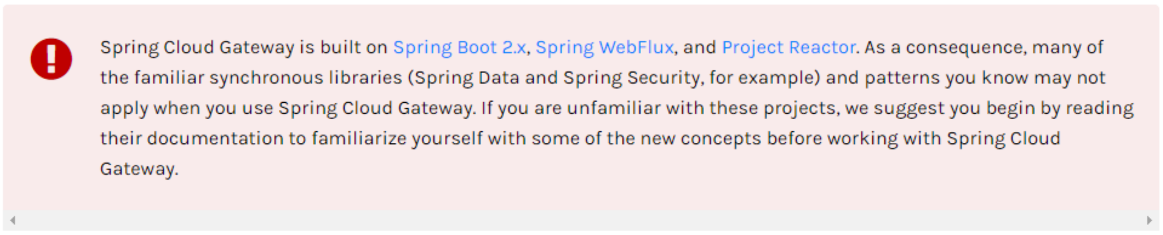
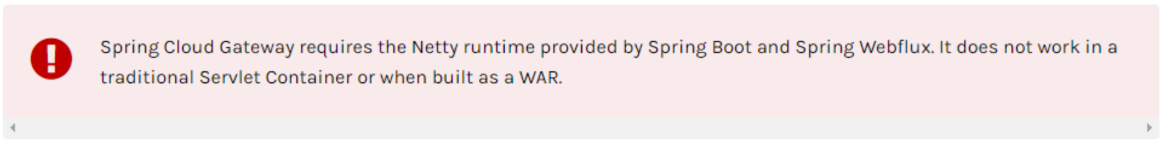
Spring Cloud Gateway Guide

> 学习技术最佳渠道，肯定是官方文档： [gateway传送门]



Spring Cloud Gateway is built on Spring Boot 2.x, Spring WebFlux, and Project Reactor. As a consequence, many of the familiar synchronous libraries (Spring Data and Spring Security, for example) and patterns you know may not apply when you use Spring Cloud Gateway. If you are unfamiliar with these projects, we suggest you begin by reading their documentation to familiarize yourself with some of the new concepts before working with Spring Cloud Gateway.



Spring Cloud Gateway requires the Netty runtime provided by Spring Boot and Spring Webflux. It does not work in a traditional Servlet Container or when built as a WAR.

主要2点

- Spring Cloud Gateway是构建在Spring Boot 2.x、Spring Webflux 以及 Reactor 之上的，因此一些同步类库可能不能和gateway同时使用。 （Spring Data/ Spring Security）
- 因为SC Gateway需要Springboot和Netty的运行支持，因此无法支持传统的Servlet 容器，也不支持编译war包。

Quick Start

和`Seata`中提到的一样，Gateway的使用同样遵循三步曲规律，首先我们创建项目`broad-gateway`。

- 第一步：添加依赖

```
<dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-gateway</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
    </dependencies>
```

  - `spring-cloud-starter-gateway` 网关的主依赖
  - `spring-boot-starter-actuator` 用于对外暴露微服务的管理端点(endpoint)

    - `spring-cloud-starter-netflix-eureka-client` 用于服务注册（因为我们的路由规则是根据 `service-name` 进行实现匹配和负载均衡的）
- 第二步：加注解(gateway不需要开始注解，这里只需添加服务发现客户端注解)

```java
@SpringBootApplication
@EnableDiscoveryClient
public class GatewayApplication {
    public static void main(String[] args) {
        new SpringApplicationBuilder()
                .sources(GatewayApplication.class)
                .run(args);
    }
}
```

