**一、什么是微服务**

微服务的概念源于 2014 年 3 月 Martin Fowler 所写的一篇文章“Microservices”。文中内容提到：微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。

每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通（通常是基于 HTTP 的 RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立地部署到生产环境、类生产环境等。

另外，应尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。

微服务是一种架构风格，一个大型复杂软件应用由一个或多个微服务组成。系统中的各个微服务可被独立部署，各个微服务之间是松耦合的。每个微服务仅关注于完成一件任务并很好地完成该任务。在所有情况下，每个任务代表着一个小的业务能力。

## 二、微服务架构优势

**1 复杂度可控**

在将应用分解的同时，规避了原本复杂度无止境的积累。每一个微服务专注于单一功能，并通过定义良好的接口清晰表述服务边界。

由于体积小、复杂度低，每个微服务可由一个小规模开发团队完全掌控，易于保持高可维护性和开发效率。

**2 独立部署**

由于微服务具备独立的运行进程，所以每个微服务也可以独立部署。当某个微服务发生变更时无需编译、部署整个应用。

由微服务组成的应用相当于具备一系列可并行的发布流程，使得发布更加高效，同时降低对生产环境所造成的风险，最终缩短应用交付周期。

**3 技术选型灵活**

微服务架构下，技术选型是去中心化的。每个团队可以根据自身服务的需求和行业发展的现状，自由选择最适合的技术栈。

由于每个微服务相对简单，所以需要对技术栈进行升级时所面临的风险就较低，甚至完全重构一个微服务也是可行的。

**4 容错**

当某一组件发生故障时，在单一进程的传统架构下，故障很有可能在进程内扩散，形成应用全局性的不可用。

在微服务架构下，故障会被隔离在单个服务中。若设计良好，其他服务可通过重试、平稳退化等机制实现应用层面的容错。

**5 扩展**

单块架构应用也可以实现横向扩展，就是将整个应用完整的复制到不同的节点。当应用的不同组件在扩展需求上存在差异时，微服务架构便体现出其灵活性，因为每个服务可以根据实际需求独立进行扩展。

## 三、Spring Cloud 都做了哪些事

Spring Cloud 是一系列框架的有序集合，它利用 Spring Boot 的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用 Spring Boot 的开发风格做到一键启动和部署。

Spring 并没有重复制造轮子，它只是将目前各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过 Spring Boot 风格进行再封装、屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包。

