一、ribbon

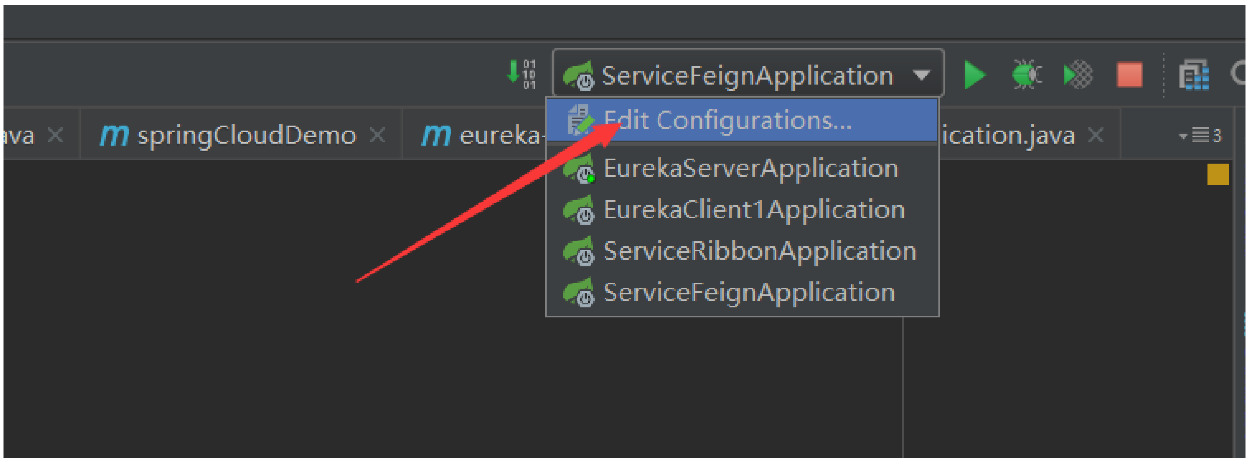
ribbon是一个负载均衡客户端，可以很好的控制htt和tcp的一些行为。Feign默认集成了ribbon。

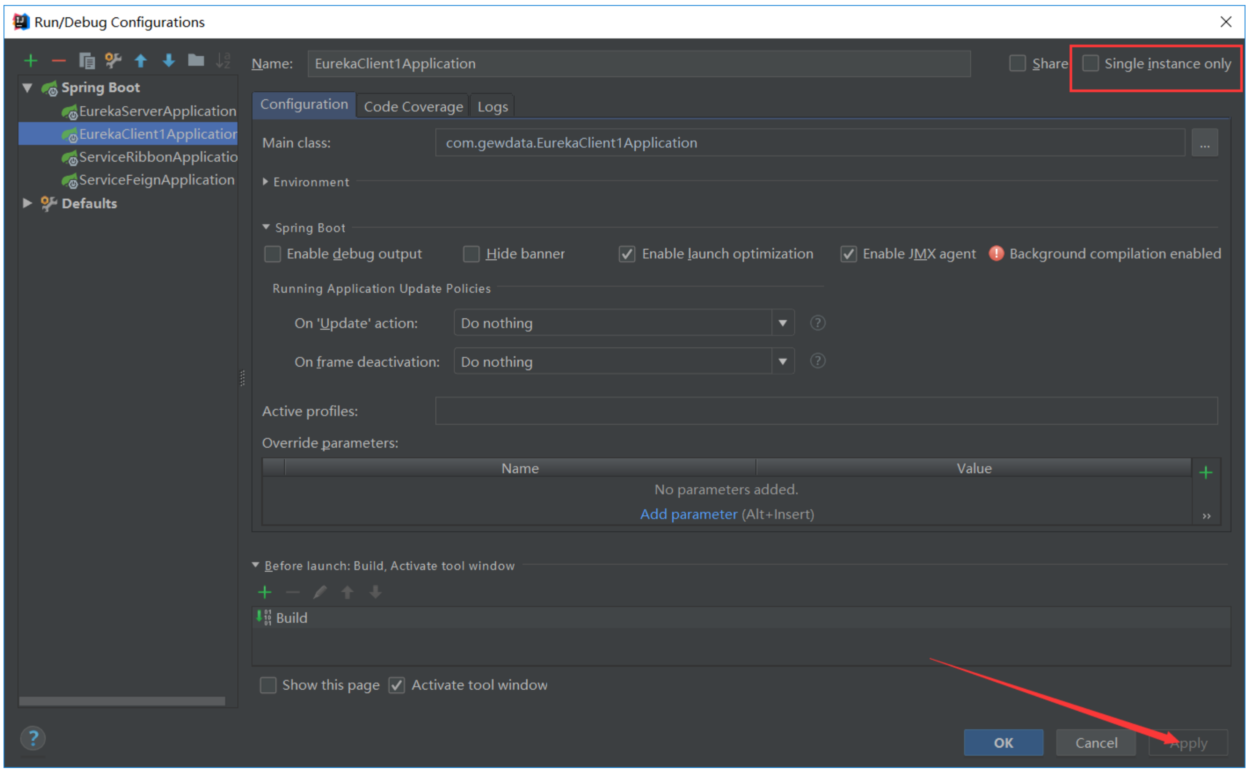
准备工作

1.启动eureka-server；

2.启动eureka-client1，端口为8762；

3.将eureka-client1配置文件端口改为8763并启动，这时候eureka-client1在eureka-client1注册了两个实例，相当于一个小集群。idea启动多个Spring Boot工程实例：





