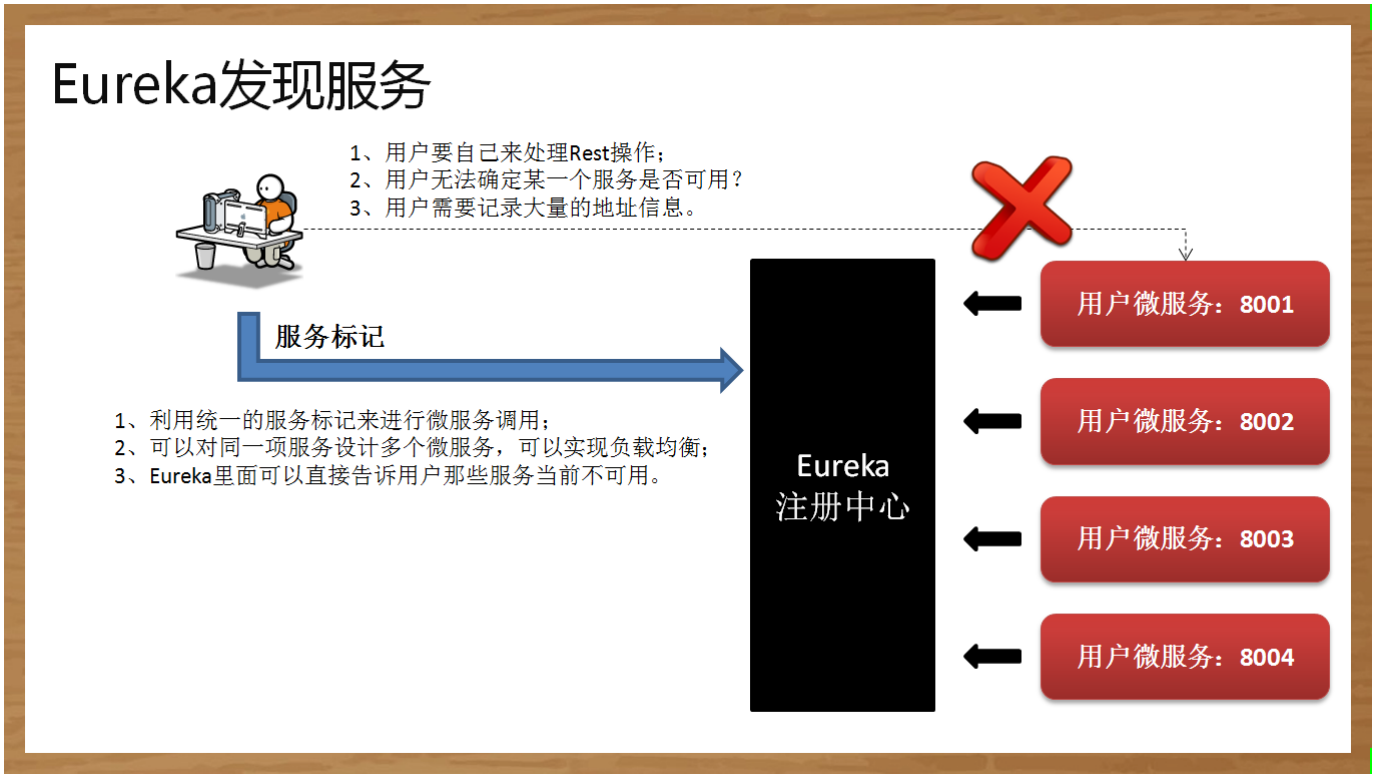
1、课程名称：Eureka 服务发现框架

2、具体内容

对于服务发现框架可以简单的理解为服务的注册以及使用操作步骤，例如：在最初学习过的 ZooKeeper 组件，这个组件里面已经明确的描述了一个服务的注册以及发现操作流程，在整个 Rest 架构里面，会存在有大量的微服务的信息。

