# CAP

1. C-Consistency：强一致性
2. A-Availability：高可用性
3. P-Partition tolerance：分区容错性

# 服务注册中心

1. Eureka
2. Zookeeper
3. Consul

## Eureka

### EurekaServer

1. @EnableEurekaServer
2. 默认端口8761
3. **Eureka服务器自己不要注册自己**

### EurekaClient

1. @EnableDiscoveryClient/@EnableEurekaClient

# 负载均衡（Load Balance）

将用户的请求平摊到多个服务器上，实现HA（高可用）；

1. 本地负载均衡/进程内LB
2. 服务器负载均衡/集中式LB

## Ribbon

负载均衡+RestTemplate调用

### 工作流程

1. 优先选择服务较少的EurekaServer
2. 根据用户指定的策略，从Sever获取服务注册列表中的一个地址
3. Ribbon提供了轮询、随机和根据响应时间加权等策略

### 核心接口：IRule

根据特定算法从服务列表中选取一个要访问的服务

#### 七种负载均衡策略

1. RoundRibbonRule：轮询
2. RandomRule：随机
3. RetryRule：轮询重试
4. WeightedResponseTimeRule：相应速度决定权重
5. BestAvailableRule：最优可用
6. AvailabilityFilteringRule：可用性过滤原则
   1. 过滤掉不可用的Server实例
   2. 再选择并发量最小的实例
7. ZoneAvoidanceRule：区域内可用性能最优

### 轮询算法核心

1. 接口请求次数 % 服务器集群总数量 = 实际调用服务器下标位置。
2. 每次服务重启后，接口请求次数重新从1开始。

# 服务调用

## OpenFeign

1. 声明式的webService客户端
2. 定义一个服务接口，然后在上面添加注解即可
3. 消费端使用

### 简单使用

1. 开启feign支持：@EnableFeignClients
2. 定义feign接口，接口上加上@FeignClient

# 服务降级

## Hytrix

1. 处理分布式系统延迟和容错的开源库
2. 不会导致服务整体失败，避免级联故障，提高分布式系统的弹性

### 服务降级-fallback

服务器忙，请稍后再试。不让客户端等待并立刻返回一个友好提示-fallback

#### 服务降级出现场景：

1. 程序运行异常
2. 超时
3. 服务熔断触发降级
4. 线程池、信号量打满触发

##### 简单使用

1. 方法上加上@HystrixCommand
2. 定义一个fallback方法
3. 启动类上加上@EnableCircuitBreaker

### 服务熔断-break

服务降级🡪进而熔断🡪恢复调用链路

1. 调用失败会触发降级，进而调用fallback方法
2. 无论如何，服务降级一定会先运行业务方法，如果不行再运行fallback方法
3. 假如单位时间内调用失败次数过多，也就时降级次数过多，就会触发熔断
4. 熔断之后就会跳过业务方法，直接运行fallback方法
5. 当检测到微服务调用恢复正常后，恢复链路调用。

#### 断路器开、关条件

1. 满足阈值时开启（默认10秒超过20次请求）并且失败率也超过阈值（默认10秒内超过50%请求失败）时开启断路器。开启后，所有请求都不会进行转发
2. 一段时间后（默认5秒），断路器为半开状态，会让其中一个请求进行转发。如果成功，断路器关闭，如果不成功，继续开启，并重复此步骤。

### 服务限流-flowlimit

高并发等操作，请求进行排队，有序进行

# 网关

## gateway

### 核心概念

1. Route（路由）
   1. 路由时构建网关的基本模块
   2. 由ID、目标uri、一系列的断言和过滤器组成
   3. 如果断言为true则匹配该路由
2. Predicate（断言）
   1. 匹配http请求中的内容（比如请求头或者请求参数），如果请求和断言相匹配则进行路由
3. Filter（过滤）
   1. 再请求被路由前、后对请求进行修改