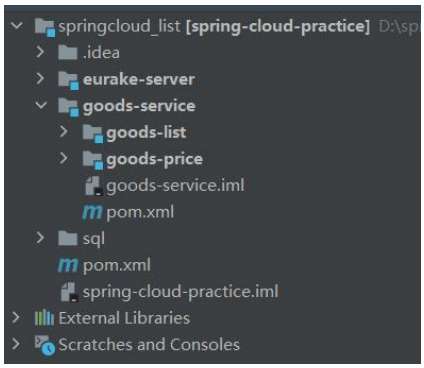
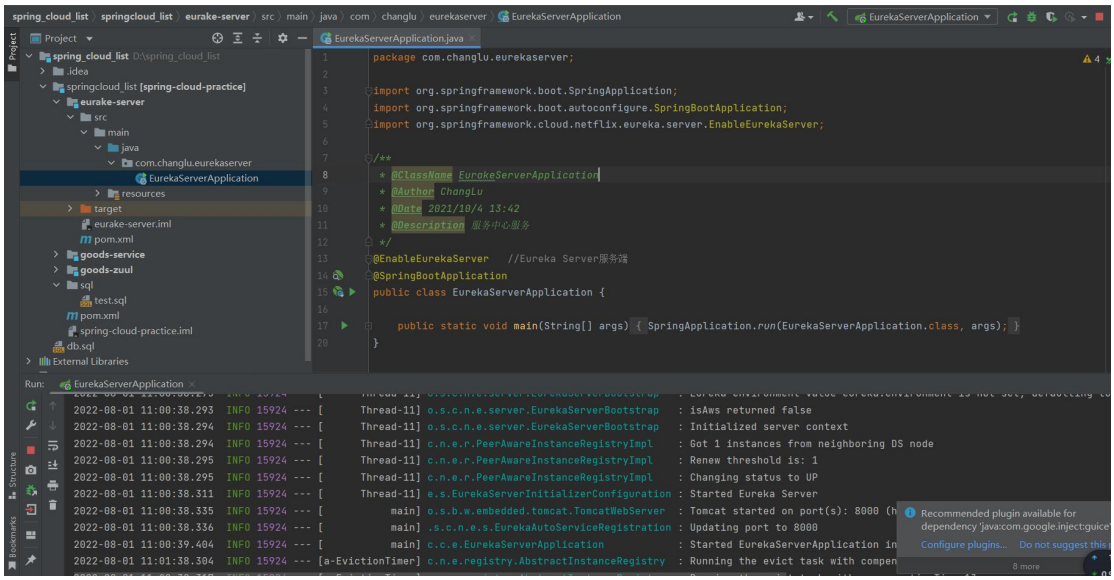
本项目选用 wms 系统的货品查询功能，包括查询货品列表和货品价格等。项目结构如下图所示：