Name: EurekaClient1Application

☐ Share ☐ Single instance only

Main class: com.gewdata.EurekaClient1Application

Spring Boot

☐ Enable debug output ☐ Hide banner ☑ Enable launch optimization ☑ Enable JMX agent Background compilation enabled

Running Application Update Policies

On 'Update' action: Do nothing

On frame deactivation: Do nothing

Active profiles:

Override parameters:

Name	Value
No parameters added.	

Add parameter (Alt+Insert)

Before launch: Build, Activate tool window

Build

☐ Show this page ☑ Activate tool window

创建一个服务消费者

1.新建一个spring boot工程，取名为service-ribbon；

2.在他的pom.xml继承父pom文件件，引入一下依赖：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.gewdata</groupId>
 <artifactId>service-ribbon</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <packaging>jar</packaging>

  <name>service-ribbon</name>
  <description>Demo project for Spring Boot</description>

  <parent>
  <groupId>com.gewdata</groupId>
  <artifactId>eureka-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  </parent>

  <dependencies>
  <dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>

  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
  </dependency>
  </dependencies>
 </project>
```

3.配置文件application.yml如下：

```yaml
eureka:
  client:
  serviceUrl:
  defaultZone: http://localhost:8761/eureka/

server:
  port: 8764

spring:
  application:
  name: service-ribbon
```

4.在工程的启动类中,通过@EnableDiscoveryClient向服务中心注册；并且向程序的ioc注入一个bean: restTemplate;并通过@LoadBalanced注解表明这个restRemplate开启负载均衡的功能：

```java
package com.gewdata;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.hystrix.EnableHystrix;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
@EnableEurekaClient
@EnableDiscoveryClient
public class ServiceRibbonApplication {

 public static void main(String[] args) {
 SpringApplication.run(ServiceRibbonApplication.class, args);
 }

 @Bean
 @LoadBalanced // 通过@LoadBalanced注解表明这个restRemplate开启负载均衡的功能
 RestTemplate restTemplate() {
 return new RestTemplate();
 }
}
```

4.写一个测试类HelloService，通过之前注入ioc容器的restTemplate来消费service-hi服务的“/hi”接口，在这里我们直接用的程序名替代了具体的url地址，在ribbon中它会根据服务名来选择具体的服务实例，根据服务实例在请求的时候会用具体的url替换掉服务名，代码如下：

```java
package com.gewdata.service;

import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

/**
 * @author: JunYaoWang
  * @create: 2018-12-05 11:21
  **/
@Service
public class HelloService {

  @Autowired
  private RestTemplate restTemplate;

  public String hiService(String name) {
  // 直接用的程序名替代了具体的url地址，在ribbon中它会根据服务名来选择具体的服务实例，根据服务实例在请求的时候会用具体的url替换掉服务名
  return restTemplate.getForObject("http://eureka-client1/hi?name="+name,String.class);
  }
}
```

5.写一个controller，在controller中用调用HelloService 的方法，代码如下：

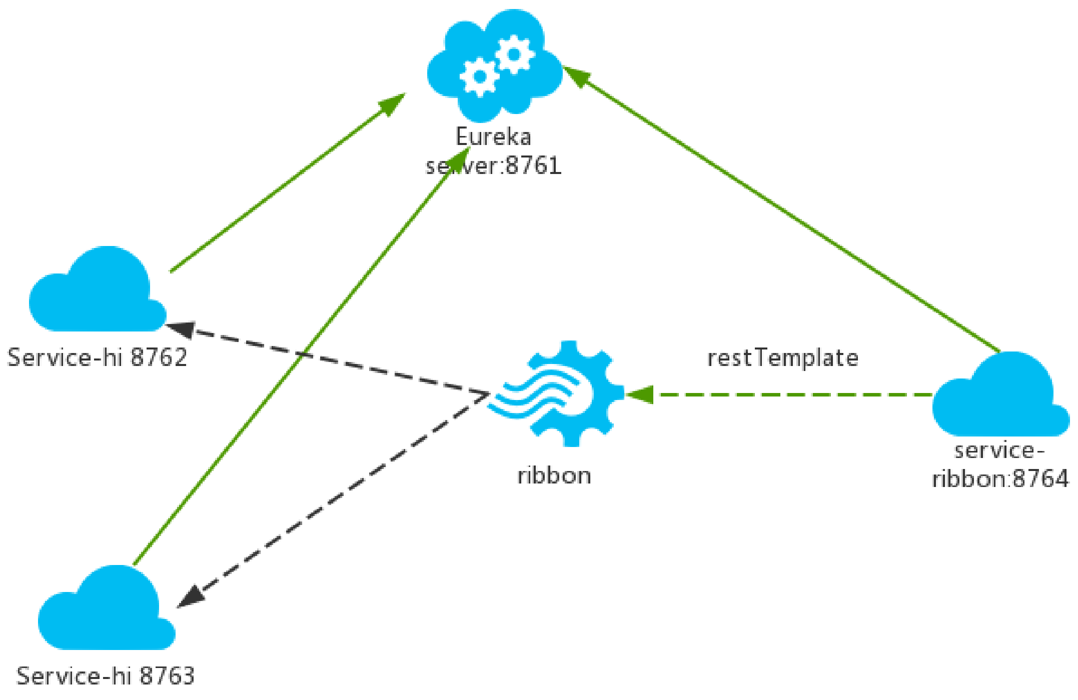
```java
package com.gewdata.controller;

