1、Eureka 服务注册 服务提供者 服务消费

# 1、服务注册中心的搭建

## 1、项目名称为one-eurekas-erver

### 1、pom.xml依赖

*<?*xml version="1.0" encoding="UTF-8"*?>*<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>  
  
 <groupId>com.eureka.server.first</groupId>  
 <artifactId>one-eurekas-erver</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>one-eurekas-erver</name>  
 <description>Demo project for Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.5.3.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 <spring-cloud.version>Dalston.RELEASE</spring-cloud.version>  
 </properties>  
  
 <dependencies>  
 <!--eureka 服务端，服务注册中心-->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka-server</artifactId>  
 </dependency>  
   
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka-server</artifactId>  
 </dependency>  
  
  
 </dependencies>  
  
 <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>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
  
</project>

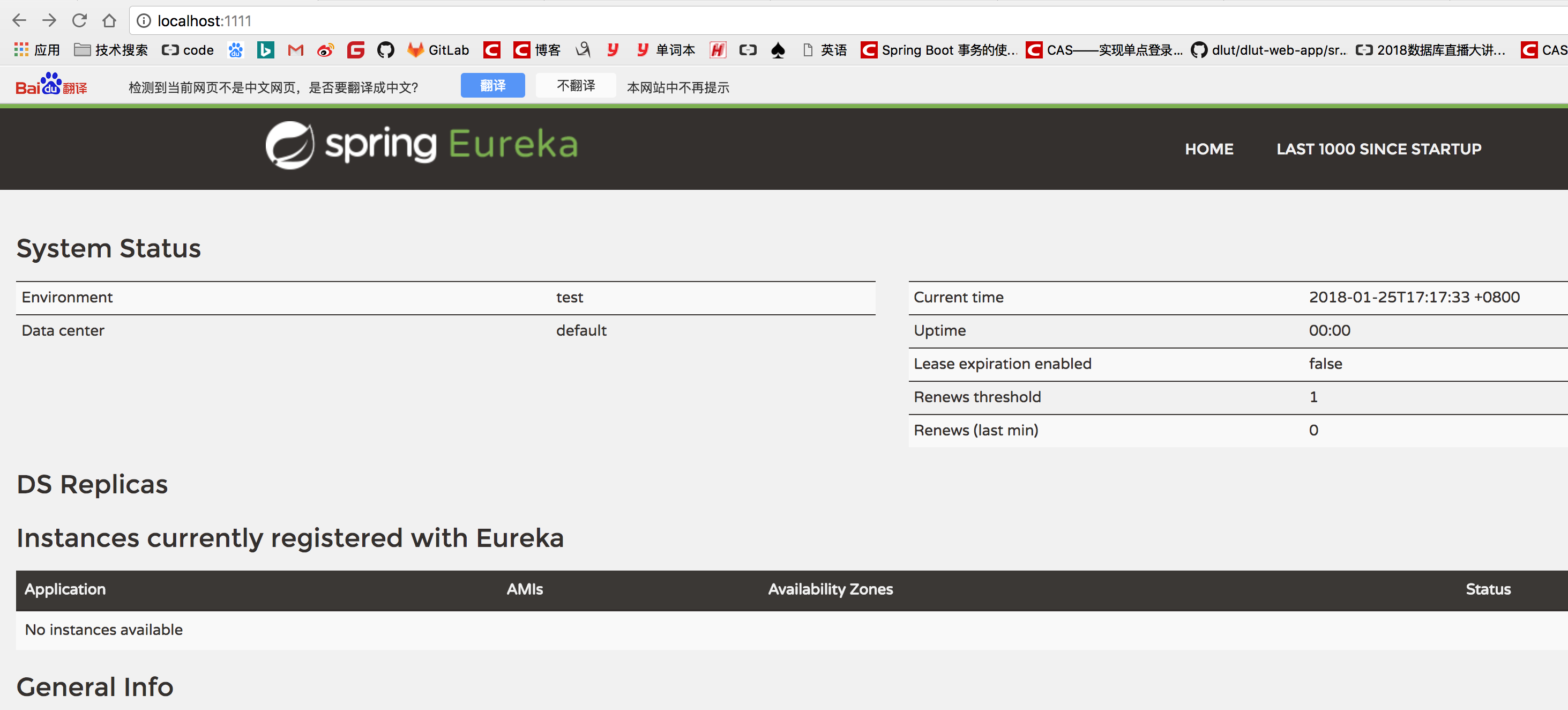
## 2、注解启动服务注册中心为其他的应用提供服务

@EnableEurekaServer //开启Eureka服务注册中心的支持,用来供给其他应用进行对话  
@SpringBootApplication  
public class OneEurekasErverApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(OneEurekasErverApplication.class, args);  
 }  
}

## 3、开始配置和服务注册中心

spring.application.name=one-eurekas-erver  
  
  
#默认设置下，这个服务注册中心会将自己作为客户端，所以需要禁用它的客户端注册行为 下面5个  
# 服务注册中心 端口  
server.port=1111  
#主机  
eureka.instance.hostname=localhost  
#这个为注册中心，false代表不向注册中心注册自己 ##直接启动之后会发现这个是空的，说明注册中心没有注册任何服务  
eureka.client.register-with-eureka=false   
#注册中心的职责就是维护服务实例，并不需要它去检索服务，所以将它关闭  
eureka.client.fetch-registry=false  
eureka.client.serviceUrl.defaultZone=http://${eureka.instance.hostname}:${server.port}/eureka/

