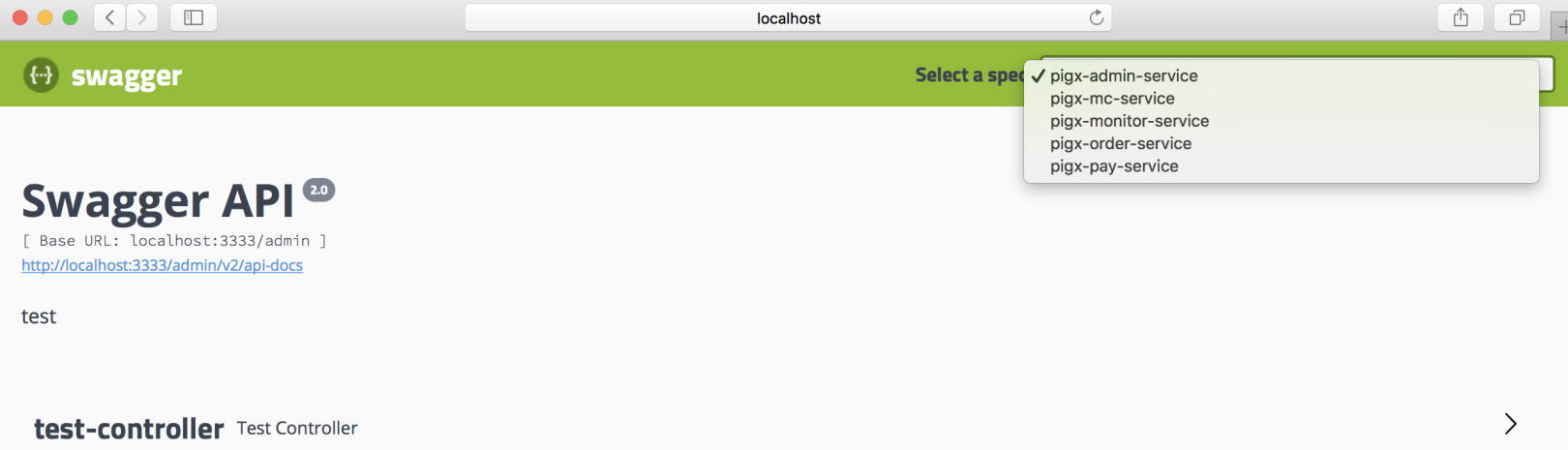
在微服务架构下，通常**每个微服务**都会使用Swagger来管理我们的接口文档，当微服务越来越多，接口查找管理无形中要浪费我们不少时间，毕竟懒是程序员的美德。

​ 由于swagger2暂时不支持webflux 走了很多坑，完成这个效果感谢 @dreamlu @世言。

[**#**](https://pig4cloud.com/#%E6%96%87%E6%A1%A3%E8%81%9A%E5%90%88%E6%95%88%E6%9E%9C) **文档聚合效果**

通过访问网关的 host:port/swagger-ui.html，即可实现: [pig聚合文档效果预览传送门](http://preview.pig4cloud.com/#/myiframe/urlPath?src=http%3A%2F%2F139.224.200.249%3A9999%2Fswagger-ui.html&name=%E6%8E%A5%E5%8F%A3%E6%96%87%E6%A1%A3)

通过右上角的Select a spec 选择服务模块来查看swagger文档 

[**#**](https://pig4cloud.com/#pig%E7%9A%84zuul-%E6%A0%B8%E5%BF%83%E5%AE%9E%E7%8E%B0) **Pig的Zuul 核心实现**

获取到zuul配置的路由信息，主要到**SwaggerResource**

/\*\*

\* 参考jhipster

\* GatewaySwaggerResourcesProvider

\*/

@Component

@Primary

public class RegistrySwaggerResourcesProvider implements SwaggerResourcesProvider {

private final RouteLocator routeLocator;

public RegistrySwaggerResourcesProvider(RouteLocator routeLocator) {

this.routeLocator = routeLocator;

