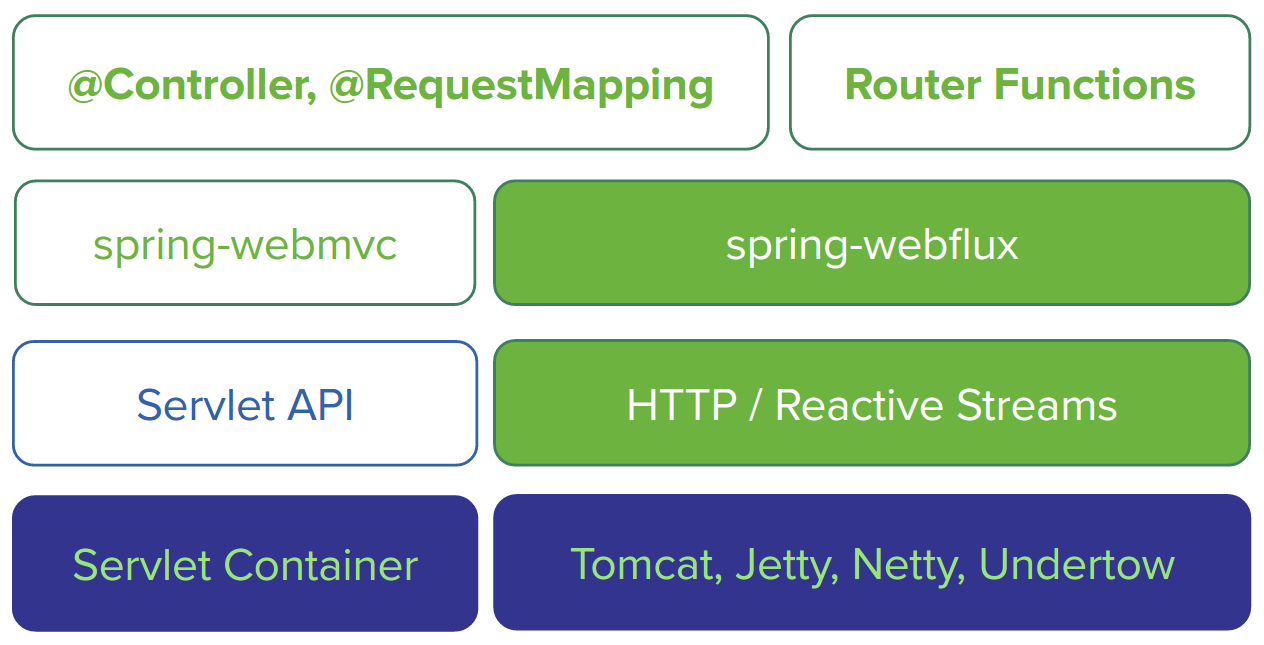
# SpringCloud Gateway

SpringCloud Gateway是建立在Spring生态(Spring 5、Spring Boot 2、Project Reactor)之上的Api网关。旨在提供一种简单有效的方式来对Api进行路由，并提供横切关注点的功能，如：安全校验、监控、统计指标和弹性扩展。

SpringCloud Gateway是基于Netty和WebFlux构建的。

## 0.WebFlux

Spring WebFlux是Spring5推出的响应式Web框架。



1）服务端技术栈

Spring提供了完整的支持响应式的服务端技术栈

如上图所示，左侧为基于spring-webmvc的技术栈，右侧为基于spring-webflux的技术栈，

Spring WebFlux是基于响应式流的，因此可以用来建立异步的、非阻塞的、事件驱动的服务。它采用Reactor作为首选的响应式流的实现库，不过也提供了对RxJava的支持。

由于响应式编程的特性，Spring WebFlux和Reactor底层需要支持异步的运行环境，比如Netty和Undertow；也可以运行在支持异步I/O的Servlet 3.1的容器之上，比如Tomcat（8.0.23及以上）和Jetty（9.0.4及以上）。

从图的纵向上看，spring-webflux上层支持两种开发模式：

类似于Spring WebMVC的基于注解（@Controller、@RequestMapping）的开发模式；

Java 8 lambda 风格的函数式开发模式。

Spring WebFlux也支持响应式的Websocket服务端开发。

2）响应式Http客户端

此外，Spring WebFlux也提供了一个响应式的Http客户端API WebClient。它可以用函数式的方式异步非阻塞地发起Http请求并处理响应。其底层也是由Netty提供的异步支持。

我们可以把WebClient看做是响应式的RestTemplate，与后者相比，前者：

是非阻塞的，可以基于少量的线程处理更高的并发；

可以使用Java 8 lambda表达式；

支持异步的同时也可以支持同步的使用方式；

可以通过数据流的方式与服务端进行双向通信。

Mono和Flux是Reactor提供的响应式模型，Mono 是一个用来发送 0 或者单值数据的发布器，Flux 可以用来发送 0 到 N 个值。

## 1.引入SpringCloud Gateway

在pom文件添加下面启动器：

<**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-gateway</**artifactId**>  
</**dependency**>

SpringBoot版本：2.0.6.RELEASE

SpringCloud版本：Finchley.SR2

当添加了starter，由于某些原因又不想启动网关，则可以设置：

spring.cloud.gateway.enabled=false

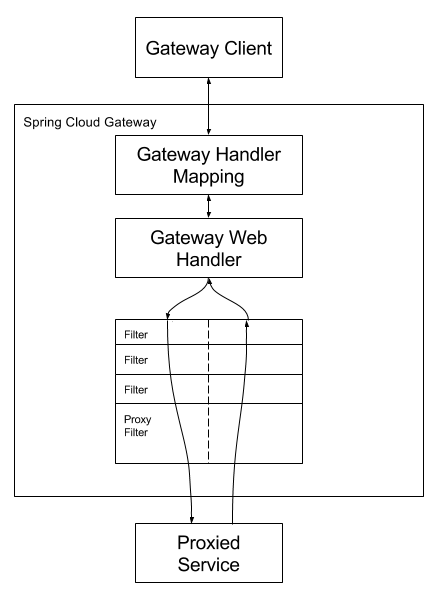
## 2.相关词汇

Route:路由是网关的基本组成，包含id、目标Url、谓词集合和过滤器集合，当聚合谓词返回true时，则匹配到对应路由。

Predicate:是java8的函数谓词,输入参数类型为: org.springframework.web.server. ServerWebExchange,允许开发人员匹配Http请求中的任何内容,如:请求头和参数。