## 4、启动这个服务注册中心



# 2、搭建一个简单的服务提供者

## 1、项目名称为 eureka-server-ha

### 1、pom.xml依赖为

*<?*xml version="1.0" encoding="UTF-8"*?>*<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>  
  
 <groupId>com.didispace</groupId>  
 <artifactId>eureka-server-ha</artifactId>  
 <version>1.0.0</version>  
 <packaging>jar</packaging>  
  
 <name>eureka-server-ha</name>  
 <description>Spring Cloud In Action</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.3.7.RELEASE</version>  
 <relativePath/>  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
   
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka-server</artifactId>  
 </dependency>  
 </dependencies>

<!--eureka 客户端，处理服务的注册和发现-->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eurek</artifactId>  
 </dependency>

<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>Brixton.SR5</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
   
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

## 2、注解启动服务发现

//支持服务发现  
@EnableDiscoveryClient  
@SpringBootApplication  
public class Application {  
  
 public static void main(String[] args) {  
 new SpringApplicationBuilder(Application.class).web(true).run(args);  
 }  
  
}

## 3、自动化配置DiscoveryClient发现服务后，提供一个http请求接口，💺作为服务提供者提供的接口

@RestController  
public class HomeController {  
  
  
 @Autowired  
 private DiscoveryClient client;  
  
 @RequestMapping(value = "hello",method = RequestMethod.***GET***)  
 public ServiceInstance index(){  
 ServiceInstance serviceInstance = client.getLocalServiceInstance();  
 System.***out***.println(  
 "host："+serviceInstance.getHost()+"/n " +  
 "service\_id :"+serviceInstance.getServiceId());  
 return serviceInstance;  
 }  
  
}

## 4、指定提供的服务命名、指定服务注册中心

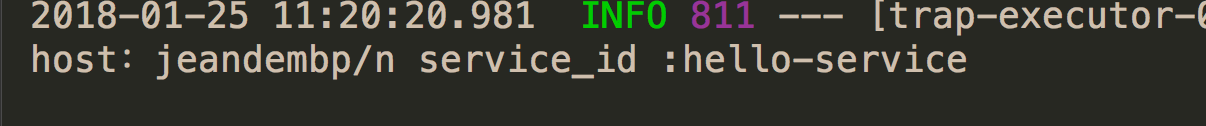
spring.application.name=hello-service  
eureka.client.serviceUrl.defaultZone=http://localhost:1111/eureka/  
server.port=8080

