

## 第五章 Eureka 服务注册中心

### (Spring Cloud 初级)

#### 一、 什么是服务注册中心

服务注册中心是服务实现服务化管理的核心组件,类似于目录服务的作用,主要用来存储服务信息,譬如提供者 url 串、路由信息等。服务注册中心是 SOA 架构中最基础的设施之一。

#### 1 服务注册中心的作用

- 1, 服务的注册
- 2, 服务的发现



#### 2 常见的注册中心有哪些

- 1, Dubbo 的注册中心 Zookeeper
- 2, Springcloud 的注册中心 Eureka

#### 3 服务注册中心解决了什么问题

- 
1. 服务管理
  2. 服务的依赖关系管理

## 4 什么是 Eureka 注册中心

Eureka 是 Netflix 开发的服务发现组件，本身是一个基于 REST 的服务。Spring Cloud 将它集成在其子项目 spring-cloud-netflix 中，以实现 Spring Cloud 的服务注册与发现，同时还提供了负载均衡、故障转移等能力。

## 5 Eureka 注册中心三种角色

### 5.1 Eureka Server

通过 Register、Get、Renew 等接口提供服务的注册和发现。

### 5.2 Application Service (Service Provider)

服务提供方

把自身的服务实例注册到 Eureka Server 中

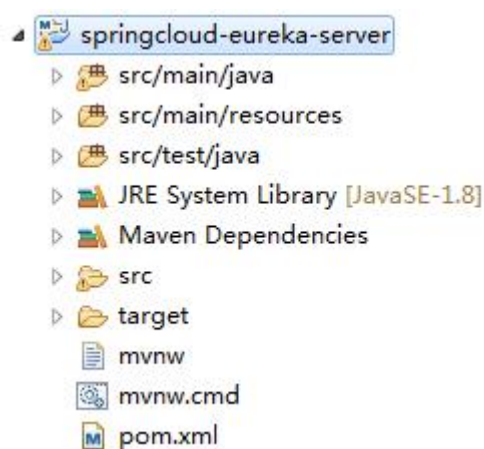
### 5.3 Application Client (Service Consumer)

服务调用方

通过 Eureka Server 获取服务列表，消费服务。

## 二、 Eureka 入门案例

### 1 创建项目



## 2 修改 pom 文件添加依赖

```
<?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.bjsxt</groupId>
  <artifactId>springcloud-eureka-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>springcloud-eureka-server</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.13.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEn
coding>

    <project.reporting.outputEncoding>UTF-8</project.reporting.
outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>
        <version>Dalston.SR5</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
```

```

        </dependencies>
    </dependencyManagement>
    <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-config</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka-server</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>

```

### 3 修改启动类

```

@EnableEurekaServer
@SpringBootApplication
public class EurekaApplication {
    public static void main(String[] args) {
        SpringApplication.run(EurekaApplication.class, args);
    }
}