在 SpringCloud 之中使用了大量的 Netflix 的开源项目，而其中 Eureka 就属于 Netflix 提供的发现服务组件，所有的微服务在使用之中全部向 Eureka 之中进行注册，而后客户端直接利用 Eureka 进行服务信息的获得。



Eureka 的主要作用实际上和 ZooKeeper 是非常类似的，但是在 SpringCloud 虽然支持有 ZooKeeper，不过从官方的宣传角度来说并不支持这样处理，推荐使用 Eureka，因为速度更快，同时该服务组件是以程序的形式出现的，也就是说你只需要编写一个程序的项目类，而后就可以启动 Eureka 注册服务了。

2.1、定义 Eureka 服务端

1、 为了方便进行统一的微服务的管理，建议创建一个新的项目：microcloud-eureka-7001；

2、 【microcloud-eureka-7001】的 pom.xml 配置文件，追加相应的依赖支持库

```
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-eureka-server</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
        </dependency>
```

3、 【microcloud-eureka-7001】现在修改 application.yml 配置文件，在这个配置文件里面主要进行 eureka 服务的定义。

```
server:
  port: 7001
eureka:
```

```
  instance: # eureak实例定义
    hostname: eureka-7001.com # 定义 Eureka 实例所在的主机名称
```

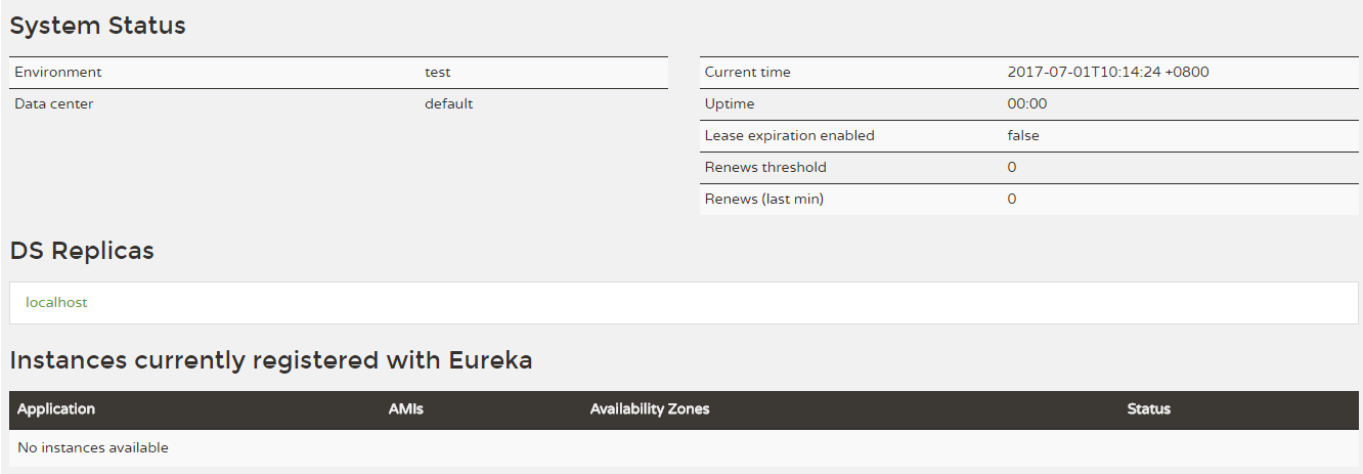
4、 【microcloud-eureka-7001】修改 hosts 配置文件，追加 eureka 的映射地址。

```
127.0.0.1   eureka-7001.com
```

5、 【microcloud-eureka-7001】修改 Eureka 程序启动类，追加有 Eureka 服务声明：

```
package cn.mldn.microcloud;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
@SpringBootApplication
@EnableEurekaServer
public class Eureka_7001_StartSpringCloudApplication {
    public static void main(String[] args) {
        SpringApplication.run(Eureka_7001_StartSpringCloudApplication.class, args);
    }
}
```

