链滴

springcloud hystrix 熔断器

作者：ws
原文链接：https://ld246.com/article/1536824569569
来源网站：链滴

<h2 id="toc_h2_0">hystrix 熔断器</h2>
<h3 id="toc_h3_1">1.Maven导入</h3>
<pre><code class="language-xml"><dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
</code></pre>
<h3 id="toc_h3_2">2.开启熔断</h3>
<blockquote>
<code>@EnableHystrix</code>激活： 单纯的功能
<code>@EnableCircuitBreaker</code>激活：<code>@ EnableHystrix</code>+springcl
ud功能
</blockquote>
<h3 id="toc_h3_3">3.对应controller增加</h3>
<pre><code class="language-java">@RestController
public class SpringCloudHystrixApplication {
    public static final Random random = new Random();
    public static void main(String[] args) {
        SpringApplication.run(SpringCloudHystrixApplication.class, args);
    }
    @GetMapping("/hello")
    @HystrixCommand(fallbackMethod="fail",commandProperties = {
            @HystrixProperty(name = "execution.isolation.thread.timeoutInMilliseconds", value =
100")
    })
    public String hello() throws InterruptedException {
        int anInt = random.nextInt(200);
        System.out.println("随机数睡眠时间："+anInt);
        Thread.sleep(anInt);
        return "hello";
    }
    public String fail(){
        return "fail";
    }
}
</code></pre>
<h3 id="toc_h3_4">4.开启监控</h3>
<h5 id="toc_h5_5">开启启动器访问权限</h5>

application.properties

<pre><code class="language-properties">management.endpoints.web.exposure.include: hys
rix.stream
</code></pre>
<ol start="2">
访问：<a href="https://ld246.com/forward?goto=http%3A%2F%2Flocalhost%3A8080%2F
ctuator%2Fhystrix.stream" target="_blank" rel="nofollow ugc">http://localhost:8080/actuator
hystrix.stream 即可

<blockquote>

注意在Application.java中使用的是<code>@EnableCircuitBreaker</code>注解
</blockquote>
<h2 id="toc_h2_6">hystrix dashboard 监控器</h2>
<h3 id="toc_h3_7">1.Maven导入</h3>
<pre><code class="language-xml"><dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
</dependency>
</code></pre>
<h3 id="toc_h3_8">2.Application.java开启注解<code>@EnableHystrixDashboard</code></h
>
<h3 id="toc_h3_9">3.访问配置</h3>

访问<a href="https://ld246.com/forward?goto=http%3A%2F%2Flocalhost%3A7070%2Fh
strix" target="_blank" rel="nofollow ugc">http://localhost:7070/hystrix
配置url：<a href="https://ld246.com/forward?goto=http%3A%2F%2Flocalhost%3A8080
2Factuator%2Fhystrix.stream" target="_blank" rel="nofollow ugc">http://localhost:8080/actu
tor/hystrix.stream

<h2 id="toc_h2_10">Turbine监控</h2>
<blockquote>
在复杂的分布式系统中，相同服务的节点经常需要部署上百甚至上千个，很多时候，运维人员希
能够把相同服务的节点状态以一个整体集群的形式展现出来，这样可以更好的把握整个系统的状态。
此，Netflix提供了一个开源项目（Turbine）来提供把多个hystrix.stream的内容聚合为一个数据源供
ashboard展示。
</blockquote>
<h3 id="toc_h3_11">1.Maven导入</h3>
<pre><code class="language-xml"><dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-turbine</artifactId>
</dependency>
</code></pre>
<h3 id="toc_h3_12">2.Application.java增加注解</h3>
<pre><code class="language-java">@SpringBootApplication
@EnableTurbine
@EnableHystrixDashboard
public class SpringCloudHystrixDashboardApplication {

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

}

</code></pre>

<h3 id="toc_h3_13">3.application.properties配置</h3>
<pre><code class="language-properties">server.port = 7070
spring.application.name=hystrix-dashboard-turbine
##消费方的名称
turbine.appConfig=eureka.client.consumer
turbine.aggregator.clusterConfig= default
turbine.clusterNameExpression= new String("default")
因为它需要去寻找服务,所以也需要注册进eureka server中
eureka.client.serviceUrl.defaultZone=http://localhost:9090/eureka/
</code></pre>

<code>turbine.appConfig</code> ： 配置Eureka中的serviceId列表，表明监控哪些服务
<code>turbine.aggregator.clusterConfig</code> ： 指定聚合哪些集群，多个使用”,”分割默认为default。可使用<code>http://.../turbine.stream?cluster={clusterConfig之一}</code>访
<code>turbine.clusterNameExpression</code> ： 1. clusterNameExpression指定集群名，默认表达式appName；此时：<code>turbine.aggregator.clusterConfig</code>需要配置想监控的应用名称；2. 当clusterNameExpression: default时，<code>turbine.aggregator.clusterCofig</code>可以不写，因为默认就是default；3. 当clusterNameExpression: metadata[‘cluster’时，假设想要监控的应用配置了<code>eureka.instance.metadata-map.cluster: ABC</code>，需要配置，同时<code>turbine.aggregator.clusterConfig: ABC</code>

<h3 id="toc_h3_14">4.分别启动项目,访问http://localost:7070/turbine.stream .</h3>