```

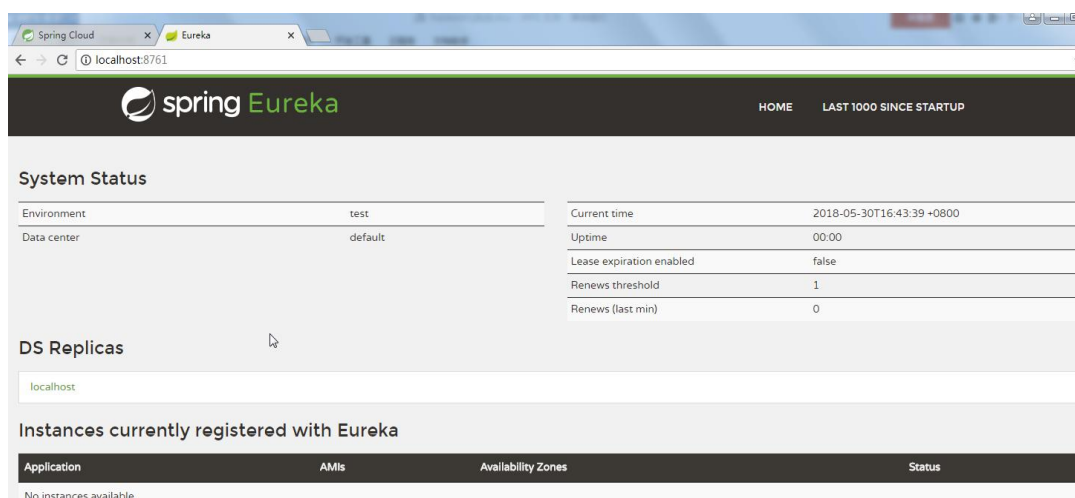
## 4 修改 application.properties 全局配置文件

```
spring.application.name=eureka-server
server.port=8761

#是否将自己注册到 Eureka-Server 中，默认的为 true
eureka.client.registerWithEureka=false

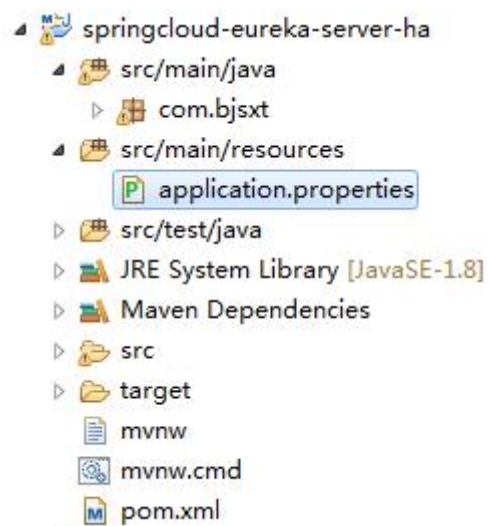
#是否从 Eureka-Server 中获取服务注册信息，默认为 true
eureka.client.fetchRegistry=false
```

## 5 通过浏览器访问 Eureka-Server 服务管理平台



## 三、 搭建高可用 Eureka 注册中心（Eureka 集群）

### 1 创建项目



---

## 2 配置文件

```
#设置 eureka 实例名称，与配置文件的变量为主
eureka.instance.hostname=eureka2

#设置服务注册中心地址，指向另一个注册中心
eureka.client.serviceUrl.defaultZone=http://eureka1:8761/eureka/
```

在搭建 Eureka 集群时，需要添加多个配置文件，并且使用 SpringBoot 的多环境配置方式。集群中需要多少节点就添加多少个配置文件。

## 3 在配置文件中配置集群节点

### 3.1 eureka1

```
spring.application.name=eureka-server
server.port=8761

#设置 eureka 实例名称，与配置文件的变量为主
eureka.instance.hostname=eureka1

#设置服务注册中心地址，指向另一个注册中心
eureka.client.serviceUrl.defaultZone=http://eureka2:8761/eureka/
```

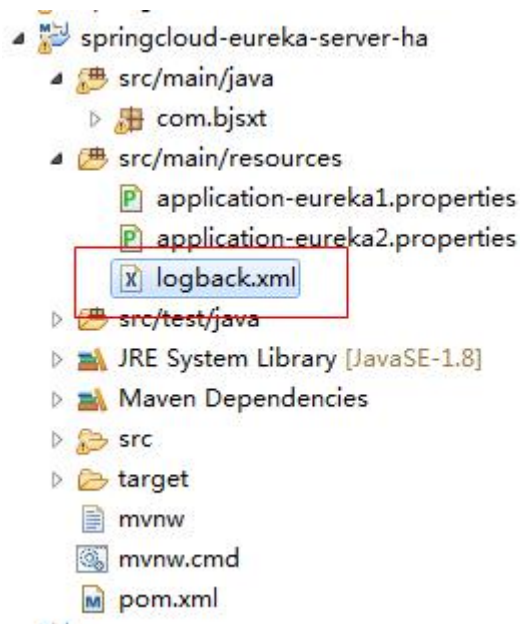
### 3.2 eureka2

```
spring.application.name=eureka-server
server.port=8761

#设置 eureka 实例名称，与配置文件的变量为主
eureka.instance.hostname=eureka2

#设置服务注册中心地址，指向另一个注册中心
eureka.client.serviceUrl.defaultZone=http://eureka1:8761/eureka/
```