6、 运行程序后通过浏览器执行路径：http://eureka-7001.com:7001/



2.2、向 Eureka 中注册部门微服务

现在 Eureka 已经可以正常启用了，那么随后就需要在项目之中将所有的微服务信息注册到 Eureka 服务之中，那么这样就可以被客户端执行并且调用了。

1、 【microcloud-provider-dept-8001】修改 pom.xml 配置文件，追加有 eureka 的相关依赖支持包；

```
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-eureka</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
```

2、 【microcloud-provider-dept-8001】修改 application.yml 配置文件，在这个配置文件之中主要是定义要进行注册的 Eureka 服务的地址，而这个地址就是 Eureka 的客户端配置。

```
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://eureka-7001.com:7001/eureka
```

3、 【microcloud-provider-dept-8001】修改项目的运行主类，在这个主类上追加有 Eureka 客户端的启用注解：

```
package cn.mldn.microcloud;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
@SpringBootApplication
@EnableEurekaClient
public class Dept_8001_StartSpringCloudApplication {
    public static void main(String[] args) {
        SpringApplication.run(Dept_8001_StartSpringCloudApplication.class, args);
    }
}
```

此时由于存在有“@EnableEurekaClient”注解信息，所以当服务启动之后该服务会自动注册到Eureka服务器之中；

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
UNKNOWN	n/a (1)	(1)	UP (1) - mldn-PC:8001

4、 【microcloud-provider-dept-8001】修改application.yml配置文件，为此微服务设置一个名字（这个名字将作为日后负载均衡）

```
spring:
  application:
    name: microcloud-provider-dept
```

Application	AMIs	Availability Zones	Status
MICROCLOUD-PROVIDER-DEPT	n/a (1)	(1)	UP (1) - mldn-PC:microcloud-provider-dept:8001

2.3、Eureka 服务信息

现在虽然成功的实现了微服务的 Eureka 注册，但是所表现出来的微服务的信息并不完整，因为给定的地址信息是你的主机名称，而我们现在是一个自定义的路径地址。

Instances currently registered with Eureka

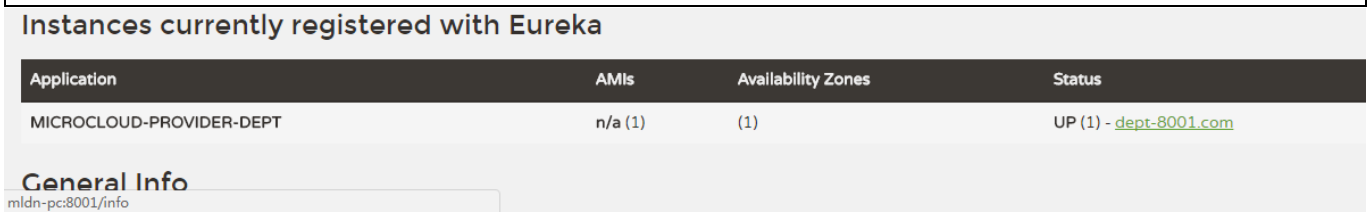
Application	AMIs	Availability Zones	Status
MICROCLOUD-PROVIDER-DEPT	n/a (1)	(1)	UP (1) - mldn-PC:microcloud-provider-dept:8001

mldn-pc:8001/info

1、 【microcloud-provider-dept-8001】修改 application.yml 配置文件，追加主机名称的显示：

```
eureka:
```

```yaml
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://eureka-7001.com:7001/eureka
  instance:
    instance-id: dept-8001.com
```



2、 【microcloud-provider-dept-8001】在服务信息查看的时候应该以 IP 地址作为连接项。

```yaml
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://eureka-7001.com:7001/eureka
  instance:
    instance-id: dept-8001.com  # 在信息列表时显示主机名称
    prefer-ip-address: true     # 访问的路径变为 IP 地址
```

3、 【microcloud-provider-dept-8001】如果现在要想查看所有的微服务详细信息，则需要修改 pom.xml 文件，追加监控配置：

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
```

4、 【microcloud】修改 pom.xml 文件，追加一个信息匹配的插件：

```xml
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <filtering>true</filtering>
            </resource>
        </resources>
```

