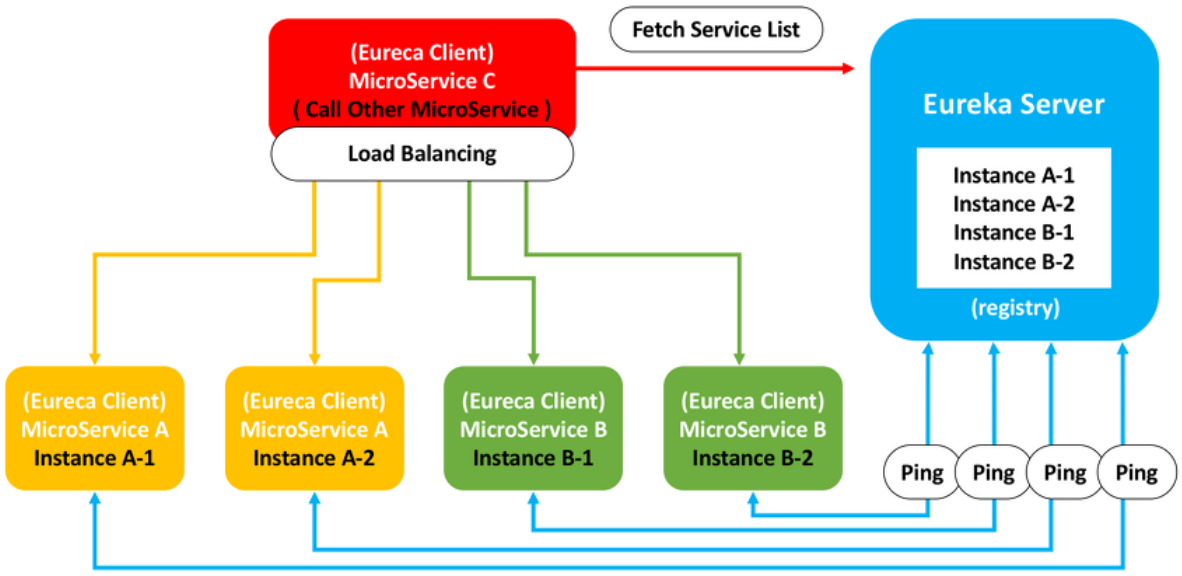
Eureka 자료조사

목차

1. Eureka

Eureka란?

마이크로서비스들의 정보를 레지스트리에 등록할 수 있도록 하고 마이크로서비스의 도적인 탐색과 로드 밸런싱을 제공해준다.



Eureka는 Eureka Server와 Eureka Client로 구성되며, Eureka Server는 Eureka Client에 해당하는 마이크로서비스들의 상태 정보가 등록되어있는 레지스트리를 갖는다.

Eureka Client의 서비스가 시작될 때 Eureka Server에 자신의 정보를 등록한다. 등록된 후에는 30초마다 레지스트리에 ping을 전송하여 자신이 가용 상태임을 알리는데 일정 횟수 이상 ping이 확인되지 않으면 Eureka Server에서 해당 서비스를 레지스트리에서 제외시킨다.

레지스트리의 정보는 모든 Eureka Client에 복제되어 있어 필요할 때마다 가용 상태인 모든 서비스들의 목록을 확인할 수 있고 이 목록은 30초마다 갱신된다. 가용 상태의 서비스 목록을 확인할 경우에는 서비스의 이름을 기준으로 탐색하며 로드밸런싱을 위해 내부적으로 Ribbon(클라이언트 측의 로드밸런서)을 사용한다.

Eureka 적용

1. Dependency를 추가한다. 어노테이션을 통하여 Eureka Server로 사용이 가능하다. (Server)

```
implementation("org.springframework.cloud:spring-cloud-starter-netflix-eureka-server:2.1.2.RELEASE")
```

```java
@EnableEurekaServer
@SpringBootApplication
public class EurekaServiceApplication {

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

2. Application.yml 파일에 Eureka Server 설정에 필요한 정보를 추가한다. (Server)

```yaml
eureka:
  instance:
    hostname: localhost
  client:
    register-with-eureka: false
    fetch-registry: false
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

3. Dependency를 추가한다.(Client)

```
implementation("org.springframework.cloud:spring-cloud-starter-netflix-eureka-client:2.1.2.RELEASE")
```

4. 기본적인 ping을 구현하는 controller

```java
@SpringBootApplication
public class OrderApplication {
    public static void main(String[] args) {
        SpringApplication.run(OrderApplication.class, args);
    }
}
```

```java
@RestController
public class HealthCheckController {
    @GetMapping("/ping")
    public ResponseEntity<String> healthCheck() {
        log.info("###health check");
        return ResponseEntity.ok("pong");
    }
}
```

5. Application.yml 파일에 Client 설정 기입

```yaml
server:
  port: 8080

spring:
  application:
    name: service-1

eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

2. Feign

1. Feign란?

넷플릭스에서 개발된 HTTP client binder로 웹 서비스 클라이언트 작성을 보다 용이하게 해준다.

2. Feign 적용

1. Dependcy 추가

```
dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:Hoxton.SR8"
    }
}

dependencies {
    implementation "org.springframework.cloud:spring-cloud-starter-openfeign"
}
```

2. FeignClient 선언

```
@FeignClient(name = "azureClient" url = "https://api.cognitive.microsofttranslator.com")
public interface AzureClient {

  @PostMapping(value = "/translate", produces = "application/json", consumes = "application/json")
  String translateSingleSentence(
      @RequestHeader("Ocp-Apim-Subscription-Key") String key,
      @RequestHeader("Ocp-Apim-Subscription-Region") String region,
      @RequestParam("api-version") String version,
      @RequestParam("from") Locale from,
      @RequestParam("to") Locale to,
      @RequestBody String content);
}
```

속성

- Name: 서비스 이름(필수)
- url: interface baseUrl
- qualifier: beanName
- configuration: 커스텀마이징한 configuration을 넣을 수 있다.
- Fallback: Hystrix fallback 매서드