## 4 添加 logback 日志配置文件



## 5 Eureka 集群部署

部署环境：需要安装 jdk1.8，正确配置环境变量。

注意：需要关闭 linux 的防火墙，或者是开放 8761 端口

### 5.1 将项目打包

Maven install

### 5.2 上传实例

在/usr/local/创建一个 eureka 的目录

将项目的 jar 包拷贝到/usr/local/eureka

## 6 编写一个启动脚本文件

```
#!/bin/bash

cd `dirname $0`

CUR_SHELL_DIR=`pwd`
CUR_SHELL_NAME=`basename ${BASH_SOURCE}`
```

```
JAR_NAME="项目名称"
JAR_PATH=$CUR_SHELL_DIR/$JAR_NAME

#JAVA_MEM_OPTS="-server -Xms1024m -Xmx1024m -XX:PermSize=128m"
JAVA_MEM_OPTS=""

SPRING_PROFILES_ACTIV="-Dspring.profiles.active=配置文件变量名称"
#SPRING_PROFILES_ACTIV=""
LOG_DIR=$CUR_SHELL_DIR/logs
LOG_PATH=$LOG_DIR/${JAR_NAME%..}.log

echo_help()
{
    echo -e "syntax: sh $CUR_SHELL_NAME start|stop"
}

if [ -z $1 ];then
    echo_help
    exit 1
fi

if [ ! -d "$LOG_DIR" ];then
    mkdir "$LOG_DIR"
fi

if [ ! -f "$LOG_PATH" ];then
    touch "$LOG_PATH"
fi

if [ "$1" == "start" ];then

    # check server
    PIDS=`ps --no-heading -C java -f --width 1000 | grep $JAR_NAME | awk
'{print $2}'`
    if [ -n "$PIDS" ]; then
        echo -e "ERROR: The $JAR_NAME already started and the PID is
${PIDS}."
        exit 1
    fi

    echo "Starting the $JAR_NAME..."

    # start
```



```

nohup java $JAVA_MEM_OPTS -jar $SPRING_PROFILES_ACTIV
$JAR_PATH >> $LOG_PATH 2>&1 &

COUNT=0
while [ $COUNT -lt 1 ]; do
    sleep 1
    COUNT=`ps --no-heading -C java -f --width 1000 | grep
"$JAR_NAME" | awk '{print $2}' | wc -l`
    if [ $COUNT -gt 0 ]; then
        break
    fi
done
PIDS=`ps --no-heading -C java -f --width 1000 | grep "$JAR_NAME" |
awk '{print $2}'`
echo "${JAR_NAME} Started and the PID is ${PIDS}."
echo "You can check the log file in ${LOG_PATH} for details."

elif [ "$1" == "stop" ];then

    PIDS=`ps --no-heading -C java -f --width 1000 | grep $JAR_NAME | awk
'{print $2}'`
    if [ -z "$PIDS" ]; then
        echo "ERROR:The $JAR_NAME does not started!"
        exit 1
    fi

    echo -e "Stopping the $JAR_NAME..."

    for PID in $PIDS; do
        kill $PID > /dev/null 2>&1
    done

    COUNT=0
    while [ $COUNT -lt 1 ]; do
        sleep 1
        COUNT=1
        for PID in $PIDS ; do
            PID_EXIST=`ps --no-heading -p $PID`
            if [ -n "$PID_EXIST" ]; then
                COUNT=0
                break
            fi
        done
    done
done

```

```

        echo -e "${JAR_NAME} Stopped and the PID is ${PIDS}."
    else
        echo_help
        exit 1
    fi