```xml
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <configuration>
                    <delimiters>
                        <delimiter>$</delimiter>
                    </delimiters>
                </configuration>
```

```
            </plugin>
```

新的插件配置完成之后需要进行项目的更新处理。

5、 【microcloud-provider-dept-8001】修改 application.yml 配置文件，追加 info 的相关信息：

```
info:
  app.name: mldn-microcloud
  company.name: www.mldn.cn
  build.artifactId: $project.artifactId$
  build.version: $project.verson$
```

由于项目之中微服务可能会有成白上千个，所以微服务的信息你一定要认真填写，否则你的项目维护会非常痛苦。

2.4、Eureka 发现管理

在实际的项目运行过程之中需要通过 Eureka 作为所有微服务的监控处理程序，但是对于监控程序那么就必然要面临以下问题：

- 新服务追加的时候应该立刻可以进行注册；
- 当某一个服务下线之后应该可以进行清理；

1、 【microcloud-eureka-7001】设置服务的清理间隔，修改 application.yml 配置文件

```
server:
  port: 7001
spring:
  application:
    name: microcloud-eureka-7001
eureka:
  server:
    eviction-interval-timer-in-ms: 1000 # 设置清理的间隔时间，而后这个时间使用的是毫秒单位（默认是60秒）
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://eureka-7001.com:7001/eureka
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7001.com # 定义 Eureka 实例所在的主机名称
```

一旦配置了清理的间隔为 1 秒的时间，则会在每秒的时候进行一次服务的清理过程，会出现如下错误提示信息：

```
2017-07-01 10:58:44.894  INFO 6628 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry  :
Running the evict task with compensationTime 0ms
```

一般情况下，该配置不建议进行修改，默认就是 60 秒，也就是说你的微服务如果 60 秒没有心跳了，那么就认为可以清理掉。

2、 【microcloud-eureka-7001】在 Eureka 里面有一个问题，这个问题就是它默认支持有保护模式的概念，所谓的保护模式指的是即便现在某一个微服务不可用了，eureka 不会清理，依然会进行该微服务信息的保存。

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

如果现在要想去改变这种保护模式的启用，则可以修改 application.yml 配置文件：

```
eureka:
  server:
```

```
    enable-self-preservation: false # 设置为false表示关闭保护模式
    eviction-interval-timer-in-ms: 1000 # 设置清理的间隔时间，而后这个时间使用的是毫秒单位（默认是 60 秒）
```

理论上只有关闭了保护模式之后才可以进行无效微服务的清理操作，但是很多时候 Eureka 里面也会自带有清除过程。

THE SELF PRESERVATION MODE IS TURNED OFF.THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.

3、 【microcloud-provider-dept-8001】微服务客户端之所以可以与 Eureka 之间保持联系，依靠的是心跳机制，也就是说你客户端可以自己来进行心跳的配置处理，修改 application.yml 配置文件：

```
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://eureka-7001.com:7001/eureka
  instance:
    lease-renewal-interval-in-seconds: 2 # 设置心跳的时间间隔（默认是30秒）
    lease-expiration-duration-in-seconds: 5 # 如果现在超过了5秒的间隔（默认是90秒）
    instance-id: dept-8001.com  # 在信息列表时显示主机名称
    prefer-ip-address: true     # 访问的路径变为 IP 地址
```

由于所有的服务都注册到了 Eureka 之中，这样如果配置了“lease-expiration-duration-in-seconds”此选项，表示距离上一次发送心跳之后等待下一次发送心跳的间隔时间，如果超过了此间隔时间，则认为该微服务已经宕机了。

4、 【microcloud-provider-dept-8001】现在对于注册到 Eureka 上的微服务端也可以通过发现服务来进行一些服务信息的获取，修改 DeptRest 程序类，追加一个控制调用方法： 提供服务发现功能

```
    @Resource
    private DiscoveryClient client ;  // 进行Eureka的发现服务
    @RequestMapping("/dept/discover")
    public Object discover() {    // 直接返回发现服务信息
        return this.client ;
    }
```

5、 【microcloud-provider-dept-8001】在主程序之中启用 Eureka 发现服务项：

