

SpringBoot2.7.18升级至3.2.x后的Knife4j的系列问题

一、问题描述

项目采用SpringBoot开发，原来是SpringBoot2.7.18+knife4j-spring-boot-starter(3.0.3)，使用非常完美。不过由于升级JDK至21之后（原先JDK为8或者15），需要同步升级SpringBoot，所以索性升级了下knife4j，经过数天数月的调测，还是有些不太完美的地方，寻找了官方文档和所能搜寻到的CSDN等别人写的示例文档后，仍有部分问题。汇总如下，如作者有空，还请帮忙看下，不胜感谢！

二、项目配置

以下问题均在此配置下测试获得，配置包括pom引入的依赖、yml中knife4j配置以及配置类

1. pom.xml 相关配置

升级后配置，以下问题均在此版本中描述，SpringBoot3.2.4+knife4j-openapi3-jakarta-spring-boot-starter(4.5.0)

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.4</version><!-- 2.7.18↑-->
    <relativePath/>
</parent>
<dependencies>
    <dependency>
        <groupId>com.github.xiaoymin</groupId>
        <artifactId>knife4j-openapi3-jakarta-spring-boot-starter</artifactId>
        <version>4.5.0</version>
    </dependency>
</dependencies>
```

2. application.yml 中相关配置

```
# springdoc-openapi项目配置
springdoc:
    # get请求多参数时不需要添加额外的@ParameterObject和@Parameter注解
    default-flat-param-object: true
    # 启用swaggerUI
    swagger-ui:
        #自定义swagger前端请求路径，输入http://127.0.0.1:8080/swagger-ui.html会自动重定向到swagger
        #页面
        path: /swagger-ui.html
        enabled: true
```

```

tags-sorter: alpha # 标签的排序方式 alpha:按照子母顺序排序 (@ApiSupport注解排序不生效, 因此需要设置)
#operations-sorter: alpha # 接口的排序方式 alpha:按照子母顺序排序 (@ApiOperationSupport注解排序生效, 因此这里不作设置)
# 启用文档, 默认开启
api-docs:
  path: /v3/api-docs    #swagger后端请求地址
  enabled: true
#knife4j相关配置 可以不用改
knife4j:
  enable: true      #开启knife4j, 无需添加@EnableKnife4j注解
  setting:
    language: ZH_CN   # 中文:ZH_CN 英文:EN
#    enable-swagger-models: true
#    enable-dynamic-parameter: false
    footer-custom-content: "<strong>Copyright © 2024 Keyidea. All Rights Reversed</strong>"
    enable-footer-custom: true
    enable-footer: true
    enable-document-manage: true

```

3、Knife4j配置类

```

package cn.keyidea.common.config;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Contact;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.info.License;
import org.springdoc.core.models.GroupedOpenApi;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * Knife4jConfig
 * 注意: 分组名称使用英文或数字, 在4.0.0~4.5.0的Knife4j版本中使用中文分组会出现页面访问异常
 * <p>
 * Springboot 3.0 集成 Knife4j 4.0版本, 排序无效 · Issue #I6FB9I · 萧明/knife4j - Gitee.com
 * https://gitee.com/xiaoym/knife4j/issues/I6FB9I
 * 参见 {@link com.github.xiaoymin.knife4j.spring.extension.Knife4jJakartaOperationCustomizer}
 *
 * @author qyd
 * @date 2024-04-13
 */
@Configuration
public class Knife4jConfig {

  @Bean
  public GroupedOpenApi api1() {
    // 创建了一个api接口的分组
    return GroupedOpenApi.builder()
      // 分组名称, 使用英文, 中文访问异常
      .group("01-sys-api")
      // .group("01-系统接口")
      // 接口请求路径规则

```

```

        .pathsToMatch( "/sys/**" )
        .build();
    }

    @Bean
    public GroupedOpenApi api2() {
        return GroupedOpenApi.builder()
            .group("02-business-api")
            .packagesToScan("cn.keyidea.business")
            .build();
    }

    @Bean
    public OpenAPI openAPI() {
        return new OpenAPI()
            .info(new Info()
                .title("项目接口管理")
                .description("接口文档")
                .version("v1")
                .contact(new Contact().name("Keyidea").email("support@qq.cn"))
                .license(new License().name("Apache
2.0")).url("http://springdoc.org"))
            );
    }
}

