SpringCloud之二
Spring Cloud Eureka学习

看很多网上文档都要使用Eureka来进行服务注册发现，所以先选择Eureka来进行学习。

Spring Cloud Eureka
一、Eureka介绍

Spring Cloud Eureka 是 Spring Cloud Netflix 微服务套件的一部分，基于 Netflix Eureka 做了二次封装，主要负责完成微服务架构中的服务治理功能。

除了用Eureka来做注册中心，我们还可以使用Consul,Etcd,Zookeeper等等来作为服务的注册中心。

有用过dubbo的同学应该清楚，dubbo中也有几种注册中心，有基于Zookeeper的，有基于redis的等等，用的最多的还是Zookeeper方式。

至于使用哪种方式，其实都是可以的，**注册中心无非就是管理所有服务的信息和状态**。

用我们生活中的列子来说明的话，我觉得12306比较合适。

首先12306就好比一个注册中心，N量火车都注册在了12306上面，我们顾客就好比调用的客户端，当我们需要坐火车时，我们会去12306上看有没有票，有票就可以购买，然后拿到火车的班次，时间等等，最后出发。

程序也是一样，当你需要调用某一个服务的时候，你会先去Eureka中去拉取服务列表，查看你调用的服务在不在其中，在的话就拿到服务地址，端口，等等信息，然后调用。

注册中心带来的好处就是你不需要知道有多少提供方，你只需要关注注册中心即可，你不必关系有多少火车在运行，你只需要去12306上看有没有票可以买就可以。

二、Eureka Server

1、创建项目

- 首先创建一个maven工程(或者用http://start.spring.io/来创建一个spring cloud项目)
- 在pom.xml增加依赖(如果下载包特别慢可以考虑使用阿里云的maven镜像服务器http://cxytiandi.com/blog/detail/5321)

代码示例如下，pom.xml：

```
<!-- Spring Boot -->
<parent>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-parent</artifactId>
 <version>1.5.4.RELEASE</version>
 <relativePath />
</parent>
<dependencies>
 <!-- eureka -->
  <dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-eureka-server</artifactId>
 </dependency>
</dependencies>
```

```
<!-- Spring Cloud -->
<dependencyManagement>
 <dependencies>
 <dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-dependencies</artifactId>
 <version>Dalston.SR1</version>
 <type>pom</type>
 <scope>import</scope>
 </dependency>
 </dependencies>
</dependencyManagement>
```

无需怀疑，这个肯定的使用了springboot来进行构建。

2、创建启动类

EurekaServerApplication.java

```
package xyz.jiangnanke.eureka;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

/**
 * @Auther: zhengfeng
 * @Date: 2018\12\18 0018 16:20
 * @Description: 服务注册中心
 */
@EnableEurekaServer
@SpringBootApplication
public class EurekaServerApplication {

 public static void main(String[] args) {
 SpringApplication.run(EurekaServerApplication.class, args);
 }

}
```

3、设置Eureka监听类

EurekaStateChangeListener .java

```java
package xyz.jiangnanke.eureka;

import com.netflix.appinfo.InstanceInfo;
import org.springframework.cloud.netflix.eureka.server.event.EurekaInstanceCanceledEvent;
import org.springframework.cloud.netflix.eureka.server.event.EurekaInstanceRegisteredEvent;
import org.springframework.cloud.netflix.eureka.server.event.EurekaInstanceRenewedEvent;
import org.springframework.cloud.netflix.eureka.server.event.EurekaRegistryAvailableEvent;
import org.springframework.cloud.netflix.eureka.server.event.EurekaServerStartedEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;

/**
 * @Auther: zhengfeng
 * @Date: 2018\12\18 0018 16:21
 * @Description: Eureka事件监听
 */
@Component
public class EurekaStateChangeListener {

 @EventListener
 public void listen(EurekaInstanceCanceledEvent event) {
 System.err.println(event.getServerId() + "\t" + event.getAppName() + "服务下线");
 }

 @EventListener
 public void listen(EurekaInstanceRegisteredEvent event) {
 InstanceInfo instanceInfo = event.getInstanceInfo();
 System.err.println(instanceInfo.getAppName() + "进行注册");
 }

 @EventListener
 public void listen(EurekaInstanceRenewedEvent event) {
 System.err.println(event.getServerId() + "\t" + event.getAppName() + "服务进行续约");
 }
```

```
35
36  @EventListener
37  public void listen(EurekaRegistryAvailableEvent event) {
38  System.err.println("注册中心 启动");
39  }
40
41  @EventListener
42  public void listen(EurekaServerStartedEvent event) {
43  System.err.println("Eureka Server 启动");
44  }
45 }
```

