Soyez une “feign”-asse quand vous écrivez un client REST java

Alexandre Navarro @alex_j_navarro

Introduction

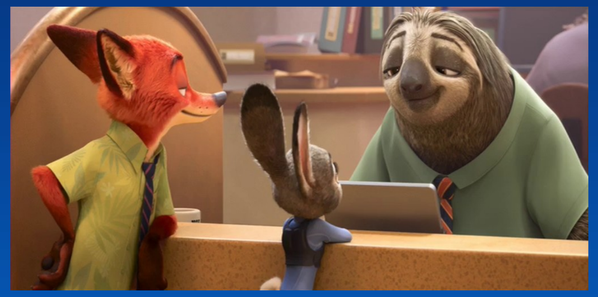
Feign, c’est quoi?

Feign Customisation

Feign + Spring

Plus loin avec Feign

Moi en train de coder sans Feign



Feign, c’est quoi?

Lib de Netflix pour appeler un service REST en étant productif

3 Étapes pour le développeur :

1) Créer une interface du service REST avec des Annotations + POJO

2) Binder votre interface pour pouvoir appeler votre interface

3) Appeler la méthode de l’interface votre service REST

Feign Sample : Resource

```
public interface GithubWithFeignResource {

    @RequestLine("GET /repos/{owner}/{repo}/contributors")
    List<Contributor> contributors(@Param("owner") String owner, @Param("repo") String repo);

}
```

Feign Sample : Binding / Appel

```
GithubWithFeignResource githubWithFeignResource =
        Feign.builder()
        .decoder(new JacksonDecoder())
        .target(GithubWithFeignResource.class, "https://api.github.com");


List<Contributor> contributors = githubWithFeignResource.contributors("OpenFeign", "feign");
```

Feign Customisation

Ajouter des paramètres à la request : Annotation (@Body / @Header)

Sérialiser en json / xml : Encoder / Decoder (jackson / gson / jaxb)

Utiliser un client HTTP/2 ou load balancer : Client (OkHttp / Ribbon)

Pour logger / monitorer ou ajouter un token : Request Interceptor

Pour formaliser vos contrats : Contract (Feign, JaxRs ou Spring MVC)

Feign + Spring = ❤️

Contract : Spring : @RequestMapping / @RequestParam ...

Facilité avec Spring Boot / Spring Cloud

- Binding via Annotation @EnableFeignClients / @FeignClient

- Discovery Service (Eureka / Consul)

- Spring Client load balancing (Ribbon)

- Circuit Breaker (Hystrix)

Feign/Spring : Resource

```
@FeignClient(url = "https://api.github.com", name = "github")
// remove url if github is your name in your eureka
public interface GithubWithSpringMvcResource {

    @RequestMapping(method = RequestMethod.GET, value = "/repos/{owner}/
{repo}/contributors")
    List<Contributor> contributors(@PathVariable("owner") String owner, @PathVariable("repo")
String repo);
}
```

Feign/Spring : Binding

```
@SpringBootApplication
@EnableFeignClients(basePackageClasses = {GithubWithSpringMvcResource.class})
public class GithubSpringBootApplication {

    public static void main(String[] args) {
        SpringApplication.run(GithubSpringBootApplication.class, args);
    }
}
```

Feign/Spring : Injection / Appel

```
@Component
public class GithubCommandLineRunner implements CommandLineRunner {

    private final GithubWithSpringMvcResource githubWithSpringMvcResource;

    public GithubCommandLineRunner(final GithubWithSpringMvcResource
githubWithSpringMvcResource) {
        this.githubWithSpringMvcResource = githubWithSpringMvcResource;
    }

    @Override
    public void run(String... strings) throws Exception {
        final List<Contributor> contributors =
githubWithSpringMvcResource.contributors("OpenFeign", "feign");
    }
}
```

Plus loin avec Feign / Spring

Partagez vos clients feign entre vos microservices (ou pas) ?

Testez vos microservices avec vos clients Feigns

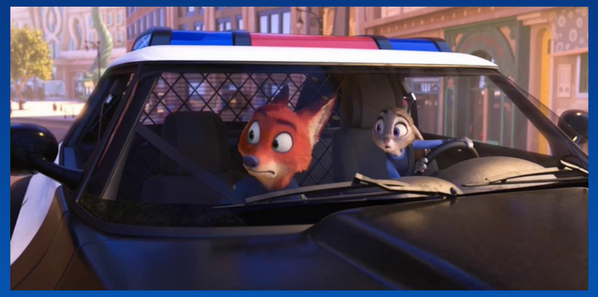
Attention certains pièges

- Regex in path non supporté in @FeignClient (spring-cloud-netflix/issues/816)

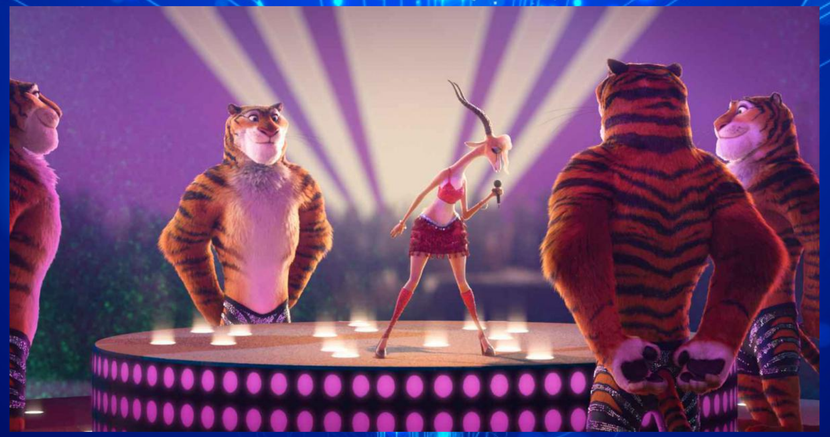
- @GetMapping non supporté @FeignClient (spring-cloud-netflix/issues/1813)

- @RequestParam non supporté via des interfaces dans un @RestController (SPR-11055)

Moi en train de coder avec Feign



Merci
A vous de jouer les “feign-asses” et les paresseux



Alexandre Navarro @alex_j_navarro