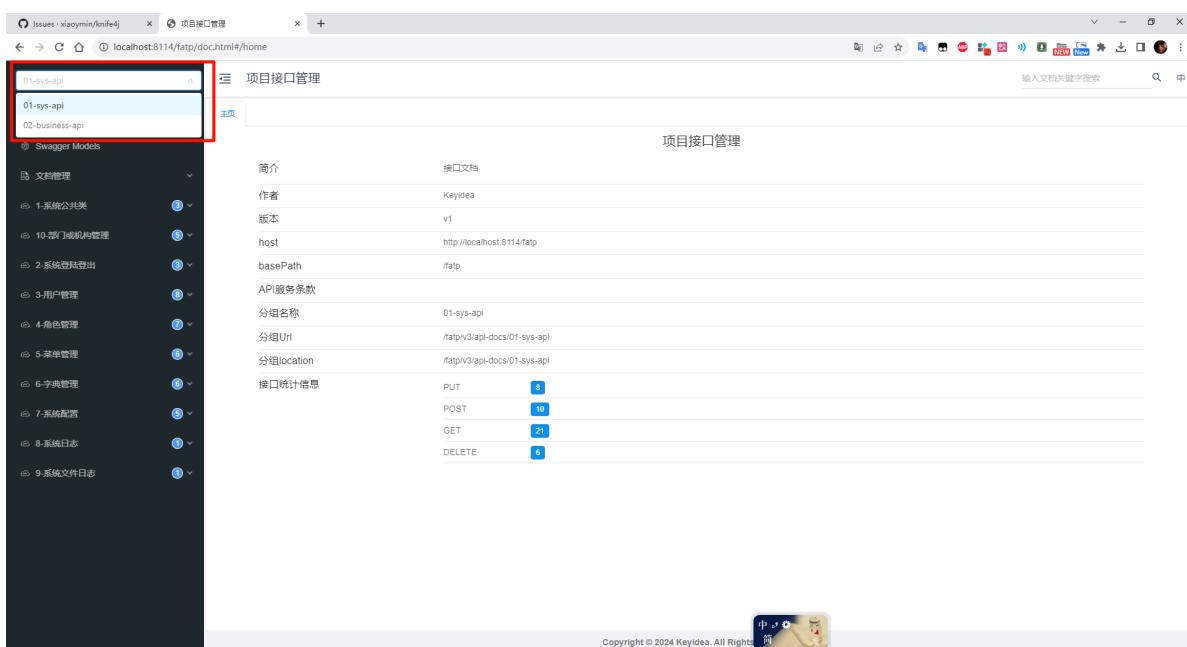
```

三、问题汇总

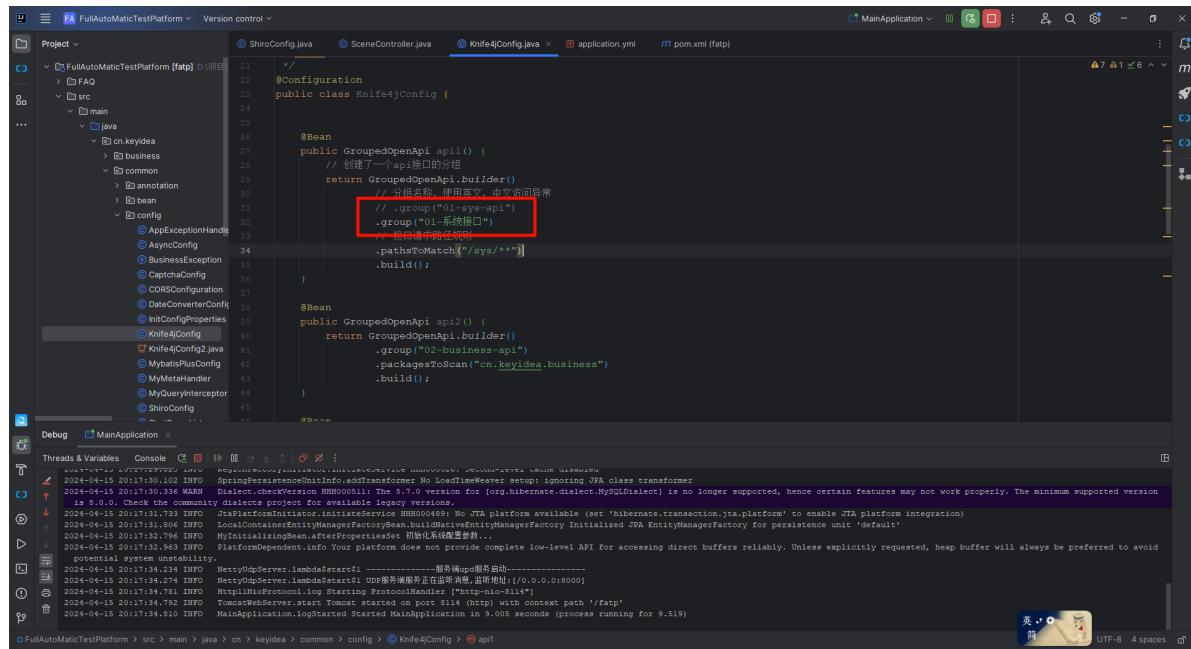
影响指数：0/5完全不影响，5/5完全受影响

1、分组名称不支持中文名称[影响指数:4/5]

使用英文分组名，访问正常

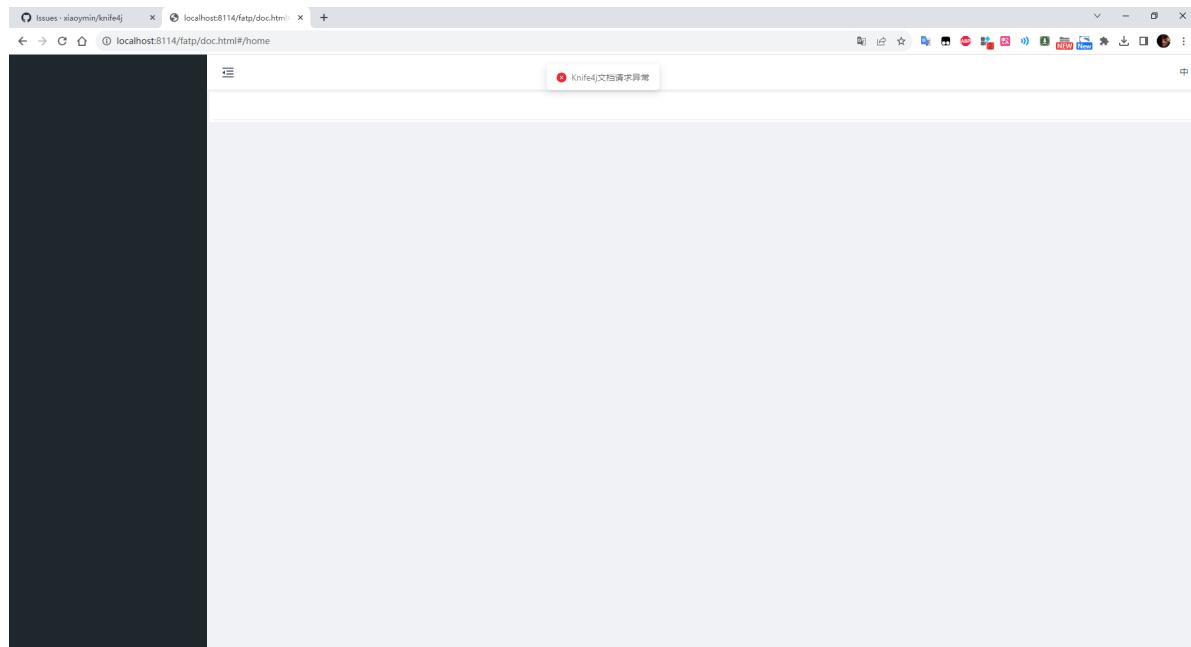


使用中文分组名后访问异常

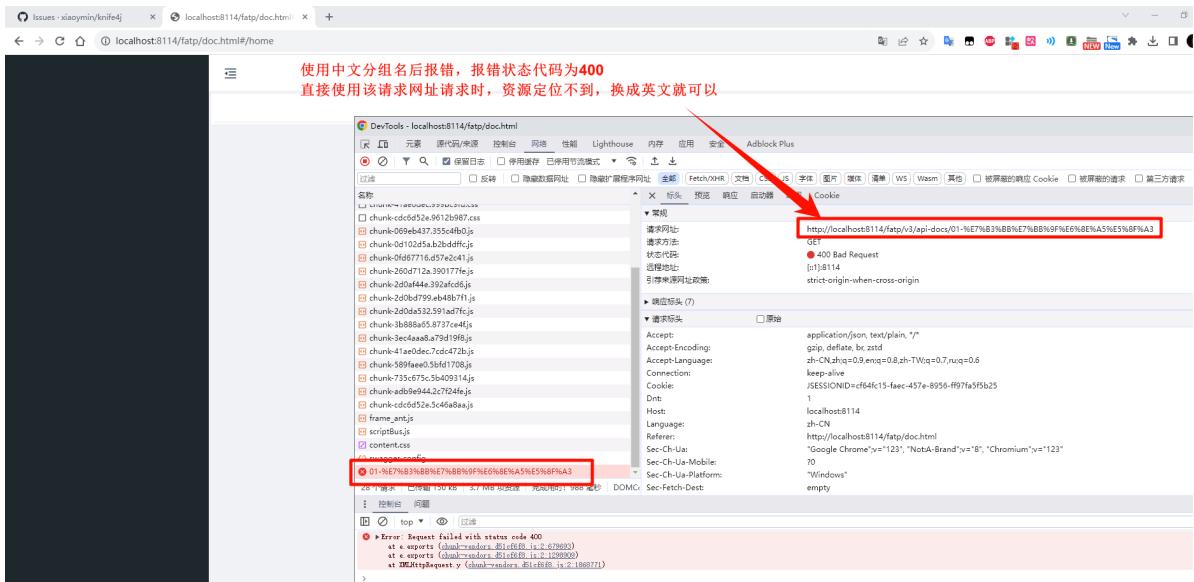


```
21     /**
22      * @Configuration
23      * public class Knife4jConfig {
24      *
25      *     @Bean
26      *     public GroupedOpenApi api() {
27      *         // 创建了一个api接口的分组
28      *         return GroupedOpenApi.builder()
29      *             // 分组名称，使用英文，中文访问异常
30      *             .group("01-sys-api") // .group("01-系统接口")
31      *             .pathToMatch("/sys/**")
32      *             .build();
33      *     }
34
35     @Bean
36     public GroupedOpenApi api2() {
37         return GroupedOpenApi.builder()
38             .group("02-business-api")
39             .packagesToScan("cn.keyidea.business")
40             .build();
41     }
42 }
```

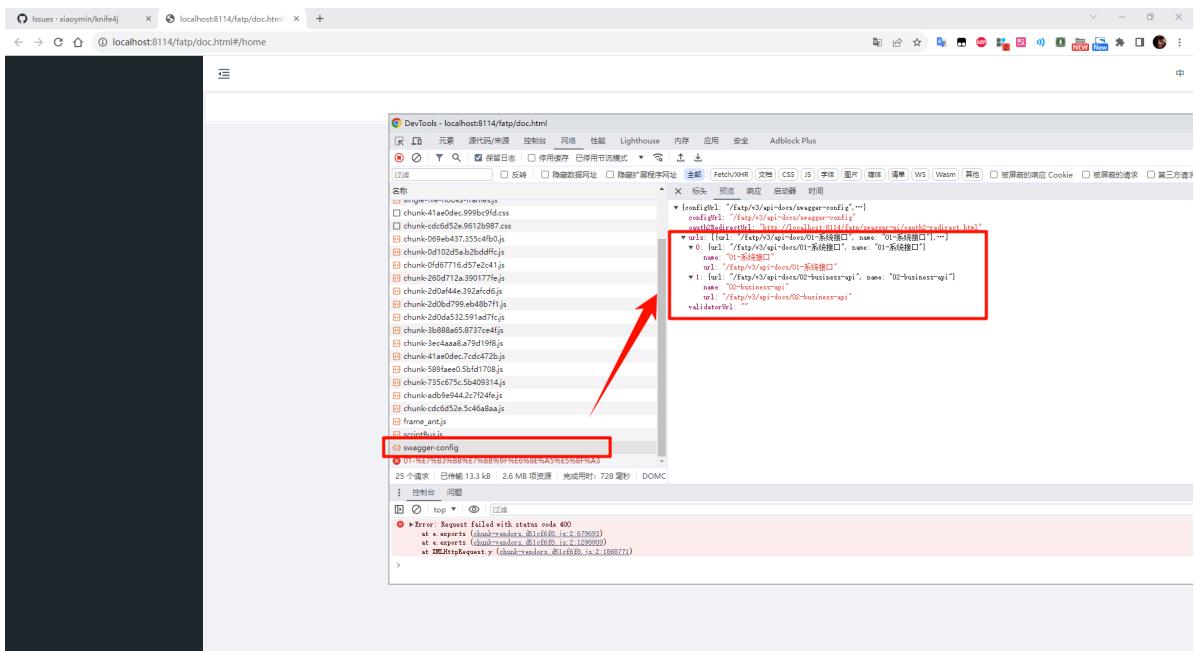
异常界面



异常详情



请求异常时swagger-config请求信息详情



2、使用@Tag中order字段对控制器排序不生效[影响指数:5/5]

将yml中tags-sorted排序注释