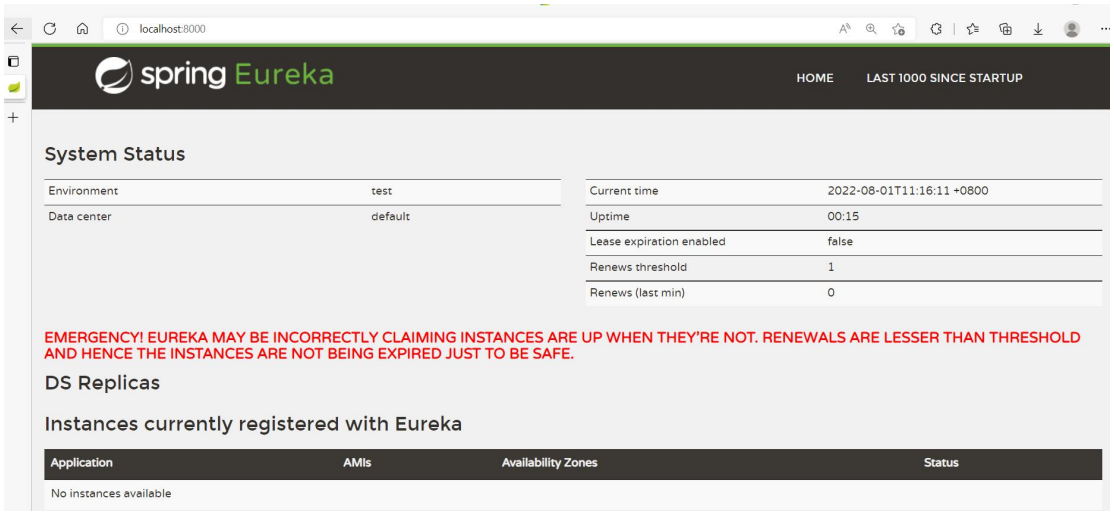


一、创建 SpringCloud 项目

1. eureka 配置成功（提供注册服务）



2. 进入 eureka 管理界面



General Info

Name	Value
total-avail-memory	336mb
environment	test
num-of-cpus	8
current-memory-usage	80mb (23%)
server-uptime	00:15
registered-replicas	
unavailable-replicas	
available-replicas	