Filter:使用特定工厂类创建的org.springframework.cloud.gateway.filter. GatewayFilter实例,在过滤器中,可以在请求发往下游时，修改请求和响应。

## 3.SpringCloud Gateway是如何运作的



客户端向SpringCloud Gateway发生请求,如果网关的Handler Mapping能够确定请求匹配到了路由,则会将请求发生给网关Web Handler。该处理程序会将请求在过滤器链上执行，过滤器有请求和响应两个阶段。执行完所有的pre类型的过滤器逻辑之后，会进行代理请求，在发出代理请求后，执行post类型的过滤器逻辑。

## 4.路由谓词工厂

SpringCloud Gateway将路由作为Spring WebFlux HandlerMapping的基础部分。Gateway包含了很多内置的Route Predicate Factory。所有的这些Predicate都会匹配Http请求的不同属性。可以组合多个Predicate，最后的结果是逻辑And。

### 4.1 After 路由谓词工厂

After路由谓词工厂需要一个datetime类型的参数,当请求时间发生在之后都会返回true。

spring:

cloud:

gateway:

routes:

- id: after\_route

uri: http://example.org

predicates:

- After=2017-01-20T17:42:47.789-07:00[America/Denver**]**

内置的路由谓词在包：

org.springframework.cloud.gateway.handler.predicate

类似的谓词工厂有：Before、Between

总结：匹配请求时间Before、After、Between

### 4.2 匹配Cookie

Cookie路由谓词工厂需要两个参数：cookie name和正则表达式。当请求含有指定名字的cookie并且值匹配了正则表达式，则返回true：

spring:

cloud:

gateway:

routes:

- id: cookie\_route

uri: http://example.org

predicates:

- Cookie=chocolate**,** ch.p

### 4.3 匹配请求头Header

Header路由谓词工厂需要两个参数：header name和正则表达式。当请求含有指定名字的header并且值匹配了正则表达式，则返回true：

spring:

cloud:

gateway:

routes:

- id: header\_route

uri: http://example.org

predicates:

- Header=X-Request-Id**,** \d+

### 4.4 匹配host

Host路由谓词工厂需要一个参数：host名称的匹配模式，该模式是Ant风格的，. 表示分隔。当请求含有Host的header并且值匹配了指定，则返回true：

spring:

cloud:

gateway:

routes:

- id: host\_route

uri: http://example.org

predicates:

- Host=\*\*.somehost.org

### 4.5 匹配请求方法

Method路由谓词工厂需要一个参数：Http请求方法,当请求方法与配置匹配了，则返回true。

spring:

cloud:

gateway:

routes:

- id: method\_route

uri: http://example.org

predicates:

- Method=GET

### 4.6 匹配路径

Path路由谓词工厂需要两个参数：一个PathMatcher列表和一个可选的matchOptionalTrailingSeparator标志。

spring:

cloud:

gateway:

routes:

- id: host\_route

uri: http://example.org

predicates:

- Path=/foo/{segment},/bar/{segment**}**

上述的定义会将Url路径变量segment，映射到ServerWebExchange的Attributes中，以map的形式存在，这些变量在GatewayFilter Factory中可以使用到：

Map<String, String> uriVariables = ServerWebExchangeUtils.getPathPredicateVariables(exchange);

String segment = uriVariables.get("segment");

### 4.7 查询参数匹配

Query路由谓词工厂需要两个参数：查询参数名称和值的正则模式(可选)

spring:

cloud:

gateway:

routes:

- id: query\_route

uri: http://example.org

predicates:

- Query=foo**,** ba.

上述的配置会匹配到查询参数中含有foo参数,并且参数值为baz，bas的请求。

### 4.8 请求地址匹配

RemoteAddr用于匹配请求的来源地址

spring:

cloud:

gateway:

routes:

- id: remoteaddr\_route

uri: http://example.org

predicates:

- RemoteAddr=192.168.1.1/24

192.168.1.1是ip，24是子网掩码。

上述的匹配会匹配到来之192.168.1.10的请求。

默认情况下，RemoteAddr会使用请求的remote address作为参数进行判断，如果SpringCloud Gateway是在代理层之后，则可能匹配不到真正的客户端ip地址。

可以通过自定义RemoteAddressResolver来解析remote address。SpringCloud Gateway附带了一个非默认的远程ip解析器：XForwardedRemoteAddressResolver基于X-Forwarded-For请求头进行解析。

XForwardedRemoteAddressResolver有两个静态构造函数：trustAll和maxTrustedIndex。trustAll会获取X-Forwarded-For的第一个ip地址作为客户端ip地址，这种方式容易被欺骗，因为客户端可以为该请求头设置一个初始值；maxTrustedIndex可以获取X-Forwarded-For指定下标的值，当请求经过多次代理的时候，X-Forwarded-For会有多个值，如：X-Forwarded-For: 0.0.0.1, 0.0.0.2, 0.0.0.3。

maxTrustedIndex为1时取0.0.0.3,2时取0.0.0.2，大于等于3时取0.0.01.下标是从1开始的。

## 5.GatewayFilter工厂