# 前言

之前咱们说过**SpringCloud是分布式微服务架构的一站式解决方案**，是一个全家桶式的技术栈，包含了很多组件。接下来咱们就开始学习其最核心的几个组件入手，也就是**Eureka、Ribbon、Feign、Hystrix、Zuul**这几个组件，并且会剖析一下每个组件其底层的工作原理。

# Eureka(优瑞卡)是什么

Eureka是基于REST（代表性状态转移）的服务，主要在AWS云中用于定位服务，以实现负载均衡和中间层服务器的故障转移。我们称此服务为Eureka服务器。Eureka还带有一个基于Java的客户端组件Eureka Client，它使与服务的交互变得更加容易。客户端还具有一个内置的负载平衡器，可以执行基本的循环负载平衡。在Netflix，更复杂的负载均衡器将Eureka包装起来，以基于流量，资源使用，错误条件等多种因素提供加权负载均衡，以提供出色的弹性。

Spring Cloud 封装了 Netflix 公司开发的 Eureka 模块来实现**服务注册和发现**。Eureka 采用了 C-S 的设计架构。Eureka Server 作为服务注册功能的服务器，它是**服务注册中心**。而系统中的其他微服务，使用 Eureka 的客户端连接到 Eureka Server，并维持心跳连接。这样系统的维护人员就可以通过 Eureka Server 来监控系统中各个微服务是否正常运行。Spring Cloud 的一些其他模块（比如Zuul）就可以通过 Eureka Server 来发现系统中的其他微服务，并执行相关的逻辑。

|  |
| --- |
| 举一个生活中的例子，就比如我们平时租房子找中介的事情。  在没有中介的时候我们需要一个一个去寻找是否有房屋要出租的房东，这显然会非常的费力，一你找凭一个人的能力是找不到很多房源供你选择，再者你也懒得这么找下去(找了这么久，没有合适的只能将就)。这里的我们就相当于微服务中的 Consumer(消费者) ，而那些房东就相当于微服务中的 Provider(提供者) 。消费者 Consumer 需要调用提供者 Provider 提供的一些服务，就像我们现在需要租他们的房子一样。  但是如果只是租客和房东之间进行寻找的话，他们的效率是很低的，房东找不到租客赚不到钱，租客找不到房东住不了房。所以，后来房东肯定就想到了广播自己的房源信息(比如在街边贴贴小广告)，这样对于房东来说已经完成他的任务(将房源公布出去)，但是有两个问题就出现了。第一、其他不是租客的都能收到这种租房消息，这在现实世界没什么，但是在计算机的世界中就会出现资源消耗的问题了。第二、租客这样还是很难找到你，试想一下我需要租房，我还需要东一个西一个地去找街边小广告，麻不麻烦？  那怎么办呢？我们当然不会那么傻乎乎的，第一时间就是去找 中介 呀，它为我们提供了统一房源的地方，我们消费者只需要跑到它那里去找就行了。而对于房东来说，他们也只需要把房源在中介那里发布就行了。  那么现在，我们的模式就是这样的了。    但是，这个时候还会出现一些问题。  房东注册之后如果不想卖房子了怎么办？我们是不是需要让房东定期续约 ？如果房东不进行续约是不是要将他们从中介那里的注册列表中移除 。  租客是不是也要进行注册呢？不然合同乙方怎么来呢？  中介可不可以做连锁店呢？如果这一个店因为某些不可抗力因素而无法使用，那么我们是否可以换一个连锁店呢？  针对上面的问题我们来重新构建一下上面的模式图 |

好了，举完这个例子我们就可以来看关于 [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 的一些基础概念了，你会发现这东西理解起来怎么这么简单。

# Eureka两大组件

* **Eureka服务器(房屋中介)**

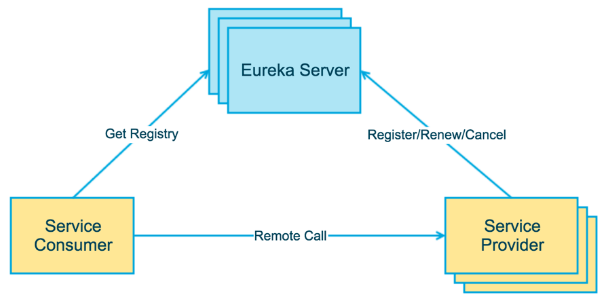
**Eureka Server提供服务注册功能**，各个节点启动后，会在EurekaServer中进行注册，这样EurekaServer中的**服务注册表**中将会存储所有可用服务节点的信息，服务节点的信息可以在界面中直观的看到