- 第三部：改配置

```yaml
server:
  port: 9999
eureka:
  instance:
    prefer-ip-address: true
  client:
    service-url:
      defaultZone: http://172.16.1.187:21001/eureka
spring:
  application:
    name: broadway-gateway
  cloud:
    #划重点
    gateway:
      #wiretap是gateway排错高级功能，从Greenwich SR3开始支持
      httpserver:
        wiretap: true
      httpclient:
        wiretap: true
      discovery:
        locator:
          # gateway通过discovery自动发现其他微服务
          enabled: true
      routes:
        - id:  ws_tally_route
          uri: lb://ws-tally-service
          # 决定是否返回404的关键
          predicates:
            - TimeBetween=上午6:00,下午11:00
            - Path=/v1/ws/tally/**
          # 对请求进行过滤处理
          filters:
            # 局部过滤器配置顺序默认数值从1开始递增
            - AddRequestHeader=CompanyKey,123456 # sort(1)
            - AddResponseHeader=Success,Isaac    # sort(2)
            - PreLog=CustomLogKey,CustomLogValue # sort(3)
# springboot 下监控的核心基础
management:
  endpoints:
    web:
      exposure:
        include: '*'
  endpoint:
    health:
      show-details: always
logging:
  level:
    org.springframework.cloud.gateway: debug
```

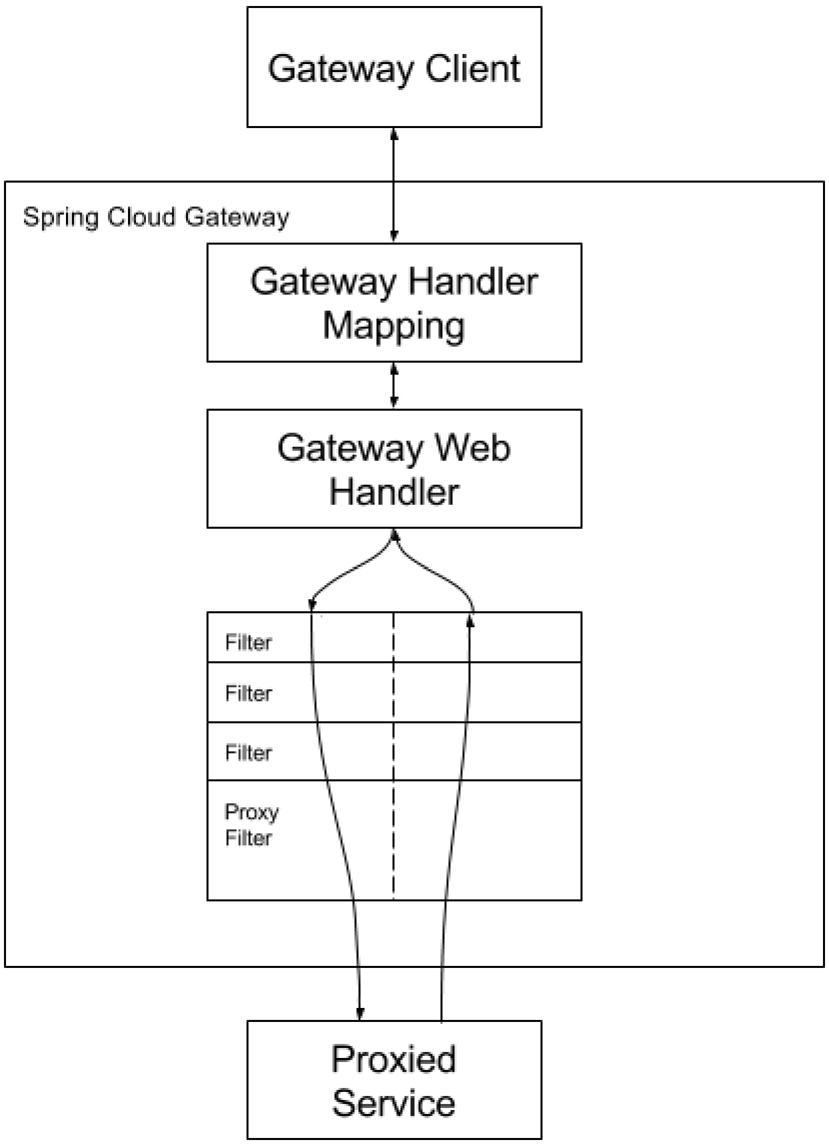
术语

- **Route**： 网关最基本的模块。（路由是否匹配是由所有的`predicate==true`决定的）

  Route 包含以下几个核心属性配置：

  - id
  - uri
  - predicates
  - filters
- **Predicate**：这是java8 引入的函数式编程的新特性Predicate。输入的类型是`org.springframework.web.server.ServerWebExchange`，它允许匹配任何来自于HTTP request的内容，例如：`headers` / `parameters`。
- **Filter**：这是使用特定创建工厂构建的一些`org.springframework.cloud.gateway.filter.GatewayFilter`的实例，可以对请求之前或者返回之前的请求/响应对象进行修改。

