

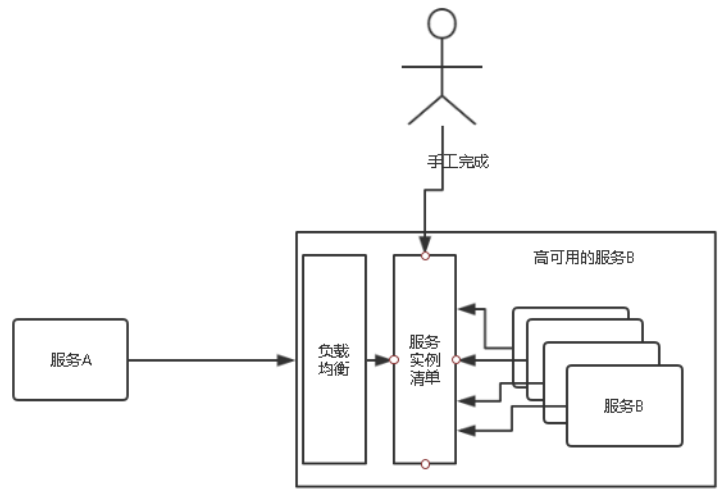
服务治理：Spring Cloud Eureka

什么是服务治理

服务治理可以说是微服务架构中最为核心和基础的模块，它主要用来实现各个微服务实例的自动化注册与发现。

为什么需要服务治理模块

在最初构建微服务系统的时候可能服务并不多，我们可以通过做一些静态配置来完成服务调用

