通过上节课程，我们已经学会如何通过LoadBalancerClient接口来获取某个服务的具体实例，并根据实例信息来发起服务接口消费请求。但是这样的做法需要我们手工的去编写服务选取、链接拼接等繁琐的工作，对于开发人员来说非常的不友好。所以，下来我们看看Spring Cloud中针对客户端负载均衡的工具包：Spring Cloud Ribbon。

Spring Cloud Ribbon

Spring Cloud Ribbon是基于Netflix Ribbon实现的一套客户端负载均衡的工具。它是一个基于HTTP和TCP的客户端负载均衡器。它可以通过在客户端中配置ribbonServerList来设置服务端列表去轮询访问以达到均衡负载的作用。

当Ribbon与Eureka联合使用时，ribbonServerList会被DiscoveryEnabledNIWSServerList重写，扩展成从Eureka注册中心中获取服务实例列表。同时它也会用NIWSDiscoveryPing来取代IPing，它将职责委托给Eureka来确定服务端是否已经启动。

而当Ribbon与Consul联合使用时，ribbonServerList会被ConsulServerList来扩展成从Consul获取服务实例列表。同时由ConsulPing来作为IPing接口的实现。

我们在使用Spring Cloud Ribbon的时候，不论是与Eureka还是Consul结合，都会在引入Spring Cloud Eureka或Spring Cloud Consul依赖的时候通过自动化配置来加载上述所说的配置内容，所以我们可以快速在Spring Cloud中实现服务间调用的负载均衡。

下面我们通过具体的例子来看看如何使用Spring Cloud Ribbon来实现服务的调用以及客户端均衡负载。

实战练习

下面的例子，我们将利用之前构建的eureka-server作为服务注册中心、eureka-client作为服务提供者作为基础。而基于Spring Cloud Ribbon实现的消费者，我们可以根据eureka-consumer实现的内容进行简单改在就能完成，具体步骤如下：

- 根据eureka-consumer复制一个服务消费者工程，命名为：eureka-consumer-ribbon。在pom.xml中增加下面的依赖：

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

- 修改应用主类。为RestTemplate增加@LoadBalanced注解：

```
@EnableDiscoveryClient
@SpringBootApplication
public class EurekaConsumerRibbonApplication {

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
```

```
    public static void main(String[] args) {
        new
SpringApplicationBuilder(EurekaConsumerRibbonApplication.class).web(true).run(args);
    }
}
```

- 修改Controller。去掉原来通过LoadBalancerClient选取实例和拼接URL的步骤，直接通过RestTemplate发起请求。

```
@RestController
public class DcController {

    @Autowired
    RestTemplate restTemplate;

    @GetMapping("/consumer")
    public String dc() {
        String url =  "http://eureka-client/dc";
        return restTemplate.getForObject(url, String.class);
    }
}
```

可以看到这里，我们除了去掉了原来与LoadBalancerClient相关的逻辑之外，对于RestTemplate的使用，我们的第一个url参数有一些特别。这里请求的host位置并没有使用一个具体的IP地址和端口的形式，而是采用了服务名的方式组成。那么这样的请求为什么可以调用成功呢？因为Spring Cloud Ribbon有一个拦截器，它能够在这里进行实际调用的时候，自动的去选取服务实例，并将实际要请求的IP地址和端口替换这里的服务名，从而完成服务接口的调用。

在完成了上面你的代码编写之后，读者可以将eureka-server、eureka-client、eureka-consumer-ribbon都启动起来，然后访问http://localhost:2102/consumer ，来跟踪观察eureka-consumer-ribbon服务是如何消费eureka-client服务的/dc接口的，并且也可以通过启动多个eureka-client服务来观察其负载均衡的效果。

具体工程说明如下：

- eureka的服务注册中心：eureka-server
- eureka的服务提供方：eureka-client
- eureka的服务消费者：eureka-consumer-ribbon