在微服务系统中，我们把系统拆分成了很多小的服务，各个服务之间通过不同的方式进行依赖和调用，为了保证服务的高可用性单个服务通常会集群部署，此时，许多服务由于各种问题可能会调用失败，比如超时、异常等。如何能够保证在一个依赖出问题的情况下，不会出现故障传播，导致整体服务失败，这个就是Hystrix需要做的事情。Hystrix提供了熔断、隔离、Fallback、cache、监控等功能，能够在一个、或多个依赖同时出现问题时保证系统依然可用。

如何结合负载均衡（Ribbon），使用Hystrix来实现断路器功能。

#### 1、创建项目

新建一个springboot项目hystrix-ribbon-server1。

#### 2、引入依赖

<!--eureka client-->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

<!--hystrix-->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-hystrix</artifactId>

</dependency>

<!--eureka ribbon-->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-ribbon</artifactId>

</dependency>

<!--web-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

springcloud依赖，在标签外面：

<!--spring-cloud依赖-->

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

properties标签里设置spring-cloud版本：

<!--spring-cloud版本-->

<spring-cloud.version>Finchley.SR2</spring-cloud.version>

#### 3、修改配置文件

**server**:

**port**: 8017

**spring**:

**application**:

**name**: hystrix-ribbon-server

**eureka**:

**client**:

**service-url**:

**defaultZone**: http://localhost:8761/eureka

#### 4、添加启动注解

@EnableEurekaClient、@EnableHystrix

#### 5、在启动类中注入模板类

@EnableHystrix

@EnableEurekaClient

@SpringBootApplication

public class HystrixRibbonServer1Application {

*/\*\**

*\**

*\** ***@Bean：注入一个名为restTemplate的bean***

*\** ***@LoadBalanced：表明这个restTemplate开启了负载均衡的功能***

*\**

*\** ***@return***

*\*/*

@Bean

@LoadBalanced

RestTemplate restTemplate() {return new RestTemplate();}

public static void main(String[] args) {

SpringApplication.*run*(HystrixRibbonServer1Application.class, args);

}

}

#### 6、跨服务调用

写一套简单的业务代码，然后调用前面创建的company-server服务提供的接口。

controller：

@RestController

@RequestMapping(value = "hrs")

public class HrsController {

@Autowired

private HrsService hrsService;

@GetMapping(value = "getCompany")

public String getCompany(String id) {

String company = hrsService.getCompany(id);

System.*out*.println(company);

return company;

}

}

service：

public interface HrsService {

String getCompany(String id);

}

serviceImpl：

在实现类中，引入前面的RestTemplate模板类，来跨服务调用，并开启了负载均衡功能。

@Service

public class HrsServiceImpl implements HrsService {

*/\*\**

*\* 启动类中注入了此模板，并且开启了负载均衡功能*

*\*/*

@Autowired

RestTemplate restTemplate;

*/\*\**

*\* 给方法添加熔断器的功能，并指定熔断方法*

*\**

*\** ***@param*** *id*

*\** ***@return***

*\*/*

@HystrixCommand(fallbackMethod = "errorCompany")

@Override

public String getCompany(String id) {

//程序名替代服务地址，ribbon会根据服务名自动选择服务实例

String company = restTemplate.getForObject(

"http://company-server/company/get?id=" + id, String.class);

return company;

}

*/\*\**

*\* 熔断方法*

*\**

*\** ***@param*** *id*

*\** ***@return***

*\*/*

public String errorCompany(String id) {

return "服务出错，返回默认企业：" + id;

}

}

#### 7、启动项目，测试

此时，启动项目，访问<http://localhost:8017/hrs/getCompany?id=234>，这个项目的接口会去调用前面启动的企业服务，由于企业服务启动了两台：8011，8014，那么，通过ribbon负载均衡后，会分发到两台服务器上。

当我们把8011服务挂掉后，此时，由于eureka更新服务列表默认时间为30s，这之间，如果请求落在8011上，那么，就会调用熔断方法，返回指定的结果，如下：