```

pathmatch:
    matching-strategy: ant_path_matcher
resources:
    static-locations: classpath:/static/
throw-exception-if-no-handler-found: true

# Knife4j配置
#knife4j:
#  enable: true # 开启增强模式 (增强功能需要通过配置yml配置文件开启增强,自2.0.7开始)
#  springdoc-openapi项目配置
springdoc:
    # yes请求参数时不需要添加额外的@ParameterObject和@Parameter注解
    default-flatten-param-object: true
    # 到swaggerUI
    swagger-ui:
        # 自定义Swagger前端请求路径, 输入http://127.0.0.1:8080/swagger-ui.html会自动重定向到swagger页面
        path: /swagger-ui.html

        # tags-sorter: alpha # 标签的排序方式 alpha:按照字母顺序排序 (@ApiSupport注解排序不生效,因此需要设置)
        # operationSorter: alpha # 接口的排序方式 alpha:按照字母顺序排序 (@ApiOperationSupport注解排序生效,因此这里不作设置)

        # api-docs:
        path: /v3/api-docs # swagger后端请求地址
        enabled: true
        #knife4j相关配置 可以不用改
        knife4j:
            enable: true # 开启knife4j, 无需添加@EnableKnife4j注解
            setting:
                language: zh_CN # zh_CN 英文:EN

```

开启@ApiSupport中order排序

```

package cn.keyidea.business.controller;
import ...;

/**
 * <p>
 * 测试场景表 扇端控制器
 * </p>
 *
 * @author qyd
 * @since 2023-09-23
 */
@ApiSupport(order = 6)
@Tag(name = "6-测试场景接口", description = "测试场景管理")
@RequestMapping(value = "/v1/scenes", produces = MediaType.APPLICATION_JSON_VALUE)
public class SceneController {
    @Autowired
    private SceneService sceneService;

    @SysLogAnnotation(module = "测试场景管理", serviceDesc = "测试场景管理-分页查询", serviceType = ConstantsExpand.ServiceType.QUERY)
    @ApiOperation(summary = "分页查询222", description = "<strong>注意</strong><br>该分页查询需要输入必须的参数,如当前页或分页数等")
    @Parameters({
        @Parameter(name = "sceneName", description = "测试场景名称", required = false, example = "测试场景名称1", in = ParameterIn.QUERY),
        @Parameter(name = "status", description = "状态(0-正常,1-禁用)", required = false, example = "0", in = ParameterIn.QUERY),
        @Parameter(name = "current", description = "当前列", required = true, example = "1", in = ParameterIn.QUERY),
        @Parameter(name = "size", description = "分页数", required = true, example = "15", in = ParameterIn.QUERY)
    })
}

```

接口中显示系统接口排序错乱

The screenshot shows the knife4j documentation interface. On the left, there's a sidebar with various system management categories like User Management, Role Management, etc. A red box highlights the '1 系统公共类' section under '01-sys-api'. The main content area shows a 'File Upload' API endpoint with a POST method. It details the request body type as application/x-www-form-urlencoded, response data type as JSON, and parameters like fileType, file, and type. It also shows the response status (200 OK) and response parameters (code, msg, data). A note at the bottom states: '以系统公共类为例，前面1代表使用@ApiSupport设置为1，应该排在第一位，但是未生效'.

代码中设置

The screenshot shows an IDE environment with a Java project structure on the left. The current file is 'CommonController.java'. A red box highlights the line containing the annotation '@ApiSupport(order = 1)'. The code snippet is as follows:

```

1 package cn.keyidea.sys.controller;
2 import ...
3 /**
4 * 系统公共类
5 */
6 @Author qrd
7 @Date 2022-10-17
8
9 @ApiSupport(order = 1)
10 @Tag(name = "1-系统公共类", description = "")
11 @ApiOperationSupport(author = "qrd", order = 1)
12 @RequestMapping("/sys/common/")
13 public class CommonController
14 {
15     private final static Logger logger = LoggerFactory.getLogger(CommonController.class);
16
17     @Autowired
18     private SysFileLogService sysFileService;
19
20     @SysLogAnnotation(module = "公共类", serviceDesc = "公共类-文件上传", serviceType = ConstantsExpand.ServiceType.UPLOAD)
21     @ApiOperationSupport(author = "qrd", order = 1)
22     @Operation(summary = "文件上传", description = "")
23     @Parameters({
24         @Parameter(name = "file", description = "单文件上传", required = true, in = ParameterIn.DEFAULT),
25         @Parameter(name = "fileType", description = "文件类型", required = true, example = "txt", in = ParameterIn.DEFAULT),
26         @Parameter(name = "type", description = "是否使用文件原始名称: 1-使用, 其他-不使用(使用随机UUID)", required = false, example = "1")
27     })
28     @PostMapping("uploadFile")
29     public BaseRes uploadFile(@RequestParam(value = "file", required = true) MultipartFile file,
30                               @RequestParam(value = "fileType", required = true) String fileType,
31                               @RequestParam(value = "type", required = false) Integer type)
32     {
33         try
34         {
35             return sysFileService.uploadFile(file, fileType, type);
36         }
37         catch (Exception e)
38         {
39             logger.error("上传文件失败: " + e.getMessage());
40             return BaseRes.error("上传文件失败");
41         }
42     }
43 }

```

其他系统类类同

说明：使用@ApiSupport对控制器排序未生效，但是使用@ApiOperationSupport中的order对单个接口设置的值生效，很奇怪。

3、无法定义全局错误码信息[影响指数:5/5]

目前有两种方式在接口中显示错误码，一种是使用@ApiResponse注解，另一种是@ApiResponses注解，但是这两种注解都均需要在每个接口上进行添加，很是麻烦，以前的knife4j3.0.3的版本中可以如下配置

```

@Order(2)
@Bean
public Docket createRestApi2() {
    // 添加全局响应状态码
    List<Response> responseMessageList = new ArrayList<>();
    // 根据 StatusCode 获取自定义响应码
    Arrays.stream(StatusCode.values())
}

```

```

        .forEach(errorEnums -> responseMessageList.add(new ResponseBuilder()
            .code(errorEnums.getCode())
            .description(errorEnums.getMsg())
            .build()));

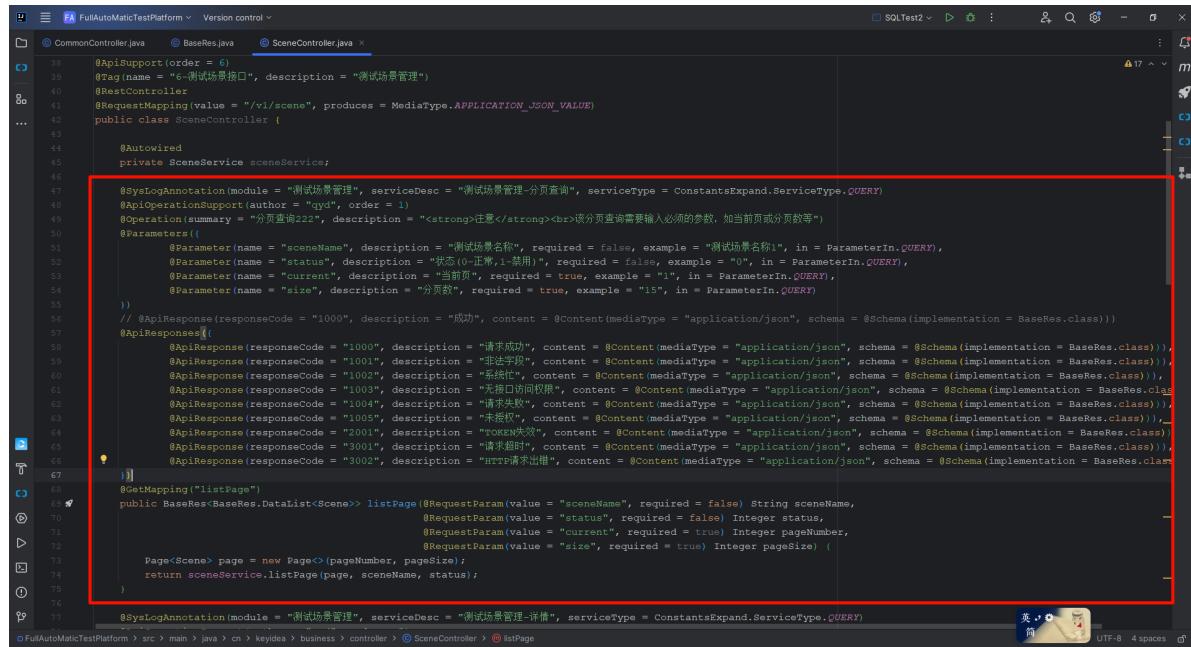
Docket build = new Docket(DocumentationType.SWAGGER_2)
    // 添加全局响应状态码，可根据不同系统定义不同的响应码信息
    .globalResponses(HttpMethod.GET, responseMessageList)
    .globalResponses(HttpMethod.PUT, responseMessageList)
    .globalResponses(HttpMethod.POST, responseMessageList)
    .globalResponses(HttpMethod.DELETE, responseMessageList)
    .apiInfo(apiInfo())
    .groupName("业务接口文档")
    .select()
    .apis(RequestHandlerSelectors.basePackage("cn.keyidea.business"))
    .build();

return build;
}

```

在Docket中添加一个globalResponses配置即可，但是新的knife4j好像没法这样添加。

目前的解决办法是可以在每个接口上写@ApiResponses或@ApiResponse注解，如下：