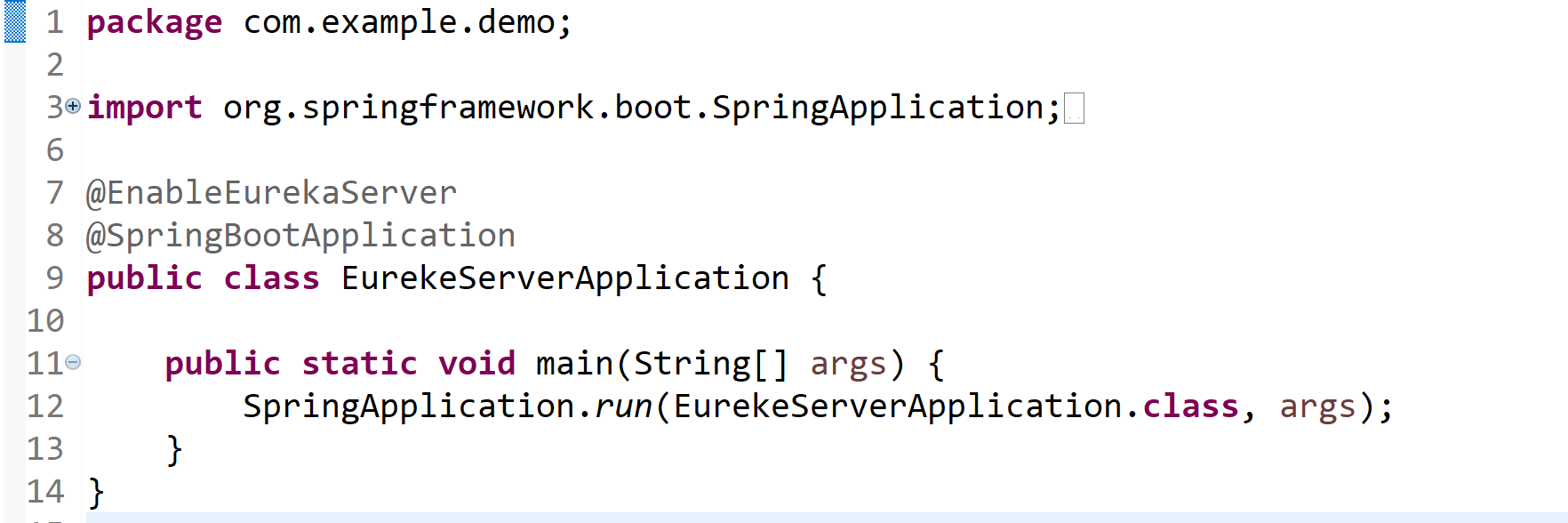
以下为 Spring Cloud 的核心功能：

* 分布式/版本化配置。
* 服务注册和发现。
* 路由。
* 服务和服务之间的调用。
* 负载均衡。
* 断路器。
* 分布式消息传递。

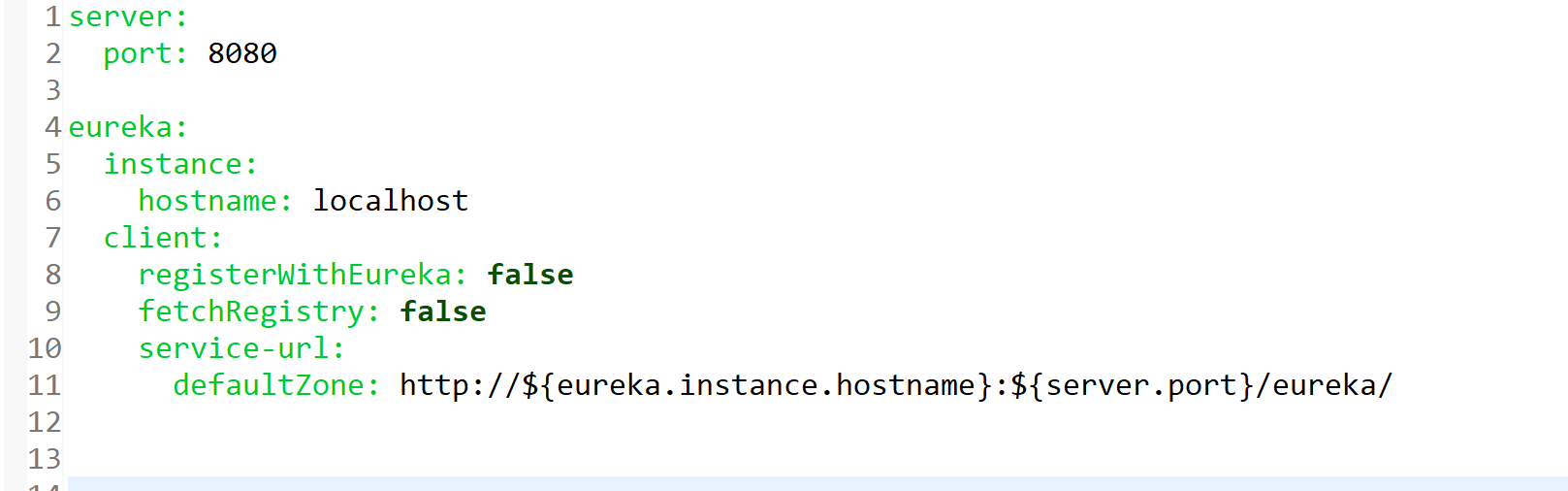
## 四、核心功能示例：

### 创建服务注册中心

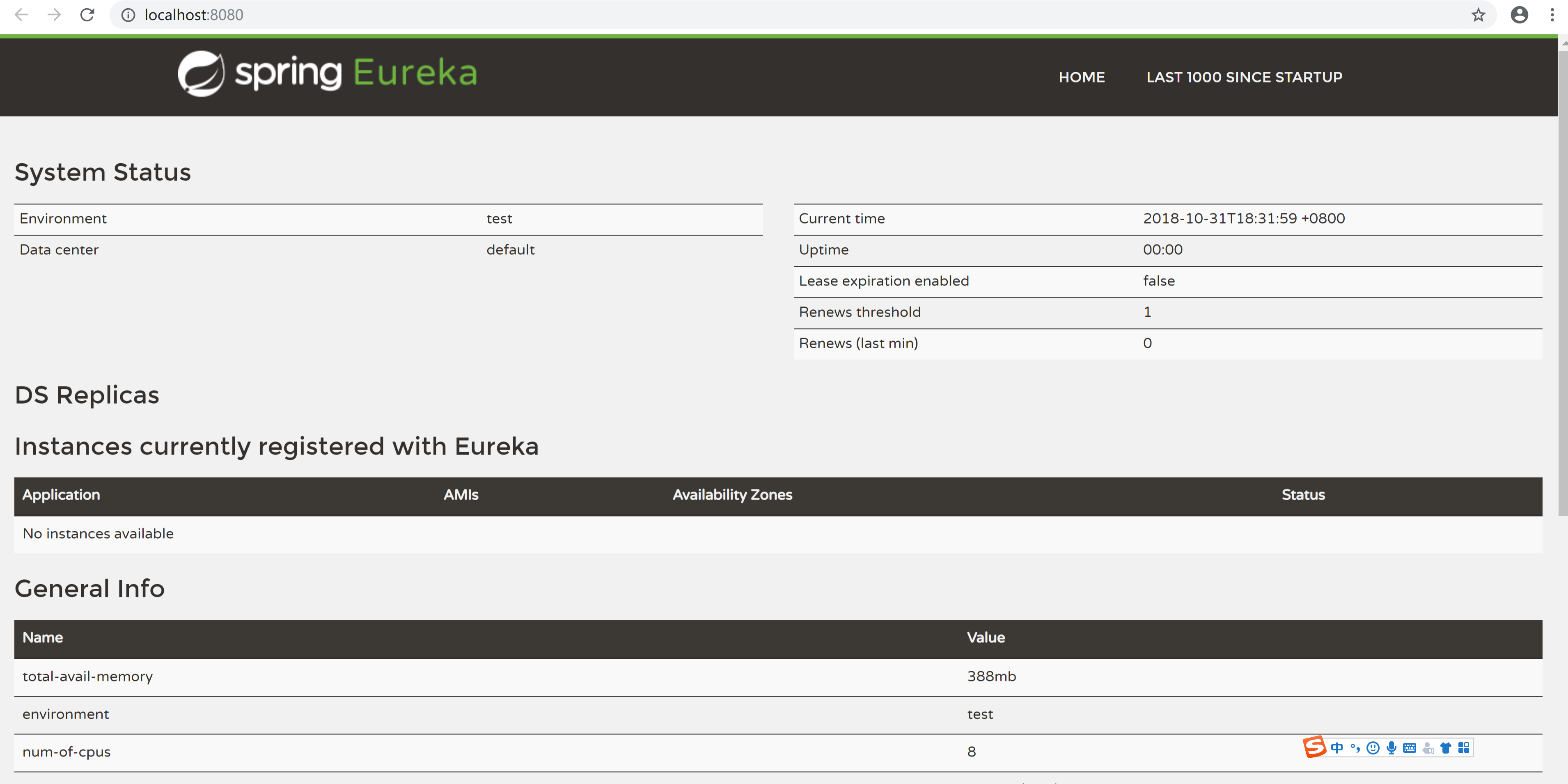
**启动一个服务注册中心**，只需要一个注解@EnableEurekaServer，这个注解需要在springboot工程的启动application类上加：



eureka是一个高可用的组件，它没有后端缓存，每一个实例注册之后需要向注册中心发送心跳（因此可以在内存中完成），在默认情况下erureka server也是一个eureka client ,必须要指定一个 server。eureka server的配置文件appication.yml



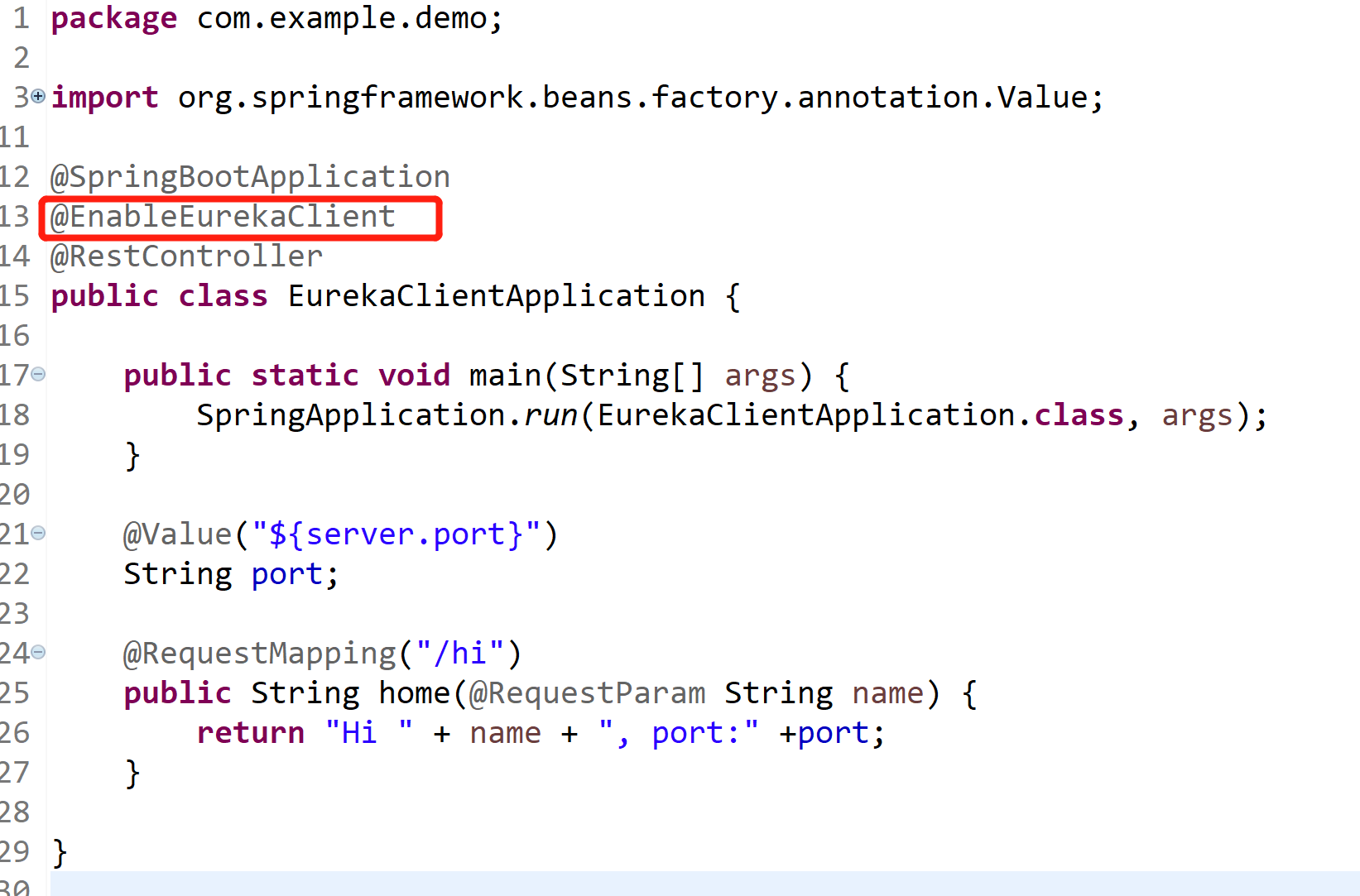
eureka server 是有界面的，启动工程,打开浏览器访问：  
http://localhost:[port] ,界面如下：



