## 组件

**Eureka** 注册服务器

**Ribbon** 负载均衡器，在客户端向注册器请求到服务列表后，客户端自行依据ribbon的均衡算法选择一个服务地址来调用。

## Eureka

1. 利用idea的springintailzer初始化eurekaserver项目后
2. 给启动类加上注解：

@EnableEurekaServer

1. 启动的时候报错：

org.springframework.context.ApplicationContextException: Unable to start web server; nested exception is org.springframework.boot.web.server.WebServerException: Unable to start embedded Tomcat

遇到错误不要慌，往上滑动查看错误信息，可以找到：

java.lang.ClassNotFoundException: javax.xml.bind.JAXBContext

这种问题很好解决：

在maven仓库中找到这个包的pom引入pom.xml即可。

如：

<dependency>  
 <groupId>javax.xml.bind</groupId>  
 <artifactId>jaxb-api</artifactId>  
 <version>2.3.0</version>  
</dependency>

1. 之后报这个错：

com.netflix.discovery.shared.transport.TransportException: Cannot execute request on any known server

但是没有关系，8080端口的eureka依旧可以被访问到。

看这个错误的信息就可以明白，它是在请求一个地址的连接，在加上官网的文件：

The combination of the two caches (client and server) and the heartbeats make a standalone Eureka server fairly resilient to failure, as long as there is some sort of monitor or elastic runtime (such as Cloud Foundry) keeping it alive. In standalone mode, you might prefer to switch off the client side behavior so that it does not keep trying and failing to reach its peers. The following example shows how to switch off the client-side behavior:

**application.yml (Standalone Eureka Server).**

server:

port: 8761

eureka:

instance:

hostname: localhost

client:

registerWithEureka: false

fetchRegistry: false

serviceUrl:

defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

**server会默认再去找peers的，所以会报错，因此，按照上面的配置关掉client即可。**

1. 接下来报错：

javax.xml.bind.JAXBException: Implementation of JAXB-API has not been found on module path or classpath.

。。。。。

Caused by: java.lang.ClassNotFoundException: com.sun.xml.internal.bind.v2.ContextFactory

。。。。。

所以maven找到pom并引入，重启。

1. 引入后又报错：

javax.xml.bind.JAXBException: Implementation of JAXB-API has not been found on module path or classpath.

。。。

Caused by: java.lang.ClassNotFoundException: com.sun.xml.internal.bind.v2.ContextFactory

。。。

<dependency>  
 <groupId>com.sun.xml.bind</groupId>  
 <artifactId>jaxb-impl</artifactId>  
 <version>2.3.0</version>  
</dependency>

1. 还报错：

java.lang.NoClassDefFoundError: com/sun/xml/bind/v2/model/annotation/AnnotationReader

。。。。。

Caused by: java.lang.ClassNotFoundException:

com.sun.xml.bind.v2.model.annotation.AnnotationReader

1. 这次还是报错，在jdk11中查看了下，没有这个java.xml.bind的包，jdk11中删除了这些。因此需要自己单独引入jar包。网上这篇文章提出了引入的pom的内容：

<https://stackoverflow.com/questions/52502189/java-11-package-javax-xml-bind-does-not-exist>

增加：

<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-core -->

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-core</artifactId>

<version>2.3.0</version>

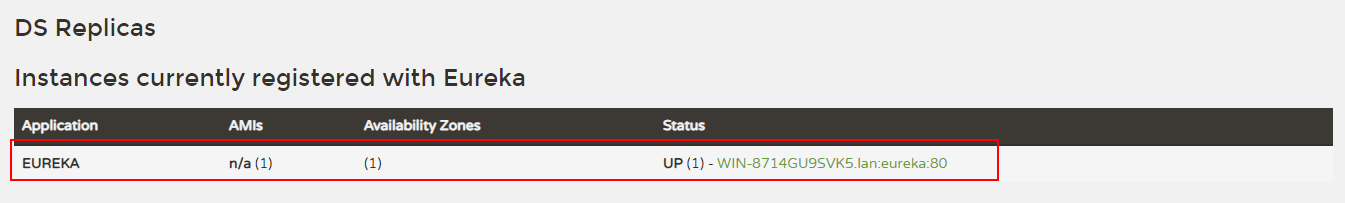
</dependency>

1. 又报错：

Caused by: java.lang.ClassNotFoundException: javax.activation.DataHandler

引入：

1. 不报错了。
2. 访问服务器主页，这时候启动会报一会错，原因是server还没有启动好client就开始去注册了，没关系，这个client其实是定时请求服务的，因此过一会等server请求好了就没错了，client就成功注册进去了。查看注册了的服务：

因为我们用的上面的spring文档说的方式，作用就是关闭client注册，所以什么都没有。

那么，我们换一种方式，让他注册到自己身上（相比于上面就只是把client属性下的两个属性注释掉即可，这样一来client依然会去找服务地址去注册自己，service-url就是指client要去寻找的服务地址）：

**server**:  
 **port**: 80  
  
**eureka**:  
 **instance**:  
 **hostname**: localhost  
 **client**:  
*# register-with-eureka: false  
# fetch-registry: false* **service-url**:  
 **defaultZone**: http://${**eureka.instance.hostname**}:${**server.port**}/eureka/

**spring**:  
 **application**:  
 **name**: eureka

