SpringCloud+SpringBoot方案

<https://blog.csdn.net/forezp/article/details/70148833/>

SpringCloud功能简介：

默认没有限流、降级等功能。

springcloud为开发人员提供了快速构建分布式系统的一些工具，包括配置管理、服务发现、断路器、路由、微代理、事件总线、全局锁、决策竞选、分布式会话等等。它运行环境简单，可以在开发人员的电脑上跑。

[**服务的注册与发现（Eureka）**](http://blog.csdn.net/forezp/article/details/69696915)

eureka是一个高可用的组件，它没有后端缓存，每一个实例注册之后需要向注册中心发送心跳（因此可以在内存中完成）。

当client向server注册时，它会提供一些元数据，例如主机和端口，URL，主页等。可以在server中对所有注册的client进行监控与管理，Eureka server 从每个client实例接收心跳消息。 如果心跳超时，则通常将该实例从注册server中删除。

[**服务消费者（Feign）**](http://blog.csdn.net/forezp/article/details/69808079)

Feign是一个声明式的伪Http客户端，它使得写Http客户端变得更简单。使用Feign，只需要创建一个接口并注解。它具有可插拔的注解特性，可使用Feign 注解和JAX-RS注解。Feign支持可插拔的编码器和解码器。Feign默认集成了Ribbon，并和Eureka结合，默认实现了负载均衡的效果。

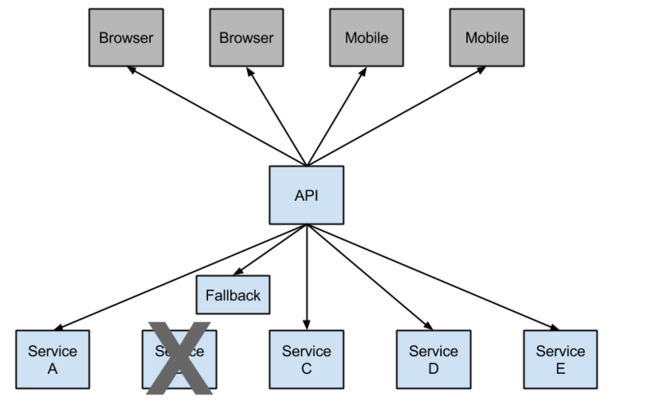
并且可以@FeignClient注解消费指定的模块或服务。



[**断路器（Hystrix）**](http://blog.csdn.net/forezp/article/details/69934399)

在微服务架构中，根据业务来拆分成一个个的服务，服务与服务之间可以相互调用（RPC），在Spring Cloud可以用RestTemplate+Ribbon和Feign来调用。为了保证其高可用，单个服务通常会集群部署。由于网络原因或者自身的原因，服务并不能保证100%可用，如果单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，导致服务瘫痪。服务与服务之间的依赖性，故障会传播，会对整个微服务系统造成灾难性的严重后果，这就是服务故障的“雪崩”效应。

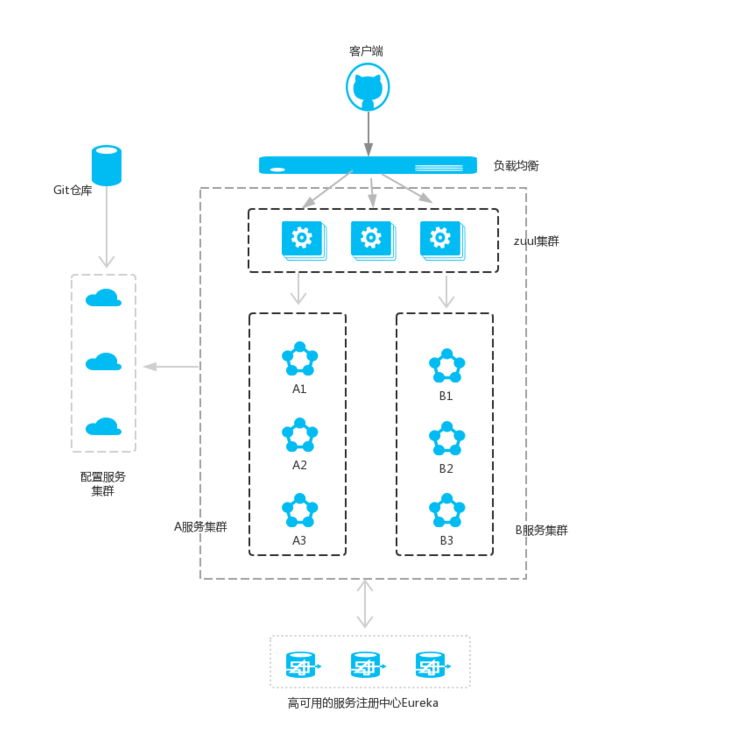
较底层的服务如果出现故障，会导致连锁故障。当对特定的服务的调用的不可用达到一个阀值（Hystric 是5秒20次） 断路器将会被打开。断路打开后，可用避免连锁故障，fallback方法可以直接返回一个固定值。