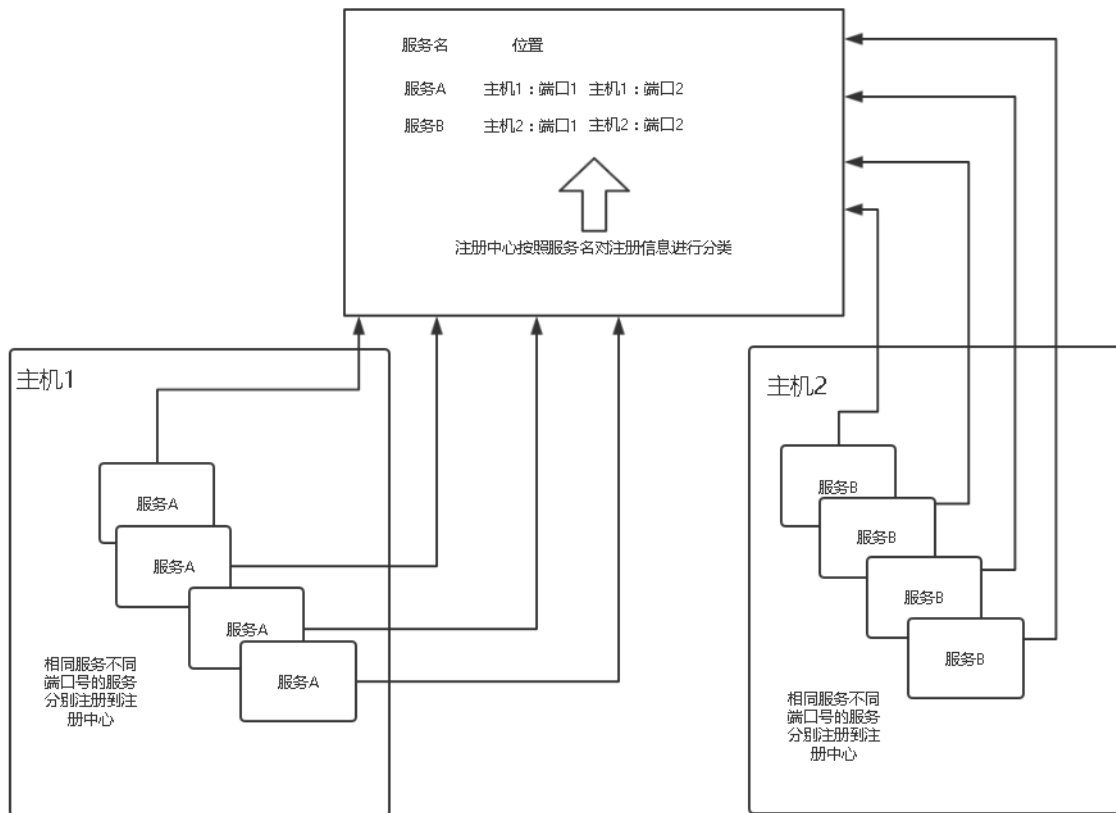


此时看着一切都还正常。

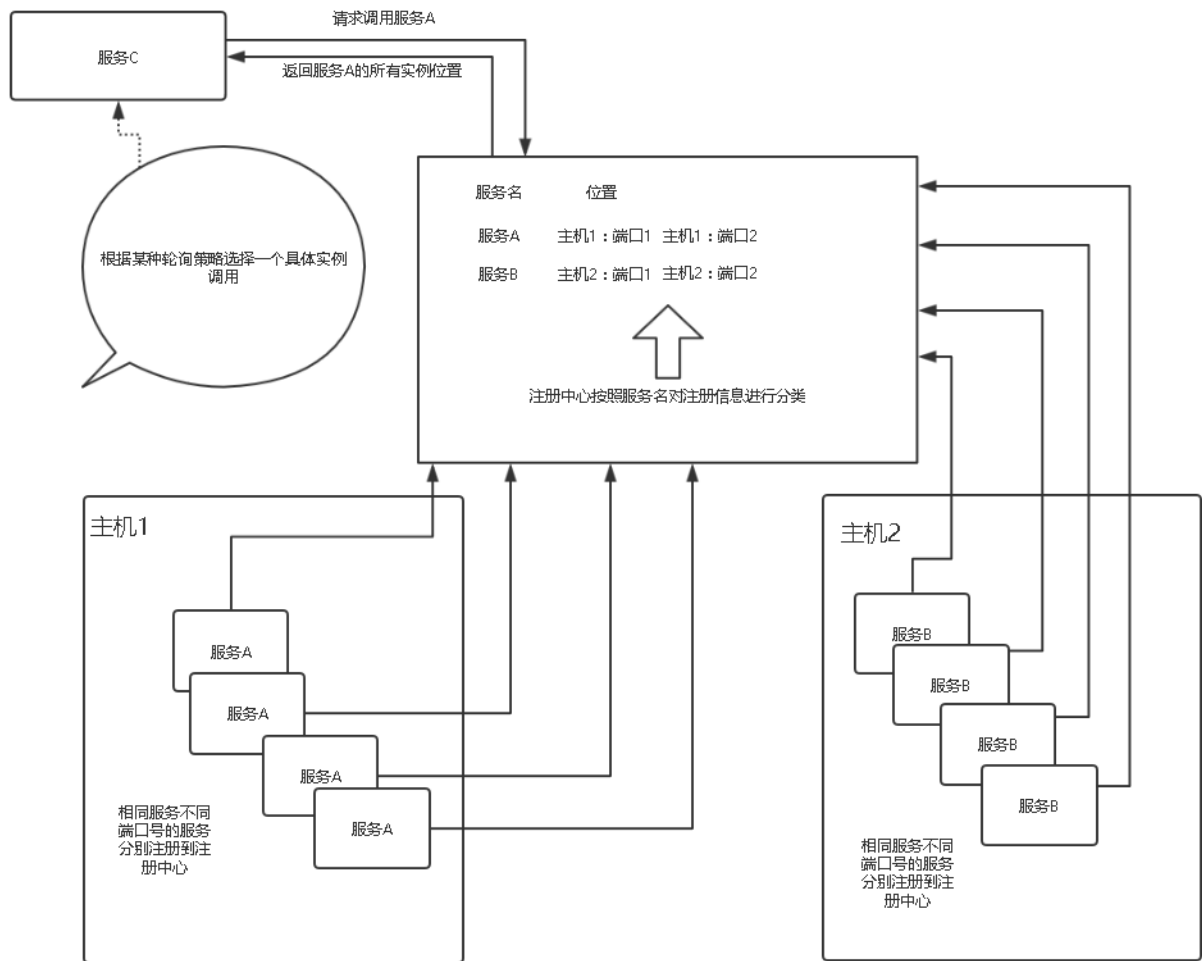
随着项目逐渐接近尾声，维护人员需要维护的服务越来越多，越来越复杂，最终形成大量的配置文件，维护将会变得越来越困难。此时，微服务应用实例自动化管理框架变得至关重要。