```

## 6.1 设置启动脚本的运行权限

```
Chmod -R 755 server.sh
```

## 7 修改 linux 的 host 文件

```

Vim    /etc/hosts
192.168.70.134 eureka1
192.168.70.135 eureka2

```

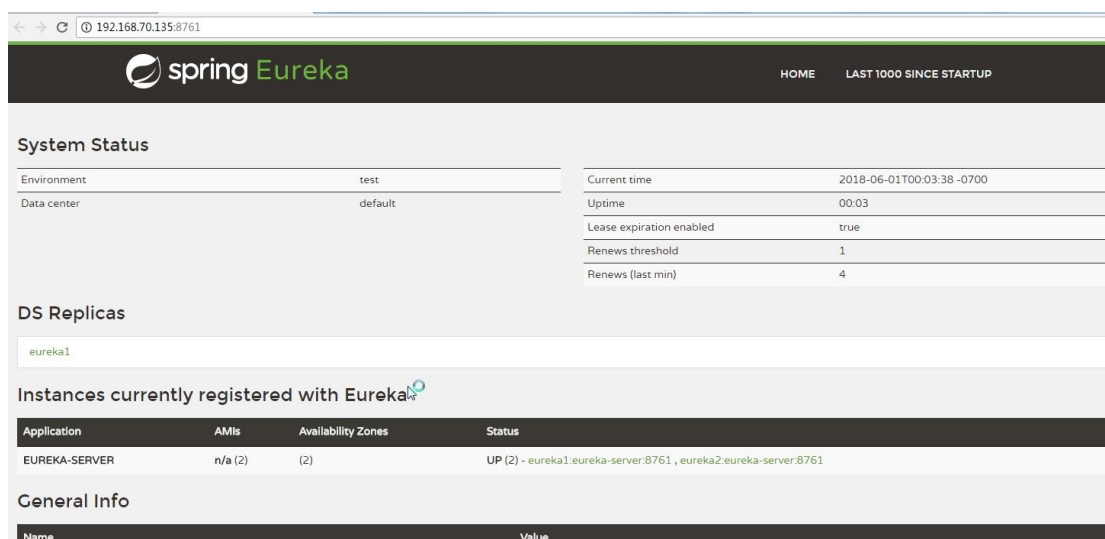
## 8 启动 eureka 注册中心

```

./server.sh start 启动
./server.sh stop 停止

```

## 9 通过浏览器访问注册中心的管理页面



The screenshot shows the Spring Eureka management interface. The browser address bar displays the URL `192.168.70.135:8761`. The page header includes the Spring Eureka logo and navigation links for HOME and LAST 1000 SINCE STARTUP.

### System Status

Environment	test	Current time	2018-06-01T00:03:38 -0700
Data center	default	Uptime	00:03
		Lease expiration enabled	true
		Renews threshold	1
		Renews (last min)	4

### DS Replicas

eureka1

### Instances currently registered with Eureka

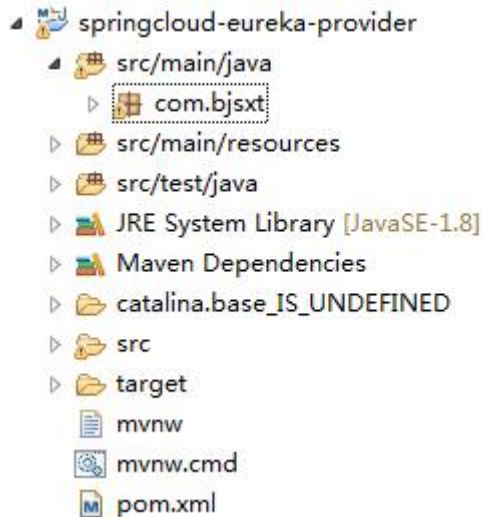
Application	AMIs	Availability Zones	Status
EUREKA-SERVER	n/a (2)	(2)	UP (2) - eureka1.eureka-server:8761, eureka2.eureka-server:8761

### General Info

Name	Value
------	-------

## 四、 在高可用的 Eureka 注册中心中构建 provider 服务

### 1 创建项目



### 2 修改 pom 文件

```
<?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.bjsxt</groupId>
  <artifactId>springcloud-eureka-provider</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>springcloud-eureka-provider</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.13.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>
```

```
<project.build.sourceEncoding>UTF-8</project.build.sourceEn
coding>

<project.reporting.outputEncoding>UTF-8</project.reporting.
outputEncoding>
    <java.version>1.8</java.version>
</properties>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>
            <version>Dalston.SR5</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<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-config</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>
    </dependency>
</dependencies>
<build>
    <plugins>
```

```
<plugin>
  <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
</project>
```