```
package cn.mldn.microcloud;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
@SpringBootApplication
@EnableEurekaClient
@EnableDiscoveryClient
public class Dept_8001_StartSpringCloudApplication {
    public static void main(String[] args) {
        SpringApplication.run(Dept_8001_StartSpringCloudApplication.class, args);
    }
}
```

6、 输入访问地址：http://dept-8001.com:8001/dept/discover

```
{"services":[],"localServiceInstance":{"host":"192.168.31.247",
"port":8001,"secure":false,"serviceId":"microcloud-provider-dept","metadata":{},"uri":"http://192.168.31.247:8001"}}
```

Eureka 里面就是根据这些信息来进行应用列表显示的。

2.5、Eureka 安全配置

现在已经成功的实现了一个 Eureka 的服务启动以及微服务的注册配置操作，但是现在的程序有一个问题，你自己公司的 Eureka 服务应该可以注册的服务只能够是满足于认证要求的微服务，所以这样一来在之前所进行的 Eureka 里面配置缺少关键性的一步：安全认证，所以应该为 Eureka 配置上安全认证处理。

1、 【microcloud-eureka-7001】修改 pom.xml 配置文件，引入 SpringSecurity 的依赖包：

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

2、 【microcloud-eureka-7001】一旦我们的项目之中导入了 Security 开发包，则每一次启动微服务的时候都会自动生成一个密码，而这个密码由于会改变，所以一般都不使用，所以要修改 application.yml 配置文件，追加密码的配置项：

```
security:                                          安全认证
  basic:
    enabled: true   # 启用安全认证处理
  user:
    name: edmin     # 用户名
    password: mldnjava  # 密码
```

```
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://edmin:mldnjava@eureka-7001.com:7001/eureka      所有注eureka的服务都需要被认证
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7001.com # 定义Eureka实例所在的主机名称
```

此时访问 eureka 的服务地址为：http://edmin:mldnjava@eureka-7001.com:7001/

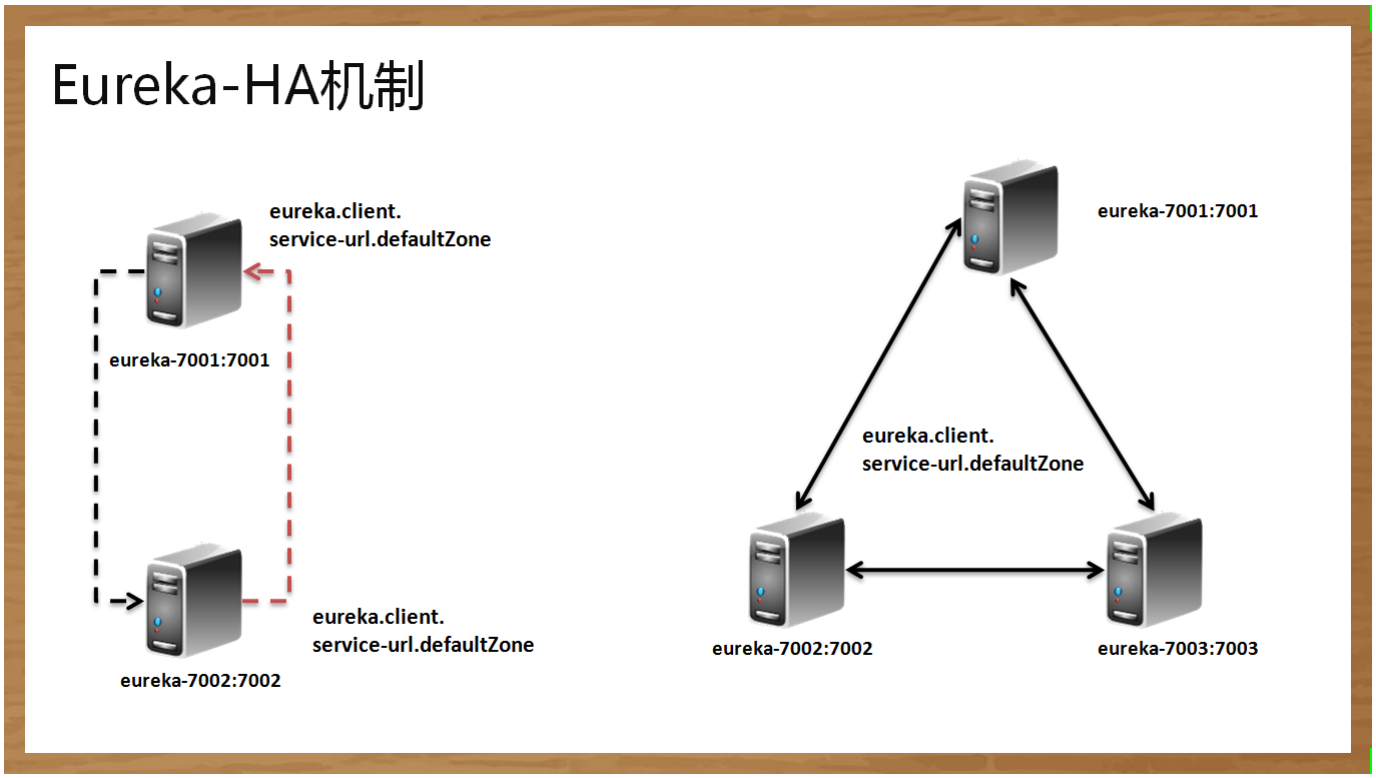
3、 【microcloud-provider-dept-8001】修改 application.yml 配置文件，进行授权的注册连接：