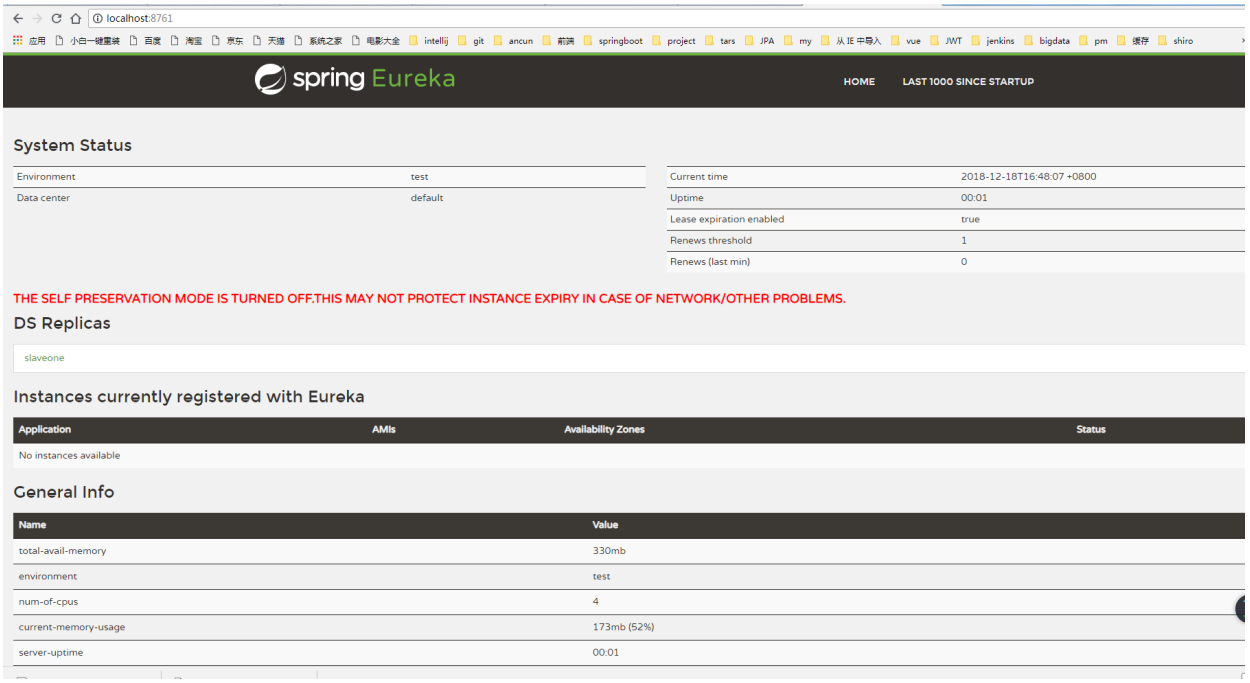
4、设置环境信息

application.properties

```
1  #服务端口
2  server.port=8761
3  #服务名称
4  spring.application.name=my-eureka
5  #服务地址绑定
6  eureka.instance.hostname=localhost
7  # 由于该应用为注册中心，所以设置为false，代表不向注册中心注册自己
8  eureka.client.register-with-eureka=false
9  # 由于注册中心的职责就是维护服务实例，他并不需要去检索服务，所以也设置为false
10  eureka.client.fetch-registry=false
11  # 关闭自我保护
12  eureka.server.enableSelfPreservation=false
13  eureka.server.eviction-interval-timer-in-ms=5000
14  security.basic.enabled=true
15  security.user.name=jiangnanke
16  security.user.password=123456
17
18  spring.profiles.active=master
```

5、启动

最后启动EurekaServerApplication，访问http://localhost:8761/就可以打开管理页面了。



(今天手痒，所以抽出了一个pom的父模块，导致代码有一点点改动)

