**一、什么是服务网关**

服务网关 = 路由转发 + 过滤器

1、路由转发：接收一切外界请求，转发到后端的微服务上去；

2、过滤器：在服务网关中可以完成一系列的横切功能，例如权限校验、限流以及监控等，这些都可以通过过滤器完成（其实路由转发也是通过过滤器实现的）。

**二、为什么需要服务网关**  
上述所说的横切功能（以权限校验为例）可以写在三个位置：

* 每个服务自己实现一遍
* 写到一个公共的服务中，然后其他所有服务都依赖这个服务
* 写到服务网关的前置过滤器中，所有请求过来进行权限校验

第一种，缺点太明显，基本不用；  
第二种，相较于第一点好很多，代码开发不会冗余，但是有两个缺点：

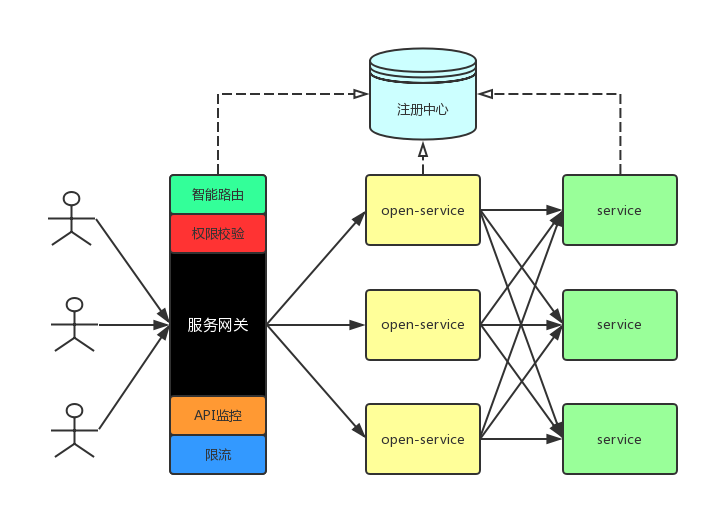
* 由于每个服务引入了这个公共服务，那么相当于在每个服务中都引入了相同的权限校验的代码，使得每个服务的jar包大小无故增加了一些，尤其是对于使用docker镜像进行部署的场景，jar越小越好；
* 由于每个服务都引入了这个公共服务，那么我们后续升级这个服务可能就比较困难，而且公共服务的功能越多，升级就越难，而且假设我们改变了公共服务中的权限校验的方式，想让所有的服务都去使用新的权限校验方式，我们就需要将之前所有的服务都重新引包，编译部署。

而服务网关恰好可以解决这样的问题：

* 将权限校验的逻辑写在网关的过滤器中，后端服务不需要关注权限校验的代码，所以服务的jar包中也不会引入权限校验的逻辑，不会增加jar包大小；
* 如果想修改权限校验的逻辑，只需要修改网关中的权限校验过滤器即可，而不需要升级所有已存在的微服务。

所以，需要服务网关！！！

**三、服务网关技术选型**

****

引入服务网关后的微服务架构如上，总体包含三部分：服务网关、open-service和service。

**1、总体流程：**

* 服务网关、open-service和service启动时注册到注册中心上去；
* 用户请求时直接请求网关，网关做智能路由转发（包括服务发现，负载均衡）到open-service，这其中包含权限校验、监控、限流等操作
* open-service聚合内部service响应，返回给网关，网关再返回给用户

**2、引入网关的注意点**

* 增加了网关，多了一层转发（原本用户请求直接访问open-service即可），性能会下降一些（但是下降不大，通常，网关机器性能会很好，而且网关与open-service的访问通常是内网访问，速度很快）；
* 网关的单点问题：在整个网络调用过程中，一定会有一个单点，可能是网关、nginx、dns服务器等。防止网关单点，可以在网关层前边再挂一台nginx，nginx的性能极高，基本不会挂，这样之后，网关服务就可以不断的添加机器。但是这样一个请求就转发了两次，所以最好的方式是网关单点服务部署在一台牛逼的机器上（通过压测来估算机器的配置），而且nginx与zuul的性能比较，根据国外的一个哥们儿做的实验来看，其实相差不大，zuul是netflix开源的一个用来做网关的开源框架；
* 网关要尽量轻。

**3、服务网关基本功能**

* 智能路由：接收**外部**一切请求，并转发到后端的对外服务open-service上去；
  + 注意：我们只转发外部请求，服务之间的请求不走网关，这就表示全链路追踪、内部服务API监控、内部服务之间调用的容错、智能路由不能在网关完成；当然，也可以将所有的服务调用都走网关，那么几乎所有的功能都可以集成到网关中，但是这样的话，网关的压力会很大，不堪重负。
* 权限校验：只校验用户向open-service服务的请求，不校验服务内部的请求。服务内部的请求有必要校验吗？
* API监控：只监控经过网关的请求，以及网关本身的一些性能指标（例如，gc等）；
* 限流：与监控配合，进行限流操作；
* API日志统一收集：类似于一个aspect切面，记录接口的进入和出去时的相关日志
* 。。。后续补充

上述功能是网关的基本功能，网关还可以实现以下功能：

* A|B测试：A|B测试时一块比较大的东西，包含后台实验配置、数据埋点（看转化率）以及分流引擎，在服务网关中，可以实现分流引擎，但是实际上分流引擎会调用内部服务，所以如果是按照上图的架构，分流引擎最好做在open-service中，不要做在服务网关中。
* 。。。后续补充

**4、技术选型**

笔者准备自建一个轻量级的服务网关，技术选型如下：

* 开发语言：java + groovy，groovy的好处是网关服务不需要重启就可以动态的添加filter来实现一些功能；
* 微服务基础框架：springboot；
* 网关基础组件：netflix zuul；
* **服务注册中心：Eureka Server；**
* 权限校验：jwt；
* API监控：prometheus + grafana；
* API统一日志收集：logback + ELK；
* 压力测试：Jmeter；