# 1、配置maven

注意maven版本最idea版本需对应，否则会下载失败

# 2、创建父子聚合项目

创建父子maven项目：<https://how2j.cn/k/idea/idea-parent-child/2051.html>

# 3、创建注册中心Eureka

## 3.1、引入依赖

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

## 3.2、创建主启动类🡪@EnableEurekaServer

@SpringBootApplication  
@EnableEurekaServer  
public class EurekaServerApplication {  
 public static void main(String[] args) {  
 //8761 这个端口是默认的，就不要修改了，后面的子项目，都会访问这个端口。  
 int port = 8761;  
 if(!NetUtil.*isUsableLocalPort*(port)) {  
 System.*err*.printf("端口%d被占用了，无法启动%n", port );  
 System.*exit*(1);  
 }  
 new SpringApplicationBuilder(EurekaServerApplication.class).properties("server.port=" + port).run(args);  
 }  
}

## 3.3、添加服务配置

**eureka**:  
 **instance**:  
 **hostname**: localhost *#localhost 表示主机名称* **client**:  
 **registerWithEureka**: false *#false. 表示是否注册到服务器。 因为它本身就是服务器，所以就无需把自己注册到服务器了* **fetchRegistry**: false *#false. 表示是否获取服务器的注册信息，和上面同理，这里也设置为 false* **serviceUrl**:  
 **defaultZone**: http://${eureka.instance.hostname}:${server.port}/eureka/ *# http://localhost:8761/eureka/***spring**:  
 **application**:  
 **name**: eureka-server

## 3.4、ip:port访问测试

http://127.0.0.1:8761/

# 4、注册数据微服务

## 4.1、引入依赖

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

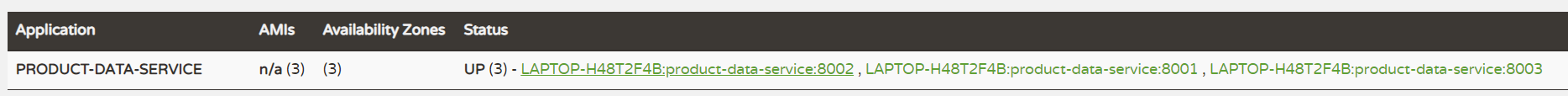
## 4.2、创建主启动类🡪@EnableEurekaClient

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

## 4.3、添加服务配置

*# server:  
# port: 因为会启动多个 product-data-service, 所以端口号由用户自动设置，推荐 8001,8002,8003***spring**:  
 **application**:  
 **name**: product-data-service  
**eureka**:  
 **client**:  
 **serviceUrl**:  
 **defaultZone**: http://localhost:8761/eureka/ *#设置注册中心的地址*

## 4.4、启动三个数据服务后注册到Eureka的视图如下：



# 5、Ribbon客户端（了解）

## 5.1、什么是Ribbon

SpringCloud提供的实现负载均衡的一种方式

## 5.2、引入依赖

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
</dependency>

## 5.3、如何使用

请求方向：controller-->service-->ribbon客户端(借助RestTemplate)

响应方向：ribbon客户端(借助RestTemplate)-->service-->controller-->html渲染

**其中Ribbon客户端源码如下：**

@Component  
public class ProductClientRibbon {  
 //Ribbon 是使用 restTemplate 进行调用，并进行客户端负载均衡  
 @Autowired  
 RestTemplate restTemplate;  
 public List<Product> listProdcuts() {  
 //product-data-service 是数据服务在 eureka 注册中心的名称  
 return restTemplate.getForObject("http://PRODUCT-DATA-SERVICE/products", List.class);  
 }  
}

注：Ribbon客户端使用的RestTemplate是在主启动类中纳入容器的，如下：

@Bean  
@LoadBalanced  
RestTemplate restTemplate() {//用 restTemplate 这个工具来做负载均衡  
 return new RestTemplate();  
}

## 5.4、配置文件

**eureka**:  
 **client**:  
 **serviceUrl**:  
 **defaultZone**: http://localhost:8761/eureka/  
**spring**:  
 **application**:  
 **name**: product-view-service-ribbon  
 **thymeleaf**:  
 **cache**: **false  
 prefix**: classpath:/templates/  
 **suffix**: .html  
 **encoding**: UTF-8  
 **mode**: HTML5  
 **servlet**:  
 **content-type**: text/html  
**server**:  
 **port**: 8010

## 5.5、数据流转说明

启动项目后访问<http://127.0.0.1:8010/products>，该服务会通过Ribbon(借助RestTemplate)访问Eureka：[http://PRODUCT-DATA-SERVICE/products](http://product-data-service/products)，再负载均衡到一个数据微服务上获取数据

