# [MSA] Filter

| | |
|---|---|
| Created | @2021년 11월 11일 |
| Tags | micro service |
| 스터디 | 개인 |

## Gateway

게이트웨이 없이 클라이언트가 서비스를 집접 연결하여 사용한다면 로깅과 보안을 구성하는 것이 번거롭다.

## 기능

- 라우팅: url과 api 경로로 알맞은 서비스를 호출한다.
- 인증과 인가: 모든 호출은 게이트웨이를 통해야하므로 클라이언트를 인증하고 권한을 확인할 수 있다.
- 로깅: 사용자의 서버에 접근에 대해서 필요한 데이터를 기록할 수 있다.

## Filter

- prefilter(사전필터): 요청이 서비스로 가기 전 게이트웨이에서 실행, 서비스의 일관된 메시지 형식을 확인하는 작업 수행 또는 사용자 인증 및 인가
- postfilter(사후필터): 요청이 서비스에서 처리된 후 응답이 게이트웨이를 지날 때 실행, 서비스의 응답을 로깅하거나 에러 처리, 민감한 정보에 대한 응답 감시
- route filter(경로필터): 서비스가 호출되기 전에 실행. 동적 라우팅 필요 여부를 결정하는 데에 사용.

Route Filter를 사용하면 클라이언트에서 같은 경로를 호출해도 다른 서비스로 연결해줄 수 있다

### Prefilter 사용 예시

- ZuulFilter를 사용
- 게이트웨이로 호출 되는 모슨 서비스에 적용
- filterOrder 메소드를 오버라이딩하여 필터 순서 설정 가능

```java
@Component
public class TrackingFilter extends ZuulFilter {
    private static final int FILTER_ORDER = 1;
    private static final boolean SHOULD_FILTER = true;
    private static final Logger logger = LoggerFactory.getLogger(TrackingFilter.class);

    @Autowired
    private FilterUtils filterUtils;

    @Override
    public String filterType(){
        return filterUtils.PRE_FILTER_TYPE;
    }

    @Override
    public int filterOrder(){
        return FILTER_ORDER;
    }
```

- run 메소드를 생성하여 zuulfilter 사용

```java
public Object run() {
    if(isCorrelationIdPresent()) {
        logger.debug("tmx-correlation-id found in tracking filter: {}.", filterUtils.getCorrelationId());
    } else {
        filterUtils.setCorrelationId(generateCorrelationId());
        logger.debug("tmx-correlation-id generated in tracking filter: {}.", filterUtils.getCorrelationId());
    }
    RequestContext ctx = RequestContext.getCurrentContext();
    logger.debug("Processing incoming request for {}.", ctx.getRequest().getRequestURL());
    return null;
}
```

→ zuul 라이브러리의 RequestContext를 사용하여 현재 요청 수정/확인 가능

## Postfilter 사용 예시

- prefilter와 동일한 방법으로 사용 가능

```java
@Override
public String filterType() {
    return FilterUtils.POST_FILTER_TYPE;
}

@Override
public int filterOrder() {
    return FILTER_ORDER;
}

@Override
public Object run() {
    RequestContext ctx = RequestContext.getCurrentContext();

    logger.debug("Adding the correlation id to the outbound headers. {}", filterUtils.getCorrelationId());
    ctx.getResponse().addHeader(FilterUtils.CORRELATION_ID, filterUtils.getCorrelationId());

    logger.debug("Completing outgoing request for {}.", ctx.getRequest().getRequestURI());

    return null;
}
```

## Route Filter 사용 예시

- AB 테스트 후 대체 서비스로 연결

```java
@Override
public Object run() {
    RequestContext ctx = RequestContext.getCurrentContext();

    // 대체 가능한 서비스 탐색
    AbTestingRoute abTestingRoute = getAbRoutingInfo(filterUtils.getServiceId());
    if(abTestingRoute!= null && useSpecialRoute(abTestingRoute)) { // useSpecialRoute로 경로 대체 결정
        String route = buildRouteString(ctx.getRequest().getRequestURI(),
                                        abTestingRoute.getEndpoint(),
                                        ctx.get("serviceId").toString());
        forwardToSpecialRoute(route); // 대체 서비스로 전달
    }
    return null;
}
```

- AB 테스트 코드 예시, 가중치와 난수를 비교했지만 새로운 버전또는 트래픽 분산으로 사용 가능

```java
public boolean useSpecialRoute(AbTestingRoute testRoute) {
    Random random = new Random();

    if(testRoute.getActive().equals("N")) return false;
    int value = random.nextInt((10-1)+1)+1;
    if(testRoute.getWeight() < value) return true;
    return false;
}
```

📌 트래픽 증가시에 도커 이미지를 사용해서 새로운 서비스 생성 후 서비스 디스커버리에 등록하여 오토 스케일 기능 구현 가능