链滴

SpringCloud 系列 --3. 负载均衡 SpringCloud 使用 Ribbon

作者：289306290

原文链接：https://ld246.com/article/1584002621162

来源网站：链滴

1.运行EurekaServer

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
ven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.crazyit.cloud</groupId>
    <artifactId>cloud-server</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Dalston.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

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

</project>
```

配置文件applicaiton.yml

```
server:
  port: 8761
eureka:
  client:
    registerWithEureka: false
    fetchRegistry: false
logging:
  level:
    com.netflix: INFO
```

启动类

```
package org.crazyit.cloud;

import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
```

```
@SpringBootApplication
@EnableEurekaServer
public class FirstServer {

    public static void main(String[] args) {
        new SpringApplicationBuilder(FirstServer.class).run(args);
    }

}
```

2.运行服务端

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
ven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.crazyit.cloud</groupId>
    <artifactId>cloud-provider</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Dalston.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

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

</project>
```

配置文件application.yml

```
spring:
  application:
```

```
    name: cloud-provider
eureka:
  instance:
    hostname: localhost
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
```

服务类

```
package org.crazyit.cloud;


import javax.servlet.http.HttpServletRequest;

import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class FirstController {

    @RequestMapping(value = "/person/{personId}", method = RequestMethod.GET,
            produces = MediaType.APPLICATION_JSON_VALUE)
    public Person findPerson(@PathVariable("personId") Integer personId, HttpServletRequest
equest) {
        Person person = new Person();
        person.setId(personId);
        person.setName("Crazyit");
        person.setAge(33);
        person.setMessage(request.getRequestURL().toString());
        return person;
    }
}
```

启动类

```
package org.crazyit.cloud;

import java.util.Scanner;

import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
@EnableEurekaClient
public class FirstServiceProvider {

    public static void main(String[] args) {
        // 读取控制台输入作为端口参数
```

```
        Scanner scan = new Scanner(System.in);
        String port = scan.nextLine();
        // 设置启动的服务器端口
        new SpringApplicationBuilder(FirstServiceProvider.class).properties(
                "server.port=" + port).run(args);
    }
}
```

然后运行时候分别启动端口8088,8099启动两个服务.

3. 调用端

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
ven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.crazyit.cloud</groupId>
    <artifactId>cloud-invoker</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Dalston.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-eureka</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-ribbon</artifactId>
        </dependency>
    </dependencies>

</project>
```

配置文件:里面指定了自定义的IRule 和 IPing 实现

```
server:
  port: 9000
spring:
  application:
    name: cloud-invoker
  cloud:
    loadbalancer:
      retry: false
eureka:
  instance:
    hostname: localhost
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
cloud-provider:
  ribbon:
    NFLoadBalancerRuleClassName: org.crazyit.cloud.MyRule
    NFLoadBalancerPingClassName: org.crazyit.cloud.MyPing
```

```
package org.crazyit.cloud;

import com.netflix.loadbalancer.IPing;
import com.netflix.loadbalancer.Server;

public class MyPing implements IPing {

    public boolean isAlive(Server server) {
        System.out.println("自定义Ping类，服务器信息：" + server.getHostPort());
        return true;
    }
}
```

```
package org.crazyit.cloud;

import java.util.List;

import com.netflix.loadbalancer.ILoadBalancer;
import com.netflix.loadbalancer.IRule;
import com.netflix.loadbalancer.Server;

public class MyRule implements IRule {

    private ILoadBalancer lb;

    public Server choose(Object key) {
        List<Server> servers = lb.getAllServers();
        System.out.println("这是自定义服务器定规则类，输出服务器信息：");
        for(Server s : servers) {
            System.out.println("        " + s.getHostPort());
        }
        return servers.get(0);
    }
```

```
    public void setLoadBalancer(ILoadBalancer lb) {
        this.lb = lb;
    }

    public ILoadBalancer getLoadBalancer() {
        return this.lb;
    }
}
```

启动类

```
package org.crazyit.cloud;

import org.crazyit.cloud.config.MyConfig;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class FirstInvoker {

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

主要看下面的服务类:

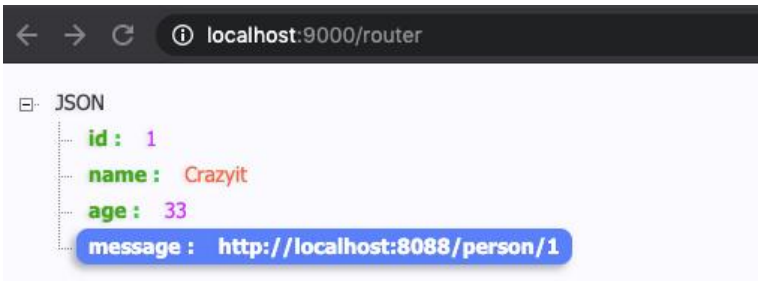
```
package org.crazyit.cloud;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.client.loadbalancer.LoadBalancerClient;
import org.springframework.cloud.client.loadbalancer.LoadBalancerRetryProperties;
import org.springframework.cloud.netflix.ribbon.SpringClientFactory;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

import com.netflix.loadbalancer.ZoneAwareLoadBalancer;

@RestController
@Configuration
public class InvokerController {

    @LoadBalanced
```