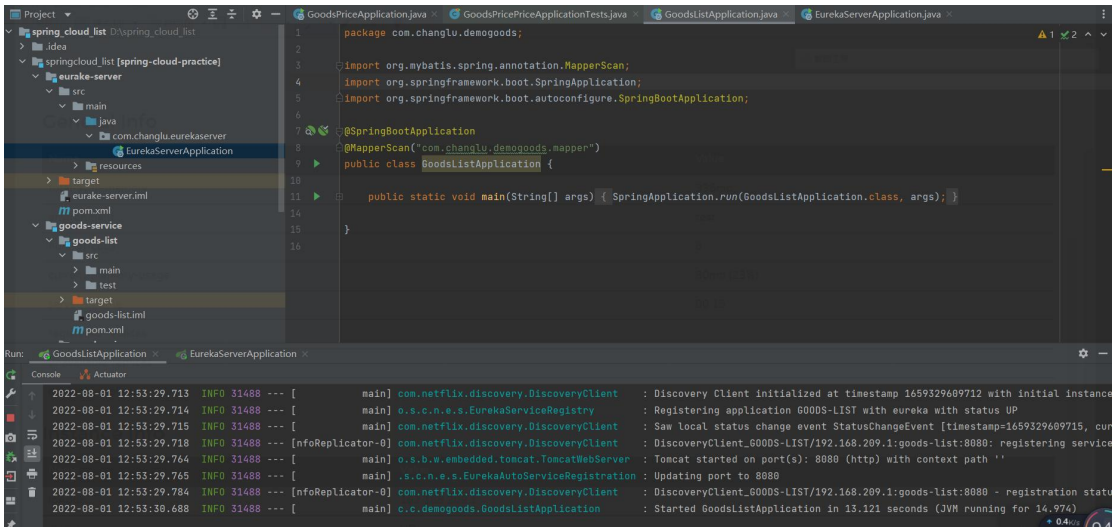
Instance Info

Name	Value
ipAddr	192.168.209.1
status	UP

二、客户端 client 配置

1. 客户端 client 配置成功

（1）货物信息列表查询



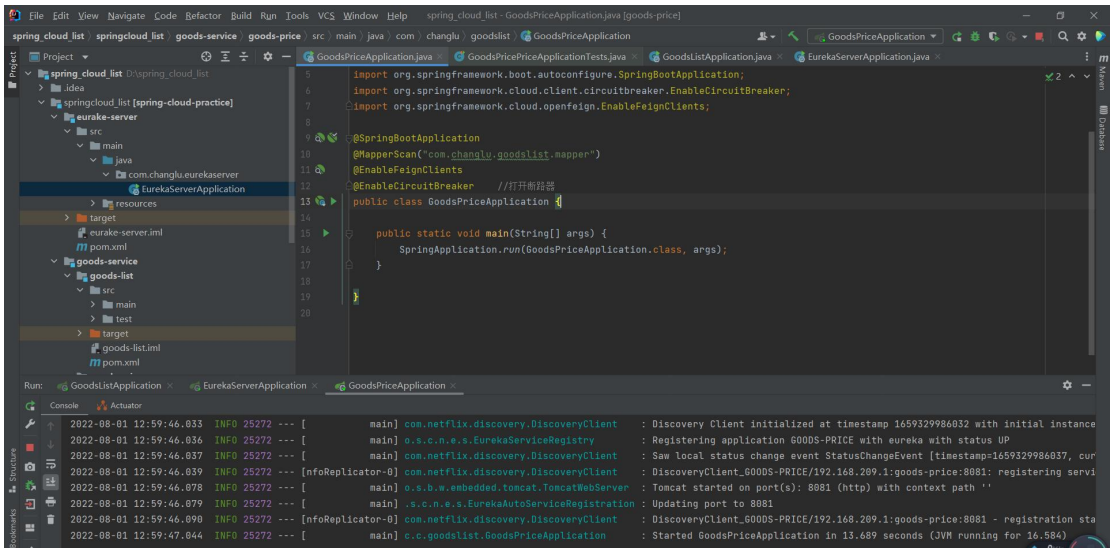
当再次进入服务注册页面时，可见服务提供者已被注册进服务注册者

DS Replicas

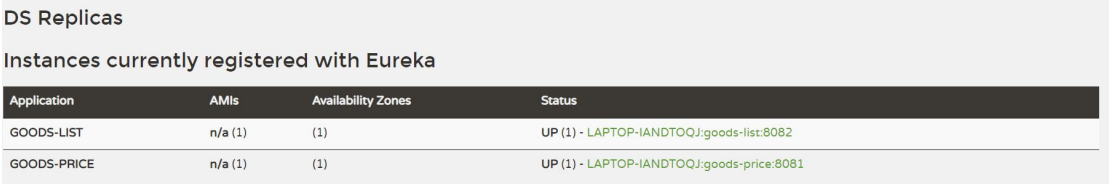
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
GOODS-LIST	n/a (1)	(1)	UP (1) - LAPTOP-IANDTOQJ:goods-list:8082

（2）货物价格查询

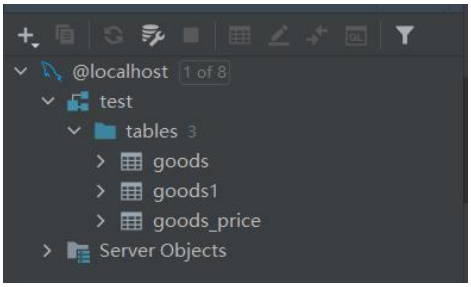


当再次进入服务注册页面时，同理可见可见服务提供者已被注册进服务注册者

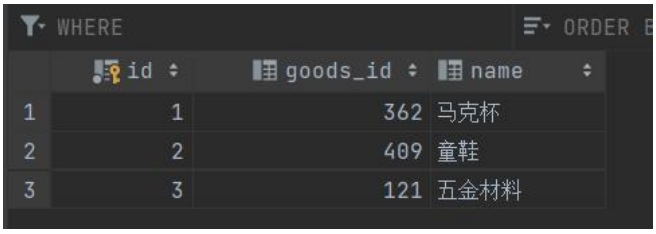


三、创建数据库（Mysql）

1. 在 MySQL 中创建数据库，包括三个表，分别为货物列表、货物有效标识表和货物价格列表。



	id	goods_id	name	valid
1	1	362	马克杯	1
2	2	409	童鞋	1
3	3	121	五金材料	0



	id	goods_id	name
1	1	362	马克杯
2	2	409	童鞋
3	3	121	五金材料

WHERE ORDER BY

	id	goods_id	price
1	1	362	348.50
2	2	409	399.68
3	3	121	266.78

四、Mybatis 使用

如在 mapper 文件中引入 mybatisplus 包中的基类

```
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.changlu.goodslist.pojo.GoodsPrice;
import org.apache.ibatis.annotations.Select;
import com.changlu.goodslist.pojo.Goods1;
```

并在继承基类的基础上构造所需的 mapper 文件

```
public interface GoodsPriceMapper extends BaseMapper<GoodsPrice> {
    @Select("select * from goods1")
    List<Goods1> getGoods1List();
}
```

五、运行测试项目

1. 先运行 Eureka 服务的启动类，然后运行货物信息查询功能（GoodsListApplication）的启动类，其中货物信息模块的 yml 文件中该服务提供方的 port 端口号设置为 8082