# 6、Feign客户端（重点）

## 6.1、什么是Feign

Feign是对Ribbon的封装，使用注解的方式，调用起来更简单

## 6.2、引入依赖

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
</dependency>

## 6.4、如何使用

请求方向：controller-->service-->Feign客户端(注解@FeignClient)

响应方向：Feign客户端(注解@FeignClient)-->service-->controller-->html渲染

**其中Feign客户端源码如下：**

@FeignClient(value = "PRODUCT-DATA-SERVICE")  
public interface ProductClientFeign {  
 @GetMapping("/products")  
 public List<Product> listProdcuts();  
}

## 6.5、配置文件

**eureka**:  
 **client**:  
 **serviceUrl**:  
 **defaultZone**: http://localhost:8761/eureka/  
**spring**:  
 **application**:  
 **name**: product-view-service-feign  
 **thymeleaf**:  
 **cache**: **false  
 prefix**: classpath:/templates/  
 **suffix**: .html  
 **encoding**: UTF-8  
 **mode**: HTML5  
 **servlet**:  
 **content-type**: text/html  
**server**:  
 **port**: 8012

## 6.6、数据流转说明

启动项目后访问[http://127.0.0.1:8012/products](http://127.0.0.1:8010/products)，该服务会通过Feign访问Eureka：[http://PRODUCT-DATA-SERVICE/products](http://product-data-service/products)，再负载均衡到一个数据微服务上获取数据

# 7、Zipkin服务链路追踪

## 7.1、什么是服务链路

当微服务较多时，他们之间的调用关系往往更加复杂，而仅仅通过观察代码，会越来越难以识别调用关系，所以就需要通过zipkin服务链路追踪服务器来进行链路查看

## 7.2、准备工作

视图微服务与数据微服务需进行简单改造，才可以被追踪到，步骤如下：

### 7.2.1、添加依赖

数据微服务与视图微服务均新增依赖

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-zipkin</artifactId>  
</dependency>

### 7.2.2、配置zipkin

数据微服务与视图微服务均新增zipkin的配置项

**spring**:  
 **zipkin**:  
 **base-url**: http://localhost:9411

### 7.2.3、配置Sampler抽样策略

在数据微服务与视图微服务的主启动类里配置Sampler抽样策略：ALWAYS\_SAMPLE表示持续抽样

@Bean  
public Sampler defaultSampler() {  
 return Sampler.*ALWAYS\_SAMPLE*;  
}

## 7.3、查看链路

下载zipkin-server.jar并启动，访问路径： <http://localhost:9411/zipkin/dependency/>

# 8、配置服务端

## 8.1、为什么需要配置服务

当微服务较多时，若修改某些公共配置则会挨个修改，使用配置服务可直接从git下拉公共配置即可

## 8.2、创建配置服务

### 8.2.1、引入依赖

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-config-server</artifactId>  
</dependency>

### 8.2.2、主启动类

主启动类上新增@EnableConfigServer注解

@EnableConfigServer //此注解表示本springboot 是个配置服务器

### 8.2.3、配置文件

**spring**:  
 **application**:  
 **name**: config-server  
 **cloud**:  
 **config**:  
 **label**: master *# label表示分支* **server**:  
 **git**:  
 **uri**: https://github.com/how2j/springcloudConfig/ *# uri为git地址* **searchPaths**: respo *# searchPaths表示目录：***eureka**:  
 **client**:  
 **serviceUrl**:  
 **defaultZone**: http://localhost:8761/eureka/  
**server**:  
 **port**: 8030

# 9、配置客户端

## 9.1、更改Feign客户端

### 9.1.1、增加依赖

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

### 9.1.2、配置bootstrap.yml

bootstrap.yml和application.yml的区别，理解为前者先启动，故一些系统配置需要在 bootstrap.yml 里进行配置（此时applicantion.yml不必再配置eureka）

**spring**:  
 **cloud**:  
 **config**:  
 **label**: master  
 **profile**: dev  
 **discovery**:  
 **enabled**: **true  
 serviceId**: config-server *# 配置服务器在 eureka server 里的服务名称* **client**:  
 **serviceUrl**:  
 **defaultZone**: http://localhost:8761/eureka/

### 9.1.3、读取配置信息

@Value("${version}")  
String version;

## 9.2、相关问题

若此时修改git上的配置信息，需重启配置服务与Feign客户端才会生效，后续可使用rabbitMQ实现动态刷新

# 10、消息总线Bus