## 5、在已经启动了服务注册中心之后可启动这个服务提供者

### 1、http://localhost:1111 观察服务注册中心浏览器状态，发现有服务注册进来的



### 2、http://localhost:8080/hello ，查看服务提供者控制台，



# 3、高可用注册中心（接1中项目）

## 解释：讲自己作为服务向其他服务注册中心注册自己，这样就会形成一组互相注册的注册中心，用来实心服务清单的互相同步

## 1、修改host文件

127.0.0.1 peer1

127.0.0.1 peer2

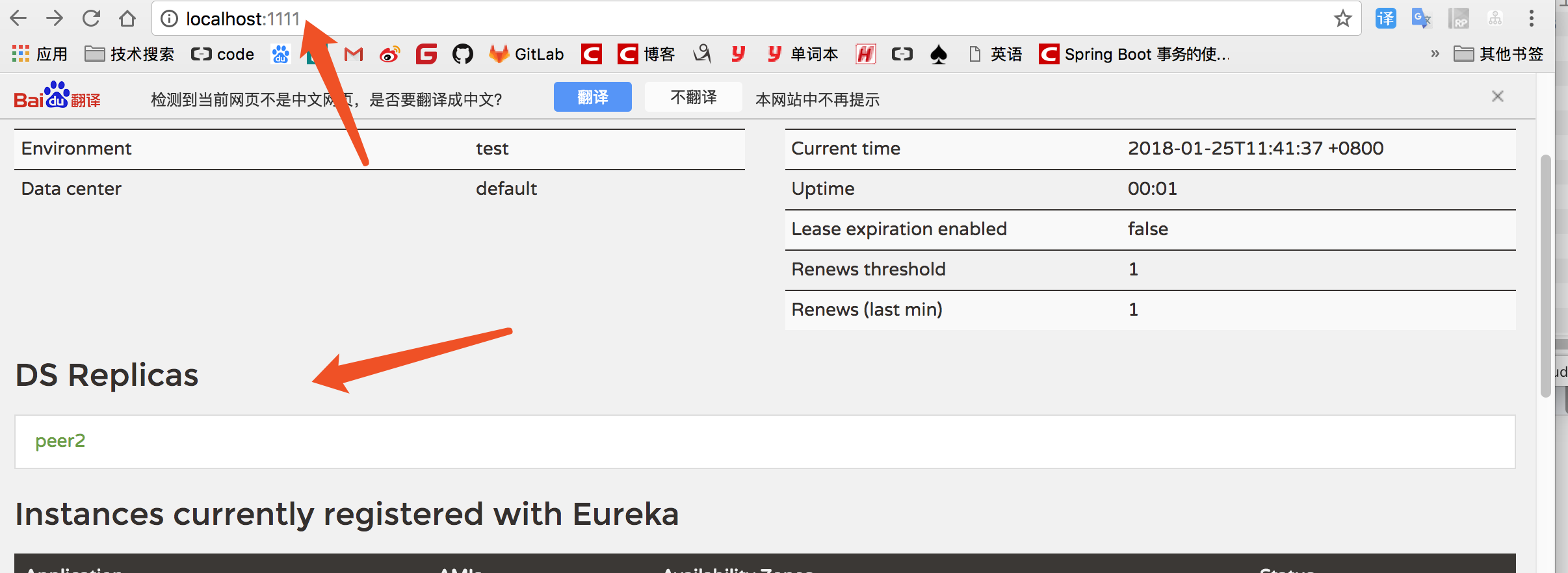