### 3 修改启动类

```
@EnableEurekaClient
@SpringBootApplication
public class EurekaApplication {
    public static void main(String[] args) {
        SpringApplication.run(EurekaApplication.class, args);
    }
}
```

### 4 修改 provider 的配置文件

```
spring.application.name=eureka-provider
server.port=9090

#设置服务注册中心地址，指向另一个注册中心
eureka.client.serviceUrl.defaultZone=http://eureka1:8761/eureka/,http://eureka2:8761/eureka/
```

### 5 修改 windows 的 host 文件

路径：C:\Windows\System32\drivers\etc

```
192.168.70.134 eureka1
192.168.70.135 eureka2
```

### 6 编写服务接口

#### 6.1 创建接口

```
@RestController
public class UserController {
```

```
@RequestMapping("/user")
public List<User> getUsers(){
    List<User> list = new ArrayList<>();
    list.add(new User(1,"zhangsan",20));
    list.add(new User(2,"lisi",22));
    list.add(new User(3,"wangwu",20));
    return list;
}
}
```

## 6.2 创建 pojo

```
public class User {

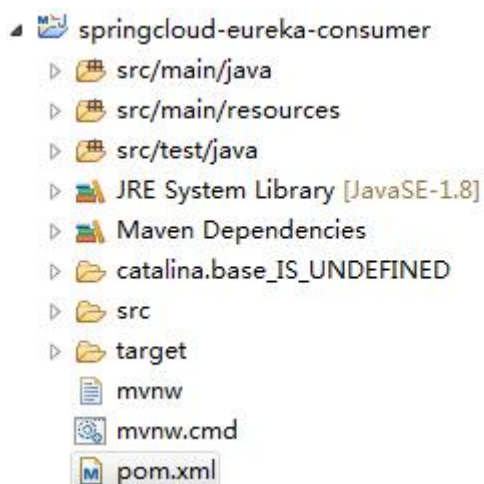
    private int userid;
    private String username;
    private int usage;
    public int getUserid() {
        return userid;
    }
    public void setUserid(int userid) {
        this.userid = userid;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public int getUsage() {
        return usage;
    }
    public void setUsage(int usage) {
        this.usage = usage;
    }
    public User(int userid, String username, int usage) {
        super();
        this.userid = userid;
        this.username = username;
        this.usage = usage;
    }
    public User() {
        super();
    }
}
```

```
        // TODO Auto-generated constructor stub
    }

}
```

## 五、 在高可用的 Eureka 注册中心中构建 consumer 服务

### 1 创建项目



服务的消费者与提供者都需要再 Eureka 注册中心注册。

### 2 Consumer 的配置文件

```
spring.application.name=eureka-consumer
server.port=9091

#设置服务注册中心地址，指向另一个注册中心
eureka.client.serviceUrl.defaultZone=http://eureka1:8761/eureka/,http://eureka2:8761/eureka/
```

### 3 在 Service 中完成服务的调用

```
@Service
public class UserService {

    @Autowired
    private LoadBalancerClient loadBalancerClient;//ribbon 负载均衡器
```

```

    public List<User> getUsers(){
        //选择调用的服务的名称
        //ServiceInstance 封装了服务的基本信息，如 IP，端口
        ServiceInstance si =
this.loadBalancerClient.choose("eureka-provider");
        //拼接访问服务的 URL
        StringBuffer sb = new StringBuffer();
        //http://localhost:9090/user

        sb.append("http://").append(si.getHost()).append(":").append(
si.getPort()).append("/user");

        //springMVC RestTemplate
        RestTemplate rt = new RestTemplate();

        ParameterizedTypeReference<List<User>> type = new
ParameterizedTypeReference<List<User>>() {};

        //ResponseBody:封装了返回值信息
        ResponseEntity<List<User>> response =
rt.exchange(sb.toString(),HttpMethod.GET, null, type);
        List<User> list =response.getBody();
        return list;
    }
}