```

38     @ApiSupport(order = 6)
39     @Tag(name = "6-测试场景接口", description = "测试场景管理")
40     @RestController
41     @RequestMapping(value = "/vi/scene", produces = MediaType.APPLICATION_JSON_VALUE)
42     public class SceneController {
43
44         @Autowired
45         private SceneService sceneService;
46
47         @SysLogAnnotation(module = "测试场景管理", serviceDesc = "测试场景管理-分页查询", serviceType = ConstantsExpand.ServiceType.QUERY)
48         @ApiOperation(summary = "分页查询222", description = "<strong>注意</strong><br>该分页查询需要输入必须的参数,如当前页数分页数等")
49         @Parameters({
50             @Parameter(name = "sceneName", description = "测试场景名称", required = false, example = "测试场景名称!", in = ParameterIn.QUERY),
51             @Parameter(name = "status", description = "状态(0-正常,1-禁用)", required = false, example = "0", in = ParameterIn.QUERY),
52             @Parameter(name = "current", description = "当前页", required = true, example = "1", in = ParameterIn.QUERY),
53             @Parameter(name = "size", description = "分页数", required = true, example = "15", in = ParameterIn.QUERY)
54         })
55         // @ApiResponse(responseCode = "1000", description = "成功", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class)))
56         @ApiResponses([
57             @ApiResponse(responseCode = "1000", description = "请求成功", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
58             @ApiResponse(responseCode = "1001", description = "非法字符", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
59             @ApiResponse(responseCode = "1002", description = "系统忙", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
60             @ApiResponse(responseCode = "1003", description = "无接口访问权限", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
61             @ApiResponse(responseCode = "1004", description = "请求失败", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
62             @ApiResponse(responseCode = "1005", description = "未授权", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
63             @ApiResponse(responseCode = "2001", description = "TOKEN失效", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
64             @ApiResponse(responseCode = "3001", description = "请求超时", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class))),
65             @ApiResponse(responseCode = "3002", description = "HTTP请求出错", content = @Content(mediaType = "application/json", schema = @Schema(implementation = BaseRes.class)))
66         ])
67         @GetMapping("listPage")
68         public BaseRes<BaseRes.DataList<Scene>> listPage(@RequestParam(value = "sceneName", required = false) String sceneName,
69             @RequestParam(value = "status", required = false) Integer status,
70             @RequestParam(value = "current", required = true) Integer pageNumber,
71             @RequestParam(value = "size", required = true) Integer pageSize) {
72
73             Page<Scene> page = new Page<景><>(pageNumber, pageSize);
74             return sceneService.listPage(page, sceneName, status);
75         }
76
77         @SysLogAnnotation(module = "测试场景管理", serviceDesc = "测试场景管理-详情", serviceType = ConstantsExpand.ServiceType.QUERY)

```

设置@ApiResponse注解时返回信息

The screenshot shows the 'Project Interface Management' interface. On the left, there's a sidebar with 'Swagger Models' and a list of API endpoints. The main area shows a table for 'Request Parameters' and another for 'Response Status'. A red box highlights the 'Response Status' table, which lists various HTTP status codes with their descriptions and schema types.

状态码	说明	schema
1000	请求成功	BaseRes
1001	非法字段	BaseRes
1002	系统忙	BaseRes
1003	无操作访问权限	BaseRes
1004	请求失败	BaseRes
1005	未授权	BaseRes
2001	TOKEN失效	BaseRes
3001	请求超时	BaseRes
3002	HTTP请求出错	BaseRes

未设置@ApiResponse或@ApiResponses注解时返回信息，而且默认返回的schema并不是接口返回的BaseRes对象

This screenshot shows the 'Project Interface Management' interface for a specific endpoint: GET /http/v1/scene/getById/{id}. The 'Details' tab is selected. A red box highlights the 'Response Status' table, which shows a single entry for status code 200 with the description 'OK'.

状态码	说明	schema
200	OK	BaseResScene

4、控制器层返回数据为xml(目前可解决,解决方式不完美)[影响指数:4/5]

登陆成功后请求返回的数据是xml格式

请求头部: x-www-form-urlencoded

```

1 <@BaseRes>
2 <code>1000</code>
3 <msg>请求成功</msg>
4 <data>
5 <sysUser>
6 <id>1</id>
7 <createTime>2023-10-10 14:46:44</createTime>
8 <updateTime>2023-10-16 09:30:59</updateTime>
9 <nickName>admin</nickName>
10 <loginName>superadmin</loginName>
11 <password>e10adc3949a59abb56e095723f883e</password>
12 <roles>超级管理员</roles>
13 <token>8c744fd4ad5748956dfe7b36402</token>
14 <mobile></mobile>
15 <status>0</status>
16 <departId></departId>
17 <departLevel></departLevel>
18 <oldPw></oldPw>
19 <newPw></newPw>
20 <newPw2></newPw2>
21 <roleNames>管理员,超级管理员</roleNames>
22 <departName></departName>
23 </data>
24 </BaseRes>
25
26
27
  
```

请求返回接口示例

请求头部: x-www-form-urlencoded

参数名称	参数值	操作
sceneName	测试场景名称1	删除
status	0	删除
current	1	删除
size	15	删除

```

1 <@BaseRes>
2 <code>1000</code>
3 <msg>请求成功</msg>
4 <data>
5 <list>
6 <@>1</list>
7 <@>2</list>
8 <@>3</list>
9 <@>4</list>
10 <@>5</list>
11 <@>6</list>
12 <@>7</list>
13 <@>8</list>
14 <@>9</list>
15 <@>10</list>
16 <@>11</list>
17 <@>12</list>
18 <@>13</list>
19 <@>14</list>
20 <@>15</list>
21 <@>16</list>
22 <@>17</list>
23 <@>18</list>
24 <@>19</list>
25 <@>20</list>
26 <@>21</list>
27 <@>22</list>
  
```

代码中设置

```

1 package cn.keyidea.business.controller;
2 import ...
3 /**
4 * <p>
5 * 测试场景表 前端控制器
6 * </p>
7 *
8 * @author qyd
9 * @since 2023-09-23
10 */
11 @ApiSupport(order = 6)
12 @Tag(name = "6-测试场景接口", description = "测试场景管理")
13 @RestController
14 @RequestMapping(value = "/v1/scenes", produces = MediaType.APPLICATION_JSON_VALUE)
15 public class SceneController {
16     ...
17     @Autowired
18     private SceneService sceneService;
19
20     @SysLogAnnotation(module = "测试场景管理", serviceDesc = "测试场景管理-分页查询", serviceType = ConstantsExpand.ServiceType.QUERY)
21     @ApiOperationSupport(author = "qyd", order = 1)
22     @Operation(summary = "分页查询222", description = "<strong>注意</strong><br>(该分页查询需要输入必须的参数, 如当前页或分页数等)")
23     @Parameters({
24         @Parameter(name = "sceneName", description = "测试场景名称", required = false, example = "测试场景名称1", in = ParameterIn.QUERY),
25         @Parameter(name = "status", description = "状态(0-正常,1-禁用)", required = false, example = "0", in = ParameterIn.QUERY),
26         @Parameter(name = "current", description = "当前页", required = true, example = "1", in = ParameterIn.QUERY),
27         @Parameter(name = "size", description = "每页大小", required = true, example = "15", in = ParameterIn.QUERY)
28     })
29 }

```

@RequestMapping注解配置produces参数为json时返回正常

Issues · xiaoymin/knife4j

localhost:8114/fatp/doc.html#02-business-api/6-测试场景接口/listPage

项目接口管理

请求头: Content-Type: application/json

请求参数: sceneName, status, current, size

响应内容:

