在分布式环境中，总会有些节点或者依赖的服务时效（宕机，网络延迟等）。Hystrix的出现就是来很好的处理这种可怕问题的，它里面增加了很多容错机制，比如修改请求的线程池大小、信号量等来帮助我们控制分布式系统间组件的交互。Spring Cloud将Hystrix整合到Netflix项目中，Hystrix通过隔离服务间的访问点、停止他们之间的级联故障、提供可回退操作来实现容错。

好了，概念啰嗦完了，现在你知道Hystrix大概是一个什么东东了，现在我们来玩玩代码。

本节演示的项目结构依旧是老三样，需要源码的童鞋请前往本人github下载：

*https://github.com/aharddreamer/chendong/tree/master/springcloud/hystrix-CSDN/*

项目结构：

eureka-server: 集群服务器

first-service-provider: 微服务提供者

first-service-invoker: 微服务调用者

只要这三样，我们几乎可以玩转整个Spring Cloud。

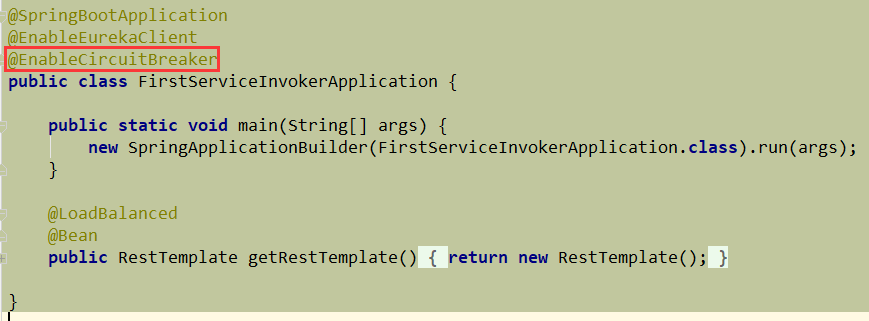
实现方式：

（不得不说Spring Cloud项目真是方便，集成一个组件就只需要三步： 加依赖，加配置，稍加代码或注解）

我们先在first-service-invoker的POM里面加入Hystrix的依赖：

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

然后在启动类加入注解@EnableCircuitBreaker启用断路器



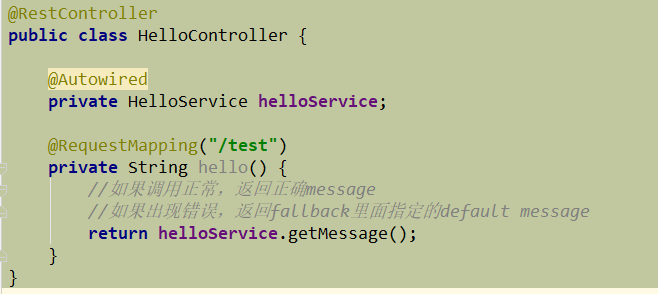
我在这里面还配置了一个restTemplate, 方便调用其他微服务。

新建一个HelloService，在里面定义一个简单的Command。

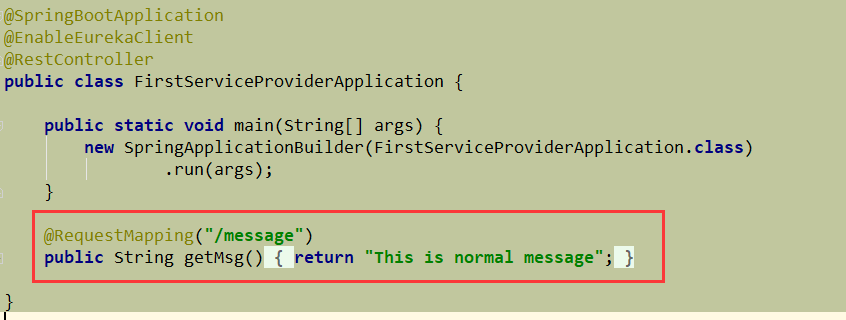


这个Command要做的事情很简单，就是当getMessage方法出错时，会执行getMessageFallback方法返回一个默认返回值。

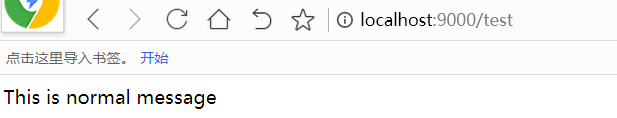
新建一个测试接口，跑跑看。



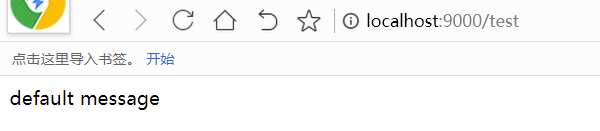
当然在first-service-provider里面要定义一个message接口供Invoker模块调用：



好了，分别启动eureka-server, first-service-provider, first-service-invoker，然后访问<http://localhost:9000/test> 试试 （service provider的端口是8080， service invoker的端口是9000）。



试着停掉first-service-provider微服务，然后再访问试试：



变成了default message

除了上述配置外，你还可以给HystrixCommand加入很多其他配置，都可以在命令注解里完成：

@HystrixCommand(fallbackMethod = "getMessageFallback", groupKey = "MyGroup",

commandKey = "MyCommandKey", threadPoolKey = "MyCommandPool",

commandProperties = {

@HystrixProperty(name = "execution.isolation.thread.timeoutInMilliseconds", value = "1000")},

threadPoolProperties = {

@HystrixProperty(name = "coreSize", value = "1")}

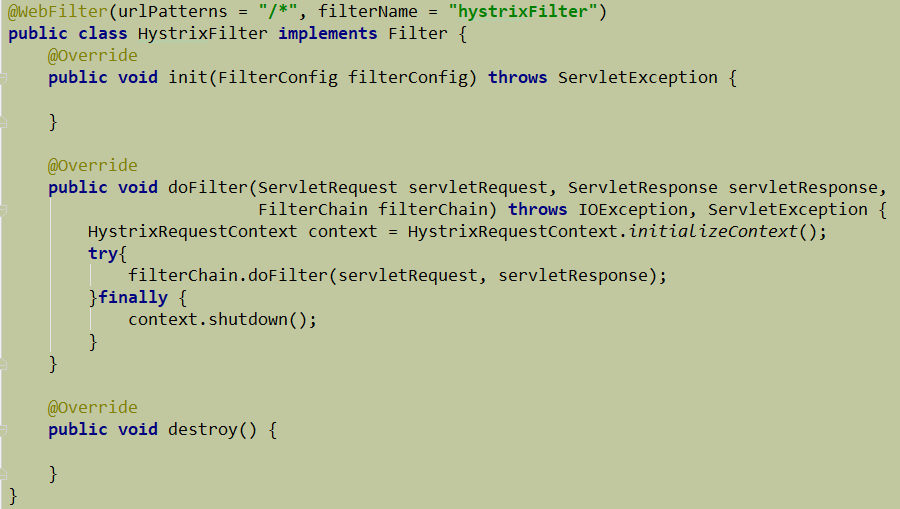
)

当然，也可以在类上面加@DefaultProperties(groupKey=”xxx”)来使用默认配置：



我们也可以使用缓存注解，这样就不用每次都去call 其他微服务了。

使用缓存注解要先加一个过滤器，在过滤器里面配置下Hystrix的context



加个注解@CacheResult

