一 微服务之间相互调用

1 导入 ribbon 依赖

```
<!--ribbon-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-ribbon</artifactId>
</dependency>
```

2 创建 RestTemplate

```
@SpringBootApplication
@EnableEurekaClient
@MapperScan("cn.shu.cart.mapper")
public class StarterCart {
    public static void main(String[] args) {
        SpringApplication.run(StarterCart.class,args);
    }
    @Bean
    @LoadBalanced
    public RestTemplate init(){
        return new RestTemplate();
    }
}
```

3 注入 RestTemplate

```
@Autowired
private RestTemplate restTemplate=null;
```

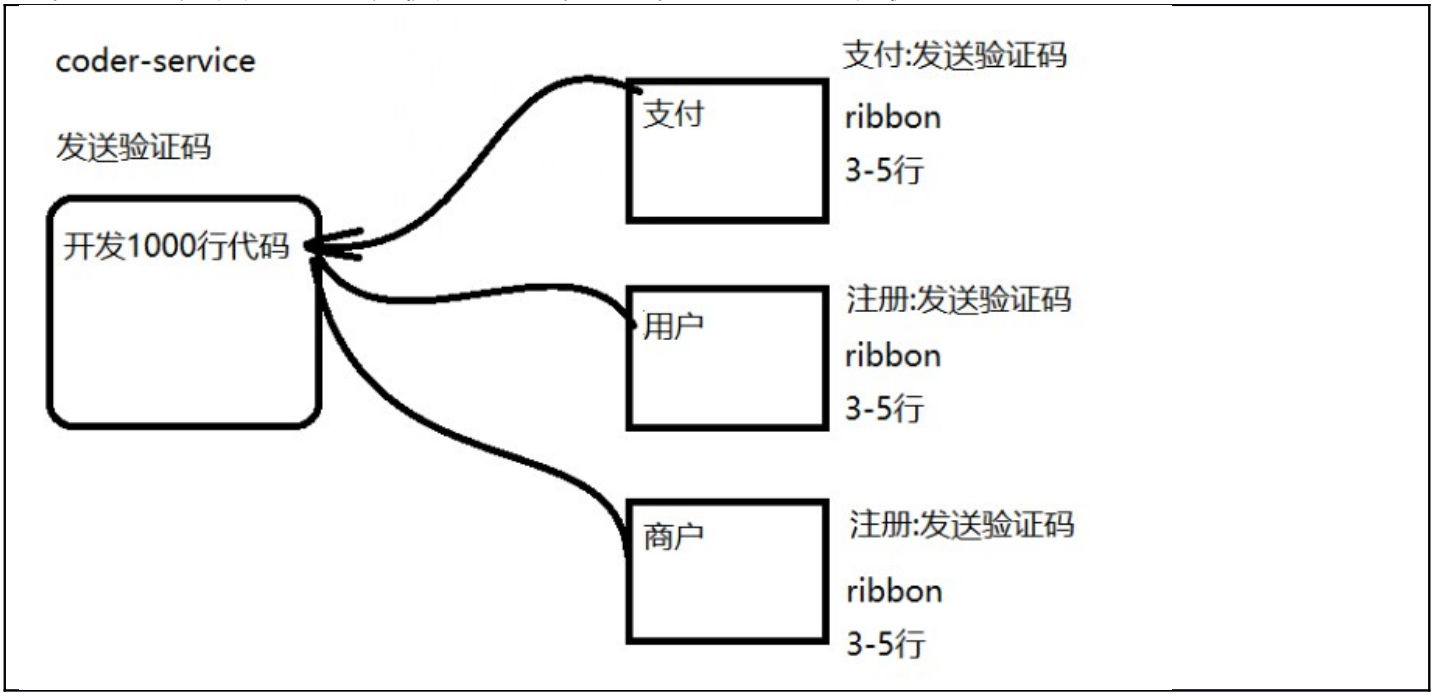
4 调用服务

```
String url="http://easymall-product/product/manage/item/"+cart.getProductId();
Product forObject = restTemplate.getForObject(url, Product.class);
```

二 feign 使用场景

之前微服务调用结构：

多个微服务调用同一服务使用 ribbon，有重复代码。能否提取出来?



在微服务框架中,对于多个微服务调用同一个微服务的功能,能否编写成更简单的形式,实现公用服务调用---feign

三　feign 介绍

也是一个服务调用的客户端,也可以实现负载均衡.和 ribbon 有关系.为了简化服务调用形式,springcloud 封装了 ribbon+restTemplate 形成了一个新的组件 feign. ribbon 和 feign 完全可以相互代替使用.

组件底层:ribbon

公用服务调用:feign

四　feign 测试案例

1　pom 文件

1.A　继承 spring-boot

1.B　依赖 feign /eureka（抓取服务信息）

```
<!--eureka 客户端-->
```

```xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-eureka</artifactId>
</dependency>
<!--feign-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-feign</artifactId>
</dependency>
```

2 启动类

```java
@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients
@MapperScan("cn.shu.cart.mapper")
public class StarterCart {
    public static void main(String[] args) {
        SpringApplication.run(StarterCart.class,args);
    }
}
```

3 编写服务接口

该接口和 easymall-product 服务的 controller 接口一致，当调用该方法时，会自动调用 easymall-product 中的指定接口

```java
//调用easymall-product 微服务
@FeignClient(name="easymall-product")
public interface ProductService {
    //单个商品查询
    @RequestMapping("/product/manage/item/{productId}")
    public Product queryOneProduct(@PathVariable String productId);
}
```

easymall-product 中的 controller 接口：

```java
@RestController
@RequestMapping("/product/manage")
public class ProductController {
    @Autowired
    private ProductService productService;

    //单个商品查询
    @RequestMapping("/item/{productId}")
    public Product queryOneProduct(@PathVariable String productId){
        return productService.queryOneProduct(productId);
    }

}
```

4 使用

```java
//注入服务对象
@Autowired
```

```
private ProductService productService=null;

//使用
Product forObject = productService.queryOneProduct(cart.getProductId());
```

五　feign 的实际应用

编写的服务接口和被调用者的 controller 一致，所以应该由被调用者来写这个接口，然后调用者添加依赖使用相关接口