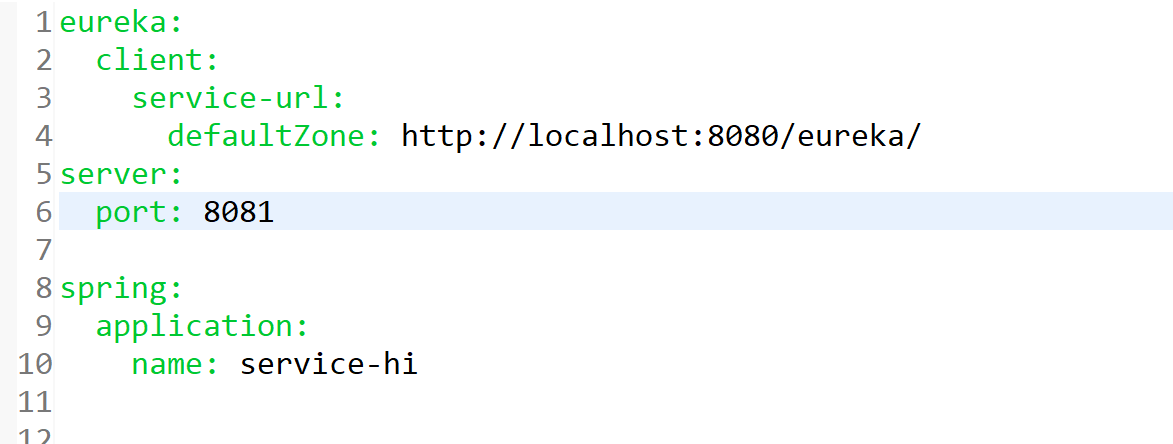
### 创建一个服务提供者 (eureka client)

当client向server注册时，它会提供一些元数据，例如主机和端口，URL，主页等。Eureka server 从每个client实例接收心跳消息。 如果心跳超时，则通常将该实例从注册server中删除。

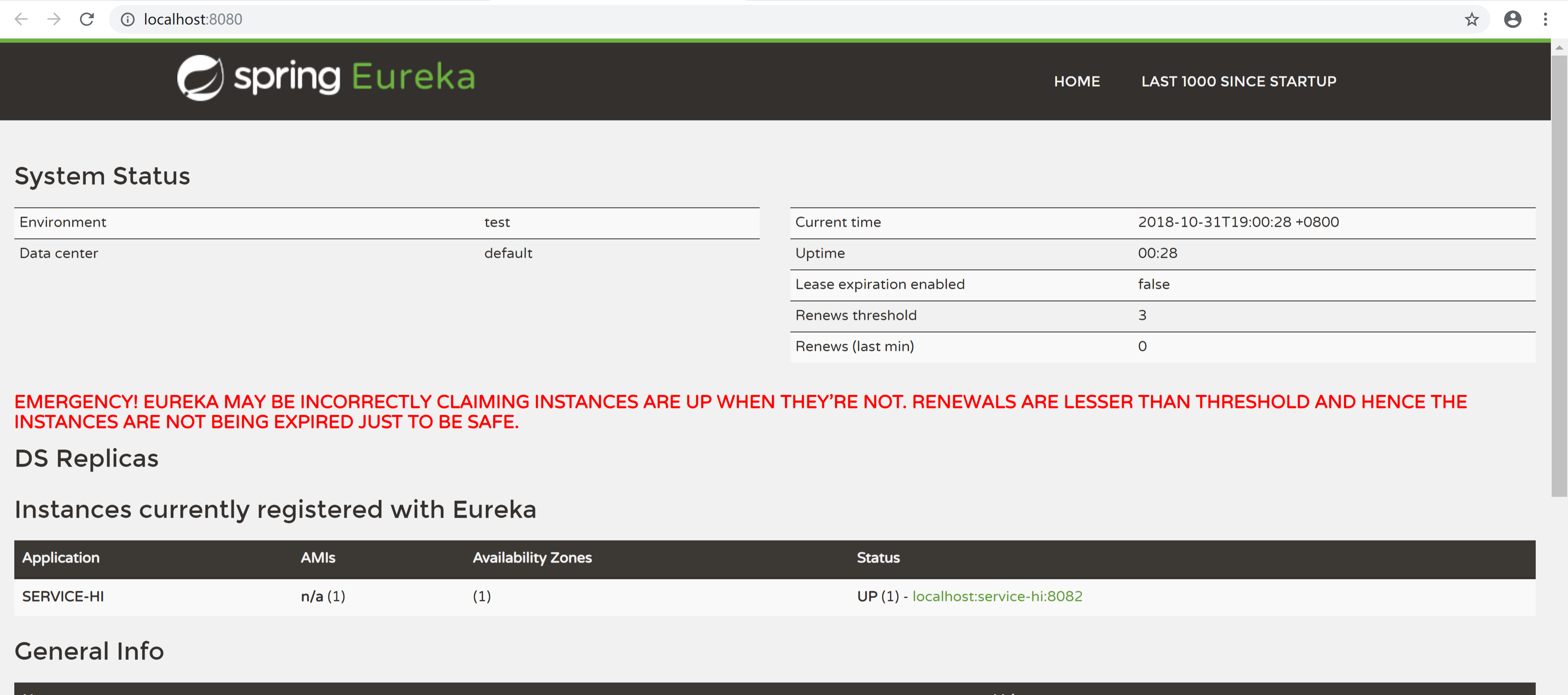
通过注解@EnableEurekaClient 表明自己是一个eurekaclient.



仅仅@EnableEurekaClient是不够的，还需要在配置文件中注明自己的服务注册中心的地址，application.yml配置文件如下：



[需要指明spring.application.name](http://xn--spring-ve8l53rst3geh2a.application.name/),这个很重要，这在以后的服务与服务之间相互调用一般都是根据这个name 。  
启动工程，打开http://localhost:8080 ，即eureka server 的网址：