## 2、创建配置文件application-peer1.properties

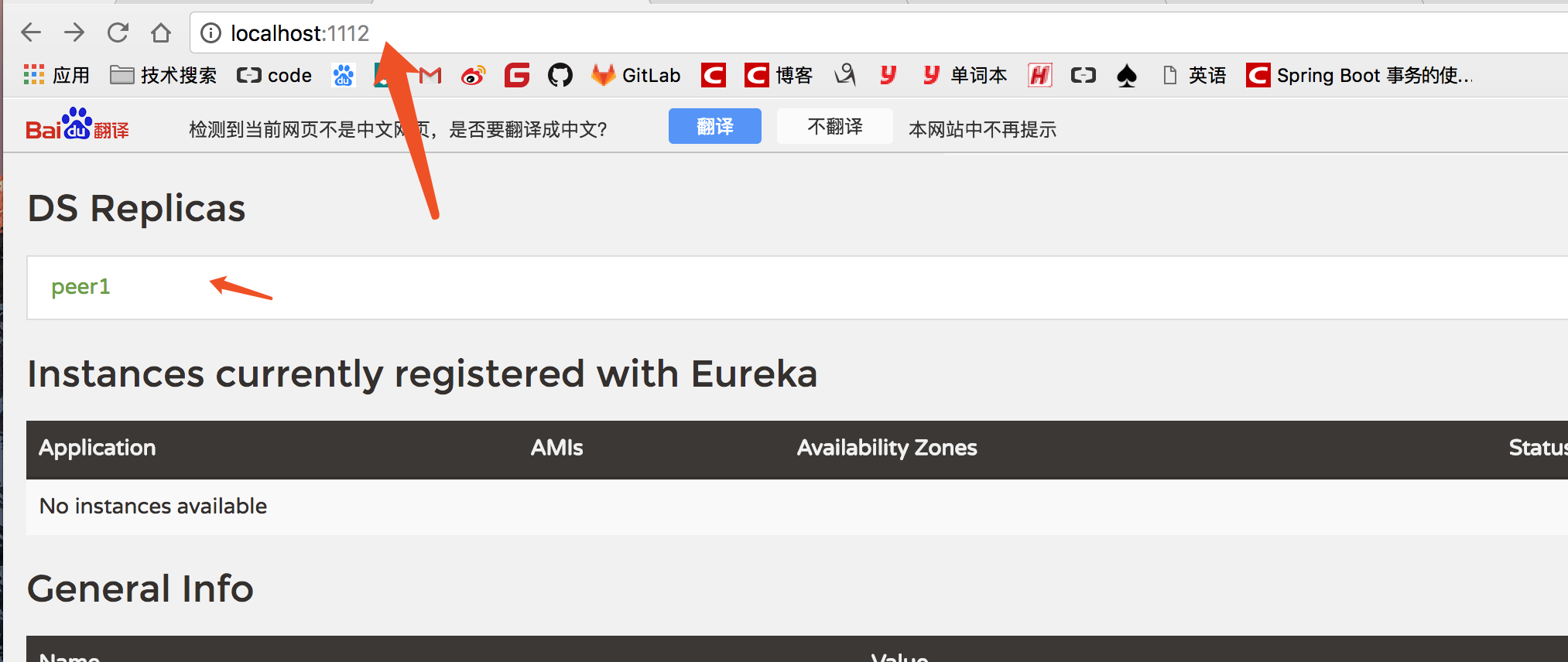
spring.application.name=one-eurekas-erver  
server.port=1111  
eureka.instance.hostname=peer1  
  
eureka.client.serviceUrl.defaultZone=http://peer2:1112/eureka/

## 3、创建配置文件

spring.application.name=one-eurekas-erver  
server.port=1112  
eureka.instance.hostname=peer2  
  