}

@Override

public List<SwaggerResource> get() {

List<SwaggerResource> resources = new ArrayList<>();

List<Route> routes = routeLocator.getRoutes();

routes.forEach(route -> {

//授权不维护到swagger

if (!StringUtils.contains(route.getId(), ServiceNameConstant.AUTH\_SERVICE)){

resources.add(swaggerResource(route.getId(), route.getFullPath().replace("\*\*", "v2/api-docs")));

}

});

return resources;

}

private SwaggerResource swaggerResource(String name, String location) {

SwaggerResource swaggerResource = new SwaggerResource();

swaggerResource.setName(name);

swaggerResource.setLocation(location);

swaggerResource.setSwaggerVersion("2.0");

return swaggerResource;

}

}

[**#**](https://pig4cloud.com/#pigx%E7%9A%84spring-cloud-gateway-%E5%AE%9E%E7%8E%B0) **PigX的Spring Cloud Gateway 实现**

[**#**](https://pig4cloud.com/#%E6%B3%A8%E5%85%A5%E8%B7%AF%E7%94%B1%E5%88%B0swaggerresource) **注入路由到SwaggerResource**

@Component

@Primary

@AllArgsConstructor

public class SwaggerProvider implements SwaggerResourcesProvider {

public static final String API\_URI = "/v2/api-docs";

private final RouteLocator routeLocator;

private final GatewayProperties gatewayProperties;

@Override

public List<SwaggerResource> get() {

List<SwaggerResource> resources = new ArrayList<>();

List<String> routes = new ArrayList<>();

routeLocator.getRoutes().subscribe(route -> routes.add(route.getId()));

gatewayProperties.getRoutes().stream().filter(routeDefinition -> routes.contains(routeDefinition.getId()))

.forEach(routeDefinition -> routeDefinition.getPredicates().stream()

.filter(predicateDefinition -> "Path".equalsIgnoreCase(predicateDefinition.getName()))

.filter(predicateDefinition -> !"pigx-auth".equalsIgnoreCase(routeDefinition.getId()))

.forEach(predicateDefinition -> resources.add(swaggerResource(routeDefinition.getId(),

predicateDefinition.getArgs().get(NameUtils.GENERATED\_NAME\_PREFIX + "0")

.replace("/\*\*", API\_URI)))));

return resources;

}

private SwaggerResource swaggerResource(String name, String location) {

SwaggerResource swaggerResource = new SwaggerResource();

swaggerResource.setName(name);

swaggerResource.setLocation(location);

swaggerResource.setSwaggerVersion("2.0");

return swaggerResource;

}

}

[**#**](https://pig4cloud.com/#%E6%8F%90%E4%BE%9Bswagger-%E5%AF%B9%E5%A4%96%E6%8E%A5%E5%8F%A3%E9%85%8D%E7%BD%AE) **提供swagger 对外接口配置**

@Slf4j

@Configuration

@AllArgsConstructor

public class RouterFunctionConfiguration {

private final SwaggerResourceHandler swaggerResourceHandler;

private final SwaggerSecurityHandler swaggerSecurityHandler;

private final SwaggerUiHandler swaggerUiHandler;

@Bean

public RouterFunction routerFunction() {

return RouterFunctions.route(

.andRoute(RequestPredicates.GET("/swagger-resources")

.and(RequestPredicates.accept(MediaType.ALL)), swaggerResourceHandler)

.andRoute(RequestPredicates.GET("/swagger-resources/configuration/ui")

.and(RequestPredicates.accept(MediaType.ALL)), swaggerUiHandler)

.andRoute(RequestPredicates.GET("/swagger-resources/configuration/security")

.and(RequestPredicates.accept(MediaType.ALL)), swaggerSecurityHandler);