在浏览器访问<http://localhost:8081/hi?name=SpringCloud>

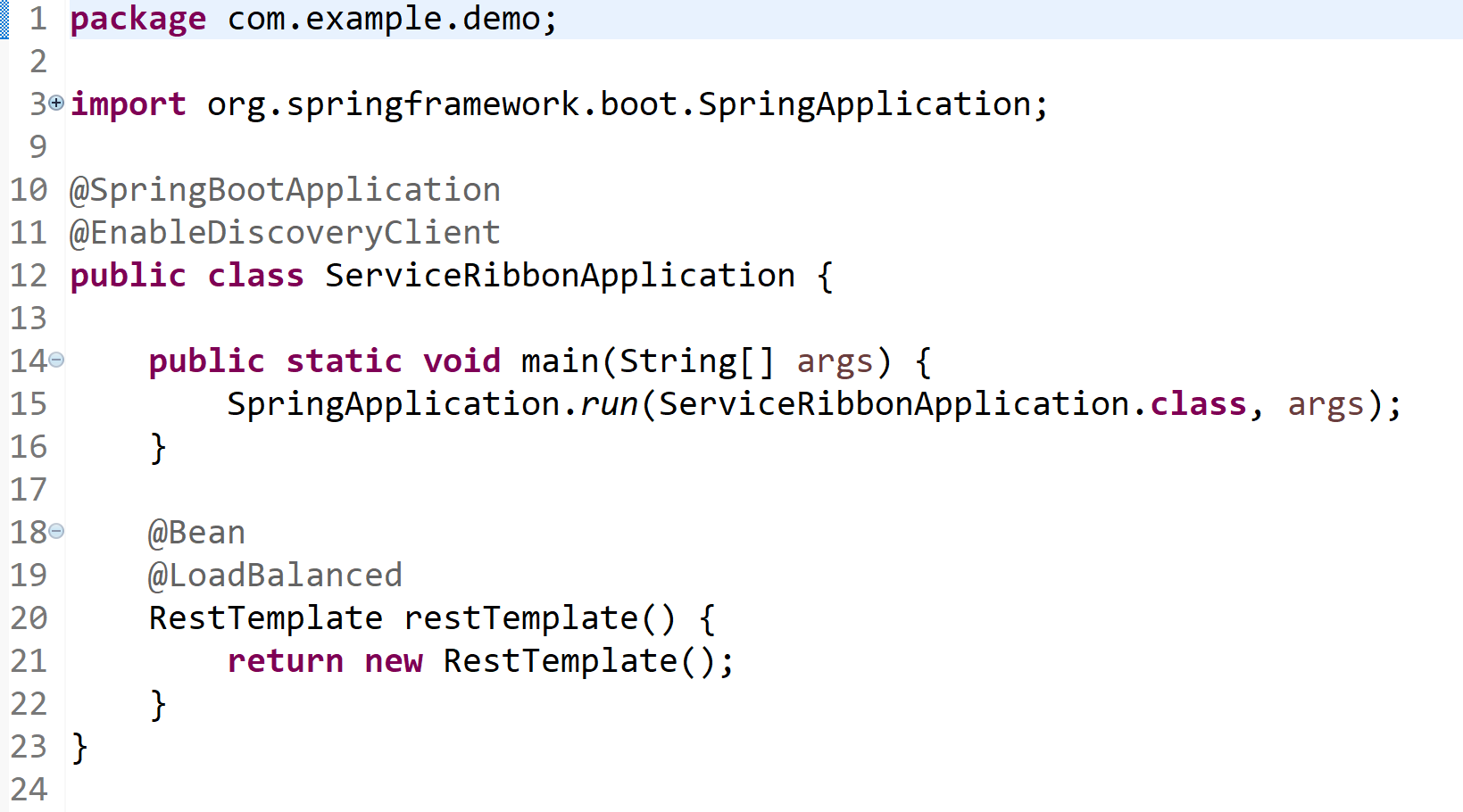
# 服务消费者

在微服务架构中，业务都会被拆分成一个独立的服务，服务与服务的通讯是基于http restful的。Spring cloud有两种服务调用方式，一种是ribbon+restTemplate，另一种是feign。

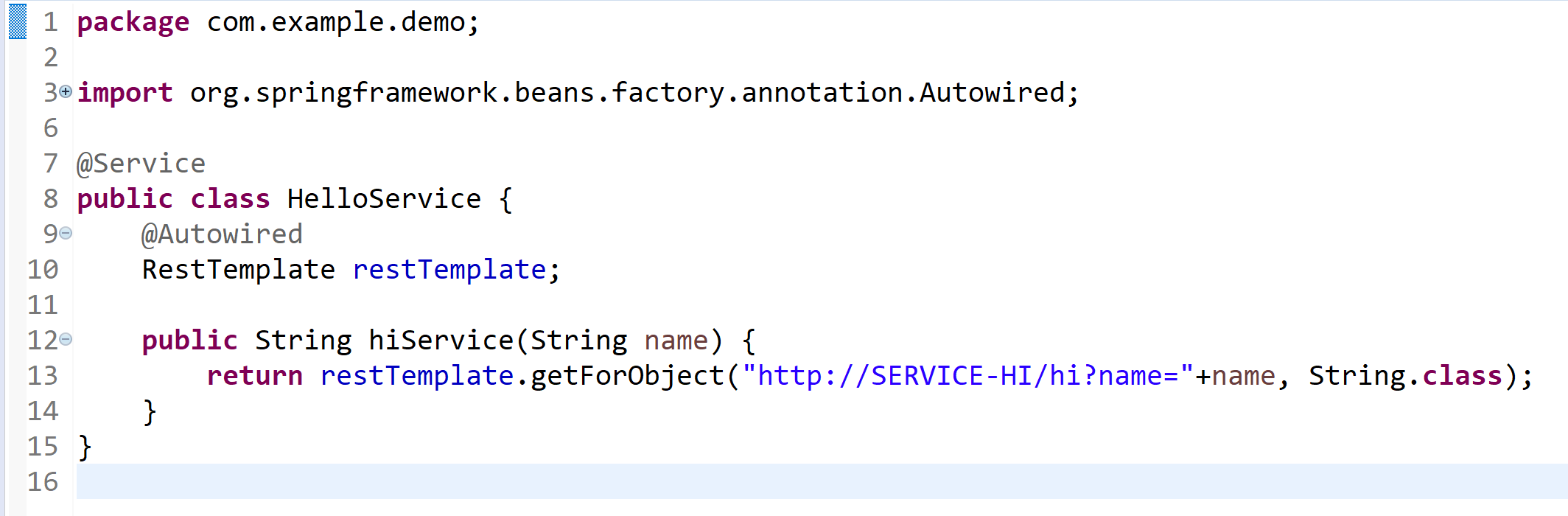
ribbon是一个负载均衡客户端，可以很好的控制http和tcp的一些行为。Feign默认集成了ribbon。

启动eureka-server 工程；启动service-hi工程，它的端口为8081；将service-hi的配置文件的端口改为8082,并启动，这时你会发现：service-hi在eureka-server注册了2个实例，这就相当于一个小的集群。

在工程的启动类中,通过@EnableDiscoveryClient向服务中心注册；并且向程序的ioc注入一个bean: restTemplate;并通过@LoadBalanced注解表明这个restRemplate开启负载均衡的功能。