eureka.client.serviceUrl.defaultZone=http://peer1:1111/eureka/

## 4、分别利用这2两个配置文件进行启动（复制项目，修改spring.profiles.active 利用idea启动）

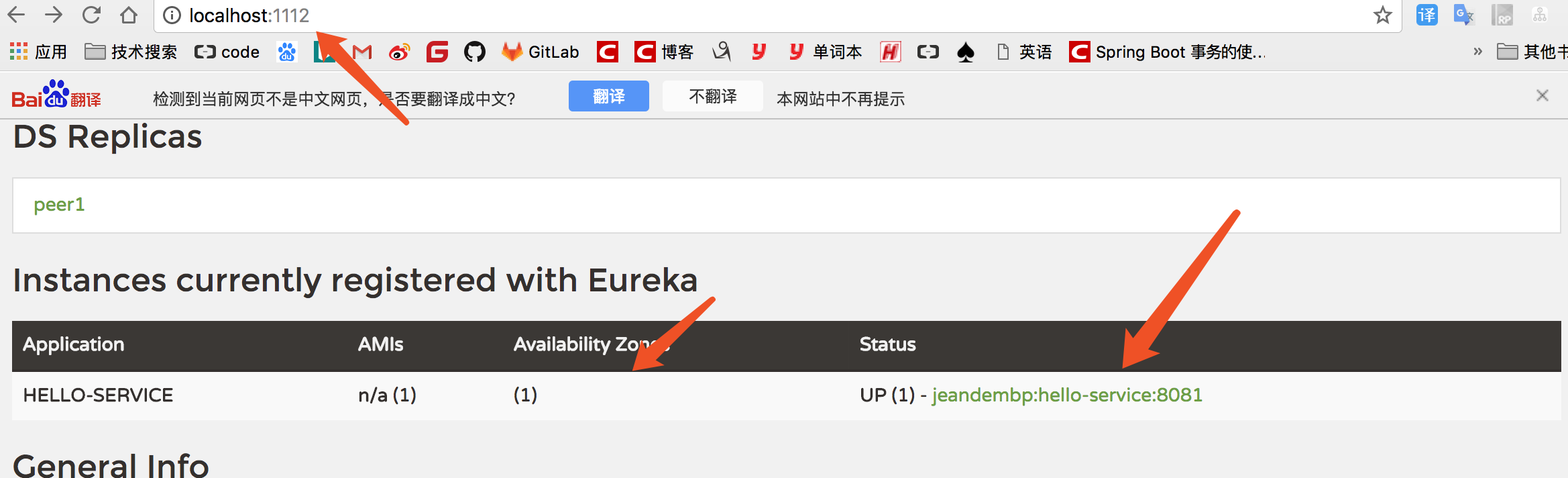


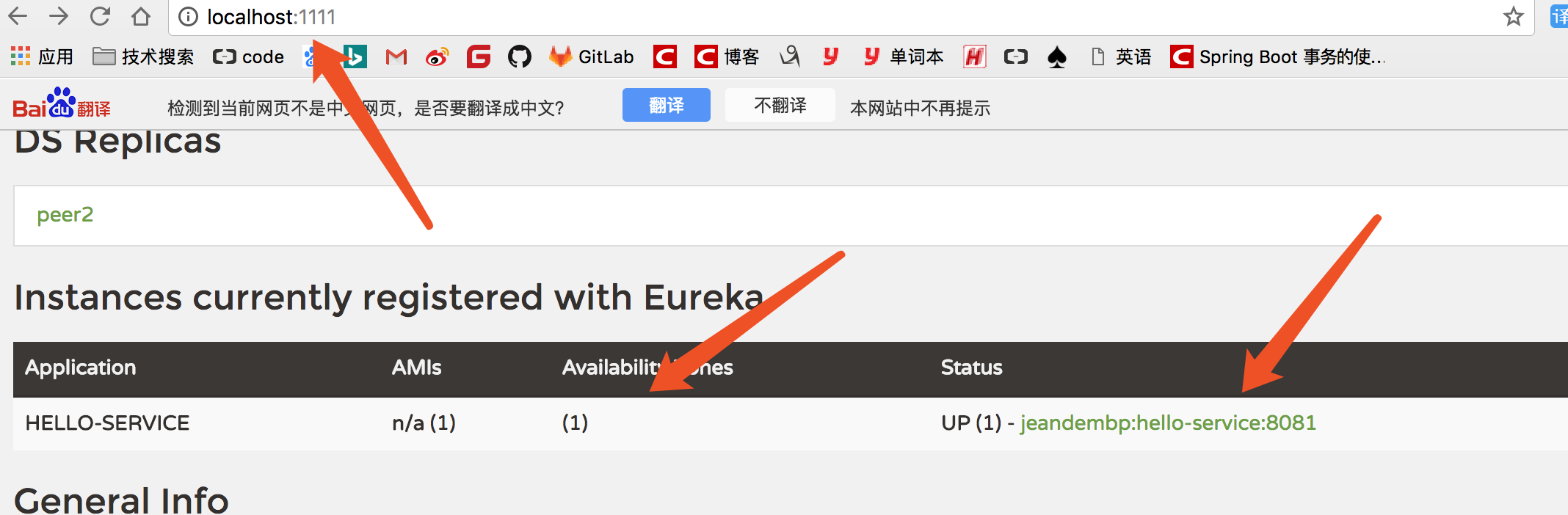


## 5、启动上面的（项目2，服务提供者hello-service）

### 