API Gateway Service-2

작성 일시	@2022년 10월 27일 오후 9:06
최종 편집 일시	@2022년 10월 31일 오후 10:41
유형	Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA)
작성자	
참석자	

Spring Cloud Gateway
Spring Cloud Gateway - Filter

Spring Cloud Gateway

Spring boot 2.4.x 이상 버전으로 올리고 ,zuul → Gateway 로 변환한다.

장점으로는 비동기 처리가 가능하다.

- application.yml

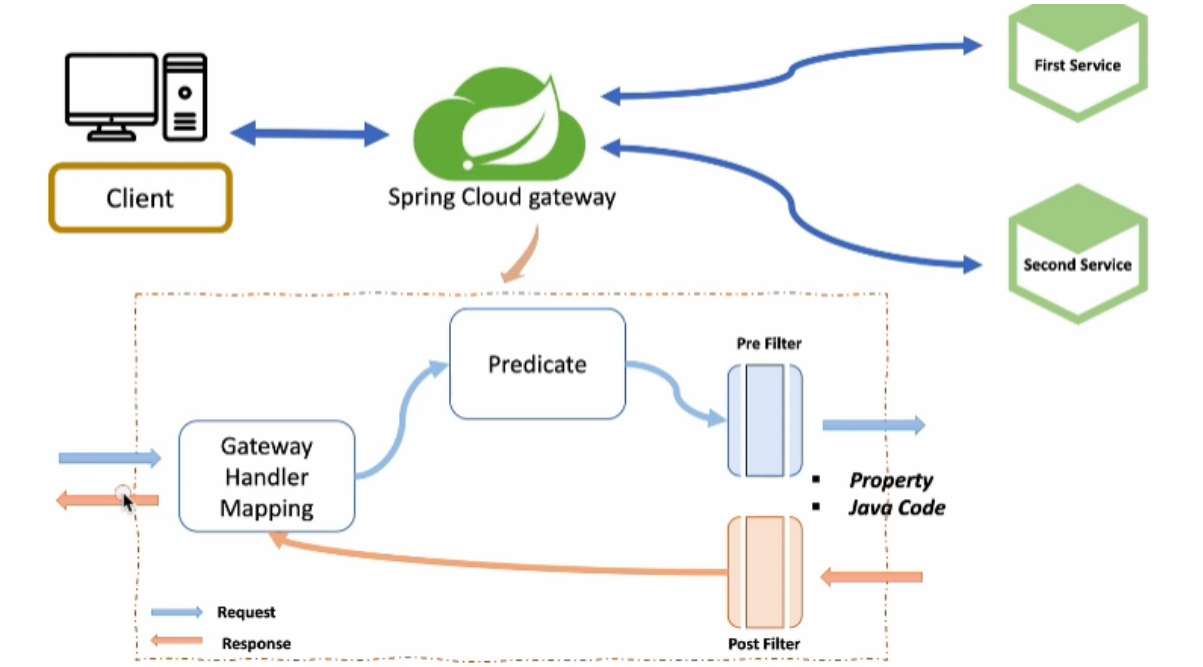
```
zuul:
  routes:
    first-service:
      path: /first-service/**
      url: http://localhost:8001
    second-service:
      path: /second-service/**
      url: http://localhost:8002
```

```
cloud:
  gateway:
    routes:
      - id: first-service
        url: http://localhost:8001
        predicates:
          -Path=/first-service/**
      - id: second-service
        url: http://localhost:8002
        predicates:
          -Path=/second-service/**
```

- 실행하게 되면, tomcat 이 아닌 netty 라는 것으로 실행된다. (tomcat - 동기 방식) (netty - 비동기 방식)
- gateway 를 사용하면서, 기존 first, second service 의 controller 에 url를 변경해줘야 한다. (/first-service)

Spring Cloud Gateway - Filter

- Spring Cloud Gateway 흐름도



- 기존 gateway(yml 내용)을 java 로 mapping 가능 (+ header 에 key, value 추가)

```
 // java filter 방식
@Configuration
public class FilterConfig {
    @Bean
    public RouteLocator gatesWayRoutes
(RouteLocatorBuilder builder) {
        return builder.routes()
            .route(r -> r.path("/first
-service/**")
                .filters(f -> f.addReq
uestHeader("first-request", "first-req
uest-header-value")
                    .addResponseHeader
("first-response", "first-response-hea
der-value"))
                .uri("http://localhos
t:8081"))
            .route(r -> r.path("/secon
d-service/**")
                    .filters(f -> f.ad
dRequestHeader("second-request", "seco
nd-request-header-value")
                            .addRespon
seHeader("second-response", "second-re
sponse-header-value"))
                    .uri("http://local
host:8082"))
            .build();
```

```
spring:
  application:
    name: apigateway-service
#  yml filter방식
cloud:
    gateway:
      routes:
        - id: first-service
          uri: http://localhost:8081/
          predicates:
            - Path=/first-service/**
          filters:
            -  AddRequestHeader=first-
request, first-request-header2
            -  AddResponseHeader=first
-response, first-response-header2
        - id: second-service
          uri: http://localhost:8082/
          predicates:
            - Path=/second-service/**
          filters:
            - AddRequestHeader=second-
request, second-request-header2
            - Ad
```

```
    }
}
```

- Custom Filter 적용 (pre, post)

```
spring:
  application:
    name: apigateway-service
#  yml filter방식
cloud:
    gateway:
      routes:
        - id: first-service
          uri: http://localhost:8081/
          predicates:
            - Path=/first-service/**
          filters:
#            -  AddRequestHeader=first
-request, first-request-header2
#            -  AddResponseHeader=firs
t-response, first-response-header2
            - CustomFilter
        - id: second-service
          uri: http://localhost:8082/
          predicates:
            - Path=/second-service/**
          filters:
#            - AddRequestHeader=second
-request, second-request-header2
#            - AddResponseHeader=secon
d-response, second-response-header2
            - CustomFilter
```

```
@Component
@Slf4j
public class CustomFilter extends Abst
ractGatewayFilterFactory<CustomFilter.
Config> {
    public CustomFilter() {
        super(Config.class);
    }

    @Override
    public GatewayFilter apply(Config
 config) {
        // Custom pre Filter
        return (exchange, chain) -> {
            ServerHttpRequest request
 = exchange.getRequest();
            ServerHttpResponse respons
e = exchange.getResponse();

log.info("Custom Pre Filter: request i
d -> {}", request.getId());

            // Custom Post Filter
            // webflux(spring 5) Mono
 단일값 전달 한다는 의미이며, 비동기적 netty 서
버에서 지원함.
            return chain.filter(exchan
ge).then(Mono.fromRunnable(() -> {
log.info("Custom Post Filter: response
code -> {}", response.getStatusCode
());
            }));
        };
    }

    public static class Config {
        // put the configuration prope
rties
    }
}
```