服务治理框架需要完成什么任务

- 服务注册：在服务治理框架中，通常会构建一个注册中心，每个服务单元向注册中心登记自己提供的服务，将主机与端口号、版本号、通信协议等一些附加信息告知注册中心，注册中心按服务名分类组织服务清单。



● 服务发现：我们的所有服务都已经注册到注册中心，并且在注册中心是按照服务名分类，并且由注册中心维护者服务的具体位置。所以调用方需要调用某个服务时，需要和注册中心咨询，注册中心会返回被调用方服务的所有具体位置，调用方在根据某种轮询策略选择一个具体位置进行服



务调用。

Netflix Eureka

Spring Cloud Eureka,使用Netflix Eureka来实现服务注册与发现，它既包含了服务端组件，也包含了客户端组件。

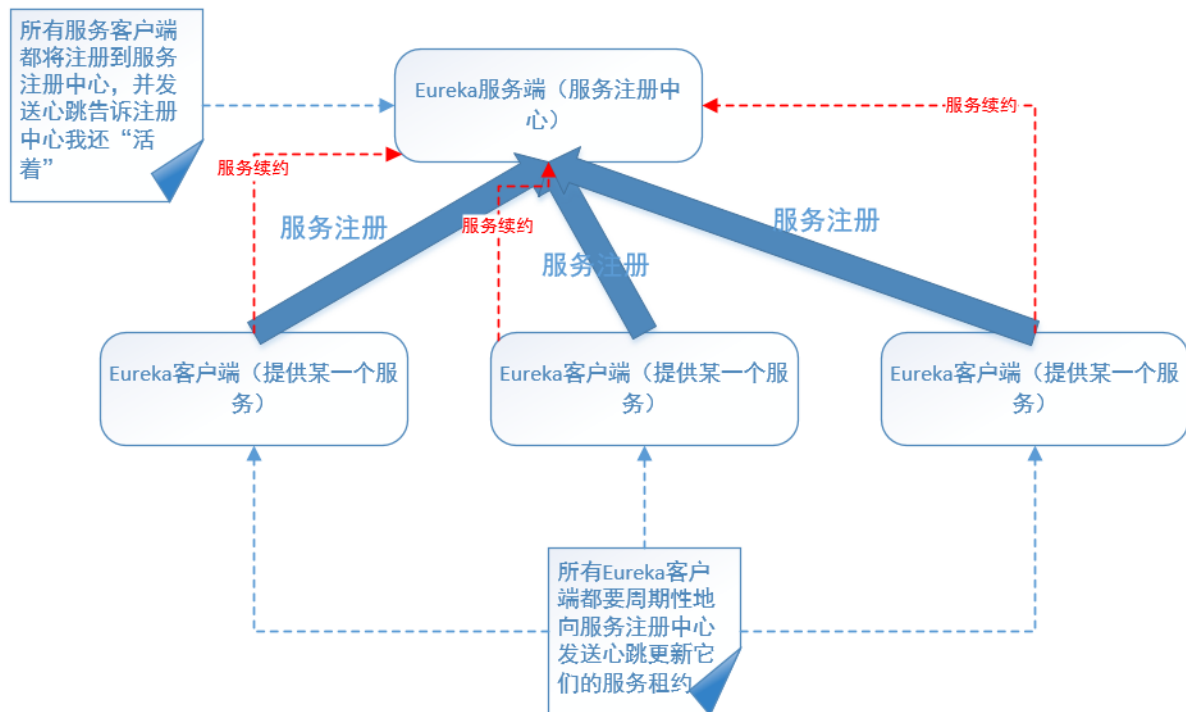
Eureka服务端

Eureka服务端，我们也称为服务注册中心，他同其他服务注册中心一样，支持高可用配置。它依托于强一致性提供良好的服务实例可用性，可以应对多种不同的故障场景。如果Eureka以集群方式部署，当集群中有分片出现故障时，那么Eureka就转入自我保护模式。它允许在分片故障期间继续提供服务的发现和注册，当故障分片恢复运行时，集群中的其他分片会把它们的状态再次同步回来。