```

## 4 Controller

```

@RestController
public class UserController {

    @Autowired
    private UserService userService;

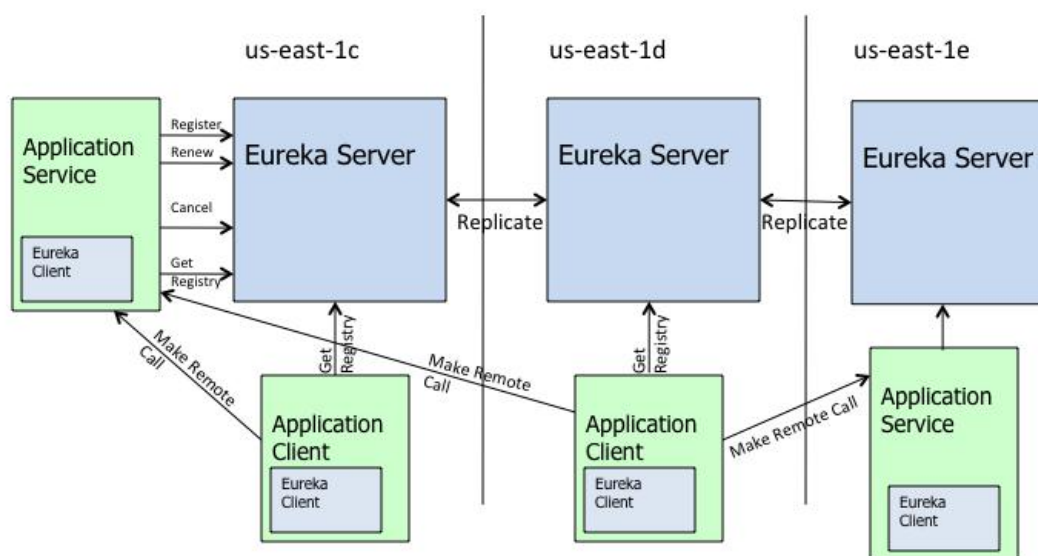
    @RequestMapping("/consumer")
    public List<User> getUsers(){
        return this.userService.getUsers();
    }
}

```



## 六、 Eureka 注册中心架构原理

### 1 Eureka 架构图



Register(服务注册): 把自己的 IP 和端口注册给 Eureka。

Renew(服务续约): 发送心跳包, 每 30 秒发送一次。告诉 Eureka 自己还活着。

Cancel(服务下线): 当 provider 关闭时会向 Eureka 发送消息, 把自己从服务列表中删除。防止 consumer 调用到不存在的服务。

Get Registry(获取服务注册列表): 获取其他服务列表。

Replicate(集群中数据同步): eureka 集群中的数据复制与同步。

Make Remote Call(远程调用): 完成服务的远程调用。

## 七、 基于分布式 CAP 定理, 分析注册中心两大主流框架: Eureka 与 Zookeeper 的区别

### 1 什么是 CAP 原则

CAP 原则又称 CAP 定理, 指的是在一个分布式系统中, **Consistency** (一致性)、**Availability** (可用性)、**Partition tolerance** (分区容错性), 三者不可兼得。

CAP 由 **Eric Brewer** 在 2000 年 PODC 会议上提出。该猜想在提出两年后被证明成立, 成为我们熟知的 CAP 定理

分布式系统 CAP 定理	
<b>C</b> <b>数据一致性</b> <b>(Consistency)</b>	也叫做数据原子性 系统在执行某项操作后仍然处于一致的状态。在分布式系统中，更新操作执行成功后所有的用户都应该读到最新的值，这样的系统被认为是具有强一致性的。等同于所有节点访问同一份最新的数据副本。
<b>A</b> <b>服务可用性</b> <b>(Availability)</b>	每一个操作总是能够在一定的时间内返回结果，这里需要注意的是“一定时间内”和“返回结果”。一定时间内指的是，在可以容忍的范围内返回结果，结果可以是成功或者是失败。
<b>P</b> <b>分区容错性</b> <b>(Partition-tolerance)</b>	在网络分区的情况下，被分隔的节点仍能正常对外提供服务(分布式集群，数据被分布存储在不同的服务器上，无论什么情况，服务器都能正常被访问)