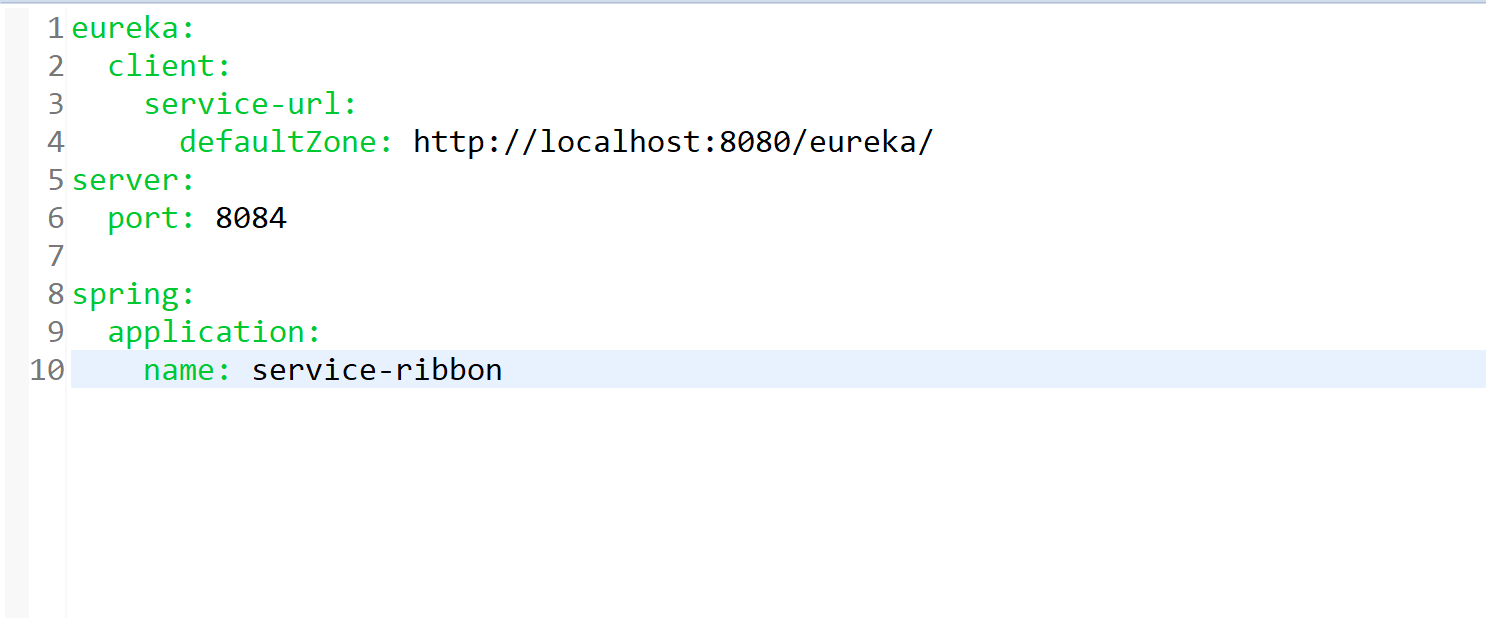


写一个测试类HelloService，通过之前注入ioc容器的restTemplate来消费service-hi服务的“/hi”接口，在这里我们直接用的程序名替代了具体的url地址，在ribbon中它会根据服务名来选择具体的服务实例，根据服务实例在请求的时候会用具体的url替换掉服务名，代码如下：



写一个controller，在controller中用调用HelloService 的方法。

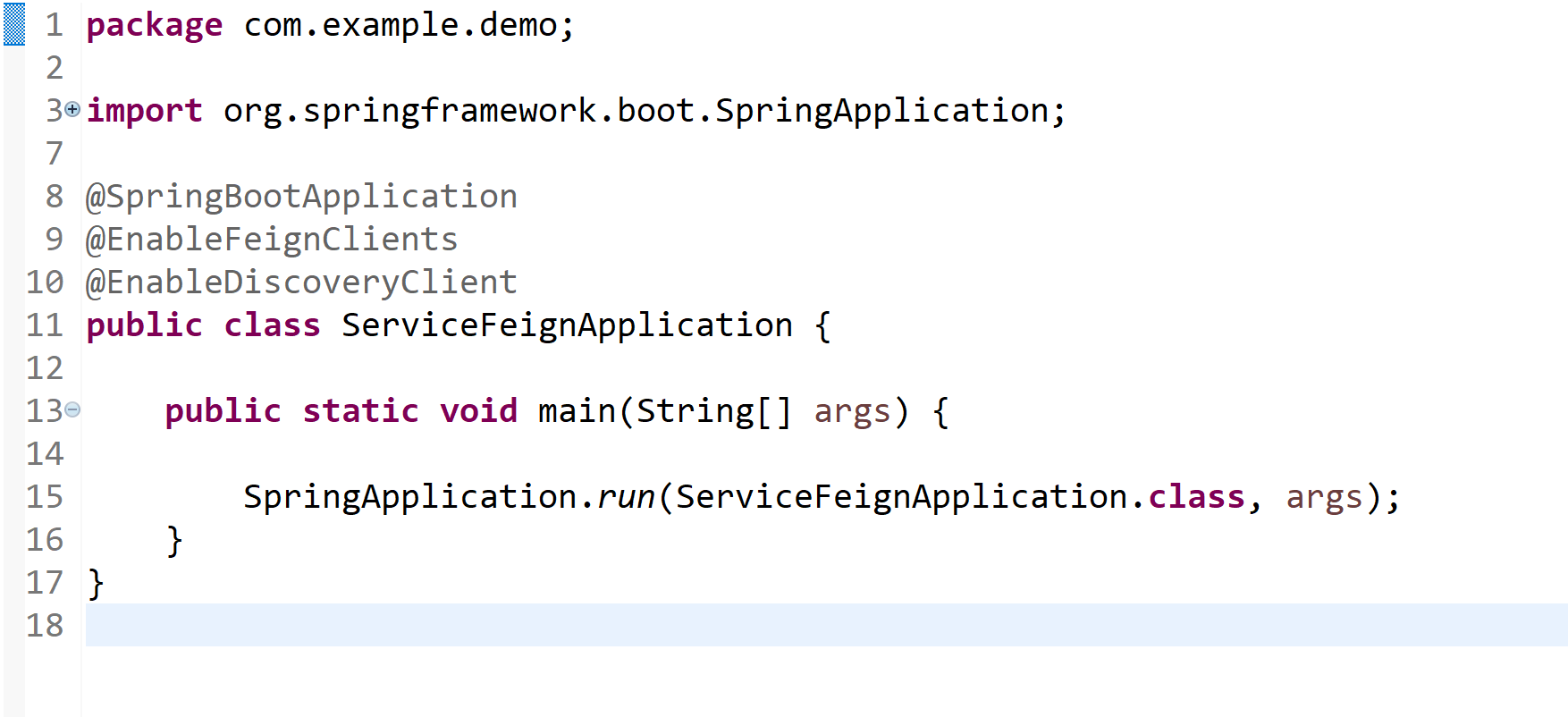
appication.yml配置：



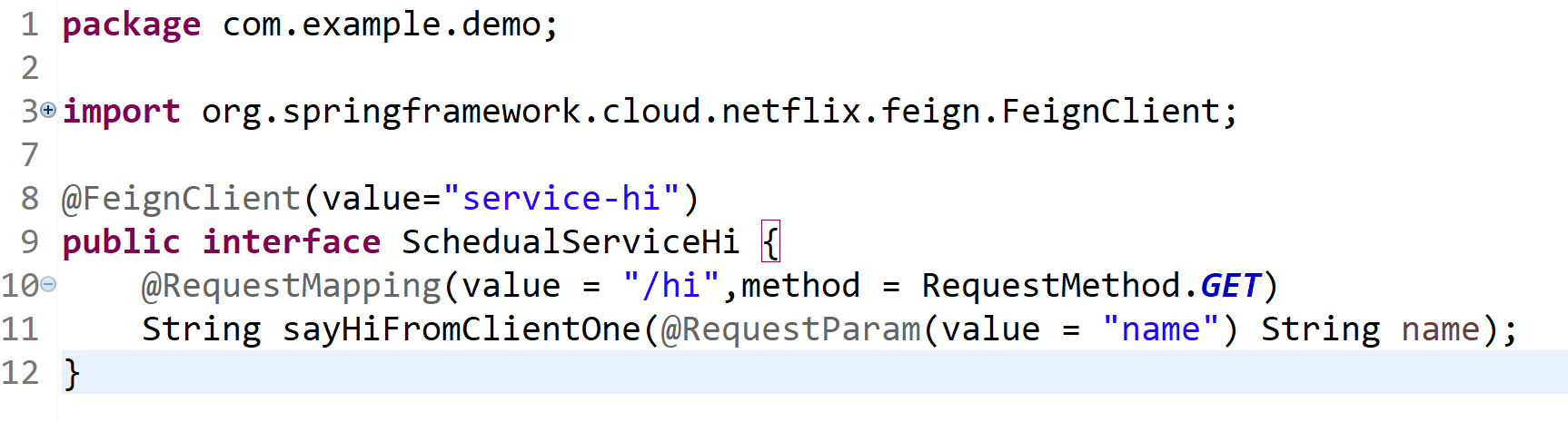
在浏览器上多次访问<http://localhost:8084/hi?name=SpringCloud> 观看效果。