[**路由网关(zuul)**](http://blog.csdn.net/forezp/article/details/69939114)

在微服务架构中，需要几个基础的服务治理组件，包括服务注册与发现、服务消费、负载均衡、断路器、智能路由、配置管理等，由这几个基础组件相互协作，共同组建了一个简单的微服务系统。

Zuul的主要功能是路由转发和过滤器。路由功能是微服务的一部分，比如／api/user转发到到user服务，/api/shop转发到到shop服务。zuul默认和Ribbon结合实现了负载均衡的功能。



SpringCloud项目基础架构

前台访问

负载均衡

Web层集群

Web服务器2

Web服务器3

Web服务器1

路由层(**zuul**)集群

**zuul**服务器3

**zuul**服务器1

**zuul**服务器2

内网

交互层(f**eign**)消费具体模块

f**eign**服务器3

f**eign**服务器2

f**eign**服务器1

应用层

系统1

系统2

系统1

数据库

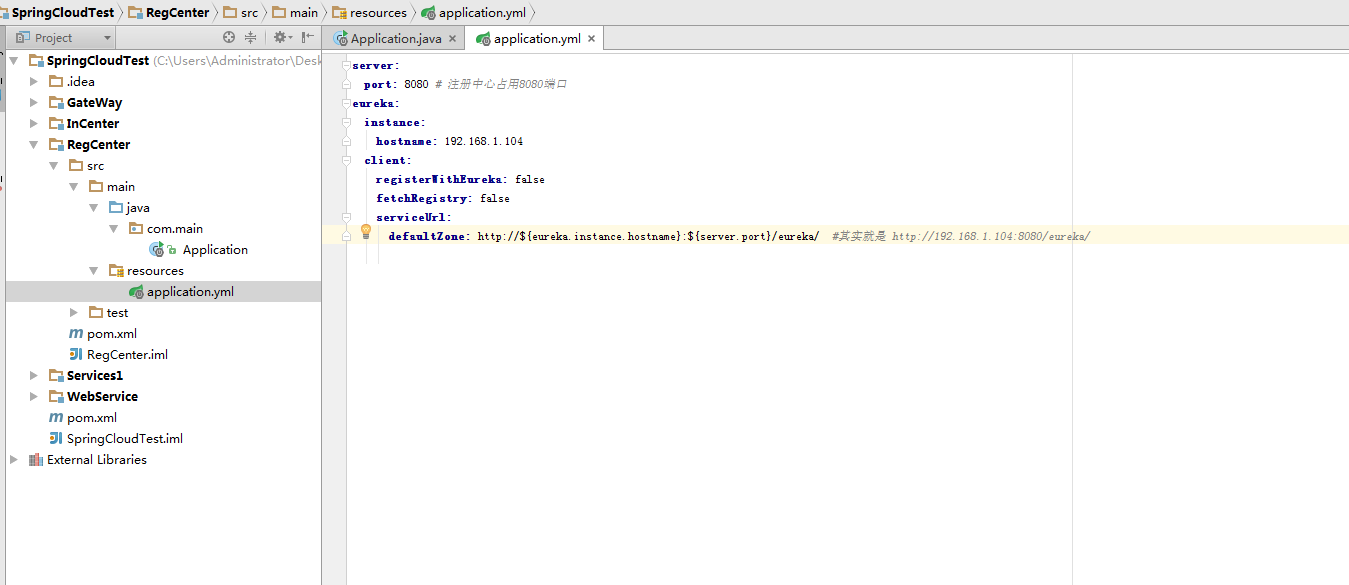
数据库2

数据库1

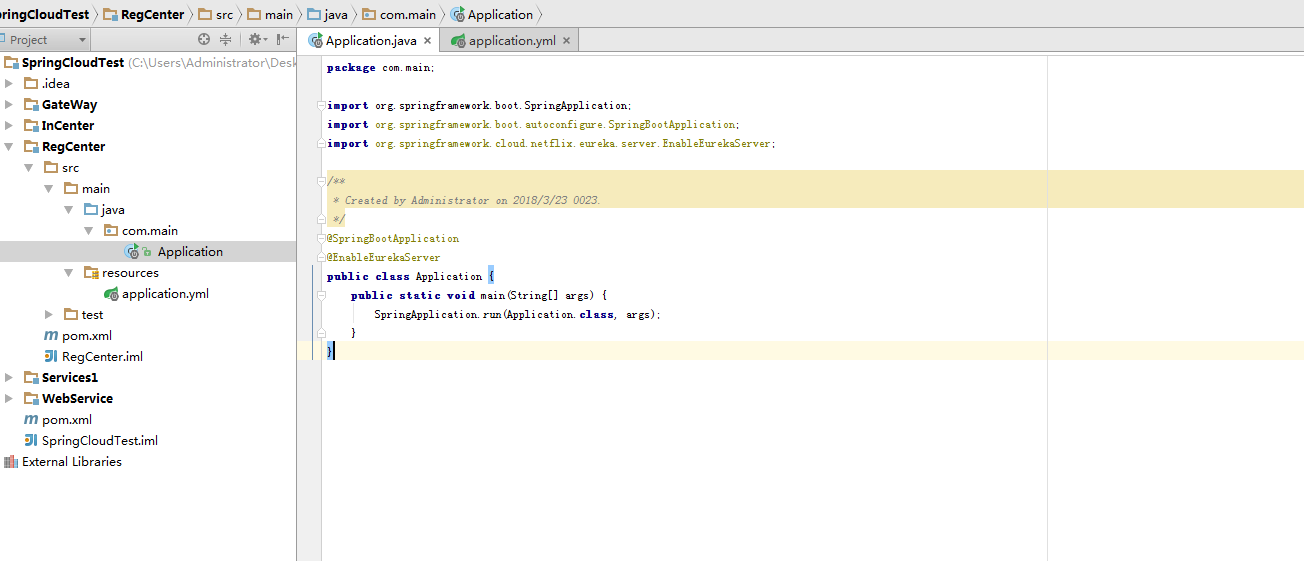
环境搭建：

1. 搭建服务注册中心

1）配置文件。



2）application



配置文件：

**spring:  
 application:  
 name:** service1  