* **Eureka客户端(租房的人和房东)**

EurekaClient是一个Java客户端，用于简化Eureka Server的交互，客户端同时也具备一个内置的、使用轮询(round-robin)负载算法的负载均衡器。在应用启动后，将会向Eureka Server发送心跳(默认周期为30秒)。如果Eureka Server在多个心跳周期内没有接收到某个节点的心跳，EurekaServer将会从服务注册表中把这个服务节点移除（默认90秒）

|  |
| --- |
| 思考：Eureka服务器是不是就是相当于中介呀，Eureka客户端是不是就是我们和房东呀，我们是不是通过中介找到的房东啊？ |

# Eureka基本架构（三个角色）



* **Eureka Server(房屋中介)**

提供服务注册和发现。同时，Eureka Server 也为我们提供了可视化的监控页面，可以直观地看到各个 Eureka Server 当前的运行状态和所有已注册服务的情况。再说白点EurekaServer其实就是服务提供者和服务消费者之间的“桥梁”，服务提供者可以把自己注册到服务中介那里，而服务消费者如需要消费一些服务(使用一些功能)就可以在服务中介中寻找注册在服务中介的服务提供者。

* **Service Provider(出租房子的)**

服务提供方。将自身服务注册到Eureka Server，从而使服务消费方能够找到

* **Service Consumer(租房子的)**

服务消费方。从Eureka Server获取注册服务列表，从而能够消费服务

# 学习SpringCloud组件的秘诀

1. **在pom.xml中添加该组件对应的依赖**
2. **修改配置文件，做一些和该组件相对应的配置**
3. **修改启动类，在启动类上加组件相对应的注解:@EnableXXXX**

# IDEA批量启动多个SpringBoot应用

<https://www.jianshu.com/p/1f840b7f7de4>

# Eureka实战

1. **搭建Eureka Server注册中心**

* **新建一个子模块**
* **在子模块的pom.xml中添加Eureka Server的依赖**

|  |
| --- |
| <!--引入EurekaServer的依赖--> <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-eureka-server --> <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-eureka-server</artifactId>  <version>1.4.7.RELEASE</version> </dependency> |

* **修改子模块的配置文件**

|  |
| --- |
| **server**:  **port**: 22222  *#进行Eureka Server的配置* **eureka**:  **instance**:  **hostname**: localhost *#Eureka Server实例的主机名* **client**:  *#这块设成false表名不向Eureka注册中心注册自己* **register-with-eureka**: **false** *#这块设成false是表名就是Eureka服务器* **fetch-registry**: **false** *#Eureka客户端想要向EurekaServer查询服务和注册服务都需要依赖于这个地址* **service-url**:  **defaultZone**: http://${**eureka.instance.hostname**}:${**server.port**}/eureka/ |

* **修改子模块的启动类**

1. **将微服务全都注册到Eureka Server中**

**添加Eureka客户端依赖**

**修改配置文件**

Spring:

Application:

Name: 名字

Eureka:

Service-url:

defaultZone: EurekaServer提供的交互地址

**修改了启动类:@EnableEurekaClient**

1. **完善注册到Eureka Server的微服务信息**

* 主机名称:服务名称修改

instance:

instance-id:

* 访问信息有IP信息提示

prefer-ip-address: true #访问路径可以显示IP地址

* 微服务info内容详细信息

父工程的pom.xml加入以下配置

|  |
| --- |
| <build>  <finalName>microservicecloud</finalName>  <resources>  <resource>  <directory>src/main/resources</directory>  <filtering>true</filtering>  </resource>  </resources>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-resources-plugin</artifactId>  <configuration>  <delimiters>  <delimit>$</delimit>  </delimiters>  </configuration>  </plugin>  </plugins>  </build> |

在服务提供者模块的pom.xml中加入

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-actutor</artifactId>  <version>2.1.2.RELEASE</version> </dependency> |

info:

app.name:

company.name: www.itfh.com

build.artifactId: $project.artifactId$

build.version: $project.version$

# Eureka工作原理

服务启动向Eureka注册中心注册服务，注册中心将注册信息同步给其他注册中心，当服务消费者需要调用服务提供者，则向服务注册中心获取服务提供者地址，然后会将获取的地址缓存在本地，下次调用的则直接从缓存中取，当服务注册中心检测到服务提供者宕机、网络不通等服务不可用时，则在注册中心将服务置为down状态，并向订阅者发布当前服务提供者状态，订阅过的服务消费者更新本地缓存。