### 创建一个feign的服务

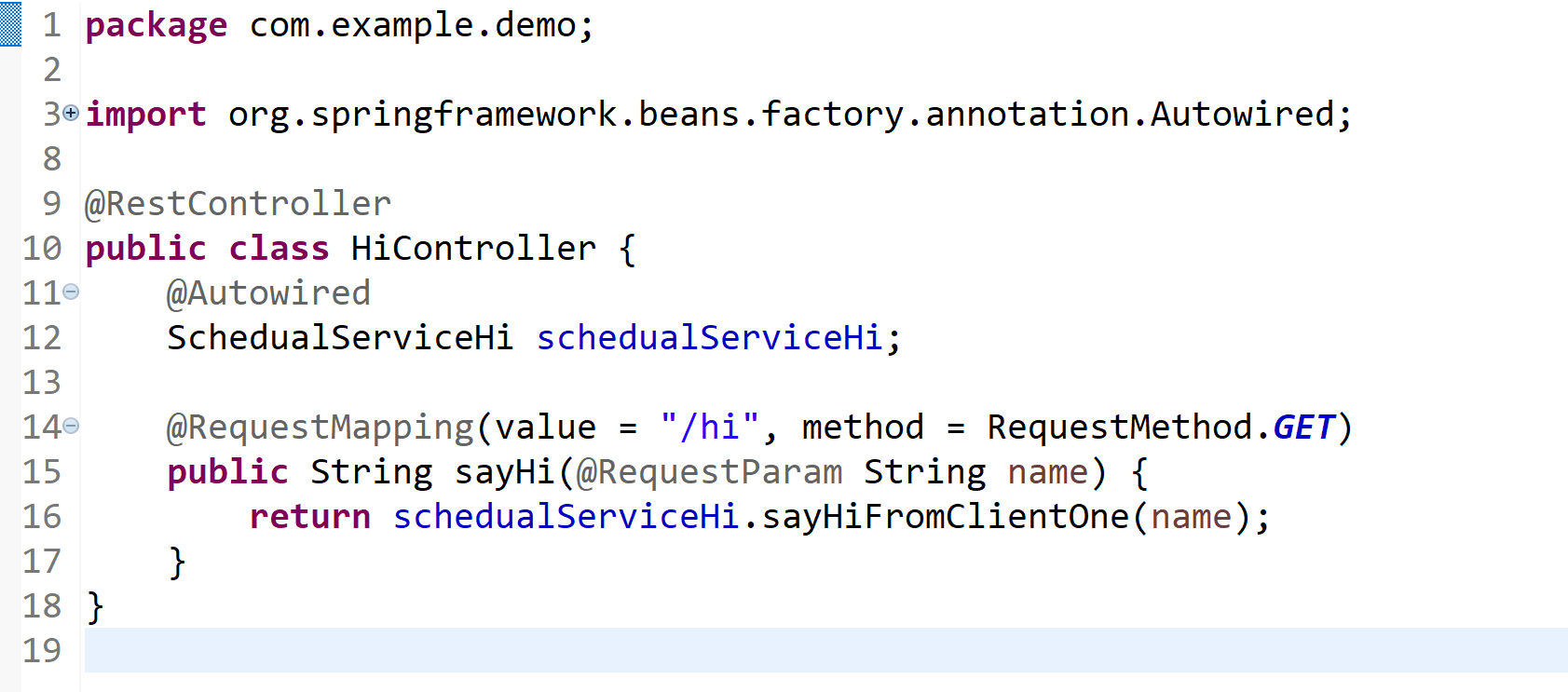
在程序的启动类ServiceFeignApplication ，加上@EnableFeignClients注解开启Feign的功能：



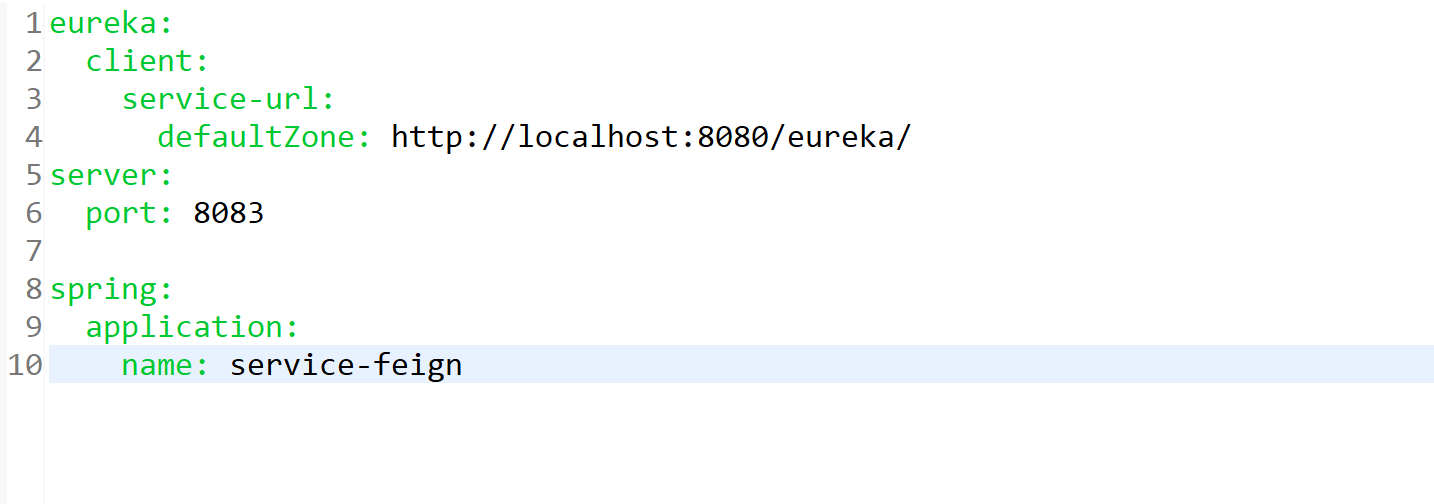
定义一个feign接口，通过@ FeignClient（“服务名”），来指定调用哪个服务。比如在代码中调用了service-hi服务的“/hi”接口，代码如下：