1、即使这个服务提供者所指定的注册中心只有一个，但是因为上面的配置的高可用，其实二者共享了

spring.application.name=hello-service  
eureka.client.serviceUrl.defaultZone=http://localhost:1111/eureka/  
server.port=8080





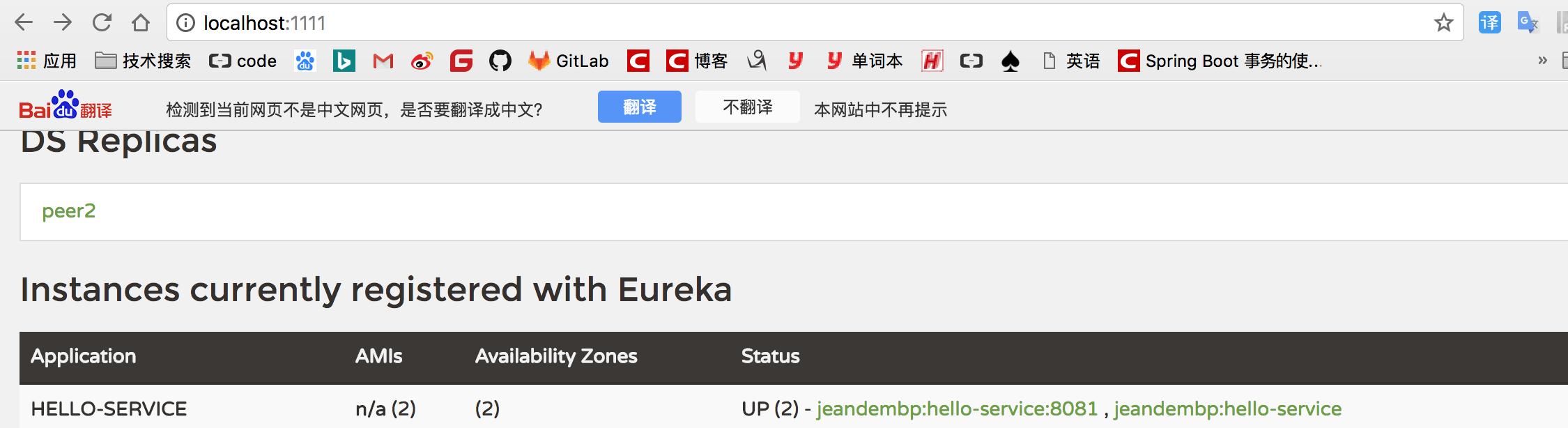
### 2、现在为这个服务提供者指定上面的两个注册中心 ，这样即使一个注册中心挂掉了，另外的一个注册中心还能够继续提供服务