}

}

[**#**](https://pig4cloud.com/#%E4%B8%9A%E5%8A%A1handler-%E7%9A%84%E5%AE%9E%E7%8E%B0) **业务handler 的实现**

@Override

public Mono<ServerResponse> handle(ServerRequest request) {

return ServerResponse.status(HttpStatus.OK)

.contentType(MediaType.APPLICATION\_JSON\_UTF8)

.body(BodyInserters.fromObject(swaggerResources.get()));

}

@Override

public Mono<ServerResponse> handle(ServerRequest request) {

return ServerResponse.status(HttpStatus.OK)

.contentType(MediaType.APPLICATION\_JSON\_UTF8)

.body(BodyInserters.fromObject(

Optional.ofNullable(securityConfiguration)

.orElse(SecurityConfigurationBuilder.builder().build())));

}

@Override

public Mono<ServerResponse> handle(ServerRequest request) {

return ServerResponse.status(HttpStatus.OK)

.contentType(MediaType.APPLICATION\_JSON\_UTF8)

.body(BodyInserters.fromObject(

Optional.ofNullable(uiConfiguration)

.orElse(UiConfigurationBuilder.builder().build())));

}

[**#**](https://pig4cloud.com/#swagger%E8%B7%AF%E5%BE%84%E8%BD%AC%E6%8D%A2) **swagger路径转换**

通过以上配置，可以实现文档的参考和展示了，但是使用swagger 的 **try it out** 功能发现路径是路由切割后的路径比如：

swagger 文档中的路径为： 主机名：端口：映射路径 少了一个 **服务路由前缀**，是因为展示handler 经过了 **StripPrefixGatewayFilterFactory** 这个过滤器的处理，原有的 路由前缀被过滤掉了！

[**#**](https://pig4cloud.com/#%E6%96%B9%E6%A1%881%EF%BC%8C%E9%80%9A%E8%BF%87swagger-%E7%9A%84host-%E9%85%8D%E7%BD%AE%E6%89%8B%E5%8A%A8%E7%BB%B4%E6%8A%A4%E4%B8%80%E4%B8%AA%E5%89%8D%E7%BC%80) **方案1，通过swagger 的host 配置手动维护一个前缀**

return new Docket(DocumentationType.SWAGGER\_2)

.apiInfo(apiInfo())

.host("主机名：端口：服务前缀") //注意这里的主机名：端口是网关的地址和端口

.select()

.apis(RequestHandlerSelectors.withMethodAnnotation(ApiOperation.class))