```

1+ {
2     "code": 1000,
3     "msg": "请求成功",
4     "data": [
5         {
6             "id": 1,
7             "list": [
8                 {
9                     "id": 2,
10                    "sceneName": "测试场景名称2",
11                    "img": null,
12                    "width": null,
13                    "height": null,
14                    "sendType": 1,
15                    "timeOrder": 0,
16                    "timeInterval": 0,
17                    "mainId": 0,
18                    "status": 1,
19                    "revert": 1,
20                    "deviceCount": null,
21                    "comCount": 0,
22                    "kendDeviceList": null
23                },
24                {
25                    "id": 1,
26                    "sceneName": "测试场景名称1",
27                    "img": null,
28                    "width": null,
29                    "height": null,
30                    "sendType": 1,
31                    "timeOrder": 0,
32                    "timeInterval": 0,
33                    "mainId": 0,
34                    "status": 0,
35                    "revert": 0,
36                    "deviceCount": 0,
37                    "comCount": 0,
38                    "kendDeviceList": null
39                }
40            ],
41            "total": 2
42        }
43    ],
44    "total": 2
45 }

```

说明：返回前端的数据在全局中并未使用AOP做二次处理。

5、新版Knife4j文件上传如何通过注解设置接口参数[影响指数:5/5]

在进行新版Knife4j设置文件上传时的测试时，咱们先看下以前knife4j3.0.3是如何实现的

旧版Knife4j实现文档上传

knife4j3.0.3中文件上传请求时的代码设置

```

32     @RestController
33     @RequestMapping("//sys/common/")
34     public class CommonController {
35
36         private final static Logger logger = LoggerFactory.getLogger(CommonController.class);
37
38         @Autowired
39         private SysFileService sysFileService;
40
41         // @SysLogAnnotation(module = "系统公共类", serviceDesc = "系统公共类-文件上传", serviceType = ConstantsExpand.ServiceType.UPLOAD)
42         @ApiOperation(author = "qyd", order = 1)
43         @ApiOperation(value = "文件上传", notes = "")
44         @ApiImplicitParams({
45             @ApiImplicitParam(name = "file", value = "单文件上传", required = true, paramType = "formdata", dataType = "File", dataTypeClass = File.class),
46             @ApiImplicitParam(name = "filetype", value = "文件类型", required = true, defaultValue = "txt", example = "txt", dataType = "String", dataTypeClass = String.class),
47             @ApiImplicitParam(name = "type", value = "是否使用文件原始名称: 1-使用, 其他-不使用(使用随机UUID)", required = false, defaultValue = "1", example = "1", dataType = "Integer")
48         })
49         @PostMapping("uploadFile")
50         public BaseRes uploadFile(@RequestPart("file") MultipartFile file,
51             @RequestParam("filetype") String filetype,
52             @RequestParam("type") Integer type) {
53             try {
54                 return sysFileService.uploadFile(file, filetype, type);
55             } catch (Exception e) {
56                 return BaseRes.InvalidParam(msg: "文件上传失败");
57             }
58         }
59
60         @ApiOperation(author = "qyd", order = 2)
61         @ApiOperation(value = "错误码列表(Map集合)", notes = "", produces = MediaType.APPLICATION_JSON_VALUE)
62         @GetMapping("errorCodeMap")
63         public BaseRes errorCodeMap() { return BaseRes.successData(HttpStatusCode.toMap()); }
64
65         @ApiOperation(author = "qyd", order = 3)
66         @ApiOperation(value = "WebSocket消息推送类型(Map集合)", notes = "", produces = MediaType.APPLICATION_JSON_VALUE)
67         @GetMapping("webSocketConstantsMap")
68         public BaseRes webSocketConstantsMap() { return BaseRes.successData(WebSocketConstants.toMap()); }
69
70     }
71 }

```

knife4j3.0.3中文件上传请求时文档显示

参数名称	参数说明	请求类型	是否必须	数据类型	schema
fileType	文件类型示例值(txt)	query	true	string	
file	是否使用文件原始名称: 1-使用, 其他-不使用(使用随机UUID)示例值(1)	formData	false	file	
type	是否使用文件原始名称: 1-使用, 其他-不使用(使用随机UUID)示例值(1)	query	false	integer(int32)	

knife4j3.0.3中文件上传调试时的界面-可正常上传文件

新版Knife4j目前的窘境（未实现）

knife4j4.5.0中文件上传请求时的代码设置

```

    /**
     * @date 2022-10-17
     */
    @ApiSupport(order = 1)
    @Tag(name = "1-系统公共类", description = "")
    @RestController
    @RequestMapping("/sys/common/")
    public class CommonController
    {
        private final static Logger logger = LoggerFactory.getLogger(CommonController.class);

        @Autowired
        private SysFileLogService sysFileService;

        @SysLogAnnotation(module = "公共类", serviceDesc = "公共类-文件上传", serviceType = ConstantsExpand.ServiceType.UPLOAD)
        @ApiOperation(summary = "文件上传", description = "")
        @Parameters({
            @Parameter(name = "file", description = "单文件上传", required = true, in = ParameterIn.DEFAULT),
            @Parameter(name = "fileType", description = "文件类型", required = true, example = "txt", in = ParameterIn.DEFAULT),
            @Parameter(name = "type", description = "是否使用文件原名命名, 1-使用, 其他-不使用(随机UUID)", required = false, example = "1", in = ParameterIn.DEFAULT)
        })
        @PostMapping("uploadFile")
        public BaseRes uploadFile(@RequestParam(value = "file", required = true) MultipartFile file,
                                  @RequestParam(value = "fileType", required = true) String fileType,
                                  @RequestParam(value = "type", required = false) Integer type)
        {
            try
            {
                return sysFileService.uploadFile(file, fileType, type);
            }
            catch (Exception e)
            {
                return BaseRes.invalidParam(msg: "文件上传失败");
            }
        }

        @ApiOperationSupport(author = "gyd", order = 2)
        @Operation(summary = "错误码列表(Map集合)", description = "")
        @GetMapping("errorCodeList")
        public BaseRes errorCodeList() | return BaseRes.successData(StatusCodes.toList());
    }

```

knife4j4.5.0中文件上传请求时文档显示

Knife4j 4.5.0 中文件上传调试时的界面-可正常上传文件

附录A：通用响应封装类代码(BaseRes.java)

```

package cn.keyidea.common.bean;

import cn.keyidea.common.constant.StatusCode;
import com.fasterxml.jackson.annotation.JsonIgnore;
import io.swagger.v3.oas.annotations.media.Schema;
import lombok.Data;
import org.apache.poi.ss.formula.functions.T;

import java.io.Serializable;

/**
 * 通用响应封装，范式返回(Swagger要求)

```

```
*  
 * @author qyd  
 */  
@Data  
public class BaseRes<T> implements Serializable {  
  
    /**  
     * 错误码  
     */  
    @Schema(name = "code", description = "错误码,当code为1000时返回正常,其余返回异常",  
required = true, example = "1000")  
    public Integer code;  
  
    /**  
     * 错误提示信息  
     */  
    @Schema(name = "msg", description = "错误提示信息,当code为非1000时返回提示信息")  
    public String msg;  
  
    /**  
     * 附加返回数据  
     */  
    @Schema(name = "data", description = "附加返回数据,当code为1000时返回数据")  
    public T data;  
  
    public static class DataList<T> {  
        /**  
         * 记录总数  
         */  
        @Schema(name = "total", description = "记录总数")  
        public Integer total;  
        /**  
         * 数据列表  
         */  
        @Schema(name = "list", description = "数据列表")  
        public T list;  
  
        public DataList(Integer total, T list) {  
            this.total = total;  
            this.list = list;  
        }  
    }  
  
    /**  
     * 给ObjectMapper用的,代码中不要调用  
     */  
    public BaseRes() {  
    }  
  
    /**  
     * 自定义返回码和提示消息  
     *  
     * @param code 错误码  
     * @param msg 提示文字  
     */  
    public BaseRes(int code, String msg) {  
        this.code = code;
```

```
        this.msg = msg;
    }

    public BaseRes(int code, String msg, T data) {
        this.code = code;
        this.msg = msg;
        this.data = data;
    }

    /**
     * 返回成功，但是没有附加数据
     *
     * @return BaseRes对象
     */
    public static BaseRes success() {
        return new BaseRes(StatusCode.SUCCESS.getCodeValue(), "请求成功");
    }

    /**
     * 返回成功，带附加数据
     *
     * @param data 附加数据
     * @return BaseRes对象
     */
    public static BaseRes successData(Object data) {
        BaseRes value = new BaseRes(StatusCode.SUCCESS.getCodeValue(), "请求成功");
        value.data = data;
        return value;
    }

    /**
     * 返回参数无效响应
     *
     * @return BaseRes对象
     */
    public static BaseRes invalidParam() {
        return new BaseRes(StatusCode.INVALID_PARAM.getCodeValue(), "参数无效");
    }

    /**
     * 返回参数无效响应，自定义错误提示
     *
     * @param msg 提示文字
     * @return BaseRes对象
     */
    public static BaseRes invalidParam(String msg) {
        return new BaseRes(StatusCode.INVALID_PARAM.getCodeValue(), msg);
    }

    /**
     * 返回系统忙无效响应
     *
     * @return BaseRes对象
     */
    public static BaseRes systemBusy() {
        return new BaseRes(StatusCode.SYSTEM_BUSY.getCodeValue(), "系统忙");
    }
```