在Web层的controller层，对外暴露一个"/hi"的API接口，通过上面定义的Feign客户端SchedualServiceHi 来消费服务。代码如下：



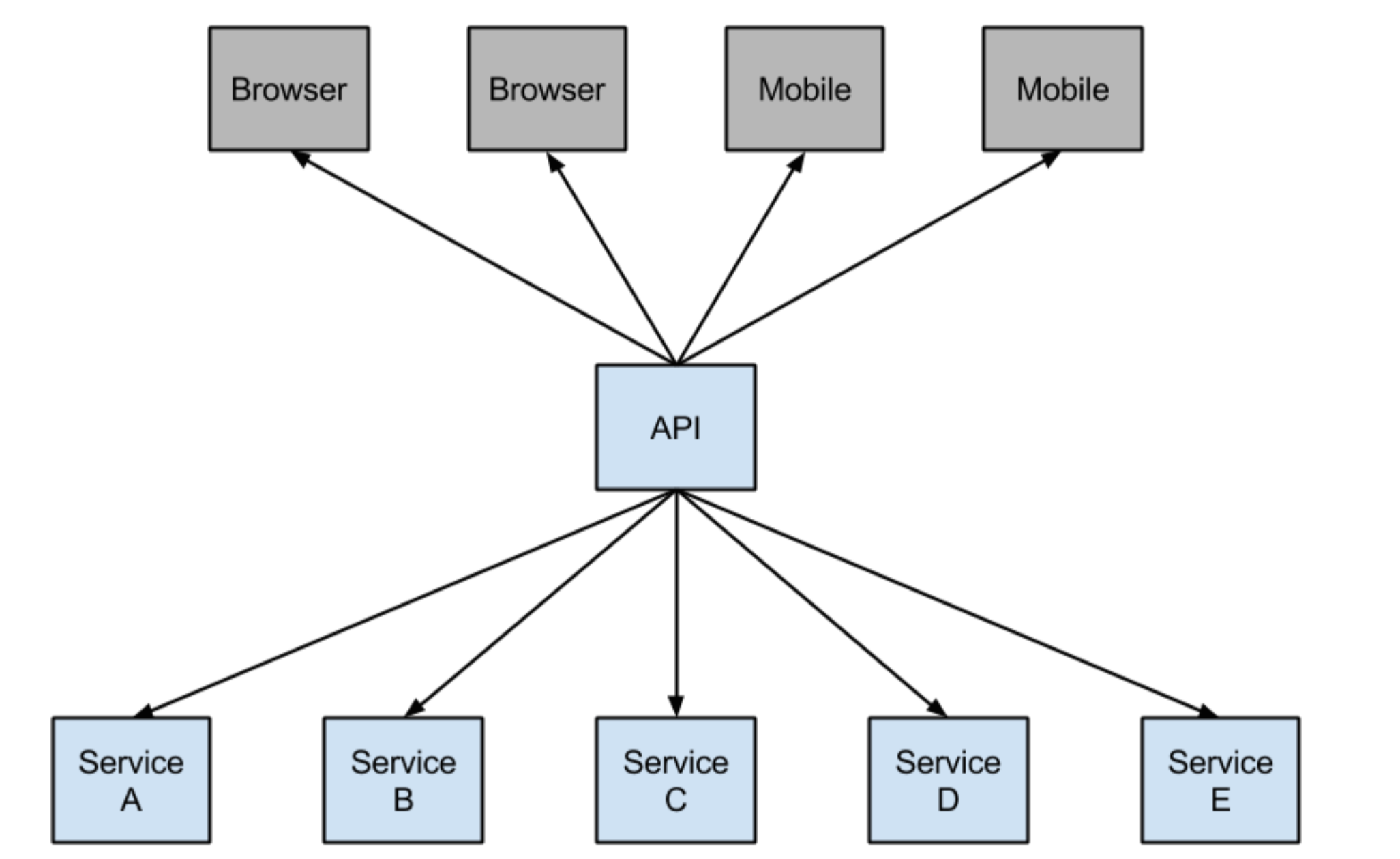
appication.yml配置：



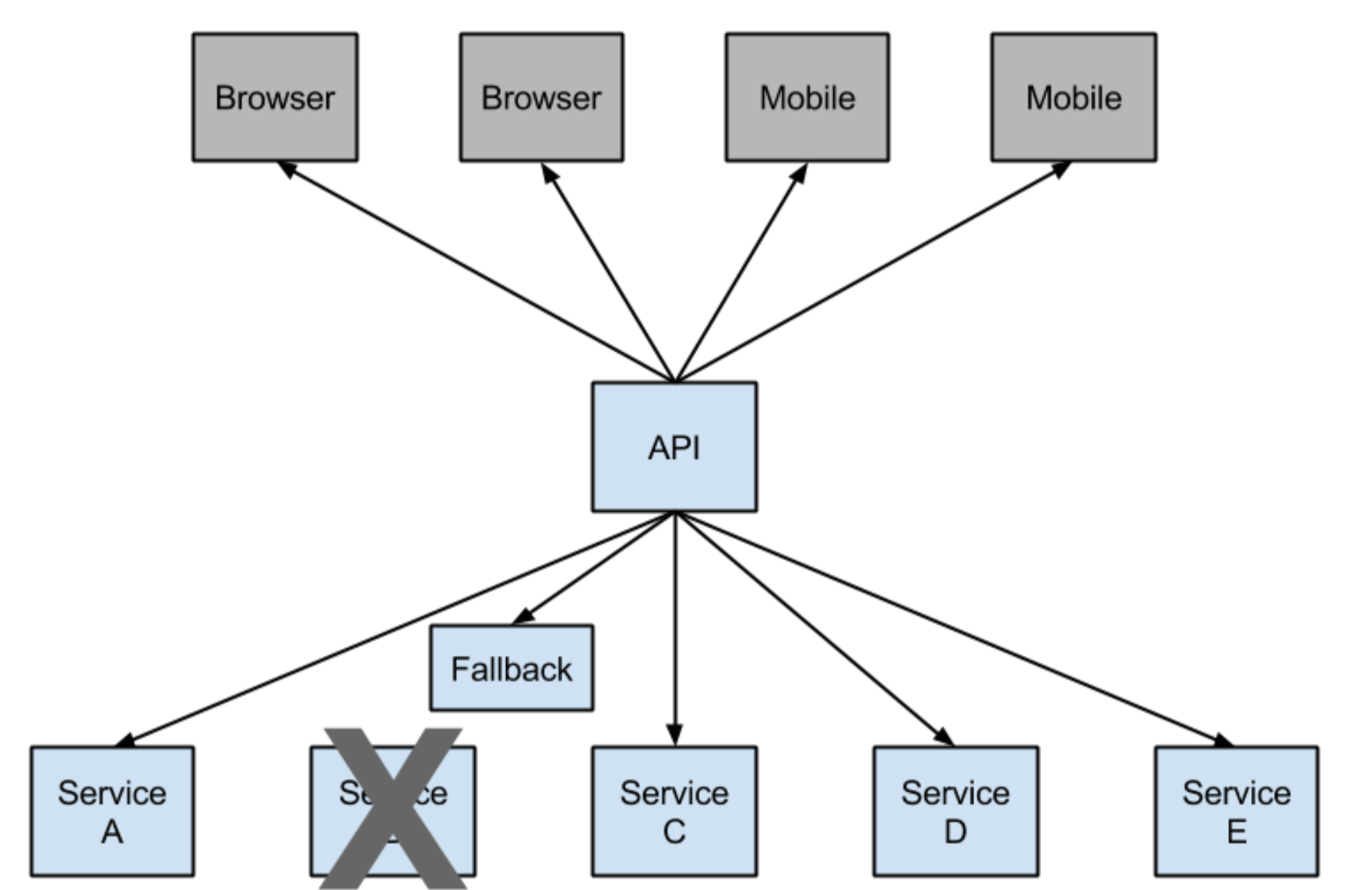
测试：<http://localhost:8083/hi?name=SpringCloud>

**如何加入Hystrix**

[Hystrix](https://github.com/Netflix/Hystrix)实现了[断路器图案](http://martinfowler.com/bliki/CircuitBreaker.html)。在微服务架构中，通常有多层服务调用。