三、Eureka Client

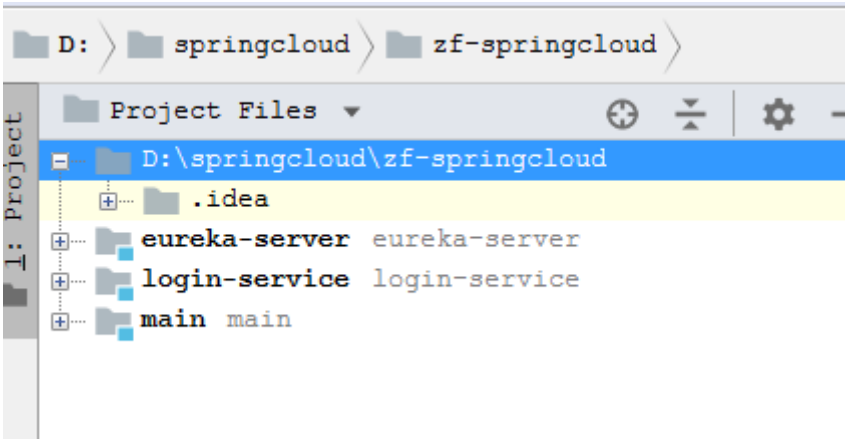
首先说明一下我的项目结构。

main

eureka-server

login-service （eureka-client）

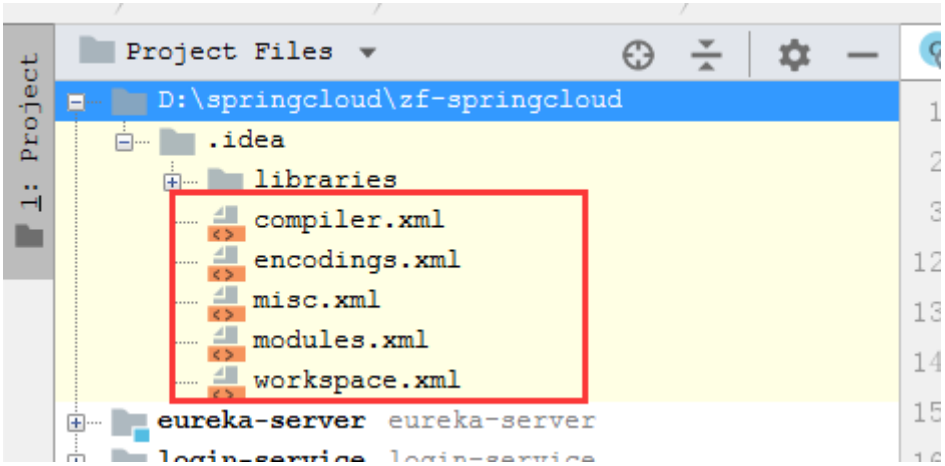
如图：



需要注意的地方

1、创建问题

如果发现删除模块之后再重新创建同名的模块，则需要删除project里面的一些配置信息，不然会发现一些很奇怪的问题，可以自己看看。



2、父子模块pom的配置

main模块的pom.xml

```
<?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>xyz.jiangnanke</groupId>
  <artifactId>main</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>

  <name>main</name>
  <description>main project for Spring Boot</description>

  <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.1.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>Finchley.RELEASE</spring-cloud.version>
```

```
</properties>

<modules>
<module>../eureka-server</module>
<module>../login-service</module>
</modules>

<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</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>
```

eureka-server的pom.xml

```
<?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>xyz.jiangnanke</groupId>
  <artifactId>eureka-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>eureka-server</name>
   <description>Demo project for Spring Boot</description>

   <parent>
   <groupId>xyz.jiangnanke</groupId>
   <artifactId>main</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   </parent>


   <dependencies>
   <dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
   </dependency>

   </dependencies>


</project>
```