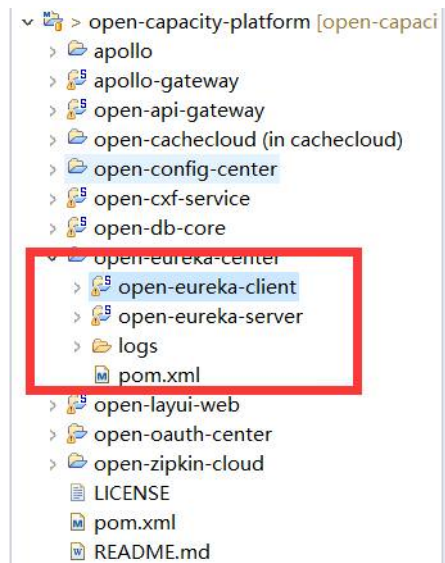
Eureka客户端

Eureka客户端，主要处理服务的注册与发现。客户端服务通过注解和参数配置的方式，嵌入在客户端应用程序的代码中，在应用程序运行时，Eureka客户端向注册中心注册自身提供的服务并周期性地发送心跳来更新它的服务租约。同时，他也能从服务端查询当前注册的服务信息并把它们缓存到本地并周期性地刷新服务状态。

服务端与客户端的关系

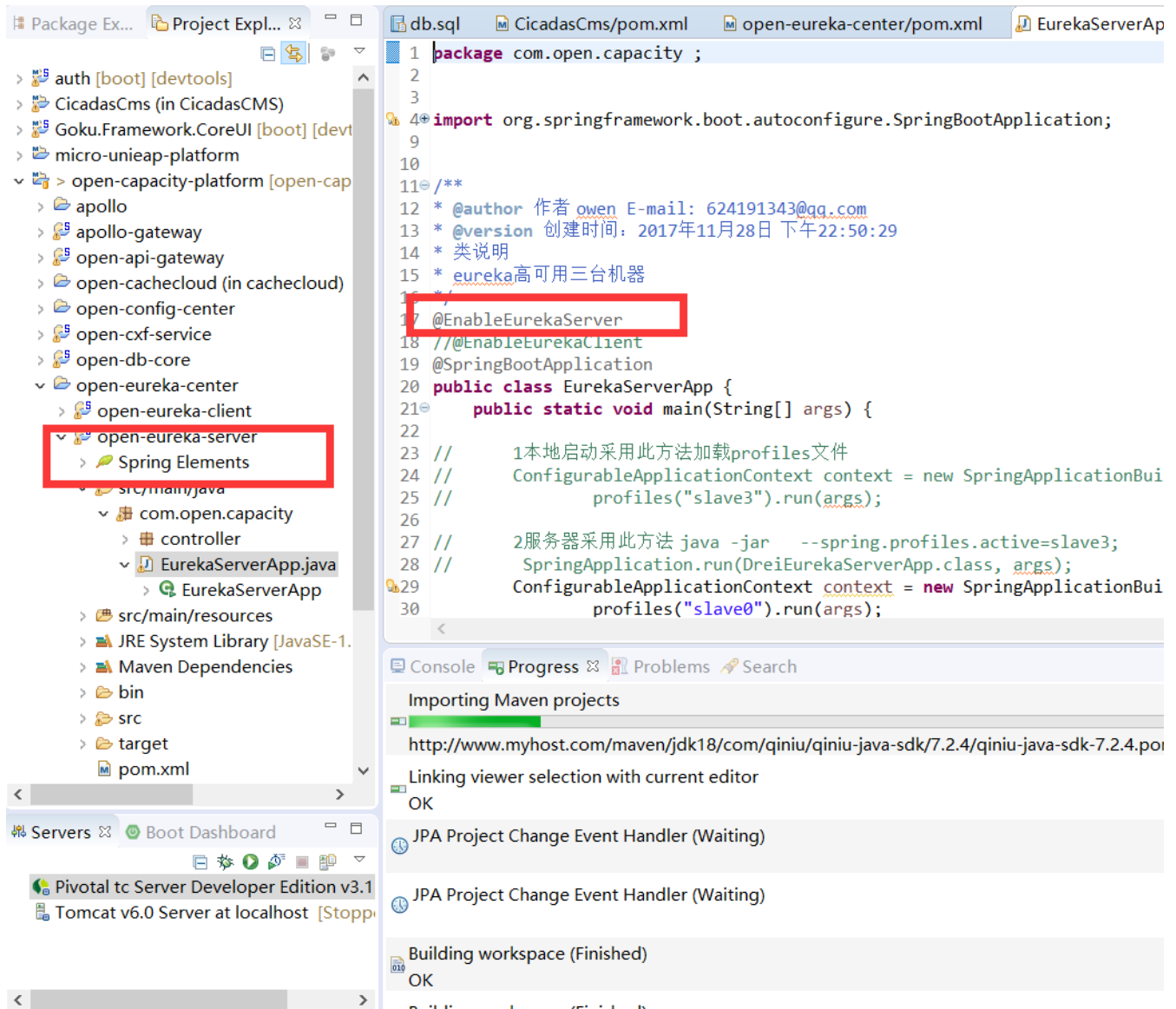


搭建服务注册中心



1. 创建Spring Boot工程, 命名为open-eureka-center, 并在pom中加入必要依赖, 如下图:

2. 通过@EnableEurekaServer注解启动一个服务注册中心提供给其他应用进行对话。在Spring boot应用中添加这个注解就能开启此功能。



在默认情况下，该服务注册中心也会将自己作为客户端来尝试注册它自己，所以我们需要禁用它的客户端注册行为，只需在application.properties中增加如下配置：

```
spring.application.name=open-eureka-server
server.port=1111
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
eureka.client.serviceUrl.defaultZone=http://127.0.0.1:1111/eureka
```

- spring.application.name是服务名称，会在服务注册中心中看到这个属性的值，也是服务间调用时使用的名字。
- server.port是该服务启动时所使用的端口号。
- eureka.client.register-with-eureka因为该应用为注册中心，所以设置为false,代表不向服务注册中心注册自己。
- eureka.client.fetch-registry因为服务注册中心的职责就是维护服务服务实例，它并不需要去检索服务，所以设置为false。

完成上面的配置，在浏览器中输入<http://127.0.0.1:1111/>如图：



系统状态

| | | |
|------|---------|--------|
| 环境 | test | 当前时间 |
| 数据中心 | default | 运行 |
| | | 启用租约 |
| | | 续订阈值 |
| | | 续订 (最) |