服务提供者启动后周期性（30秒）向服务注册中心发送心跳，以证明当前服务可用，服务中心在一定时间（默认90秒）未收到心跳，则认为该服务宕机，注销该实例。

# Eureka自我保护机制

默认情况下，如果Eureka Server在一定时间内没有接收到某个微服务实例的心跳，Eureka Server将会注销该实例（默认90秒）。但是当网络分区故障发生时，微服务与Eureka Server之间无法正常通信，以上行为可能变得非常危险了——因为微服务本身其实是健康的，此时本不应该注销这个微服务。Eureka通过“自我保护模式”来解决这个问题——当Eureka Server节点在**短时间内丢失过多客户端时**（可能发生了网络分区故障），那么这个节点就会进入自我保护模式。一旦进入该模式，Eureka Server就会保护服务注册表中的信息，不再删除服务注册表中的数据（也就是不会注销任何微服务）。当网络故障恢复后，该Eureka Server节点会自动退出自我保护模式。

在自我保护模式中，Eureka Server会保护服务注册表中的信息，不再注销任何服务实例。当它收到的心跳数重新恢复到阈值以上时，该Eureka Server节点就会自动退出自我保护模式。它的设计哲学就是宁可保留错误的服务注册信息，也不盲目注销任何可能健康的服务实例。一句话讲解：**好死不如赖活着**。

综上，自我保护模式是一种应对网络异常的安全保护措施。它的架构哲学是宁可同时保留所有微服务（健康的微服务和不健康的微服务都会保留），也不盲目注销任何健康的微服务。使用自我保护模式，可以让Eureka集群更加的健壮、稳定。

在Spring Cloud中，可以使用eureka.server.enable-self-preservation = false 禁用自我保护模式。

# 服务注册 Register

**官方解释**：当 [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户端向 Eureka Server 注册时，它提供自身的元数据 ，比如IP地址、端口，运行状况指示符URL，主页等。

**结合中介理解**：房东 (服务提供者)在中介 (服务器 Eureka Server) 那里登记房屋的信息，比如面积，价格，地段等等(元数据 metaData)。

# 服务续约 Renew

**官方解释**：[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户会每隔30秒(默认情况下)发送一次心跳来续约 。 通过续约来告知 EurekaServer 该 [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户仍然存在，没有出现问题。 正常情况下，如果 Eureka Server 在90秒没有收到 [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户的续约，它会将实例从其注册表中删除。

**结合中介理解**：房东 (服务提供者 ) 定期告诉中介 (服务器 Eureka Server) 我的房子还租(续约) ，中介 (服务器Eureka Server) 收到之后继续保留房屋的信息。

# 获取注册列表信息 Fetch Registries

**官方解释**：[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户端从服务器获取注册表信息，并将其缓存在本地。客户端会使用该信息查找其他服务，从而进行远程调用。该注册列表信息定期（每30秒钟）更新一次。每次返回注册列表信息可能与 [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户端的缓存信息不同, [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户端自动处理。如果由于某种原因导致注册列表信息不能及时匹配，[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户端则会重新获取整个注册表信息。[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 服务器缓存注册列表信息，整个注册表以及每个应用程序的信息进行了压缩，压缩内容和没有压缩的内容完全相同。[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户端和 [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 服务器可以使用JSON / XML格式进行通讯。在默认的情况下 [Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect) 客户端使用压缩 JSON 格式来获取注册列表的信息。

**结合中介理解**：租客(消费者 Eureka Client Consumer) 去中介 (服务器 Eureka Server) 那里获取所有的房屋信息列表 (客户端列表 Eureka Client List) ，而且租客为了获取最新的信息会定期向中介 (服务器 Eureka Server) 那里获取并更新本地列表。

# 服务下线 Cancel

**官方解释**：[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect)客户端在程序关闭时向[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect)服务器发送取消请求。 发送请求后，该客户端实例信息将从服务器的实例注册表中删除。该下线请求不会自动完成，它需要调用以下内容：DiscoveryManager.getInstance().shutdownComponent();

**结合中介理解**：房东 (提供者 Eureka Client Provider) 告诉中介  (服务器 Eureka Server) 我的房子不租了，中介之后就将注册的房屋信息从列表中剔除。

# 服务剔除 Eviction

**官方解释**：在默认的情况下，当[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect)客户端连续90秒(3个续约周期)没有向[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect)服务器发送服务续约，即心跳，[Eureka](https://mp.weixin.qq.com/s?__biz=MzUzMTA2NTU2Ng==&mid=2247486260&idx=1&sn=8f14c0c191d6f8df6eb34202f4ad9708&scene=21#wechat_redirect)服务器会将该服务实例从服务注册列表删除 ，即服务剔除。

**结合中介理解**：房东(提供者 Eureka Client Provider) 会定期联系中介 (服务器 Eureka Server) 告诉他我的房子还租(续约)，如果中介  (服务器 Eureka Server) 长时间没收到提供者的信息，那么中介会将他的房屋信息给下架(服务剔除)。