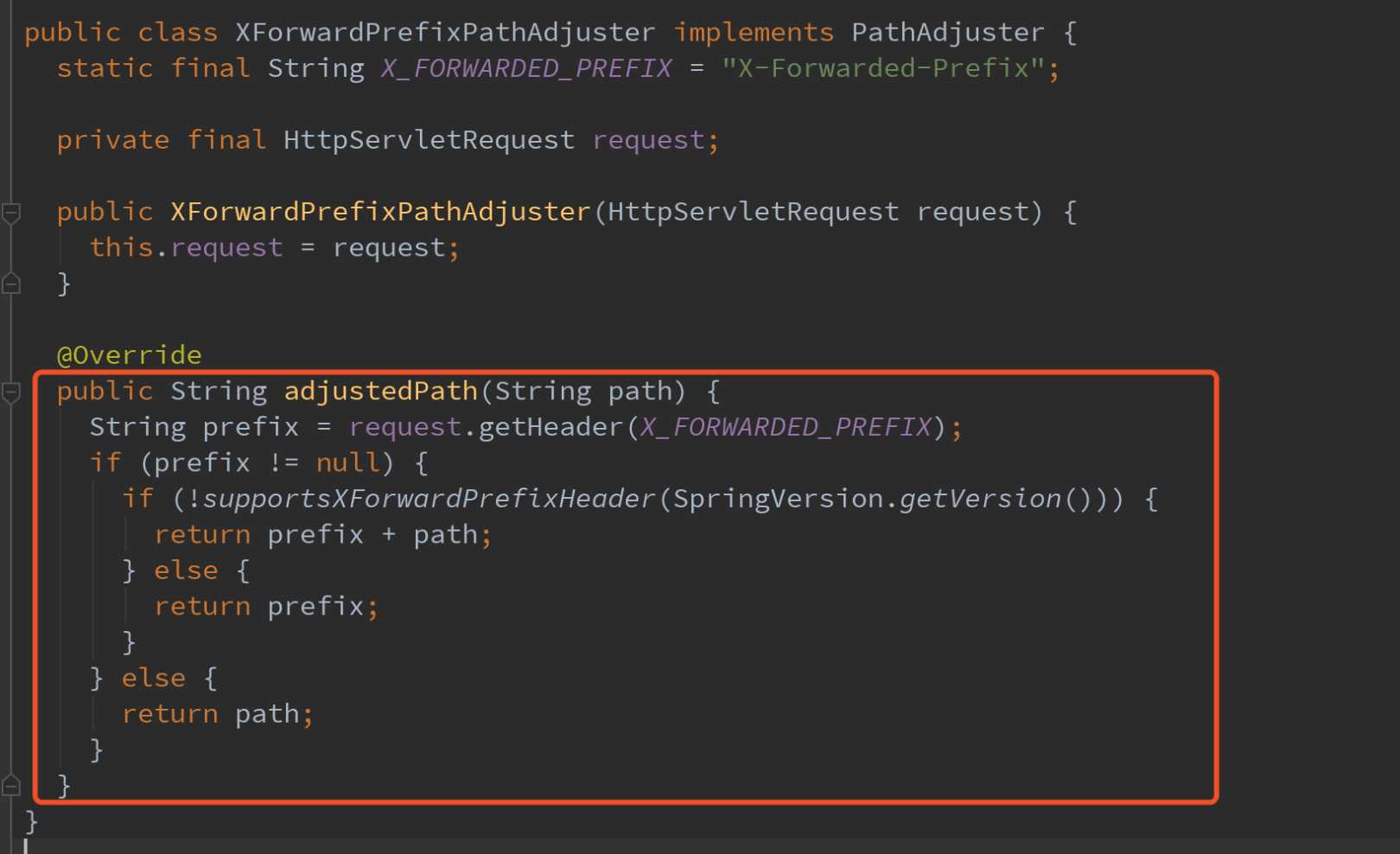
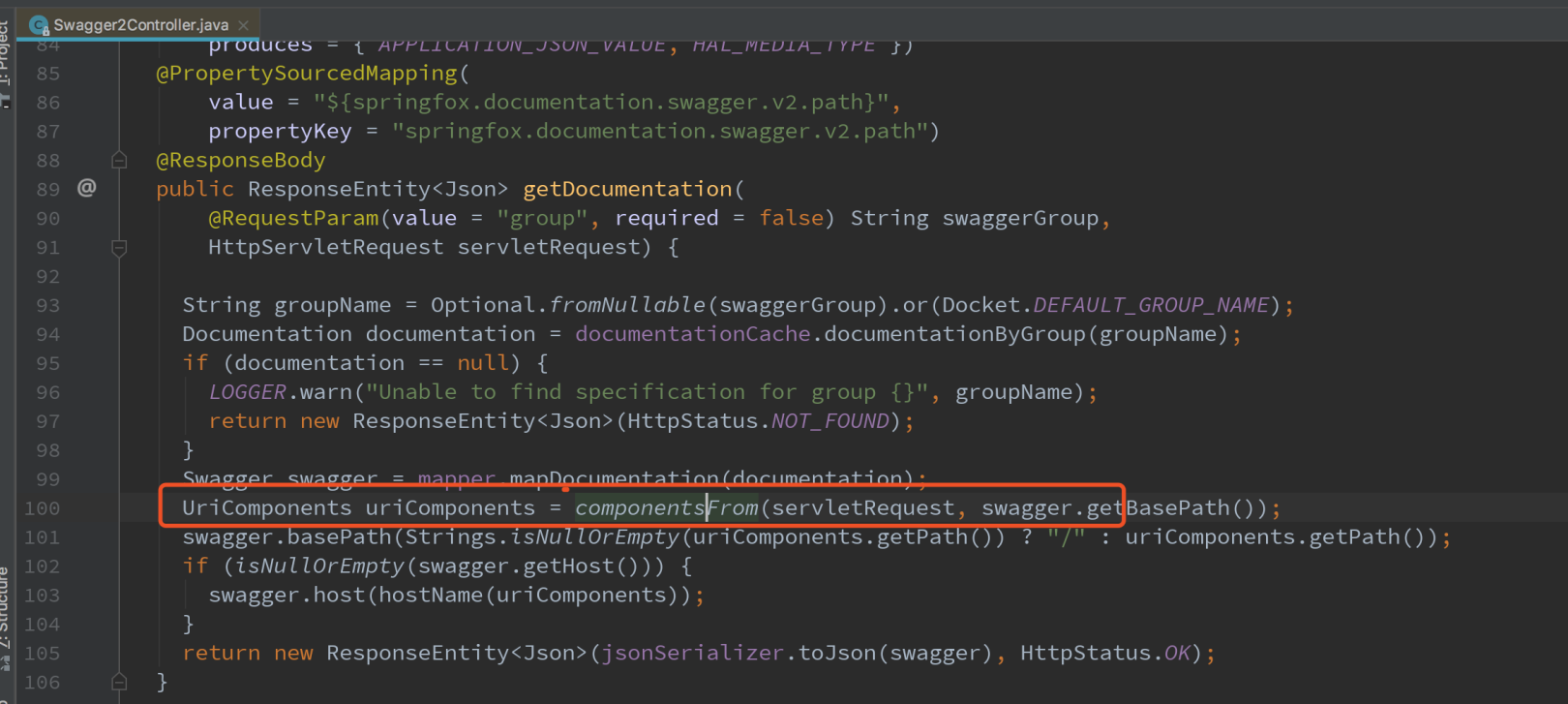
.paths(PathSelectors.any())