login-service的pom.xml

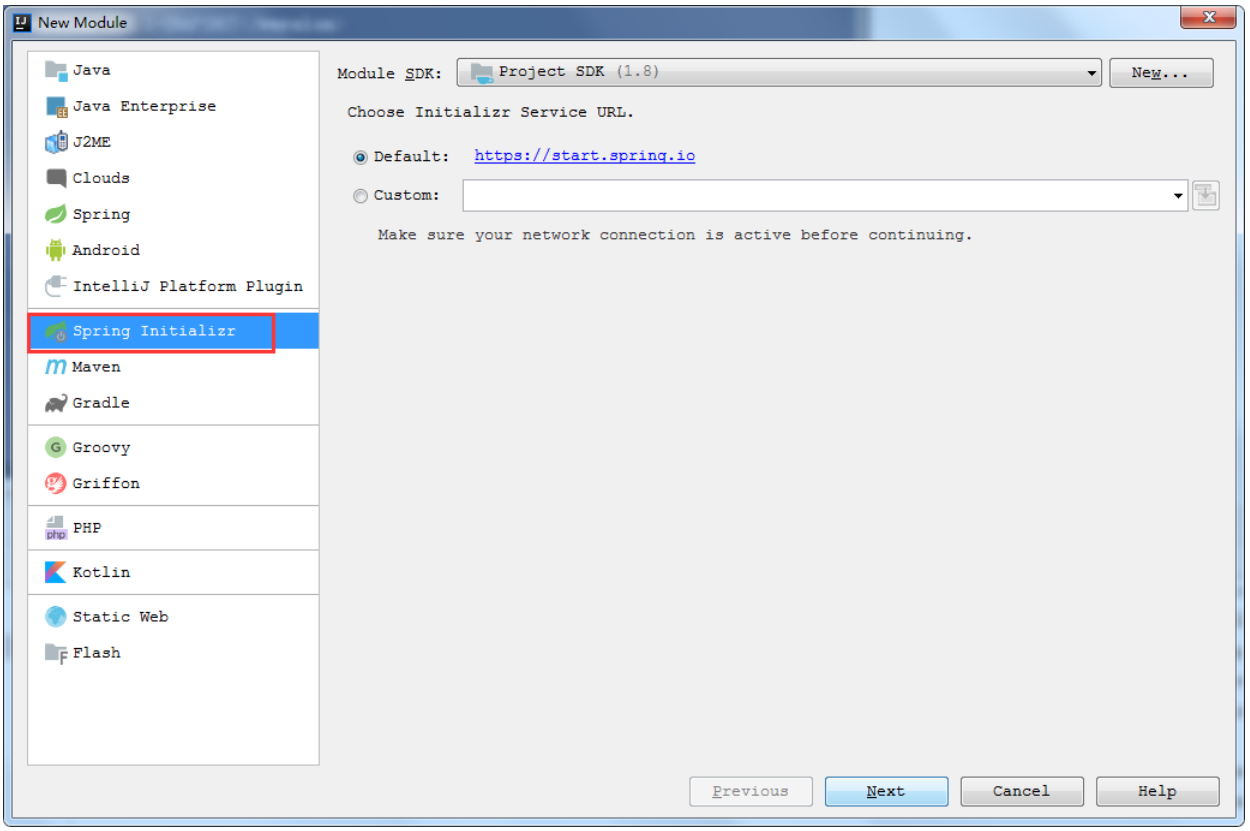
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

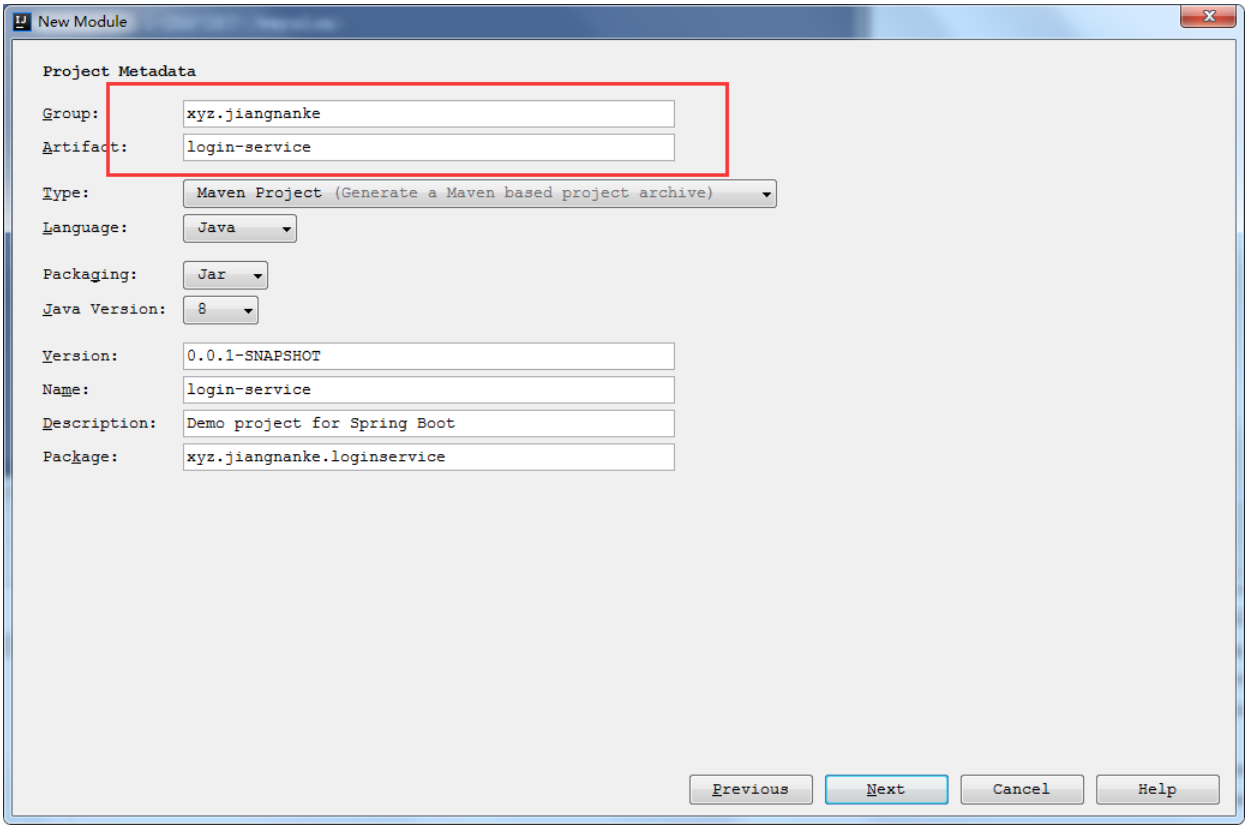
```
5
6  <groupId>xyz.jiangnanke</groupId>
7  <artifactId>login-service</artifactId>
8  <version>0.0.1-SNAPSHOT</version>
9  <name>login-service</name>
10  <description>Demo project for Spring Boot</description>
11
12  <parent>
13  <groupId>xyz.jiangnanke</groupId>
14  <artifactId>main</artifactId>
15  <version>0.0.1-SNAPSHOT</version>
16  </parent>
17
18  <dependencies>
19  <dependency>
20  <groupId>org.springframework.boot</groupId>
21  <artifactId>spring-boot-starter-web</artifactId>
22  </dependency>
23  <dependency>
24  <groupId>org.springframework.cloud</groupId>
25  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
26  </dependency>
27
28  </dependencies>
29
30
31 </project>
```