```
/**
 * 返回master key无效响应
 *
 * @return BaseRes对象
 */
public static BaseRes invalidMasterkey() {
    return new BaseRes(StatusCode.INVALID_MASTER_KEY.getCodeValue(), "没有接口访问权限");
}

/**
 * 返回失败，附带说明
 *
 * @return BaseRes对象
 */
public static BaseRes fail(String msg) {
    return new BaseRes(StatusCode.FAILURE.getCodeValue(), msg);
}

/**
 * 返回错误信息时，仍然返回数据
 *
 * @param data 数据集
 * @param msg 错误信息
 * @return BaseRes对象
 */
public static BaseRes failData(Object data, String msg) {
    return new BaseRes(StatusCode.FAILURE.getCodeValue(), msg, data);
}

/**
 * 登录失效的错误
 *
 * @return BaseRes对象
 */
public static BaseRes invalidToken() {
    return new BaseRes(StatusCode.INVALID_TOKEN.getCodeValue(), "请先登录");
}

/**
 * 检查响应处理是否成功
 *
 * @return 成功返回true，否则false
 */
@JsonIgnore
public boolean isSuccess() {

    return (this.code.equals(StatusCode.SUCCESS.getCodeValue()));
}

/**
 * 返回分页列表数据
 *
 * @param total 记录总数
 * @param list 列表数据
 * @return rsp
 */
public static BaseRes list(long total, Object list) {
```

```

        DataList data = new DataList((int) total, list);

        return BaseRes.successData(data);
    }
}

```

附录B：完整 pom.xml 文件

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.keyidea</groupId>
    <artifactId>fatp</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>FullAutoMaticTestPlatform</name>
    <description>测试平台</description>

    <!--
        spring-boot-starter-parent
        https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-
        starter-parent
        -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.4</version><!-- 2.7.18 ↑ -->
        <relativePath/>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>21</java.version>
        <!-- 更新log4j2版本
包:https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api -->
        <log4j2.version>2.23.1</log4j2.version>
        <!-- 打包时跳过测试环境 -->
        <skipTests>true</skipTests>

        <!--
            mybatis-plus:https://mvnrepository.com/artifact/com.baomidou/mybatis-plus-
            boot-starter
            -->
        <mybatis-plus.version>3.5.5</mybatis-plus.version>
        <mybatis-spring.version>3.0.3</mybatis-spring.version>
        <!--
            MyBatis生成器
            mybatis-plus-
generator:https://mvnrepository.com/artifact/com.baomidou/mybatis-plus-generator/

```

```
velocity-engine-
core:https://mvnrepository.com/artifact/org.apache.velocity/velocity-engine-core
    freemarker:https://mvnrepository.com/artifact/org.freemarker/freemarker
    说明：任选velocity-engine-core或者freemarker一种
-->
<mybatis-plus-generator.version>3.5.5</mybatis-plus-generator.version>
<velocity-engine-core.version>2.3</velocity-engine-core.version>
<freemarker.version>2.3.32</freemarker.version>
<!-- mysql:https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<mysql.version>8.0.33</mysql.version>
<!-- lombok:https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<lombok.version>1.18.30</lombok.version>
<!-- junit:https://mvnrepository.com/artifact/junit/junit/ -->
<junit.version>4.13.2</junit.version>
<!--
    netty-all:https://mvnrepository.com/artifact/io.netty/netty-all
    mina-core:https://mvnrepository.com/artifact/org.apache.mina/mina-core
    mina-core: 用于组帧
-->
<netty-all.version>5.0.0.Alpha1</netty-all.version>
<!--
    <mina-core.version>2.1.6</mina-core.version>-->
<mina-core.version>2.2.3</mina-core.version>
<!-- hutool-all:https://mvnrepository.com/artifact/cn.hutool/hutool-all -->
<hutool-all.version>5.8.26</hutool-all.version>
<!--
jackson:https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
    <jackson.version>2.16.1</jackson.version>
    <!--
fastjson2:https://mvnrepository.com/artifact/com.alibaba.fastjson2/fastjson2 -->
    <fastjson2.version>2.0.47</fastjson2.version>
    <!-- spring-
websocket:https://mvnrepository.com/artifact/org.springframework/spring-websocket -->
    <spring-websocket.version>6.1.4</spring-websocket.version>
    <!--
        SpringBoot3.x集成knife4j
        knife4j-openapi3-jakarta-spring-boot-
starter:https://mvnrepository.com/artifact/com.github.xiaoymin/knife4j-openapi3-
jakarta-spring-boot-starter
        knife4j-aggregation-spring-boot-
starter:https://mvnrepository.com/artifact/com.github.xiaoymin/knife4j-aggregation-
spring-boot-starter
    -->
    <knife4j.version>4.5.0</knife4j.version>
    <!-- druid:https://mvnrepository.com/artifact/com.alibaba/druid/ -->
    <druid.version>1.2.21</druid.version>
    <!--
        screw螺丝钉:https://mvnrepository.com/artifact/cn.smallbun.screw/screw-core
        hikaricp:https://mvnrepository.com/artifact/com.zaxxer/HikariCP
    -->
    <screw.version>1.0.5</screw.version>
    <hikaricp.version>5.1.0</hikaricp.version>
    <!-- commons-
lang3:https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
    <commons-lang3.version>3.14.0</commons-lang3.version>
    <!--
        SpringBoot集成
shiro:https://mvnrepository.com/artifact/org.apache.shiro/shiro-spring
```

```

        shiro-ehcache:https://mvnrepository.com/artifact/org.apache.shiro/shiro-
ehcache
        -->
        <shiro-spring.version>1.13.0</shiro-spring.version>
        <!-- SpringBoot集成验证码模块
kaptcha:https://mvnrepository.com/artifact/com.github.axet/kaptcha -->
        <kaptcha.version>0.0.9</kaptcha.version>
        <!-- SpringBoot集成定时器Quartz:https://mvnrepository.com/artifact/org.quartz-
scheduler/quartz -->
        <quartz.version>2.3.2</quartz.version>
        <!-- commons-io:https://mvnrepository.com/artifact/commons-io/commons-io -->
        <commons-io.version>2.15.1</commons-io.version>
        <!-- gson:https://mvnrepository.com/artifact/com.google.code.gson/gson -->
        <gson.version>2.10.1</gson.version>
        <!--
            SpringBoot集成easypoi
            easypoi:https://mvnrepository.com/artifact/cn.afterturn/easypoi-base
            poi:https://mvnrepository.com/artifact/org.apache.poi/poi
        -->
<!--
        <easypoi.version>4.4.0</easypoi.version>-->
        <easypoi.version>4.5.0</easypoi.version>
        <poi.version>5.2.5</poi.version>

        <!--
            集成spring-cloud-starter-openfeign
            spring-cloud-
starter:https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-
starter

            openfeign:https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-
starter-openfeign
            spring-cloud-starter-netflix-
hystrix:https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-
starter-netflix-hystrix

```

说明: SpringBoot2.4.x或2.5.x使用3.0.4版本的feign, SpringBoot2.6.x或2.7.x使用3.1.3版本的feign

```

        -->
<!--
        <spring-cloud.version>3.1.4</spring-cloud.version>-->
        <spring-cloud.version>4.1.0</spring-cloud.version>
        <spring-cloud-openfeign.version>4.1.0</spring-cloud-openfeign.version>
        <hystrix.version>2.2.10.RELEASE</hystrix.version>

```

```

    </properties>

<dependencies>
    <!--
        Web启动器
        注意:
        1)由于 spring-boot-starter-web 默认替我们引入了核心启动器 spring-boot-starter,
因此, 当 Spring Boot 项目中的 pom.xml 引入了 spring-boot-starter-web 的依赖后, 就无须在引入
spring-boot-starter 核心启动器的依赖了。
        2)spring-boot-starter-web 默认使用嵌入式的tomcat作为web容器对外提供HTTP服务。
    -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>