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/test?useSSL=false
    username: root
    password: ab123456
  application:
    name: goods-list   # 应用名

server:
  port: 8082
```

在浏览器中输入 url：https://localhost:8082/goods/list
可得如下运行界面：

localhost:8082/goods/list

[{"id":1,"goodsId":362,"name":"马克杯","valid":1},{"id":2,"goodsId":409,"name":"童鞋","valid":1},{"id":3,"goodsId":121,"name":"五金材料","valid":0}]

2.启动 Eureka 服务的启动类后，再启动货物价格查询功能（GoodsPriceApplication）的启动类，其中货物价格信息模块的 yml 文件中该服务提供方的 port 端口号设置为 8081

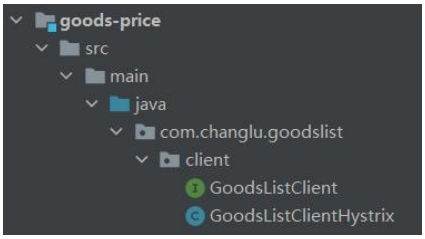
```yaml
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/test?useSSL=false
    username: root
    password: ab123456
  application:
    name: goods-price   # 应用名

server:
  port: 8081
```

在浏览器中输入 url：https://localhost:8081/goods/list
可得如下运行界面：

六、Feign 远程服务

在 goodsprice 模块的客户端（client）包中，创建了接口类 GoodsListClient 及其实现类。



在接口类 GoodsListClient 中，通过直接远程调用 goodsList 模块的 getList()方法，可以直接实现 GoodsListClient 类，不必再通过继承接口人工实现了。

```java
@FeignClient(value = "goods-list",fallback = GoodsListClientHystrix.class)
@Primary
public interface GoodsListClient {

    @GetMapping("/goods/list")
    List<Goods1> getList();

}
```

七、SpringCloud 项目打成 Jar 包

利用 maven 中的 clean 和 package 组件，将该项目的各个模块打成 jar 包存放在 target 文件夹中，方便后面进行部署。

eurake-server-1.0.0.jar	2022/8/2 13:59	Executable Jar File	46,498 KB
goods-list-1.0.0.jar	2022/8/2 13:59	Executable Jar File	48,008 KB
goods-price-1.0.0.jar	2022/8/2 13:59	Executable Jar File	51,239 KB

八、利用 Docker 部署项目

在 vmvare 虚拟机中的 ubuntu 环境中安装 docker 并进行部署。

创建 Dockerfile 文件：

Dockerfile_list	2022/8/2 14:14	文件	1 KB
Dockerfile_price	2022/8/2 14:19	文件	1 KB
Dockerfile_server	2022/8/2 14:19	文件	1 KB

在 ubuntu 中利用 docker 命令创建镜像：

```
root@ubuntu:/tmp/springcloud# docker build -t server -f Dockerfile_server .
Sending build context to Docker daemon  149.2MB
Step 1/5 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/5 : VOLUME /tmp
 ---> Using cache
 ---> d8bb3f8d2698
Step 3/5 : ADD eurake-server-1.0.0.jar /tmp/springcloud
 ---> 9650bcf04adf
Step 4/5 : EXPOSE 8000
 ---> Running in 9d7c5528b21d
Removing intermediate container 9d7c5528b21d
 ---> 8c396dcfa207
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","
/boot.jar"]
 ---> Running in f1b49bb325b3
Removing intermediate container f1b49bb325b3
 ---> 95ca99124413
Successfully built 95ca99124413
Successfully tagged server:latest
```

```
root@ubuntu:/tmp/springcloud# docker build -t list -f Dockerfile_list .
Sending build context to Docker daemon  149.2MB
Step 1/5 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/5 : VOLUME /tmp
 ---> Using cache
 ---> d8bb3f8d2698
Step 3/5 : ADD goods-list-1.0.0.jar /tpm/springcloud
 ---> 50e8fdb721e3
Step 4/5 : EXPOSE 8082
 ---> Running in 98c92097d7ea
Removing intermediate container 98c92097d7ea
 ---> 180d7602bd26
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","
/boot.jar"]
 ---> Running in f93ab529cf98
Removing intermediate container f93ab529cf98
 ---> 7fcb6131bba5
Successfully built 7fcb6131bba5
Successfully tagged list:latest
```