好了，接下来看login-service的搭建

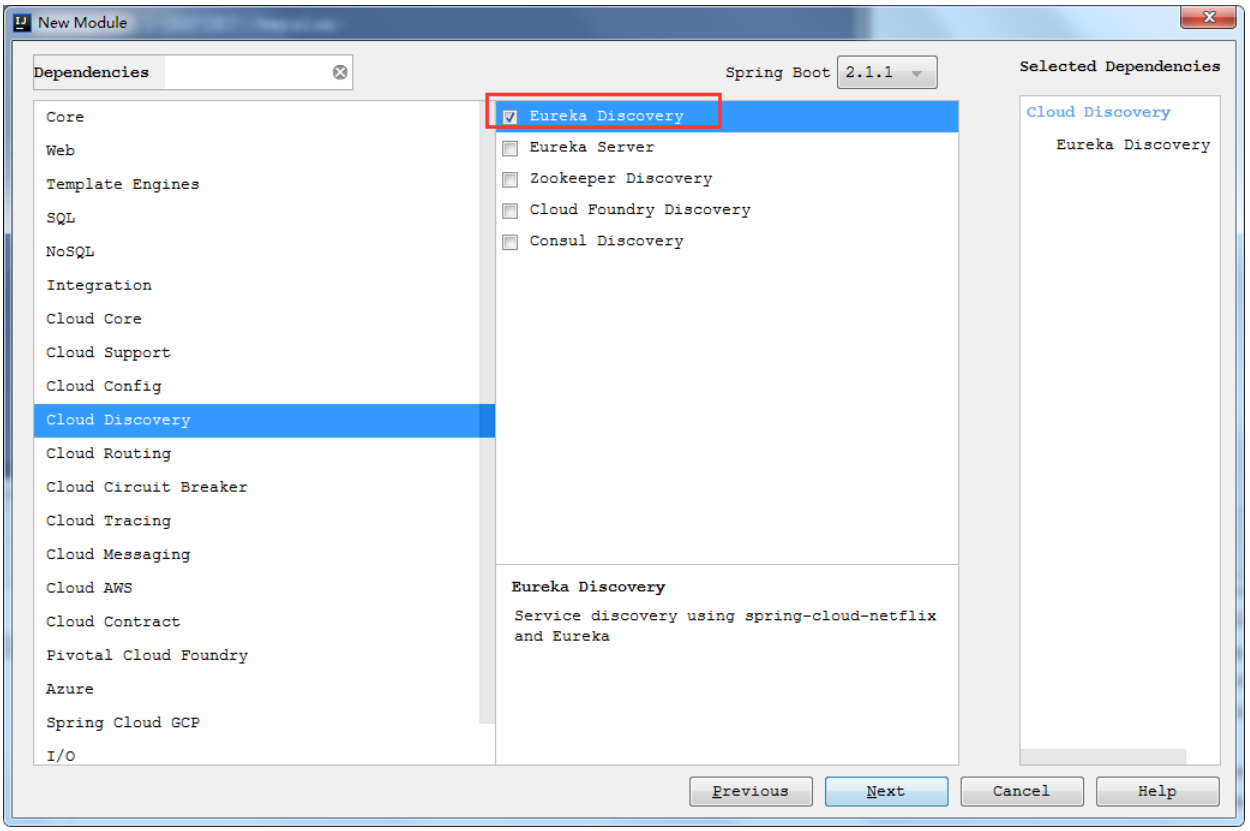
1、创建项目

类似于eureka-server创建一个springboot项目

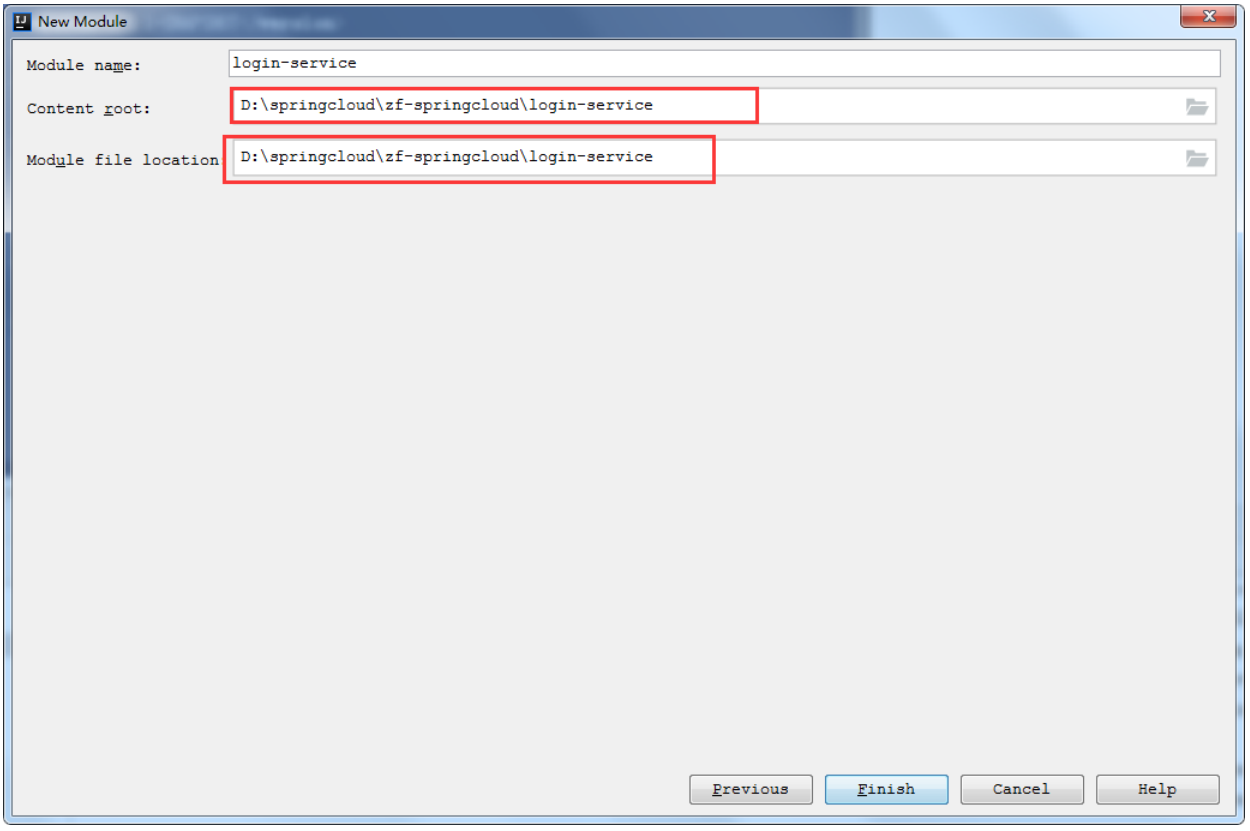




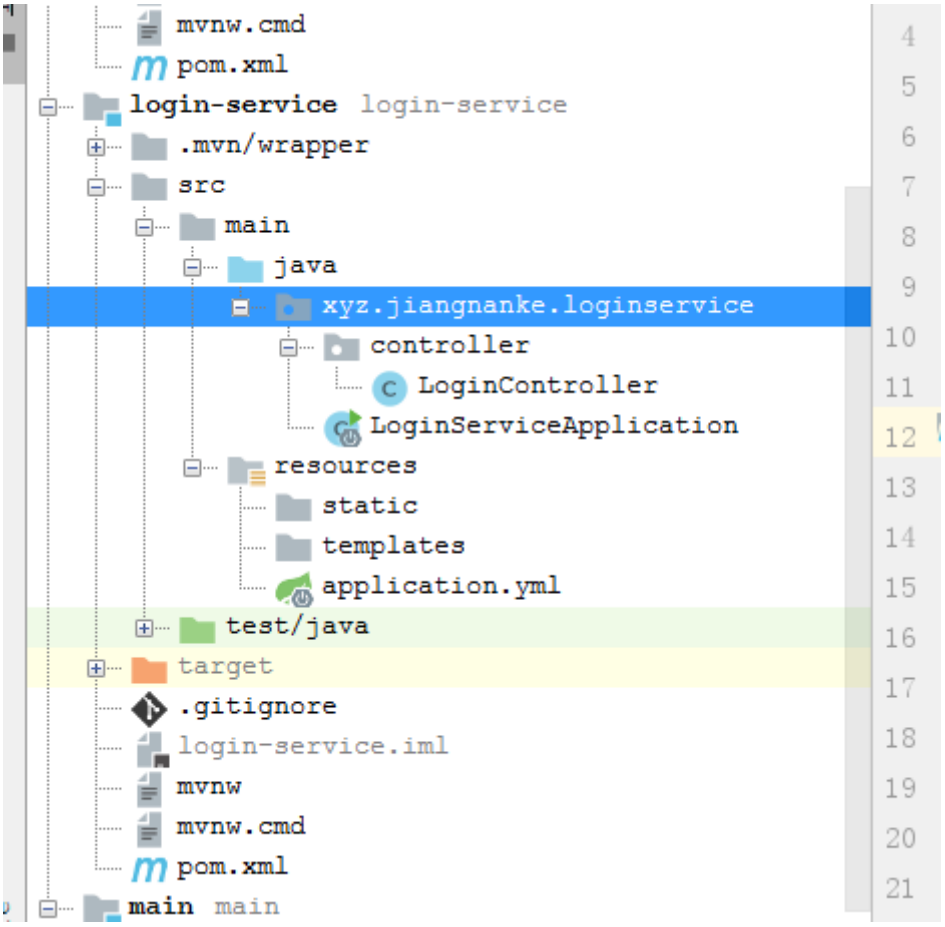
选择eureka的client依赖，当然还可以选择web依赖



一定要注意红框的路径，如果覆盖了原有的模块，那就和我一样悲剧了，暂时只能都删除了，然后重新创建



最后我的login-service的模块结构如下：



2、配置项目

关于pom.xml的配置看上面

关于**application.yml**的配置如下：

```yaml
server:
  port: 8762

spring:
  application:
  name: login-service

eureka:
  client:
   serviceUrl:
   defaultZone: http://localhost:8761/eureka/
```

3、写测试类

我这边提供了一个处理登录的路径controller

LoginController.java

```java
package xyz.jiangnanke.loginservice.controller;
```

```
2
3  import org.springframework.beans.factory.annotation.Value;
4  import org.springframework.web.bind.annotation.RequestMapping;
5  import org.springframework.web.bind.annotation.RequestParam;
6  import org.springframework.web.bind.annotation.RestController;
7
8  /**
9   * @Auther: zhengfeng
10   * @Date: 2018\12\19 0019 15:48
11   * @Description:
12   */
13  @RestController
14  public class LoginController {
15
16   @Value("${server.port}")
17   String port;
18
19   @RequestMapping("/login")
20   public String home(@RequestParam(value = "name", defaultValue = "jiangn
anke") String name) {
21   System.out.println(" this is login for name :" + name);
22   return "hi " + name + " ,login is success! the port is:" + port;
23   }
24
25  }
```