.build()

.globalOperationParameters(parameterList);

[**#**](https://pig4cloud.com/#%E6%96%B9%E6%A1%882%EF%BC%8C%E5%A2%9E%E5%8A%A0x-forwarded-prefix) **方案2，增加X-Forwarded-Prefix**

swagger 在拼装URL 数据时候，会增加X-Forwarder-Prefix 请求头里面的信息为前缀

通过如上分析，知道应该在哪里下手了吧，在 网关上追加一个请求头即可

@Component

public class SwaggerHeaderFilter extends AbstractGatewayFilterFactory {

private static final String HEADER\_NAME = "X-Forwarded-Prefix";

@Override

public GatewayFilter apply(Object config) {

return (exchange, chain) -> {

ServerHttpRequest request = exchange.getRequest();

String path = request.getURI().getPath();

if (!StringUtils.endsWithIgnoreCase(path, SwaggerProvider.API\_URI)) {

return chain.filter(exchange);

}

String basePath = path.substring(0, path.lastIndexOf(SwaggerProvider.API\_URI));

ServerHttpRequest newRequest = request.mutate().header(HEADER\_NAME, basePath).build();

ServerWebExchange newExchange = exchange.mutate().request(newRequest).build();

return chain.filter(newExchange);

};

}

}

[**#**](https://pig4cloud.com/#%E6%80%BB%E7%BB%93) **总结**

* 相对zuul的实现，核心逻辑都是一样，获取到配置路由信息，重写swaggerresource
* gateway的配置稍微麻烦，资源的提供handler，swagger url 重写的细节