请求流程



Route Predicate

Spring Cloud Gateway 内置了11种路由谓词工厂。

1. After Route Predicate Factory
   接受一个datetime类型的参数，该实例匹配在设定时间之`后`的请求，否则返回`404`.

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: after_route
        uri: https://www.life-runner.com
        predicates:
        - After=2020-01-01T17:42:47.789-07:00[America/Denver]
```

2. Before Route Predicate Factory
   接受一个datetime类型的参数，该实例匹配在设定时间之`前`的请求，否则返回`404`.

```yaml
 spring:
   cloud:
     gateway:
       routes:
       - id: before_route
         uri: https://www.life-runner.com
         predicates:
         - Before=2020-01-01T17:42:47.789-07:00[America/Denver]
```

3. Between Route Predicate Factory
   接受2个datetime类型参数，该谓词匹配发生在2个时间范围之内的请求,第二个参数必须在第一个参数日期之后。

```yaml
 spring:
   cloud:
     gateway:
       routes:
       - id: between_route
         uri: https://www.life-runner.com
         predicates:
         - Between=2017-01-20T17:42:47.789-07:00[America/Denver], 2017-01-21T17:42:47.789-07:00[America/Denver]
```

4. Cookie Route Predicate Factory
   接受2个参数： `cookie_name` 和 `正则表达式` ,该谓词匹配当前给定的cookie-name并且value符合正则表达式的请求。

```yaml
spring:
   cloud:
     gateway:
       routes:
       - id: cookie_route
         uri: https://example.org
         predicates:
         - Cookie=name, \s+
```

5. Header Route Predicate Factory
   接受2个参数: `header-name` 和 `正则表达式` ,该谓词匹配当前给定的header-name 并且value 符合正则表达式的请求。

```yaml
spring:
   cloud:
     gateway:
       routes:
       - id: header_route
         uri: https://www.life-runner.com
         predicates:
         - Header=X-Request-Id, \d+
```

6. Host Route Predicate Factory
   接受1个参数，host patterns 列表，这个patterns是 `Ant-Style` ，使用 `.` 分隔。

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: host_route
        uri: https://www.life-runner.com
        predicates:
        - Host=**.babydy.cn,**.life-runner.com
```

也支持模板变量，例如：{sub}.life-runner.com

7. Method Route Predicate Factory
   接受1-N个参数，匹配HTTP Method.

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: method_route
        uri: https://www.life-runner.com
        predicates:
        - Method=GET,POST,PUT,DELETE
```

8. Path Route Predicate Factory
   接受一个`org.springframework.util.PathMatcher`模式类型的list。

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: host_route
        uri: https://www.life-runner.com
        predicates:
        - Path=/users/**,/share/**,/setinel/{keys}
```

9. Query Route Predicate Factory
   接受2个参数：第一个`param`必填，第二个可选`regexp`,匹配一个查询路由谓词。

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: query_route
        uri: https://www.life-runner.com
        predicates:
        - Query=isaac
        #- Query=isaac,/d+
```

上述例子表示：如果请求中包含一个isaac的查询参数，则匹配，否则，返回404

10. RemoteAddr Route Predicate Factory
    接受(IPV4/IPV6)的字符串list,最少1个，例如：（192.168.1.1/16）IP+子网掩码

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: remoteaddr_route
        uri: https://www.life-runner.com
        predicates:
        - RemoteAddr=192.168.1.1/16
```

11. Weight Route Predicate Factory
    接受2个参数: `group` 和 `weight`,权重是在每一组内分别被计算.

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: weight_high
        uri: https://www.life-runner.com
        predicates:
        - Weight=group1, 8
      - id: weight_low
        uri: https://www.babydy.cn
        predicates:
        - Weight=group1, 2
```

    表示80%流量请求`life-runner`,20%流量请求到`babydy`.