```
root@ubuntu:/tmp/springcloud# docker build -t price -f Dockerfile_price .
Sending build context to Docker daemon  149.2MB
Step 1/5 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/5 : VOLUME /tmp
 ---> Using cache
 ---> d8bb3f8d2698
Step 3/5 : ADD goods-price-1.0.0.jar /tpm/springcloud
 ---> f9ce9d7d9e5b
Step 4/5 : EXPOSE 8081
 ---> Running in 7502d59c61b9
Removing intermediate container 7502d59c61b9
 ---> 8bb41c0d6052
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","
/boot.jar"]
 ---> Running in 86ff0893483a
Removing intermediate container 86ff0893483a
 ---> 88c24cb4058f
Successfully built 88c24cb4058f
Successfully tagged price:latest
```

查看镜像列表：

```
root@ubuntu:/tmp/springcloud# docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
price         latest    88c24cb4058f   2 hours ago      696MB
list          latest    7fcb6131bba5   2 hours ago      692MB
server        latest    95ca99124413   2 hours ago      691MB
mysql         8.0       38643ad93215   6 days ago       446MB
hello-world   latest    feb5d9fea6a5   10 months ago    13.3kB
java          8         d23bdf5b1b1b   5 years ago      643MB
```

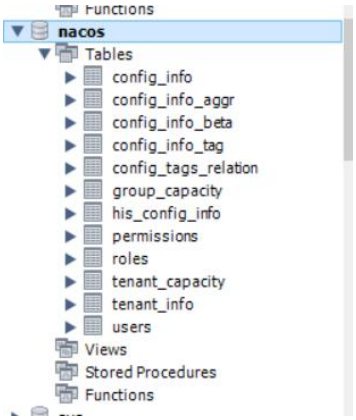
Docker 命令运行容器，并查询容器

```
root@ubuntu:/tmp/springcloud# docker run -p 8000:8000 -itd --name cserver2 --pri
vileged=true server /bin/bash
a60806f87191180cdc2a50fa7cf8639189f4e52af5794a61683e973b16f8a120
root@ubuntu:/tmp/springcloud# docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS
                    PORTS     NAMES
a60806f87191   server        "java -Djava.securit…"   14 seconds ago   Exited (1
) 10 seconds ago             cserver2
eacb85cbfc6c   server        "java -Djava.securit…"   7 minutes ago    Exited (1
```

九、nacos 配置

1. 创建 nacos 数据库，并运行 conf 文件下的 nacos-msql 文件

application.properties	2020/11/2 19:07	PROPERTIES 文件	7 KB
application.properties.example	2020/11/2 19:07	EXAMPLE 文件	7 KB
cluster.conf.example	2020/10/29 10:41	EXAMPLE 文件	1 KB
nacos-logback.xml	2020/11/2 19:07	XML 文档	26 KB
nacos-mysql.sql	2022/8/2 18:39	SQL Text File	11 KB
schema.sql	2020/10/29 10:41	SQL Text File	8 KB



2. 启动 nacos

```
C:\WINDOWS\system32\cmd.exe
"nacos is starting with cluster"

Nacos 1.4.0
Running in cluster mode, All function modules
Port: 8848
Pid: 22792
Console: http://192.168.209.1:8848/nacos/index.html

https://nacos.io

2022-08-02 18:43:30,696 INFO The server IP list of Nacos is [192.168.16.101:8847, 192.168.16.102, 192.168.16.103]
```

```
2022-08-02 18:44:02,103 INFO Nacos is starting...
2022-08-02 18:44:03,114 INFO Nacos is starting...
2022-08-02 18:44:03,567 INFO Nacos Log files: D:\nacos\nacos-server-1.4.0\nacos\logs
2022-08-02 18:44:03,568 INFO Nacos Log files: D:\nacos\nacos-server-1.4.0\nacos\conf
2022-08-02 18:44:03,574 INFO Nacos Log files: D:\nacos\nacos-server-1.4.0\nacos\data
2022-08-02 18:44:03,575 INFO Nacos started successfully in cluster mode. use external storage
```

3. 通过本地网址访问 nacos