**eureka:  
 instance:  
 hostname:** 192.168.1.106 *#指的是注册的ip*  
 **port:** 8080 *#指的是注册的端口*  
 **client:  
 serviceUrl:  
 defaultZone:** http:// 192.168.1.104:8080/eureka/ *#注册中心地址*

**server:  
 port:** 8080

Spring-cloud框架搭建

具体流程：

1.请求统一通过API网关（Zuul）来访问内部服务.

2.网关接收到请求后，从注册中心（Eureka）获取可用服务

3.由Ribbon进行均衡负载后，分发到后端具体实例

4.微服务之间通过Feign进行通信处理业务

5.Hystrix负责处理服务超时熔断

6.Turbine监控服务间的调用和熔断相关指标

具体模块：

1.服务注册中心

2.服务配置中心

3.服务管理中心

4.链路监控中心

5.路由系统

6.认证服务

7.业务启动

运用技术：

1.SpringBoot+SpringCloud微服务架构

2.Turbine集群监控

3.Rabbit消息队列

4.Oauth2安全控制

5.Zipkin:分布式链路监控

SpringCloud:

Eureka:服务注册配置中心

Config:配置文件管理中心

Hystrix:服务熔断器

Zuul:路由

Ribbon:负载均衡

\*Feign:业务通讯交互(是一种声明式、模板化的 HTTP 客户端)。