spring.application.name=hello-service  
eureka.client.serviceUrl.defaultZone=http://peer1:1111/eureka/,http://peer2:1111/eureka/  
  
server.port=8080

# 4、服务发现和消费

## 解释：根据上面的操作，我们其实已经有了服务注册中心和服务提供者，下面就是开始构建一个服务消费者，它主要完成两个目标，发现服务和消费服务，Eureka为客户端可以发现服务，Ribbon则作为消费服务，它同时也作为辅助均衡器

## 1、服务提供者复制一个出来，并设置端口设置为8081 ，这样就有两个端口了8080，8081





## 2、创建服务消费者

### 1、pom.xml为

*<?*xml version="1.0" encoding="UTF-8"*?>*<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>  
  
 <groupId>com.didispace</groupId>  
 <artifactId>ribbon-consumer</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>ribbon-consumer</name>  
 <description>Demo project for Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.3.7.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-ribbon</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-hystrix</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-netflix-hystrix-amqp</artifactId>  
 </dependency>  
 </dependencies>  
  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>Brixton.SR5</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

### 2、注解服务发现和@LoadBalanced客户端负载均衡，这样就可以随机的访问8080 和8081了

@EnableDiscoveryClient  
@SpringBootApplication  
public class ConsumerApplication {  
  
 //开启客户端负载均衡  
 @Bean  
 @LoadBalanced  
 RestTemplate restTemplate() {  
 return new RestTemplate();  
 }  
 public static void main(String[] args) {  
 SpringApplication.run(ConsumerApplication.class, args);  
 }  
  
}

### 3、对服务提供者提供的接口进行调用http://HELLO-SERVICE/hello/

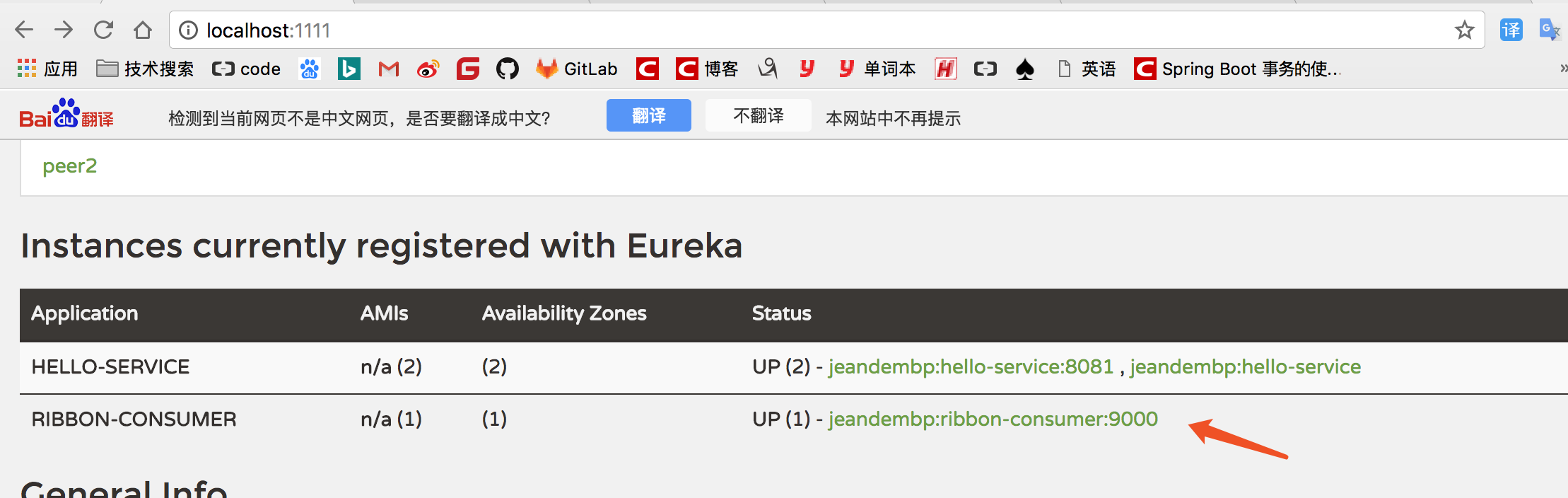
### 解释：HELLO-SERVICE 为服务提供者应用名称，hello 为提供者的url地址