<b>定律：任何分布式系统只可同时满足二点，没法三者兼顾。</b>	
<b>CA，放弃 P</b>	如果想避免分区容错性问题的发生，一种做法是将所有的数据(与事务相关的)都放在一台机器上。虽然无法 100%保证系统不会出错，单不会碰到由分区带来的负面效果。当然这个选择会严重的影响系统的扩展性。
<b>CP，放弃 A</b>	相对于放弃“分区容错性”来说，其反面就是放弃可用性。一旦遇到分区容错故障，那么受到影响的服务需要等待一定时间，因此在等待时间内系统无法对外提供服务。
<b>AP，放弃 C</b>	这里所说的放弃一致性，并不是完全放弃数据一致性，而是放弃数据的强一致性，而保留数据的最终一致性。以网络购物为例，对只剩下一件库存的商品，如果同时接受了两个订单，那么较晚的订单将被告知商品告罄。

## 2 Zookeeper 与 Eureka 的区别

对比项	Zookeeper	Eureka	
CAP	CP	AP	
Dubbo 集成	已支持	-	
Spring Cloud 集成	已支持	已支持	
kv 服务	支持	-	ZK 支持数据 存储,eureka 不支持
使用接口(多语言能 力)	提供客户端	http 多语言	ZK 的跨语 言支持比较弱
watch 支持	支持	支持	什么是 Watch 支持? 就是客户单 监听服务端 的变化情况。 zk 通过订阅监 听来实现 eureka 通过轮 询的方式来实现
集群监控	-	metrics	metrics, 运维者可以收 集并报警这些 度量信息达到 监控目的

## 八、 Eureka 优雅停机

### 1 在什么条件下，Eureka 会启动自我保护？

什么是自我保护模式
<p><b>1, 自我保护的条件</b></p> <p>一般情况下，微服务在 Eureka 上注册后，会每 30 秒发送心跳包，Eureka 通过心跳来判断服务时候健康，同时会定期删除超过 90 秒没有发送心跳服务。</p> <p><b>2, 有两种情况会导致 Eureka Server 收不到微服务的心跳</b></p> <p>a.是微服务自身的原因</p> <p>b. 是微服务与 Eureka 之间的网络故障</p> <p>通常(微服务的自身的故障关闭)只会导致个别服务出现故障，一般不会出现大面积故障，而(网络故障)通常会导致 Eureka Server 在短时间内无法收到大批心跳。</p> <p>考虑到这个区别，Eureka 设置了一个阈值，当判断挂掉的服务的数量超过阈值时，Eureka Server 认为很大程度上出现了网络故障，将不再删除心跳过期的服务。</p> <p><b>3, 那么这个阈值是多少呢？</b></p> <p>15 分钟之内是否低于 85%;</p> <p>Eureka Server 在运行期间，会统计心跳失败的比例在 15 分钟内是否低于 85%</p> <p>这种算法叫做 Eureka Server 的自我保护模式。</p>

### 2 为什么要启动自我保护

为什么要自我保护
<p>1, 因为同时保留“<b>好数据</b>”与“<b>坏数据</b>”总比丢掉任何数据要更好，当网络故障恢复后，这个 Eureka 节点会退出“自我保护模式”。</p> <p>2, Eureka 还有客户端缓存功能(也就是微服务的缓存功能)。即便 Eureka 集群中所有节点都宕机失效，微服务的 Provider 和 Consumer 都能正常通信。</p> <p>3, 微服务的负载均衡策略会自动剔除死亡的微服务节点。</p>

### 3 如何关闭自我保护

修改 Eureka Server 配置文件

```
#关闭自我保护:true 为开启自我保护, false 为关闭自我保护
eureka.server.enableSelfPreservation=false
#清理间隔(单位:毫秒, 默认是 60*1000)
eureka.server.eviction.interval-timer-in-ms=60000
```

### 4 如何优雅停服

#### 4.1 不需要再 Eureka Server 中配置关闭自我保护

#### 4.2 需要再服务中添加 actuator.jar 包

```
<?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.bjsxt</groupId>
  <artifactId>springcloud-eureka-provider</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>springcloud-eureka-provider</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.13.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>

  <project.build.sourceEncoding>UTF-8</project.build.sourceEn
coding>
```

```
<project.reporting.outputEncoding>UTF-8</project.reporting.
outputEncoding>
    <java.version>1.8</java.version>
</properties>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>
            <version>Dalston.SR5</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<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-config</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka-server</artifactId>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>
```

```
        </plugin>
    </plugins>
</build>
</project>
```

