springcloudtest

1.建立springcloud-parent maven工程 pom方式

2.建立springcloud-common 公共的

3.分别建立eureka，provide，consumer工程（都可以集群）

eureka 集群配置文件：

defaultZone跟port做修改就行

```yaml
eureka:
  instance:
    hostname: localhost
  client:
    register-with-eureka: false
    fetch-registry: false
    service-url:
     #defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/        #设置与Eureka Server交互的地址查询服务和注册服务都需要依赖
     defaultZone: http://eureka9003.com:9003/eureka/,http://eureka9002.com:9002/eureka/
server:
  port: 9001
```

provide集群配置文件（主要展示注册中心信息，mysql连接不做展示）：

instance-id跟port做修改就行

```yaml
eureka:
  client:
    service-url:
      #defaultZone: http://localhost:9001/eureka       #表示把microservice-provider微服务注册进http://localhost:9001/eureka指示的服务中
      defaultZone: http://eureka9001.com:9001/eureka/,http://eureka9002.com:9002/eureka/,http://eureka9003.com:9003/eureka/
  instance:
      instance-id: springcloud-provider8002                    #自定义服务名称信息
      prefer-ip-address: true                                  #访问路径可以显示IP地址
info:
  app.name: springcloud-provider
  company.name: www.springcloud.com
  build.artifactId: $project.artifactId$
  build.version: $project.version$
```

consumer集群配置文件：

```yaml
server:
  port: 7001

eureka:
  client:
    register-with-eureka: false
    service-url:
      defaultZone: http://eureka9003.com:9003/eureka/,http://eureka9002.com:9002/eureka/,http://eureka9001.com:9001/eureka/
```

4.Ribbon负载均衡简单

consumer配置好了Ribbon

负载均衡实际是根据RestTemplate根据均衡算法进行调度不同地址上的同一个微服务的部署。所以修改ConfigBean，在RestTemplate上加@LoadBalanced注解。

```
@Configuration
public class ConfigBean {

    @Bean
    @LoadBalanced
    public RestTemplate getRestTemplate(){
        return new RestTemplate();
    }
}
```

复制

3、通过Ribbon的核心组件IRule定义查找消费端调用提供端微服务的策略
如没有指定轮询策略，默认是消费端随机调用提供端微服的策略，下面指定轮询调用策略。只需要在microservice-consumer中的ConfigBean类添加如下声明：

```
    @Bean
    public IRule myRule(){
        return new RoundRobinRule();    //轮询策略
    }
```

复制

5.Feign负载均衡

Feign是一个声明式WebService客户端。使用Feign能让编写Web Service客户端更加简单。Feign是对Ribbon的包装，Feign集成了Ribbon。

Feign = Ribbon + RestTemplate
controller调用service方法 接口加上注解
@FeignClient(value="microservicecloud-provider")
启动类FeignConsumerApplication加上
@EnableFeignClients(basePackages="com.lzj.springcloud.service")
扫描service包

6.Hystrix断路器

Hystrix是一个用于处理分布式系统的延迟和容错的开源库，在分布式系统里，许多依赖不可避免的会调用失败，比如超时、异常等，Hystrix能够保证在一个依赖出问题的情况下，不会导致整体服务失败，避免级联故障，以提高分布式系统的弹性。
当某个服务单元发生故障之后，通过断路器的故障监控，向调用方返回一个符合预期的、可处理的备选响应（FallBack），而不是长时间的等待或者抛出调用方无法处理的异常，这样就保证了服务调用方的线程不会被长时间、不必要地占用，从而避免了故障在分布式系统中的蔓延，乃至雪崩。

Hystrix可用于服务熔断、服务降级、服务限流等作用。

服务熔断：

@HystrixCommand(fallbackMethod="hystrixGetUser") //一旦服务调用失败，就调用hystrixGetUser方法---------自定义hystrixGetUser方法-------provide

启动类HystrixProviderApplication上添加注解@EnableCircuitBreaker;

服务降级：

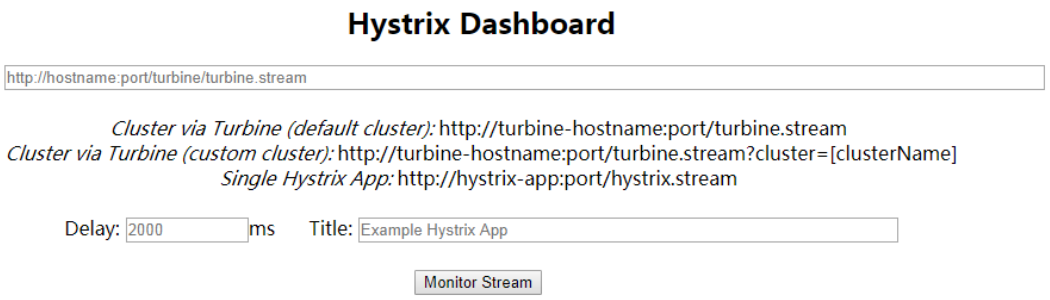
例如在分布式系统中有A、B两个服务，因为资源有限，需要关掉B服务，A服务在调用B服务时，没有调通，此时A返回指定的错误信息，注意不是在B服务端返回的，是A客户端返回的错误信息。---------返回的是客户端自定义的错误信息----consumer

服务监控：

Hystrix会持续地记录所有通过Hystrix发起的请求的执行信息，并以统计报表和图形的形式展示给用户，包括每秒执行多少请求多少成功，多少失败等

创建microservice-consumer-hystrix-dashbord微服务

输入：http://localhost:7002/hystrix 打开



输入http://localhost:8005/hystrix.stream点击Monitor stream

监听的是这个域名下的所有请求

7. Zuul路由

Zuul路由包含了对请求的路由和过滤两个功能。

路由：路由功能负责将外部请求转发到具体的微服务实例上，是实现外部访问统一入口；

过滤：过滤器功能则负责对请求的处理过程进行干预，是实现请求校验、服务聚合等功能的基础。

路由配置：http://localhost:8002/get/2 到 http://localhost:6001/springcloud-provider/get/2

http://localhost:6001为zull可配置的域名

修改服务代理名称：

```
zuul:
  routes:
    mydept.serviceId: springcloud-provider
    mydept.path: /provider/**
```

为了不暴露原来的服务名springcloud-provider

忽略带真实服务名的请求：

ignored-services: springcloud-provider 不能再通过真实的服务名

设置访问前缀：prefix: /MyDemo

8.Config配置