当然在Application启动类需要配置一些注解

如下：

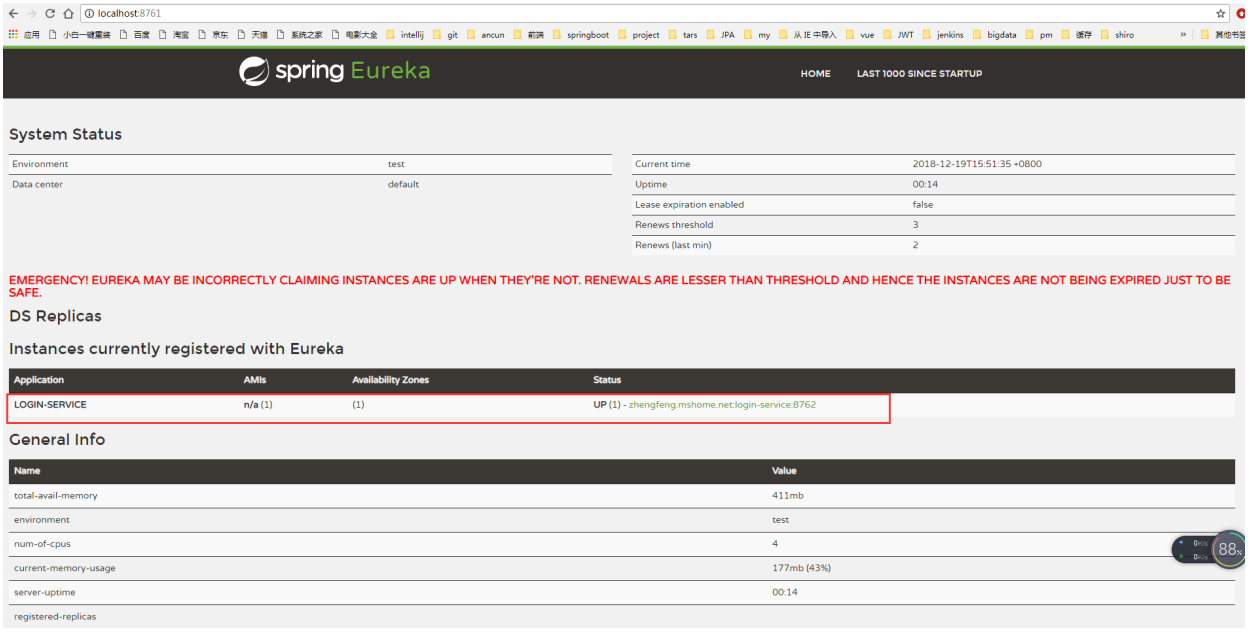
```
1  package xyz.jiangnanke.loginservice;
2
3  import org.springframework.beans.factory.annotation.Value;
4  import org.springframework.boot.SpringApplication;
5  import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
6  import org.springframework.boot.autoconfigure.SpringBootApplication;
7  import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
8  import org.springframework.context.annotation.ComponentScan;
9  import org.springframework.context.annotation.Configuration;
10  import org.springframework.web.bind.annotation.RequestMapping;
11  import org.springframework.web.bind.annotation.RequestParam;
12
13  @EnableEurekaClient
14  @SpringBootApplication
```

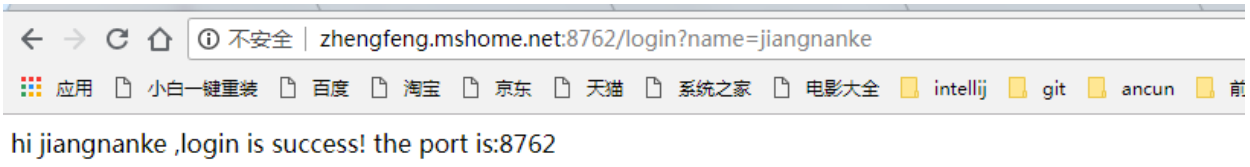
```
// @Configuration
// @ComponentScan
// @EnableAutoConfiguration
public class LoginServiceApplication {

 public static void main(String[] args) {
 SpringApplication.run(LoginServiceApplication.class, args);
 }

}

```

@Configuration， @ComponentScan， @EnableAutoConfiguration，这三个注解用SpringBootApplication来进行替代，具体三个注解是什么作用，可以去看看springboot注解

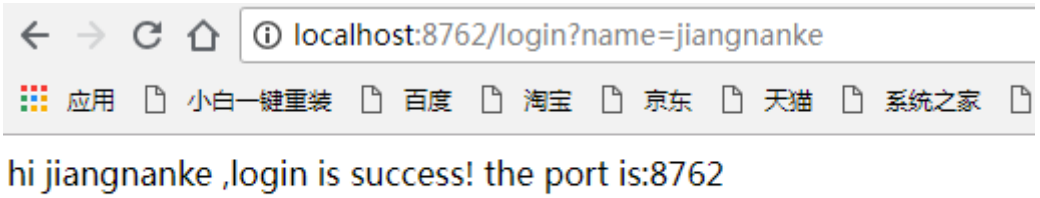
4、启动

启动后的变化以及效果，如图：





两个域名都可以



参考资料

服务的注册与发现Eureka(Finchley版本)：
https://blog.csdn.net/forezp/article/details/81040925
SpringCloud 官网： http://spring.io/projects/spring-cloud
SpringCloud 开源代码：https://github.com/spring-cloud
Spring Cloud 中文官方文档：https://springcloud.cc/
https://github.com/Netflix/eureka