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

    @RequestMapping(value = "/router", method = RequestMethod.GET, produces = MediaTy
e.APPLICATION_JSON_VALUE)
    public String router() {
        RestTemplate restTpl = getRestTemplate();
        // 根据名称调用服务
        String json = restTpl.getForObject("http://cloud-provider/person/1",
                String.class);
        return json;
    }

    @Autowired
    private LoadBalancerClient loadBalancer;

    @RequestMapping(value = "/uselb", method = RequestMethod.GET, produces = MediaTy
e.APPLICATION_JSON_VALUE)
    public ServiceInstance uselb() {
        // 查找服务器实例
        ServiceInstance si = loadBalancer.choose("cloud-provider");
        return si;
    }

    @Autowired
    private SpringClientFactory factory;

    @RequestMapping(value = "/defaultValue", method = RequestMethod.GET, produces = M
diaType.APPLICATION_JSON_VALUE)
    public String defaultValue() {
        System.out.println("==== 输出默认配置：");
        // 获取默认的配置
        ZoneAwareLoadBalancer alb = (ZoneAwareLoadBalancer) factory
                .getLoadBalancer("default");
        System.out.println("    IClientConfig: "
                + factory.getLoadBalancer("default").getClass().getName());
        System.out.println("    IRule: " + alb.getRule().getClass().getName());
        System.out.println("    IPing: " + alb.getPing().getClass().getName());
        System.out.println("    ServerList: "
                + alb.getServerListImpl().getClass().getName());
        System.out.println("    ServerListFilter: "
                + alb.getFilter().getClass().getName());
        System.out.println("    ILoadBalancer: " + alb.getClass().getName());
        System.out.println("    PingInterval: " + alb.getPingInterval());
        System.out.println("==== 输出 cloud-provider 配置：");
        // 获取 cloud-provider 的配置
        ZoneAwareLoadBalancer alb2 = (ZoneAwareLoadBalancer) factory
                .getLoadBalancer("cloud-provider");
        System.out.println("    IClientConfig: "
                + factory.getLoadBalancer("cloud-provider").getClass()
                    .getName());
        System.out.println("    IRule: " + alb2.getRule().getClass().getName());
```

```
        System.out.println("    IPing: " + alb2.getPing().getClass().getName());
        System.out.println("    ServerList: "
                + alb2.getServerListImpl().getClass().getName());
        System.out.println("    ServerListFilter: "
                + alb2.getFilter().getClass().getName());
        System.out.println("    ILoadBalancer: " + alb2.getClass().getName());
        System.out.println("    PingInterval: " + alb2.getPingInterval());
        return "";
    }

}
```

访问:

http://localhost:9000/router 由于我们自定义的负载规则只返回第一个，所以永远是



访问http://localhost:9000/defaultValue

输出:

自定义Ping类，服务器信息：localhost:8099
自定义Ping类，服务器信息：localhost:8088
==== 输出默认配置:
IClientConfig: com.netflix.loadbalancer.ZoneAwareLoadBalancer
IRule: com.netflix.loadbalancer.ZoneAvoidanceRule
IPing: com.netflix.niws.loadbalancer.NIWSDiscoveryPing
ServerList: org.springframework.cloud.netflix.ribbon.eureka.DomainExtractingServerList
ServerListFilter: org.springframework.cloud.netflix.ribbon.ZonePreferenceServerListFilter
ILoadBalancer: com.netflix.loadbalancer.ZoneAwareLoadBalancer
PingInterval: 30
==== 输出 cloud-provider 配置:
IClientConfig: com.netflix.loadbalancer.ZoneAwareLoadBalancer
IRule: org.crazyit.cloud.MyRule
IPing: org.crazyit.cloud.MyPing
ServerList: org.springframework.cloud.netflix.ribbon.eureka.DomainExtractingServerList
ServerListFilter: org.springframework.cloud.netflix.ribbon.ZonePreferenceServerListFilter
ILoadBalancer: com.netflix.loadbalancer.ZoneAwareLoadBalancer
PingInterval: 30

可见cloud-provider配置中使用的IRule,IPing是我们已定义的实现。