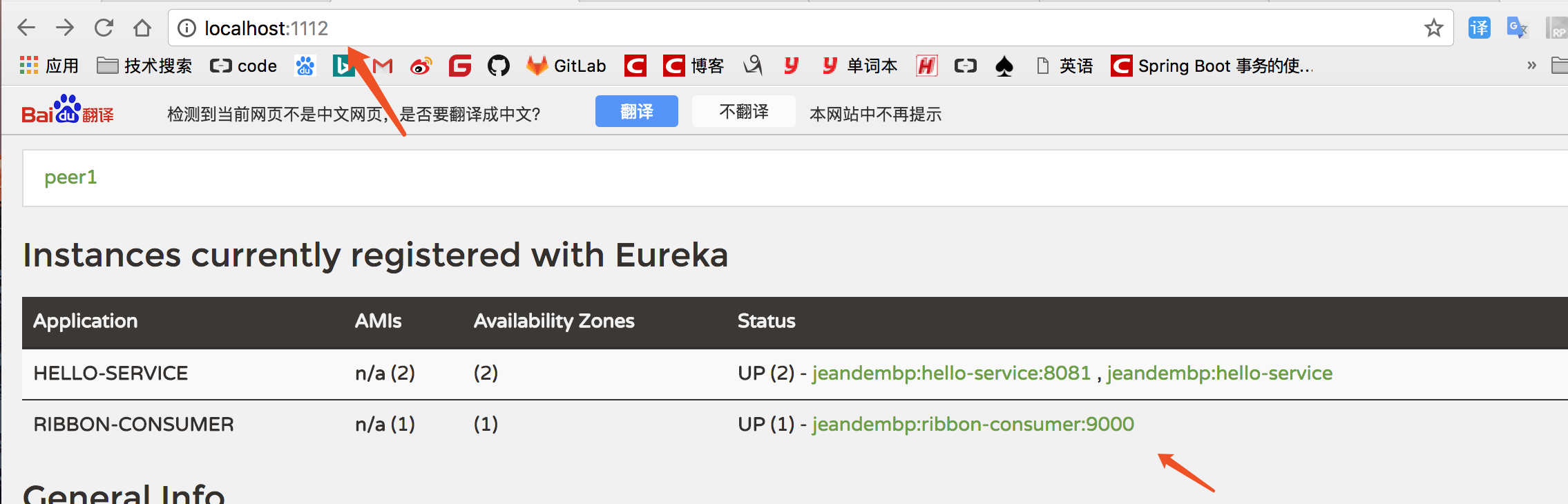
@RestController  
public class ConsumerController {  
  
  
 @Autowired  
 RestTemplate restTemplate;  
  
 @RequestMapping(value = "/ribbon-consumer", method = RequestMethod.***GET***)  
 public String helloConsumer() {  
 return restTemplate.getForEntity("http://HELLO-SERVICE/hello/",String.class).getBody();  
 }  
}

### 4、指定服服务消费名称、配置端口9000，指定注册中心，一个注册中心挂掉，还能用另外的一个

spring.application.name=ribbon-consumer  
server.port=9000  
  
eureka.client.serviceUrl.defaultZone=http://peer1:1111/eureka/,http://peer2:1112/eureka/,

### 5、启动这个服务消费者，观察1111和1112注册中心





## 3、浏览器访问这个服务消费者提供发http接口 http://localhost:9000/ribbon-consumer

#### 解释 可以观察到，访问的时候，调用的服务提供者是随机被我们访问的，8080和8081 是随机出现的，这就是负载均衡喽