服务副本

127.0.0.1

当前注册的服务实例

| 应用 | 申请 | 可用性区域 |
|---------|----|-------|
| 没有可用的实例 | | |

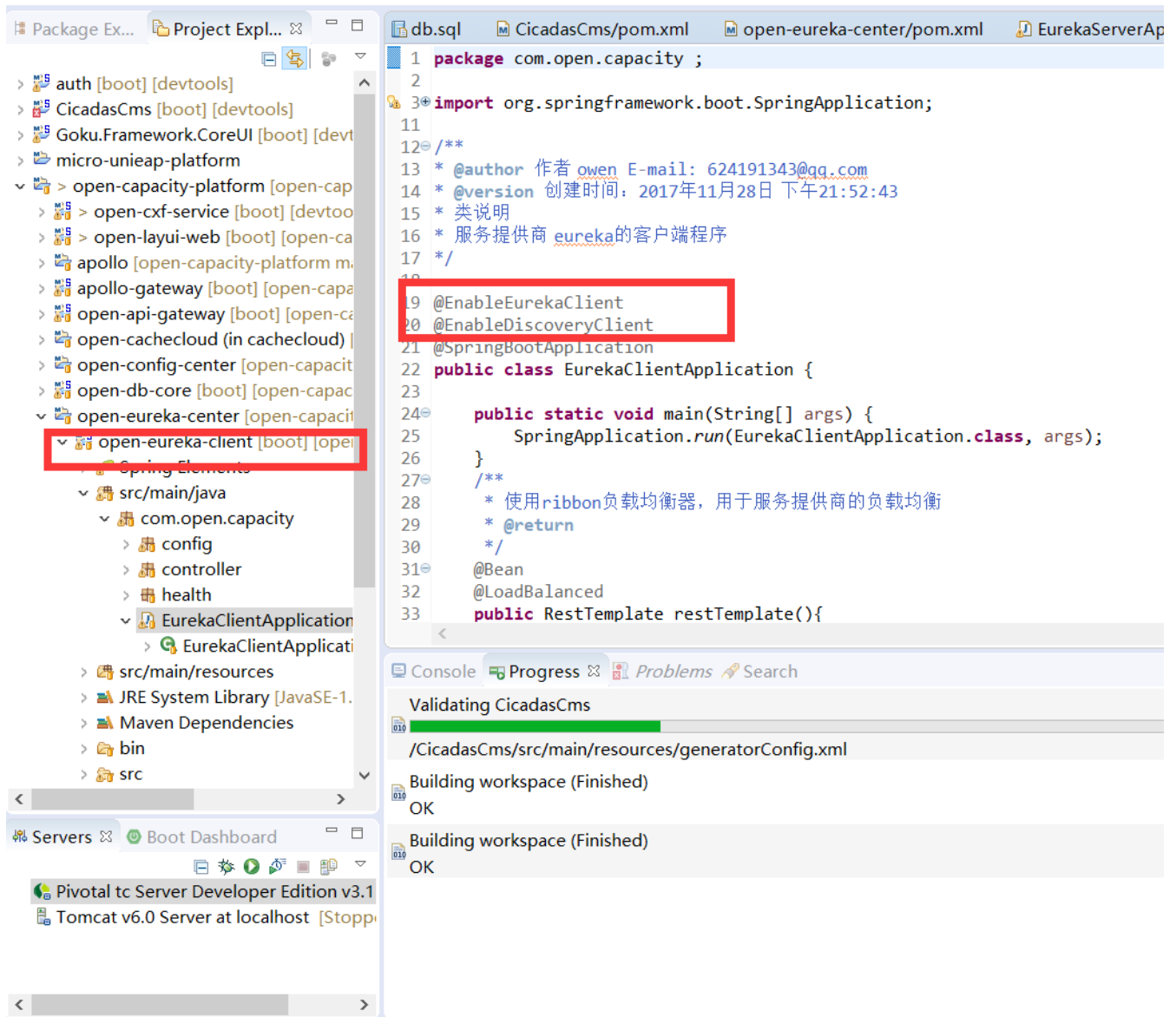
一般信息

| 名称 | 值 |
|----------------------|-------------|
| total-avail-memory | 260mb |
| environment | test |
| num-of-cpus | 4 |
| current-memory-usage | 154mb (59%) |

此时的Instances currently registered with Eureka栏是空的，因为还没有服务注册到注册中心。

注册服务提供者（Eureka客户端）

1.创建eureka客户端工程



2.修改HelloController类 在日志中打印服务的相关内容：

@RestController

public class HelloController{

private final Logger logger = Logger.getLogger(getClass());

@Autowired

private DiscoveryClient client;

@RequestMapping(value="/hello",method = RequestMethod.GET)

public String index() {

ServiceInstance instance = client.getLocalServiceInstance();

logger.info("/hello , host:" + instance.getHost() + " , service_id:" + instance.getServiceId());

return "hello world";

}

}

3.修改application.properties文件：spring.application.name=open-eureka-client eureka.client.serviceUrl.defaultZone=<http://127.0.0.1:1111/eureka/>


● eureka.client.serviceUrl.defaultZone属性指定服务注册中心的地址。 4.分别启动服务注册中心和open-eureka-client服务。结果如下图：

360安全浏览器 复制网址

http://127.0.0.1:1111/

收藏 手机收藏夹 谷歌 网址大全 游戏中心 东软数猫 杨大仙的 模仿国内 芋道源码 NATAPP Beck_ha 携程 Apo

服务注册和发现



系统状态

| | | |
|------|---------|--------|
| 环境 | test | 当前时间 |
| 数据中心 | default | 运行 |
| | | 启用租约 |
| | | 续订阈值 |
| | | 续订 (最后 |

服务副本

127.0.0.1

当前注册的服务实例

| 应用 | 申请 | 可用性区域 | 状态 |
|--------------------|---------|-------|---------------|
| OPEN-EUREKA-CLIENT | n/a (1) | (1) | UP (1) - open |