```
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://edmin:mldnjava@eureka-7001.com:7001/eureka
  instance:
    lease-renewal-interval-in-seconds: 2 # 设置心跳的时间间隔（默认是30秒）
    lease-expiration-duration-in-seconds: 5 # 如果现在超过了5秒的间隔（默认是90秒）
    instance-id: dept-8001.com  # 在信息列表时显示主机名称
    prefer-ip-address: true     # 访问的路径变为 IP 地址
```

Application	AMIs	Availability Zones	Status
MICROCLOUD-PROVIDER-DEPT	n/a (1)	(1)	UP (1) - dept-8001.com

2.6、Eureka-HA 机制

现在已经成功的实现了一个 Eureka 服务器，但是现在属于单节点的服务运行过程，如果说现在单节点的 Eureka 出现了错误，导致无法使用，那么对于所有的微服务的架构就将出现整体的瘫痪，就需要进行 Eureka 集群搭建，同时利用集群可以有效的实现 HA 的处理机制，如果要进行集群的搭建一定要选择两台或以上的电脑完成，而基本的流程如下。



1、 修改 hosts 配置文件进行多个主机名称的定义：

```
127.0.0.1  eureka-7001.com
127.0.0.1  eureka-7002.com
127.0.0.1  eureka-7003.com
```

2、 【microcloud-eureka-7001】为了方便进行 Eureka 操作，建议将“microcloud-eureka-7001”的进行复制，复制为“microcloud-eureka-7002”、“ microcloud-eureka-7003”。

3、 【microcloud-eureka-7001】修改 application.yml 配置文件，这个配置文件主要注意端口号以及 Eureka 服务注册位置；

```
server:
  port: 7001
security:
  basic:
    enabled: true   # 启用安全认证处理
  user:
    name: edmin     # 用户名
    password: mldnjava  # 密码
spring:
  application:
    name: microcloud-eureka-7001
eureka:
```

```
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone:
http://edmin:mldnjava@eureka-7002.com:7002/eureka,http://edmin:mldnjava@eureka-7003.com:7003/eureka
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7001.com # 定义 Eureka 实例所在的主机名称
```

4、 【microcloud-eureka-7002】修改 application.yml 配置文件：

