REST Client
with Spring Cloud Feign

Spring Cloud

- **Spring Cloud Contract**: supports and facilitates contract testing
- **Spring Cloud Netflix**: umbrella project offering a number of Netflix services and libraries adapted for Spring:
  - Hystrix, Eureka, Ribbon, **Feign**, Zuul

- **Spring Cloud Config Server**: configuration as a service
- **Spring Cloud Sleuth**: distributed tracing

Spring Cloud Feign

- Encapsulates the details of REST API calls behind an interface
- Integrated with Eureka and Ribbon

Procedure

1. Add dependency: `spring-cloud-starter-feign`
2. Annotate Spring Boot application class with `@EnableFeignClients`
3. Define the interface

Feign Interface: Simple Example

a. Annotate class with `@FeignClient` and indicate the service id of the backing service this interface represents

b. Map REST API calls to interface methods

c. Annotate each method with the familiar `@RequestMapping` annotation

```
 8
 9   @FeignClient("fortune")
10   public interface FortuneAPI {
11
12     @RequestMapping("/")
13     Map<String, String> getFortune();
14   }
```

Usage

- Auto-wire the interface into any client class
- Anywhere a REST API call needs to be made against the backing service, just invoke the corresponding interface method instead
- Eureka url lookup and Ribbon load balancing still take place
- In addition we've neatly encapsulated a set of related REST API calls behind an interface

```
15      private final FortuneAPI fortuneAPI;
16
17      public FortuneServiceClient(FortuneAPI fortuneAPI) {
18        this.fortuneAPI = fortuneAPI;
19      }
20
21      @HystrixCommand(fallbackMethod = "defaultFortune")
22      String getFortune() {
23        Map map = fortuneAPI.getFortune();
24        return (String) map.get("fortune");
25      }
```

Contrast with RestTemplate

```
16      private final RestTemplate restTemplate;
17
18      public FortuneServiceClient(RestTemplate restTemplate) {
19        this.restTemplate = restTemplate;
20      }
21
22      @HystrixCommand(fallbackMethod = "defaultFortune")
23      String getFortune() {
24        Map map = restTemplate.getForObject( url: "http://fortune/", Map.class);
25        return (String) map.get("fortune");
26      }
27
```

```
15      private final FortuneAPI fortuneAPI;
16
17      public FortuneServiceClient(FortuneAPI fortuneAPI) {
18        this.fortuneAPI = fortuneAPI;
19      }
20
21      @HystrixCommand(fallbackMethod = "defaultFortune")
22      String getFortune() {
23        Map map = fortuneAPI.getFortune();
24        return (String) map.get("fortune");
25      }
```