import com.gewdata.service.HelloService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

/**
  * @author: JunYaoWang
  * @create: 2018-12-05 11:22
```

```java
 **/
@RestController
public class HelloController {

 @Autowired
 HelloService helloService;

 @GetMapping(value = "/hi")
 public String hi(@RequestParam String name) {
 return helloService.hiService( name );
 }
}
```

6.在浏览器上多次访问http://localhost:8764/hi?name=gewdata，浏览器交替显示：

```
hi gewdata,i am from port:8762
hi gewdata,i am from port:8763
```

7.架构如下：



- 一个服务注册中心，eureka server,端口为8761
- eureka-client1工程跑了两个实例，端口分别为8762,8763，分别向服务注册中心注册
- sercvice-ribbon端口为8764,向服务注册中心注册

- 当sercvice-ribbon通过restTemplate调用service-hi的hi接口时，因为用ribbon进行了负载均衡，会轮流的调用eureka-client1：8762和8763 两个端口的hi接口；

二、Feign

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。

准备工作

1.启动eureka-server；
2.启动eureka-client1，端口为8762；
3.将eureka-client1配置文件端口改为8763并启动，这时候eureka-client1在eureka-client1注册了两个实例，相当于一个小集群。

创建一个feign服务

1.新建一个spring boot工程，取名为serice-feign，pom.xml如下：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.gewdata</groupId>
 <artifactId>service-feign</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <packaging>jar</packaging>

  <name>service-feign</name>
  <description>Demo project for Spring Boot</description>

  <parent>
  <groupId>com.gewdata</groupId>
  <artifactId>eureka-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  </parent>

```

```
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
</dependencies>
</project>
```

2.在工程的配置文件application.yml文件，指定程序名为service-feign，端口号为8765，服务注册地址为http://localhost:8761/eureka/ ，代码如下：

```
eureka:
 client:
 serviceUrl:
 defaultZone: http://localhost:8761/eureka/

server:
 port: 8765

spring:
 application:
 name: service-feign
```

3.在程序的启动类ServiceFeignApplication ，加上@EnableFeignClients注解开启Feign的功能：

```
package com.gewdata;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
 @EnableEurekaClient
```

```java
@EnableDiscoveryClient
@EnableFeignClients
public class ServiceFeignApplication {

 public static void main(String[] args) {
 SpringApplication.run(ServiceFeignApplication.class, args);
 }
}
```

4.定义一个feign接口，通过@ FeignClient（“服务名”），来指定调用哪个服务。比如在代码中调用了service-hi服务的“/hi”接口，代码如下：

```java
package com.gewdata.service;

import com.gewdata.service.impl.SchedualServiceHiHystric;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

/**
  * @author: JunYaoWang
  * @create: 2018-12-05 15:26
  *
  * 定义一个feign接口，通过@ FeignClient（“服务名”），
  * 来指定调用哪个服务。比如在代码中调用了eureka-client1服务的“/hi”接口
  **/
 @FeignClient(value = "eureka-client1")
 public interface SchedualServiceHi {

  @RequestMapping(value = "/hi",method = RequestMethod.GET)
  String sayHiFromClientOne(@RequestParam(value = "name") String name);
 }
```

5.在Web层的controller层，对外暴露一个"/hi"的API接口，通过上面定义的Feign客户端SchedualServiceHi 来消费服务。代码如下：

```java
package com.gewdata.controller;

import com.gewdata.service.SchedualServiceHi;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
```

```java

/**
 * @author: JunYaoWang
 * @create: 2018-12-05 15:31
 **/
@RestController
public class HiController {

 //编译器报错，无视。 因为这个Bean是在程序启动的时候注入的，编译器感知不到，所以报错。
 @Autowired
 SchedualServiceHi schedualServiceHi;

 @GetMapping(value = "/hi")
 public String sayHi(@RequestParam String name) {
 return schedualServiceHi.sayHiFromClientOne( name );
 }

}
```
6.在浏览器上多次访问http://localhost:8764/hi?name=gewdata，浏览器交替显示：
```
hi gewdata,i am from port:8762
hi gewdata,i am from port:8763
```