```
server:
  port: 7002
security:
  basic:
    enabled: true   # 启用安全认证处理
  user:
    name: edmin     # 用户名
    password: mldnjava  # 密码
spring:
  application:
    name: microcloud-eureka-7002
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone:
http://edmin:mldnjava@eureka-7001.com:7001/eureka,http://edmin:mldnjava@eureka-7003.com:7003/eureka
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7002.com # 定义 Eureka 实例所在的主机名称
```

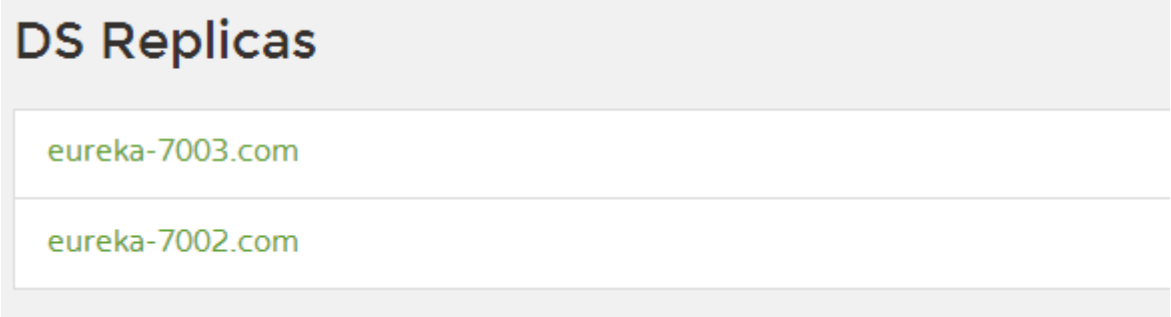
5、 【microcloud-eureka-7003】修改 application.yml 配置文件：

```
server:
  port: 7003
security:
  basic:
    enabled: true   # 启用安全认证处理
  user:
    name: edmin     # 用户名
    password: mldnjava  # 密码
spring:
  application:
    name: microcloud-eureka-7003
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
```

```
      defaultZone:
http://edmin:mldnjava@eureka-7001.com:7001/eureka,http://edmin:mldnjava@eureka-7002.com:7002/eureka
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7003.com # 定义 Eureka 实例所在的主机名称
```

6、 启动所有的 eureka 服务，而后进入到每一个服务的后台去观察运行的副本效果：

- 登录 7001 控制台：http://edmin:mldnjava@eureka-7001.com:7001/；
- 登录 7002 控制台：http://edmin:mldnjava@eureka-7002.com:7002/；
- 登录 7003 控制台：http://edmin:mldnjava@eureka-7003.com:7003/。



7、 【microcloud-provider-dept-8001】修改 application.yml 配置文件，进行多台主机注册：

```
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone: http://edmin:mldnjava@eureka-7001.com:7001/eureka,
http://edmin:mldnjava@eureka-7002.com:7002/eureka,
http://edmin:mldnjava@eureka-7003.com:7003/eureka
```

运行之后该微服务会出现在所有的 Eureka 主机之中。这样即使某一台 Eureka 出现了问题，那么依然可以保证服务的可用。

2.7、Eureka 服务发布

现在已经成功的实现了 Eureka 编写，但是在实际的运行之中，需要将 Eureka 发布到具体的服务器上进行执行，而这就需要对项目进行打包处理，同样在进行打包处理的时候也必须考虑到项目的各种环境：开发（dev）、测试（beta）、生产（product），那么下面也将基于这样的方式进行 eureka 项目打包操作。

本次的打包处理将直接基于 yml 配置文件完成，对于 properties 配置与 SpringBoot 讲解微服务发布的处理过程一样。

1、 【microcloud-eureka-server】修改 application.yml 配置文件

