参考：

Spring Cloud构建微服务架构：服务容错保护（Hystrix服务降级）【Dalston版】

前言

标题说的高深莫测，其实就是解决通信间的错误。微服务架构中系统被拆分了很多的服务单元，各个单元通过通信去相互依赖，这个过程就有可能由于网络等等原因造成故障或延迟，调用方不会知道请求被不断积压还是不断去发起请求，最后就会瘫痪。而Spring Cloud Hystrix实现了线程隔离、断路器等一系列的服务保护功能。

服务降级demo

pom。

```
<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>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>1.5.10.RELEASE</version>
	</parent>

	<groupId>com.xyz</groupId>
	<artifactId>eureka-consumer-ribbon-hystrix</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>eureka-consumer-ribbon-hystrix</name>
	<url>http://maven.apache.org</url>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<jdk.version>1.8</jdk.version>
```

```
    </properties>

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

    <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>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <!-- 指定source和target的版本 -->
                    <source>${jdk.version}</source>
                    <target>${jdk.version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

对比之前的ribbon，这边引入了新的依赖。

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-hystrix</artifactId>
</dependency>
```

启动类的变化。

```
@EnableHystrix
@EnableDiscoveryClient
@SpringBootApplication
@ComponentScan("com.xyz.controller")
public class Application {

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

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

}
```

多了一个@EnableHystrix，就是开启服务降级。

再看controller。

```
@RestController
public class DemoController {

	@Autowired
	private ConsumerService consumerService;

	@GetMapping("/consumer")
	public String dc() {
		return consumerService.consumer();
	}

}
```

这里把具体操作移交给了ConsumerService，我们看看这个类。

```
@Service
public class ConsumerService {

	@Autowired
	private RestTemplate restTemplate;

	@HystrixCommand(fallbackMethod = "fallback")
	public String consumer() {
		String url = "http://eureka-client/test";
```

```
        return restTemplate.getForObject(url, String.class);
    }

    public String fallback() {
        return "fallback";
    }
}
```

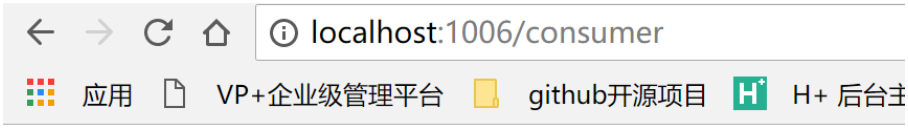
这一步终于有一些陌生的东西了。其实无非是加了@HystrixCommand注解，这个是干什么的呢？用来捕获超时异常然后调用相关方法，就如这里的fallback方法，我们在client那里加一个sleep5s（会触发hystrix的超时）。

```
@GetMapping("/test")
    public String dc() throws InterruptedException {
        Thread.sleep(5000L);
        String services = "Services: " + discoveryClient.getServices();
        System.out.println(services);
        return services;
    }
```

最后附上配置文件以及结果。

```
spring.application.name=eureka-consumer-ribbon-hystrix
server.port=1006

eureka.client.serviceUrl.defaultZone=http://localhost:1001/eureka/
```



fallback

线程隔离demo