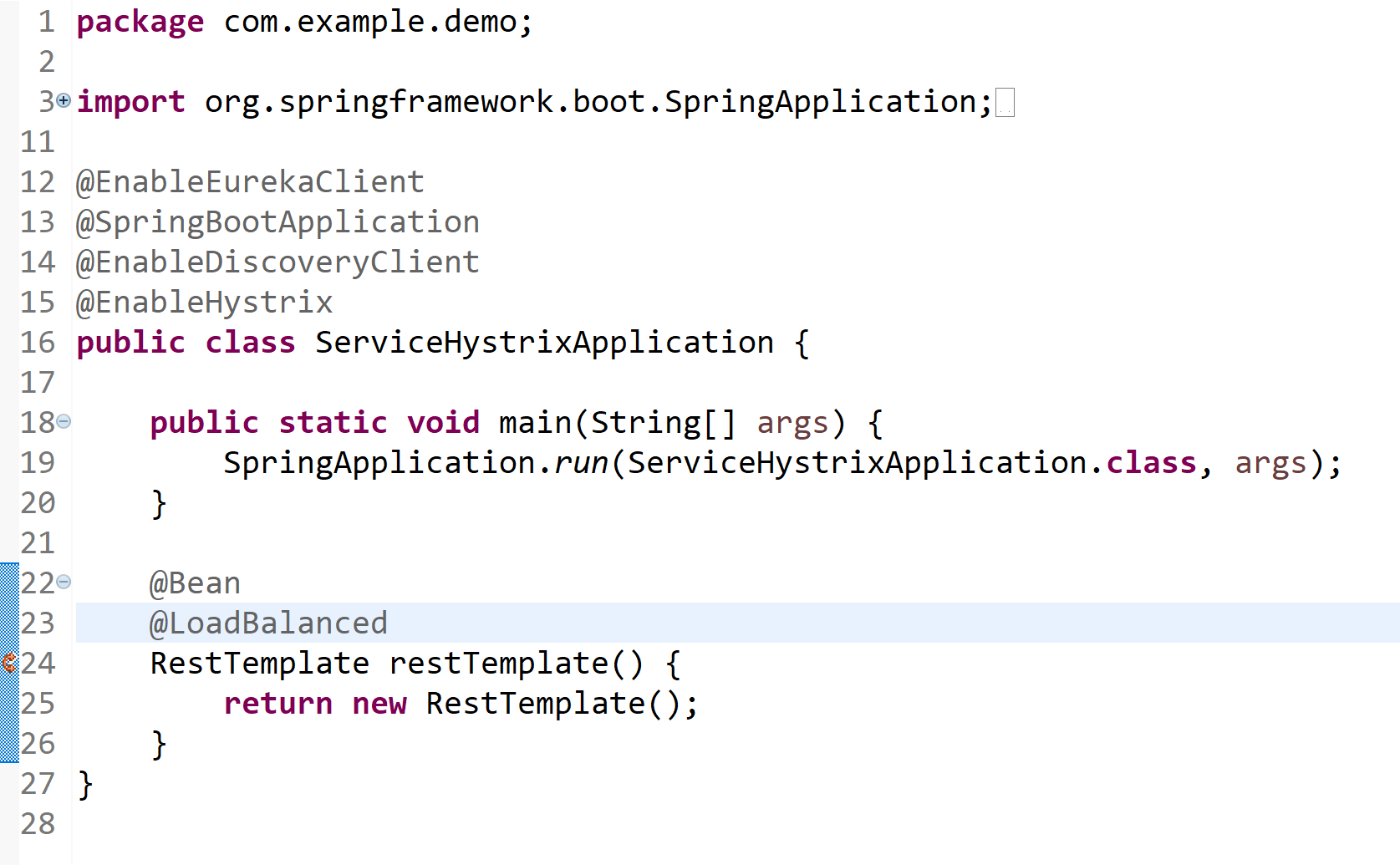
较低级别的服务中的服务故障可能导致用户级联故障。当对特定服务的呼叫达到一定阈值时（Hystrix中的默认值为5秒内的20次故障），电路打开，不进行通话。在错误和开路的情况下，开发人员可以提供后备。



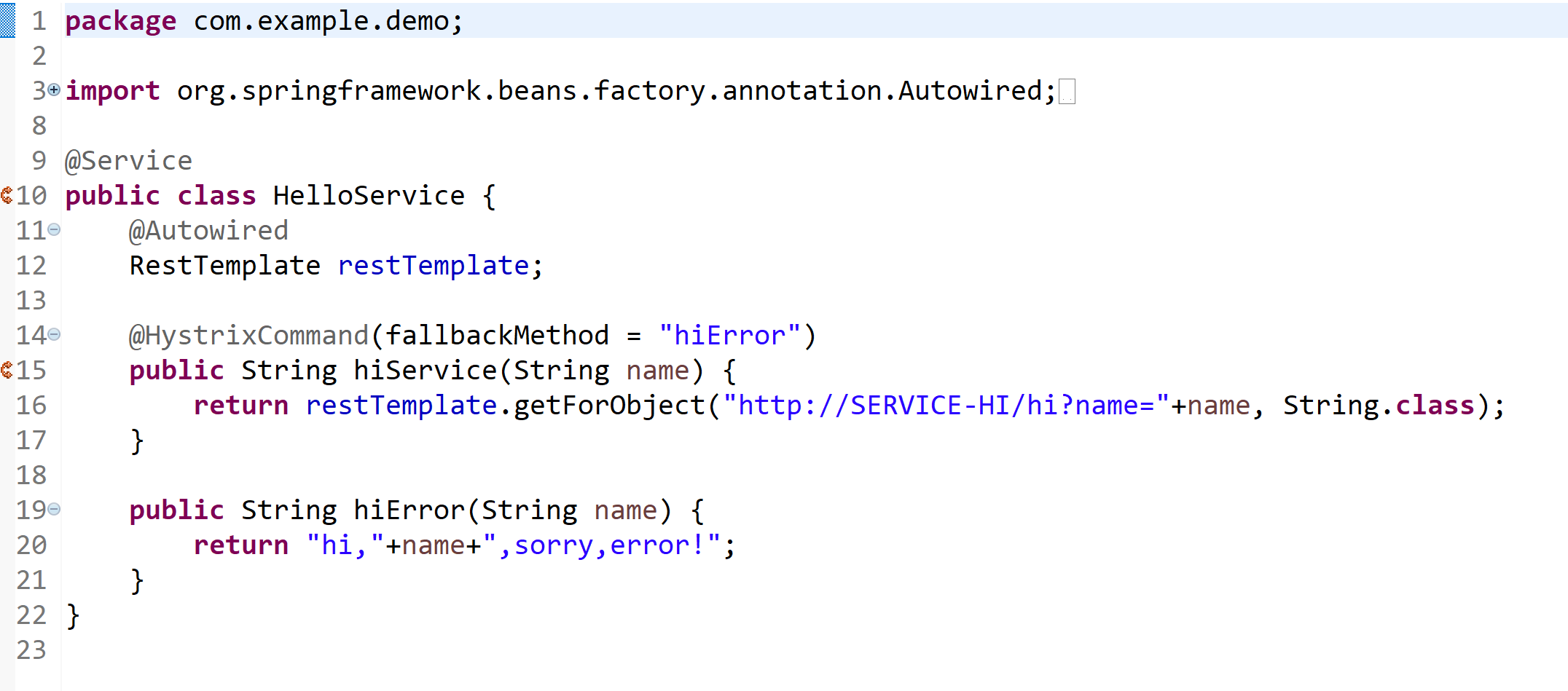
开放式电路会停止级联故障，并允许不必要的或失败的服务时间来愈合。回退可以是另一个Hystrix保护的调用，静态数据或一个正常的空值。回退可能被链接，所以第一个回退使得一些其他业务电话又回到静态数据。

首先在pox.xml文件中加入spring-cloud-starter-hystrix的起步依赖：

在程序的启动类ServiceRibbonApplication 加@EnableHystrix注解开启Hystrix：



改造HelloService类，在hiService方法上加上@HystrixCommand注解。该注解对该方法创建了熔断器的功能，并指定了fallbackMethod熔断方法，熔断方法直接返回了一个字符串，字符串为"hi,"+name+",sorry,error!"，代码如下：



访问：http://localhost:8085/hi?name=SpringCloud