```

```
</dependency>
<!-- 数据校验 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<!-- jpa -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<!--lombok-->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>${lombok.version}</version>
    <scope>provided</scope>
</dependency>

<!-- mybatis-plus依赖 -->
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>${mybatis-plus.version}</version>
    <exclusions>
        <exclusion>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<!--
主要是由于 mybatis-plus 中 mybatis 的整合包版本不够导致的：排除 mybatis-plus 中自带的 mybatis 整合包，单独引入即可
mybatis-spring:https://mvnrepository.com/artifact/org.mybatis/mybatis-
spring
-->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>${mybatis-spring.version}</version>
</dependency>

<!-- 代码生成 -->
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-generator</artifactId>
    <version>${mybatis-plus-generator.version}</version>
</dependency>
<!-- 代码生成：velocity模板-->
<dependency>
    <groupId>org.apache.velocity</groupId>
```

```
<artifactId>velocity-engine-core</artifactId>
<version>${velocity-engine-core.version}</version>
</dependency>
<!-- 代码生成: freemarker模板-->
<dependency>
    <groupId>org.freemarker</groupId>
    <artifactId>freemarker</artifactId>
    <version>${freemarker.version}</version>
</dependency>

<!-- mysql依赖 -->
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>${mysql.version}</version>
</dependency>

<!-- druid 依赖 -->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid-spring-boot-starter</artifactId>
    <version>${druid.version}</version>
</dependency>

<!-- websocket 依赖 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-websocket</artifactId>
    <version>${spring-websocket.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-messaging</artifactId>
    <version>${spring-websocket.version}</version>
</dependency>

<!-- shiro 依赖 -->
<!--<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-spring</artifactId>
    <version>${shiro-spring.version}</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-ehcache</artifactId>
    <version>${shiro-spring.version}</version>
</dependency>-->
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-spring</artifactId>
    <classifier>jakarta</classifier>
    <version>${shiro-spring.version}</version>
    <!-- 排除仍使用了javax.servlet的依赖 -->
    <exclusions>
        <exclusion>
            <groupId>org.apache.shiro</groupId>
            <artifactId>shiro-core</artifactId>
```

```
        </exclusion>
    <exclusion>
        <groupId>org.apache.shiro</groupId>
        <artifactId>shiro-web</artifactId>
    </exclusion>
</exclusions>
</dependency>
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-ehcache</artifactId>
    <version>${shiro-spring.version}</version>
</dependency>
<!-- 引入适配jakarta的依赖包 -->
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-core</artifactId>
    <classifier>jakarta</classifier>
    <version>${shiro-spring.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-web</artifactId>
    <classifier>jakarta</classifier>
    <version>${shiro-spring.version}</version>
    <exclusions>
        <exclusion>
            <groupId>org.apache.shiro</groupId>
            <artifactId>shiro-core</artifactId>
        </exclusion>
    </exclusions>
</dependency>

<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>${commons-lang3.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>${poi.version}</version>
</dependency>
<dependency>
    <groupId>cn.afterturn</groupId>
    <artifactId>easypoi-base</artifactId>
    <version>${easypoi.version}</version>
</dependency>
<dependency>
    <groupId>com.github.axet</groupId>
    <artifactId>kaptcha</artifactId>
    <version>${kaptcha.version}</version>
</dependency>

<!-- commons-io -->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
```

```
<version>${commons-io.version}</version>
<scope>compile</scope>
</dependency>

<!-- gson -->
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>${gson.version}</version>
</dependency>

<!-- junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>${junit.version}</version>
    <scope>test</scope>
</dependency>

<!-- netty -->
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-all</artifactId>
    <version>${netty-all.version}</version>
</dependency>
<!-- mina-core -->
<dependency>
    <groupId>org.apache.mina</groupId>
    <artifactId>mina-core</artifactId>
    <version>${mina-core.version}</version>
</dependency>

<!-- hutool工具类 -->
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
    <version>${hutool-all.version}</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui -->
<!--<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.5.0</version>
</dependency>-->
<dependency>
    <groupId>com.github.xiaoymin</groupId>
    <artifactId>knife4j-openapi3-jakarta-spring-boot-starter</artifactId>
    <version>${knife4j.version}</version>
</dependency>

<!--jackson-->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>${jackson.version}</version>
</dependency>
```

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>${jackson.version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>${jackson.version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
    <version>${jackson.version}</version>
</dependency>

<!-- fastjson2 -->
<dependency>
    <groupId>com.alibaba.fastjson2</groupId>
    <artifactId>fastjson2</artifactId>
    <version>${fastjson2.version}</version>
</dependency>

<!-- quartz -->
<dependency>
    <groupId>org.quartz-scheduler</groupId>
    <artifactId>quartz</artifactId>
    <version>${quartz.version}</version>
</dependency>
<dependency>
    <groupId>org.quartz-scheduler</groupId>
    <artifactId>quartz-jobs</artifactId>
    <version>${quartz.version}</version>
</dependency>

<!--HttpClient-->
<dependency>
    <groupId>commons-httpclient</groupId>
    <artifactId>commons-httpclient</artifactId>
    <version>3.1</version>
</dependency>

<!-- 可以根据两行根数据计算卫星坐标 -->
<dependency>
    <groupId>uk.me.g4dpz</groupId>
    <artifactId>predict4java</artifactId>
    <version>1.1.3</version>
</dependency>

<!-- screw核心 -->
<dependency>
    <groupId>cn.smallbun.screw</groupId>
    <artifactId>screw-core</artifactId>
    <version>${screw.version}</version>
</dependency>
<!-- HikariCP -->
<dependency>
    <groupId>com.zaxxer</groupId>
```

```

        <artifactId>HikariCP</artifactId>
        <version>${hikaricp.version}</version>
    </dependency>

    <!-- spring-cloud-starter -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter</artifactId>
        <version>${spring-cloud.version}</version>
    </dependency>
    <!-- openfeign -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
        <version>${spring-cloud-openfeign.version}</version>
    </dependency>
    <!-- spring-cloud-starter-netflix-hystrix -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
        <version>${hystrix.version}</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate.common</groupId>
        <artifactId>hibernate-commons-annotations</artifactId>
        <version>6.0.6.Final</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

```

附录C: ShiroConfig配置([ShiroConfig.java](#))

```

package cn.keyidea.common.config;

import cn.keyidea.sys.oauth2.Oauth2Filter;
import cn.keyidea.sys.oauth2.Oauth2Realm;
import jakarta.servlet.Filter;
import net.sf.ehcache.CacheManager;
import org.apache.shiro.cache.ehcache.EhCacheManager;
import org.apache.shiro.mgt.SecurityManager;
import org.apache.shiro.session.mgt.SessionManager;
import org.apache.shiro.session.mgt.eis.MemorySessionDAO;
import org.apache.shiro.spring.LifecycleBeanPostProcessor;

```

```
import org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor;
import org.apache.shiro.spring.web.ShiroFilterFactoryBean;
import org.apache.shiro.web.mgt.DefaultWebSecurityManager;
import org.apache.shiro.web.session.mgt.DefaultWebSessionManager;
import org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator;
import org.springframework.beans.factory.config.BeanDefinition;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Role;

import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;

/**
 * Shiro配置
 */
@Configuration
public class ShiroConfig
{

    /**
     * 加盐参数
     */
    public final static String hashAlgorithmName = "MD5";

    /**
     * 循环次数
     */
    public final static int hashIterations = 1024;

    @Role(BeanDefinition.ROLE_INFRASTRUCTURE)
    @Bean("cacheManager2")
    CacheManager cacheManager()
    {
        return CacheManager.create();
    }

    @Role(BeanDefinition.ROLE_INFRASTRUCTURE)
    @Bean
    public EhCacheManager ehCacheManager()
    {
        EhCacheManager em = new EhCacheManager();
        em.setCacheManager(cacheManager());
        return em;
    }

    @Role(BeanDefinition.ROLE_INFRASTRUCTURE)
    @Bean("sessionManager")
    public SessionManager sessionManager()
    {
        DefaultWebSessionManager sessionManager = new DefaultWebSessionManager();
        sessionManager.setSessionValidationSchedulerEnabled(true);
        sessionManager.setSessionIdCookieEnabled(true);
        sessionManager.setSessionDAO(new MemorySessionDAO());
        return sessionManager;
    }
}
```

```
@Role(BeanDefinition.ROLE_INFRASTRUCTURE)
@Bean("securityManager")
public SecurityManager securityManager(Oauth2Realm oAuth2Realm, SessionManager
sessionManager)
{
    DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();
    securityManager.setRealm(oAuth2Realm);
    securityManager.setCacheManager(ehCacheManager());
    securityManager.setSessionManager(sessionManager);

    return securityManager;
}

@Role(BeanDefinition.ROLE_INFRASTRUCTURE)
@Bean("shiroFilter")
public ShiroFilterFactoryBean shiroFilter(SecurityManager securityManager)
{
    ShiroFilterFactoryBean shiroFilter = new ShiroFilterFactoryBean();
    shiroFilter.setSecurityManager(securityManager);

    //oauth过滤
    Map<String, Filter> filters = new HashMap<>(16);
    filters.put("oauth2", new Oauth2Filter());
    shiroFilter.setFilters(filters);

    /* 拦截器 */
    // 配置不会被拦截的链接 顺序判断
    Map<String, String> filterMap = new LinkedHashMap<>();
    filterMap.put("/sys/sysLogin/login", "anon");
    filterMap.put("/sys/sysLogin/captcha.jpg", "anon");
    filterMap.put("/websocket/**", "anon");
    // // Shiro放行swagger2(Knife4j)
    // filterMap.put("/doc.html", "anon");
    // filterMap.put("/swagger-resources/**", "anon");
    // filterMap.put("/v2/**", "anon");
    // filterMap.put("/webjars/**", "anon");