一般信息

| 名称 | 值 |
|--------------------|-------|
| total-avail-memory | 260mb |
| environment | test |
| num-of-cpus | 4 |

通过访问<http://127.0.0.1:7760/client/hello>直接向该服务发起请求

什么叫高可用

高可用一般指服务的冗余，一个服务挂了，可以自动切换到另一个服务上，不会影响到客户体验。

高可用注册中心

在微服务架构这样的分布式环境中，我们需要充分考虑发生故障的情况，所以在生产环境中必须对各个组件进行高可用部署，对于微服务如此，对于服务中心也一样。Eureka Server的设计一开始就考虑了高可用问题，在Eureka的服务治理设计中，所有节点既是服务提供方，也是服务消费方，服务注册中心也不例外。在前一篇随笔中用到过这样的配置：

eureka.client.register-with-eureka=false

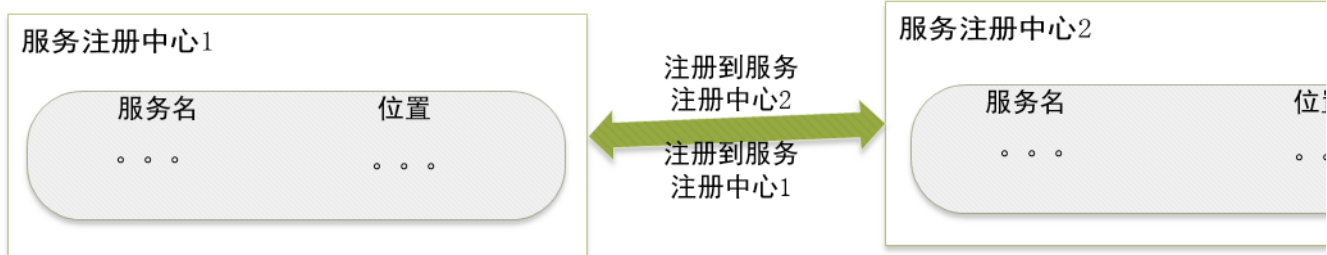
eureka.client.fetch-registry=false

现在回顾一下上面提到的这两个属性的作用：

● eureka.client.register-with-eureka=false 设置为不将自己注册到服务注册中心（默认是true）

● eureka.client.fetch-registry=false 设置为不检索服务（默认是true,在单节点服务注册中心的情况下，服务注册中心并不需要检索自己的服务）

Eureka server的高可用实际上就是将自己作为服务向其他服务注册中心注册自己，这样就可以形成一组互相注册的服务注册中心，以实现服务清单的互相同步，达到高可用的效果。



常见问题

Eureka 常见问题

踢出已关停的节点

由于Eureka自我保护模式，以及心跳周期长的原因，常常会遇到Eureka Server不踢出已关停的节点的问题。eureka.server.enable-self-preservation、eureka.server.eviction-interval-timer-in-ms

多网卡环境下的IP选择问题

Eureka会选择IP合法(标准ipv4地址)、索引值最小(eth0, eth1中eth0优先)且不在忽略列表中(可在application.properties中配置忽略哪些网卡)的网卡地址作为服务IP。

eureka.instance.ip-address、eureka.instance.prefer-ip-address

服务感知慢

Eureka的wiki上有一句话，大意是一个服务启动后最长可能需要2分钟时间才能被其它服务感知。

eureka.instance.leaseRenewalIntervalInSeconds。（在生产中，最好坚持使用默认值，因为在服务器内部有一些计算，他们对续约做出假设。）



eureka restful api

查看eureka的状态

<http://127.0.0.1:7768/eureka/status>

查看有多少服务

<http://127.0.0.1:7778/eureka/apps>

查看某个服务多少实例

<http://127.0.0.1:7768/eureka/apps?name=OPEN-EUREKA-CLIENT>

查看某个实例的状态

<http://127.0.0.1:7768/eureka/apps/OPEN-EUREKA-CLIENT/open-eureka-client:192.168.45.1:7778>

暂停微服务

<http://127.0.0.1:7768/pause>

查看某个实例的状态

<http://127.0.0.1:7768/resume>