```
spring:
  profiles:
    active:
    - dev-7001
---
spring:
  profiles: dev-7001
  application:
    name: microcloud-eureka-7001
```

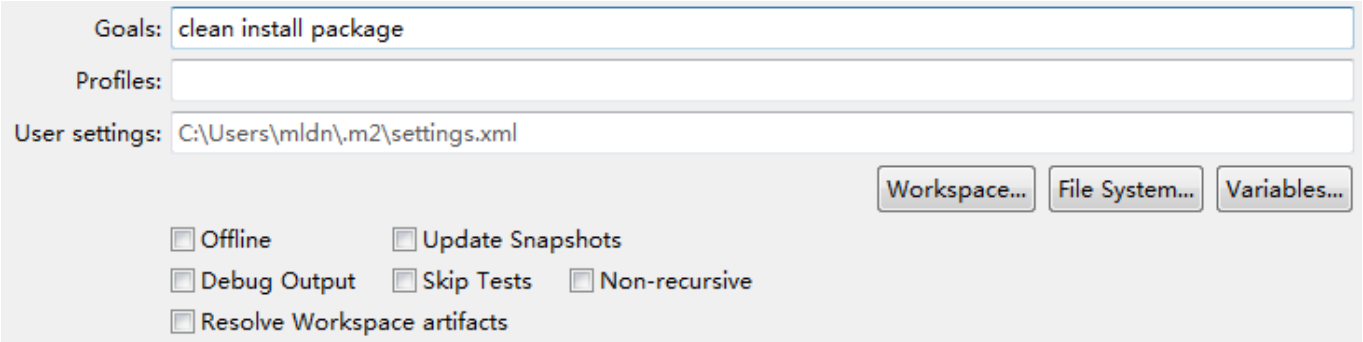
```
server:
  port: 7001
security:
  basic:
    enabled: true   # 启用安全认证处理
  user:
    name: edmin     # 用户名
    password: mldnjava  # 密码
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone:
http://edmin:mldnjava@eureka-7002.com:7002/eureka,http://edmin:mldnjava@eureka-7003.com:7003/eureka
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7001.com # 定义Eureka实例所在的主机名称
---
spring:
  profiles: dev-7002
  application:
    name: microcloud-eureka-7002
server:
  port: 7002
security:
  basic:
    enabled: true   # 启用安全认证处理
  user:
    name: edmin     # 用户名
    password: mldnjava  # 密码
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone:
http://edmin:mldnjava@eureka-7001.com:7001/eureka,http://edmin:mldnjava@eureka-7003.com:7003/eureka
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7002.com # 定义Eureka实例所在的主机名称
---
spring:
  profiles: dev-7003
  application:
    name: microcloud-eureka-7003
```

```
server:
  port: 7003
security:
  basic:
    enabled: true   # 启用安全认证处理
  user:
    name: edmin     # 用户名
    password: mldnjava  # 密码
eureka:
  client: # 客户端进行Eureka注册的配置
    service-url:
      defaultZone:
http://edmin:mldnjava@eureka-7001.com:7001/eureka,http://edmin:mldnjava@eureka-7002.com:7002/eureka
    register-with-eureka: false     # 当前的微服务不注册到eureka之中
    fetch-registry: false      # 不通过eureka获取注册信息
  instance: # eureak实例定义
    hostname: eureka-7003.com # 定义Eureka实例所在的主机名称
```

2、 【microcloud-eureka-server】添加一个打包的处理插件，修改 pom.xml 配置文件：

```
    <build>
        <finalName>eureka-server</finalName>
        <plugins>
            <plugin> <!-- 该插件的主要功能是进行项目的打包发布处理 -->
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>  <!-- 设置程序执行的主类 -->
                    <mainClass>cn.mldn.microcloud.Eureka_StartSpringCloudApplication</mainClass>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

3、 运行 maven：clean install package；



随后就可以在项目的目录之中发现生成的“eureka-server.jar”文件。

4、 采用默认的方式执行 eureka-server.jar，那么此时将运行在 7001 端口上：java -jar eureka-server.jar

5、 运行其它的两个 profile 配置：

- 运行“dev-7002”profile：java -jar eureka-server.jar --spring.profiles.active=dev-7002；
- 运行“dev-7003”profile：java -jar eureka-server.jar --spring.profiles.active=dev-7003。

3、总结

所有微服务的打包都是基于 Maven 处理的，所以整体的部署都很容易。