    // Shiro放行swagger3(Knife4j)
    filterMap.put("/doc.html", "anon");
    filterMap.put("/swagger-resources/**", "anon");
    filterMap.put("/v3/**", "anon");
    filterMap.put("/webjars/**", "anon");
    filterMap.put("/swagger-ui/**", "anon");

    // 后台接口列表(示例)
    // filterMap.put("/v1/**", "anon");
    filterMap.put("/v1/health", "anon");
    filterMap.put("/v1/backgroundPicture/**", "anon"); // 测试

    filterMap.put("/v1/networkInfo/**", "anon");
    filterMap.put("/v1/networkConf/**", "anon");

    // 模板资源下载目录配置
    filterMap.put("/template/**", "anon");
    filterMap.put("/markdown/**", "anon");

    filterMap.put("/**", "oauth2");
}
```

```

        shiroFilter.setFilterChainDefinitionMap(filterMap);

        return shiroFilter;
    }

    @Bean("lifecycleBeanPostProcessor")
    public LifecycleBeanPostProcessor lifecycleBeanPostProcessor()
    {
        return new LifecycleBeanPostProcessor();
    }

    @Bean
    public DefaultAdvisorAutoProxyCreator defaultAdvisorAutoProxyCreator()
    {
        DefaultAdvisorAutoProxyCreator proxyCreator = new
DefaultAdvisorAutoProxyCreator();
        proxyCreator.setProxyTargetClass(true);
        return proxyCreator;
    }

    /**
     * 开启shiro aop注解支持.
     * 使用代理方式;所以需要开启代码支持;
     *
     * @param securityManager
     * @return
     */
    @Role(BeanDefinition.ROLE_INFRASTRUCTURE)
    @Bean
    public AuthorizationAttributeSourceAdvisor
authorizationAttributeSourceAdvisor(SecurityManager securityManager)
    {
        AuthorizationAttributeSourceAdvisor advisor = new
AuthorizationAttributeSourceAdvisor();
        advisor.setSecurityManager(securityManager);
        return advisor;
    }

}

```

附录D：完整 application.yml 配置

```

# Tomcat
server:
  tomcat:
    uri-encoding: UTF-8
    max-threads: 1024
    min-spare-threads: 30
    accept-count: 5000
    connection-timeout: 1000ms
  port: 8114
  servlet:
    context-path: /fatp

# MyBatis Plus

```

```
mybatis-plus:
  check-config-location: true
  configuration:
    #是否开启自动驼峰命名规则（camel case）映射
    map-underscore-to-camel-case: true
    #全局地开启或关闭配置文件中的所有映射器已经配置的任何缓存
    cache-enabled: false
    call-setters-on-nulls: true
    #配置JdbcTypeForNull, oracle数据库必须配置
    jdbc-type-for-null: 'null'
    #MyBatis 自动映射时未知列或未知属性处理策略 NONE: 不做任何处理（默认值）, WARNING: 以日志的形式
    打印相关警告信息, FAILING: 当作映射失败处理, 并抛出异常和详细信息
    auto-mapping-unknown-column-behavior: warning
    # 打印SQL
    log-impl: org.apache.ibatis.logging.stdout.StdoutImpl
  global-config:
    banner: false
    db-config:
      #主键类型 0:"数据库ID自增", 1:"未设置主键类型",2:"用户输入ID（该类型可以通过自己注册自动填充插件进行填充）", 3:"全局唯一ID (idWorker), 4:全局唯一ID (UUID), 5:字符串全局唯一ID (idWorker的字符串表示)";
      id-type: auto
      #字段验证策略 IGNORED:"忽略判断", NOT_NULL:"非NULL判断", NOT_EMPTY:"非空判断", DEFAULT
      默认的,一般只用于注解里(1. 在全局里代表 NOT_NULL,2. 在注解里代表 跟随全局)
      field-strategy: NOT_NULL
      #驼峰下划线转换
      column-underline: true
      #数据库大写下划线转换
      capital-mode: true
      #逻辑删除值
      logic-delete-value: 1
      #逻辑未删除值
      logic-not-delete-value: 0
      db-type: mysql

  spring:
    # 环境 dev|local
    profiles:
      active: local
    datasource:
      druid:
        initialSize: 2
        minIdle: 2
        maxActive: 8 #业务忙可以加大
        validationQuery: select 1 from dual
        filters: stat ,wall,slf4j
        connectProperties: druid.stat.slowSqlMillis=50
        connection-init-sqls: set names utf8mb4
        # 合并多个DruidDataSource的监控数据
        useGlobalDataSourceStat: true
        stat-view-servlet:
          enabled: true #是否启用StatViewServlet默认值true
          url-pattern: /druid2/*
          reset-enable: false
          login-username: admin
          login-password: KEYM0ZcjPYk9aUCIeH21CWyYhcTK6te0
          allow: #IP
          deny: #IP
```

```
    web-stat-filter:
        exclusions: "*.js,*.gif,*.jpg,*.png,*.css,*.ico,/druid2/*"
    main:
        allow-bean-definition-overriding: true
    # jackson时间格式化
    jackson:
        time-zone: GMT+8
        date-format: yyyy-MM-dd HH:mm:ss
    servlet:
        multipart:
            max-file-size: 100MB
            max-request-size: 100MB
            enabled: true
    #配置自动建表: update:没有表新建, 有表更新操作, 控制台显示建表语句
    jpa:
        hibernate:
            ddl-auto: update
            show-sql: true
            # spring.jpa.open-in-view 属性被默认启用, 需要手动配置该属性, 去掉这个警告
            # open-in-view 是指延时加载的一些属性数据, 可以在页面展现的时候, 保持session不关闭, 从而保证能在页面进行延时加载
            open-in-view: false
        # properties:
        #     hibernate:
        #         dialect: org.hibernate.dialect.MySQL57Dialect
    # 设置静态下载资源目录
    mvc:
        # 会覆盖默认配置, 需要在static-locations添加默认的内部目录和自定义目录
        static-path-pattern: /**
        # 修复SpringBoot2.6+与Swagger3.0+兼容问题: https://zhuanlan.zhihu.com/p/450064507
        pathmatch:
            matching-strategy: ant_path_matcher
        # resources:
        #     static-locations: classpath:/static/*
        throw-exception-if-no-handler-found: true

    # springdoc-openapi项目配置
    springdoc:
        # get请求多参数时不需要添加额外的@ParameterObject和@Parameter注解
        default-flat-param-object: true
        # 启用swaggerUI
        swagger-ui:
            #自定义swagger前端请求路径, 输入http: 127.0.0.1:8080/swagger-ui.html会自动重定向到swagger页面
            path: /swagger-ui.html
            enabled: true
        #     tags-sorter: alpha # 标签的排序方式 alpha:按照字母顺序排序 (@ApiSupport注解排序不生效, 因此需要设置)
            #operations-sorter: alpha # 接口的排序方式 alpha:按照字母顺序排序 (@ApiOperationSupport注解排序生效, 因此这里不作设置)
        # 启用文档, 默认开启
        api-docs:
            path: /v3/api-docs      #swagger后端请求地址
            enabled: true
    #knife4j相关配置 可以不用改
    knife4j:
        enable: true      #开启knife4j, 无需添加@EnableKnife4j注解
        setting:
```

```
language: ZH_CN    # 中文:ZH_CN 英文:EN
#   enable-swagger-models: true
#   enable-dynamic-parameter: false
  footer-custom-content: "<strong>Copyright © 2024 Keyidea. All Rights
Reversed</strong>"
  enable-footer-custom: true
  enable-footer: true
  enable-document-manage: true
#logging:
#  level:
#    # 日志以什么样的级别监控该接口 error | debug | info
#    cn.keyidea: debug

# ----- 配置netty 开始 -----
# 监听船端TCP请求
netty:
  tcp:
    # 需要开启时置为 true
    enable: false
    # 本程序接收TCP数据包IP: 本地环境: 172.16.1.230
    host: 0.0.0.0
    # 监听端口号
    port: 8090
# ----- 配置netty 结束 -----

# 全环境参数设置
keyidea:
  upload:
    #   path: /home/www/fileupload # 230环境
    path: D:\ # 本地测试环境
    sysName: fatp
  default:
    password: 123456
  clear-dirty-data: false # 启动时是否清除脏数据(包括手动任务和自动任务): false-不清理, true-清理, 默认不清理

---
spring:
  config:
    activate:
      on-profile: dev
  datasource:
    driverClassName: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://172.16.1.230:3306/fatp?
    zeroDateTimeBehavior=convertToNull&autoReconnect=true&tinyInt1isBit=false&useSSL=false
    &allowMultiQueries=true&useUnicode=true&characterEncoding=utf8
    username: root
    password: 123456
---
spring:
  config:
    activate:
      on-profile: local
  datasource:
    driverClassName: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/fatp?
    zeroDateTimeBehavior=convertToNull&autoReconnect=true&tinyInt1isBit=false&useSSL=false
    &allowMultiQueries=true&useUnicode=true&characterEncoding=utf8
```

username: root
password: 123456

END