### 4.3 修改配置文件

```
#启用 shutdown
endpoints.shutdown.enabled=true
#禁用密码验证
endpoints.shutdown.sensitive=false
```

### 4.4 发送一个关闭服务的 URL 请求

```
public class HttpClientUtil {

    public static String doGet(String url, Map<String, String>
param) {

        // 创建 HttpClient 对象
        CloseableHttpClient httpClient =
HttpClientBuilder.createDefault();

        String resultString = "";
        CloseableHttpResponse response = null;
        try {
            // 创建 uri
            URIBuilder builder = new URIBuilder(url);
            if (param != null) {
                for (String key : param.keySet()) {
                    builder.addParameter(key, param.get(key));
                }
            }
            URI uri = builder.build();

            // 创建 http GET 请求
            HttpGet httpGet = new HttpGet(uri);

            // 执行请求
            response = httpClient.execute(httpGet);
            // 判断返回状态是否为 200
```

```

        if (response.getStatusLine().getStatusCode() ==
200) {
            resultString =
EntityUtils.toString(response.getEntity(), "UTF-8");
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (response != null) {
                response.close();
            }
            httpClient.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return resultString;
}

public static String doGet(String url) {
    return doGet(url, null);
}

public static String doPost(String url, Map<String, String>
param) {
    // 创建 HttpClient 对象
    CloseableHttpClient httpClient =
HttpClientUtils.createDefault();
    CloseableHttpResponse response = null;
    String resultString = "";
    try {
        // 创建 Http Post 请求
        HttpPost httpPost = new HttpPost(url);
        // 创建参数列表
        if (param != null) {
            List<NameValuePair> paramList = new
ArrayList<>();
            for (String key : param.keySet()) {
                paramList.add(new BasicNameValuePair(key,
param.get(key)));
            }
            // 模拟表单
            UrlEncodedFormEntity entity = new

```



```
UrlEncodedFormEntity(paramList, "utf-8");
        httpPost.setEntity(entity);
    }
    // 执行 http 请求
    response = httpClient.execute(httpPost);
    resultString =
EntityUtils.toString(response.getEntity(), "utf-8");
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            response.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    return resultString;
}

public static String doPost(String url) {
    return doPost(url, null);
}

public static String doPostJson(String url, String json) {
    // 创建 HttpClient 对象
    CloseableHttpClient httpClient =
HttpClientBuilder.create().build();
    CloseableHttpResponse response = null;
    String resultString = "";
    try {
        // 创建 Http Post 请求
        HttpPost httpPost = new HttpPost(url);
        // 创建请求内容
        StringEntity entity = new StringEntity(json,
ContentType.APPLICATION_JSON);
        httpPost.setEntity(entity);
        // 执行 http 请求
        response = httpClient.execute(httpPost);
        resultString =
EntityUtils.toString(response.getEntity(), "utf-8");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
    } finally {
        try {
            response.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    return resultString;
}

public static void main(String[] args) {
    String url = "http://127.0.0.1:9090/shutdown";
    //该 url 必须要使用 dopost 方式来发送
    HttpClientUtil.doPost(url);
}
}
```

## 九、 如何加强 Eureka 注册中心的安全认证

### 1 在 Eureka Server 中添加 security 包

```
<dependency>
    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

### 2 修改 Eureka Server 配置文件

```
#开启 http basic 的安全认证
security.basic.enabled=true
security.user.name=user
security.user.password=123456
```

---

### 3 修改访问集群节点的 url

```
eureka.client.serviceUrl.defaultZone=http://user:123456@eureka2:8761/eureka/
```

### 4 修改微服务的配置文件添加访问注册中心的用户名与密码

```
spring.application.name=eureka-provider
server.port=9090

#设置服务注册中心地址，指向另一个注册中心
eureka.client.serviceUrl.defaultZone=http://user:123456@eureka1:8761/eureka/,http://user:123456@eureka2:8761/eureka/

#启用 shutdown
endpoints.shutdown.enabled=true
#禁用密码验证
endpoints.shutdown.sensitive=false
```