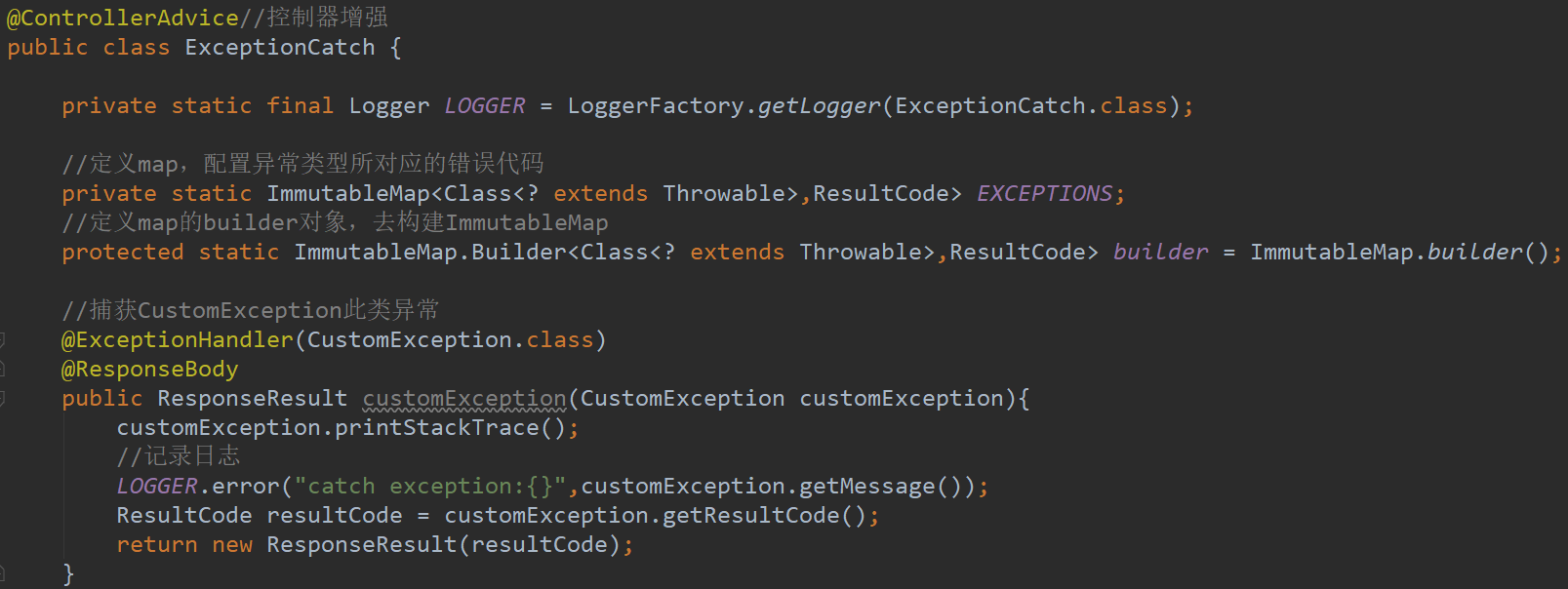


异常捕获类

使用 @ControllerAdvice和@ExceptionHandler注解来捕获指定类型的异常

@ControllerAdvice 注解，可以用于定义@ExceptionHandler、@InitBinder、@ModelAttribute，并应用到所有被@RequestMapping注解标记的方法上。

捕获自定义异常(可预知异常)



捕获不可预知异常

方法：定义一个map，配置异常类型所对应的错误代码

从这个map中找异常类型所对应的错误代码，如果找到了将错误代码响应给用户，如果找不到给用户响应99999异常

