SpringBoot学习总结

------Delta 2019.01.23

# 一：Spring Boot 之 SpringCloud 分布式

建立一个空项目，空项目中分别建立三个springboot模块（moduel）,分别是food、customer、server其中server引入：

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

表示他是一个服务端（类似于zookeeper的作用）

另外两个引入

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

表示他是客户端。

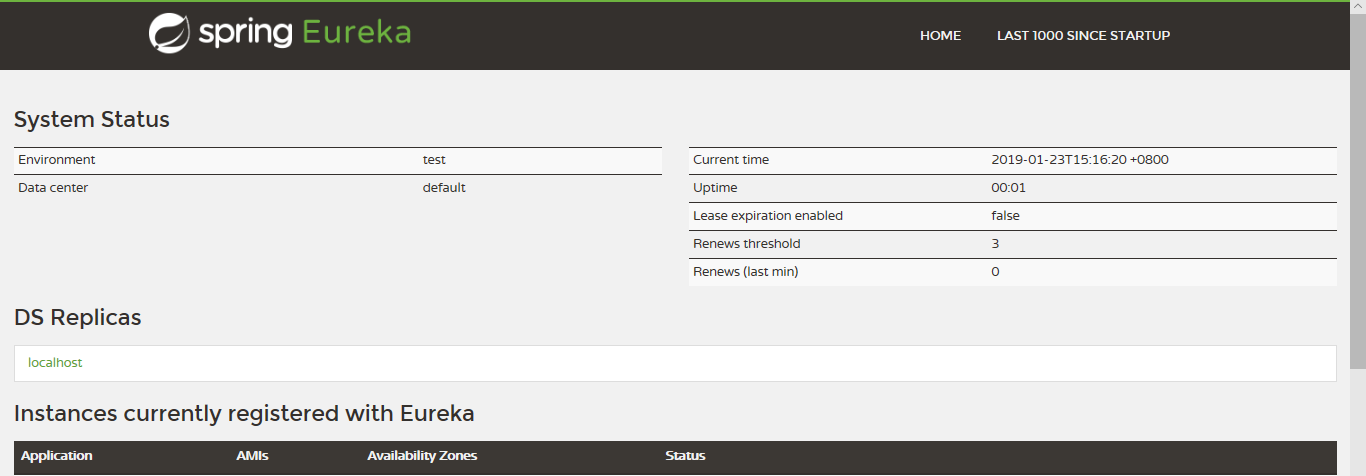
server的properties中作如下配置：

server.port=8761 //eureka的端口  
eureka.instance.hostname= eureka  
*eureka.client.fetch-registry=false  
eureka.client.register-with-eureka=false //不将本模块注册*  
eureka.client.service-

url.defaultZone=http://localhost:8761/eureka/

最后在Application类中加上@EnableEurekaServer

配置好后浏览器输入：<http://localhost:8761>将出现如下画面=>



另外编写功能被调用者，这里形象为food :

Properties中做如下配置：

server.port=8001//应用端口号  
spring.application.name=food //应用名称  
#注册时候 使用IP地址注册  
eureka.instance.prefer-ip-address=true  
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

编写food的service层和controller层，这里给出controller层：

@RestController  
public class FoodController {  
 @Autowired  
 FoodService foodService;  
   
 @GetMapping("/getfood") *//请注意这里的url地址！（\*）*  
 public String foodGet(){  
 return foodService.foodName();  
 }  
}

编写好后运行food浏览器输入localhost:8001/getfood 如果能够输出正常的结果页面出来表示该food已经配置完成。

编写customer模块:

Properties 引入

spring.application.name=customer  
server.port=8200 //服务运行端口号  
#注册时候 使用IP地址注册  
eureka.instance.prefer-ip-address=true  
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

控制层代码如下：

public class CustomerController {  
  
 @Autowired  
 RestTemplate restTemplate;  
  
 @GetMapping("/getfood")  
 public String hello\_test(){  
 return "点一份："+ restTemplate.getForObject("http://FOOD/getfood",String.class);  
 }  
  
}

其中*FOOD是我们运行food后在*[*http://localhost:876*](http://localhost:876)*出现的服务名，getfood是food的控制层url，即（\*）。String.class代表返回类型*。

[customer的Application类中加上@EnableDiscouveryClient](mailto:5.customer的Application类中加上@EnableDiscouveryClient)注解

并且添加方法：

public RestTemplate restTemplate(){  
 return new RestTemplate();  
}

在这个方法的上面加上注解：@Bean注入容器中，@LoadBalanced 负载均衡

6.运行customer 浏览器输入：localhost:8200/getfood 得到页面结果表示成功

另注：在IDEA中将food package 修改端口号再一次package得到的两个jar包

在cmd中分别执行java -jar \*\*\*\*\*.jar 再次请求可以看到两个jar执行的相应顺序

这是负载均衡的效果

# 二：SpringBoot 之Dubbo

创建一个空的工程 并分别在其中创建两个模块，暂且叫做food 和 customer

在food的pom文件中引入以下两个依赖：

<dependency>  
 <groupId>com.github.sgroschupf</groupId>  
 <artifactId>zkclient</artifactId>  
 <version>0.1</version>  
</dependency>  
  
<dependency>  
 <groupId>com.alibaba.boot</groupId>  
 <artifactId>dubbo-spring-boot-starter</artifactId>  
 <version>0.1.0</version>  
</dependency>

一个是dubbo的另一个是zookeeper的。（详见spring官网）

food的模块中编写service层，其中实现类如下：

@Component   
@Service//注意：这个Service注解是：

//Alibaba.dubbo底下的  
public class FoodServiceImp implements FoodService {  
 @Override  
 public String getFood() {  
 return "扬州炒饭...";  
 }  
}

food的配置文件如下：

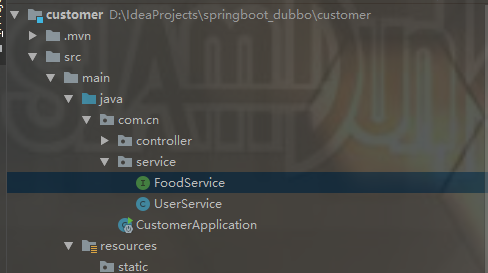
#扫描哪些包 由customer来调用  
dubbo.scan.base-packages=com.cn.service  
#服务名称  
dubbo.application.name=food  
#服务地址  
dubbo.registry.address=zookeeper://192.168.1.103:2181

编写customer模块

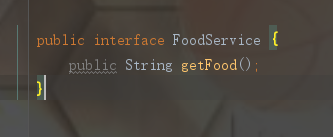
Pom文件中引入和food一样的依赖

配置文件内容如上，并且不加包扫描那段代码

将扫描的包复制到相同路径底下的文件中：比如之前food的service就是在com.cn.service下面那就直接复制到customer的相同路径下，如果没有就手动创建，注意，serviceImp就不创建了。



Service代码如下：



8.可以编写customer的测试代码测试 如果测试成功表明dubbo已被打通如果启动出问，请检查dubbo的依赖版本是不是和springboot对应。