# Spring Cloud Zuul 开发手册（1.4.x 稳定版）

## 1. Spring Cloud Zuul 概述

### 1.1 Zuul 简介

Spring Cloud Zuul 是 Netflix 开源的 API 网关组件，与 Spring Cloud 深度集成，提供动态路由、监控、鉴权、熔断等核心能力。Zuul 1.x 版本基于 Servlet 2.5 阻塞式模型，适用于中小规模微服务架构。

#### 核心功能：

* **路由转发**：将外部请求映射到内部微服务
* **统一鉴权**：权限校验、安全控制
* **请求过滤**：数据预处理/后处理
* **熔断降级**：集成 Hystrix 实现服务保护

### 1.2 Zuul 核心架构

Zuul 1.x 基于 Servlet 模型，使用多线程处理请求：

HTTP Request → Zuul Servlet → Filter Chain → 微服务

**线程模型特点**：

* 每个请求分配独立线程
* 线程池大小影响吞吐量（默认 200 线程）
* 适合低并发场景，高并发需谨慎调优

## 2. 快速入门

### 2.1 环境要求

* JDK 1.8+
* Spring Boot 1.5.x~2.0.x
* Spring Cloud Edgware~Finchley

### 2.2 基础项目搭建

**步骤 1：添加依赖**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-zuul</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

**步骤 2：启用 Zuul 代理**

@SpringBootApplication

@EnableZuulProxy

public class ZuulApplication {

public static void main(String[] args) {

SpringApplication.run(ZuulApplication.class, args);

}

}

**步骤 3：配置路由规则（application.yml）**

zuul:

routes:

user-service:

path: /api/users/\*\*

serviceId: user-service

order-service:

path: /api/orders/\*\*

url: http://localhost:8082

## 3. 路由配置详解

### 3.1 配置方式对比

|  |  |  |
| --- | --- | --- |
| 配置方式 | 示例 | 适用场景 |
| 服务发现集成 | serviceId: user-service | Eureka 注册中心 |
| 直接 URL 映射 | url: http://localhost:8080 | 非注册服务/外部服务 |
| 全局忽略模式 | ignoredPatterns: /\*\*/admin/\*\* | 屏蔽敏感路径 |

### 3.2 高级路由配置

zuul:

prefix: /v1

stripPrefix: false

routes:

legacy:

path: /legacy/\*\*

url: http://old-service

sensitiveHeaders: Cookie,Authorization

#### 参数说明：

* stripPrefix：是否移除路径前缀（默认 true）
* sensitiveHeaders：屏蔽敏感请求头
* retryable：是否启用重试机制

## 4. 过滤器开发

### 4.1 过滤器类型

|  |  |  |
| --- | --- | --- |
| 类型 | 执行阶段 | 典型场景 |
| pre | 路由前执行 | 鉴权、限流、日志记录 |
| route | 路由时执行 | 服务转发、HTTP 客户端 |
| post | 路由后执行 | 响应加工、指标收集 |
| error | 发生异常时执行 | 异常处理、熔断响应 |

### 4.2 自定义过滤器示例

**鉴权过滤器（pre 类型）**

public class AuthFilter extends ZuulFilter {

@Override

public String filterType() {

return "pre";

}

@Override

public int filterOrder() {

return 0;

}

@Override

public boolean shouldFilter() {

return true;

}

@Override

public Object run() {

RequestContext ctx = RequestContext.getCurrentContext();

String token = ctx.getRequest().getHeader("Authorization");

if (!validateToken(token)) {

ctx.setSendZuulResponse(false);

ctx.setResponseStatusCode(401);

ctx.setResponseBody("Unauthorized");

}

return null;

}

private boolean validateToken(String token) {

// 实现鉴权逻辑

}

}

**注册过滤器**

@Bean

public AuthFilter authFilter() {

return new AuthFilter();

}

## 5. 高级功能配置

### 5.1 熔断降级（Hystrix 集成）

hystrix:

command:

default:

execution.isolation.thread.timeoutInMilliseconds: 5000

zuul:

routes:

user-service:

path: /users/\*\*

serviceId: user-service

stripPrefix: false

retryable: true

### 5.2 动态路由

通过 /refresh 端点实现配置热更新：

@RefreshScope

@Configuration

public class DynamicRouteConfig {

// 实现动态路由逻辑

}

### 5.3 性能优化配置

# 调整线程池参数

zuul:

host:

maxTotalConnections: 500

maxPerRouteConnections: 100

# 禁用某些过滤器

zuul:

SendErrorFilter:

error:

disable: true

## 6. 最佳实践

### 6.1 安全建议

* 始终启用 sensitiveHeaders
* 在 pre 过滤器实现 JWT/OAuth2 鉴权
* 限制 /actuator/routes 等管理端口的访问

### 6.2 性能调优

# 调整连接池参数

httpclient:

maxTotalConnections: 500

maxConnectionsPerRoute: 50

# 启用响应压缩

server:

compression:

enabled: true

### 6.3 常见问题排查

**问题 1：路由失效**

* 检查服务是否在注册中心上线
* 确认 stripPrefix 配置是否符合预期

**问题 2：过滤器不执行**

* 检查 filterOrder() 优先级
* 确认 shouldFilter() 返回值

## 7. 版本说明

* **推荐版本**：1.4.7.RELEASE（Spring Cloud Finchley）
* **停止维护时间**：2020 年 12 月（建议新项目使用 Gateway）
* **性能基准**：单实例支撑 500~1000 TPS（视业务复杂度）

**注意事项**

1. Zuul 2.x 需要自行集成，官方未提供 Spring Cloud 支持
2. 高并发场景建议结合 Ribbon 做负载均衡
3. 生产环境必须配置 Hystrix 熔断策略

本手册适用于 Zuul 1.x 稳定版本开发，如需动态扩展、灰度发布等高级功能，建议结合 Spring Cloud Config 实现配置中心化管理。