API Gateway Service-1

작성 일시	@2022년 10월 26일 오후 11:32
최종 편집 일시	@2022년 10월 27일 오후 10:05
유형	Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA)
작성자	
참석자	

API Gateway 란

API Gateway Service

- 정의
  - 사용자가 설정한 라우팅 설정에 따라 클라이언트 대신해서 요청하고 응답을 다시 전달하는 Proxy 역할을 한다.
  - 시스템 내부 구조는 숨기고 외부 요청에 대해서 적절한 형태로 변형후 응답 할 수 있음.
  - 마이크로 서비스의 내용을 변경해서 사용하는 것은 문제가 되지 않음, 다만 client side에 이전 정보를 갖고 있는 경우로 인해 문제가 발생될 수 있음.
  - 해결책으로, 단일 진입로를 만들게되는데 이를 API GateWay 라고 한다. client 에서는 API Gateway로만 요청한다.
- 장점
  - 인증 및 권한 부여
  - 서비스 검색 통합
  - 응답 캐싱
  - 정책, 회로 차단 및 QoS 다시 시도

- 속도 제한
- 부하 분산 (Load Balancing)
- 로깅, 추적, 상관 관계
- 헤더, 쿼리 문자열 및 청구 변환
- IP 허용 목록에 추가

Netflix Ribbon

- Spring Cloud 에서의 MSA 간의 통신 (구 방식)
  - RestTemplate

```
RestTemplate restTemplate = new RestTemplate();
restTemplate.getForObject("http://localhost", User.class, 200);
```

  - Feign Client

```
// 호출이 필요한 마이크로 서비스 이름을 넣는다.
@FeignClient("stores")
public interface StoreClinet {
  @RequestMapping(method=RequestMethod.GET, value="/stores")
  List<Store> getStores();
```

- Ribbon: Client side Load Balancer
  - MSA Service 이름으로 호출
  - Health Check
  - API Gateway 가 client 단에 같이 있다고 보면 됨

Netflix Zuul

- Routing
- API Gateway

Netfilx Zuul 프로젝트 생성

Spring Boot 2.4.X 이상에선 Zuul, Ribben이 사용 불가능이다.

- First Service, Second Service
  - Spring boot: 2.3.10
  - Dependencies
    - Lombok, Spring web, Eureka Discovery Client
  - 간단한 Controller 구현 (@RestController)
- Zuul Service (API Gateway)
  - Spring boot: 2.3.10
  - Dependencies
    - Lombok, Spring web, Spring Cloud Gateway

Netfilx Zuul - filter 적용

- Zuul filter 간단히 적용

```
@Slf4j
@Component
public class ZuulLogginFilter extends ZuulFilter {

  // 필터에 적용될 내용
  @Override
  public Object run() throws ZuulException {
    log.info("***** printing logs: ");
    RequestContext ctx = RequestContext.getCurrentContext();
    HttpServletRequest request = ctx.getRequest();
    log.info("***** " + request.getRequestURL());
    return null;
  }

  // "pre" 호출 이전 - 인증
  // "post" 호출 이후 - 로깅
  @Override
  public String filterType() {
      return "pre";
  }

  // 여러 filter 가 있을시 순서를 지정
  @Overrid
  public int filterOrder() {
    return 1;
  }

  // 필터를 사용할 건지 지정
  @Override
  public boolean shouldFilter() {
```

```
    return true;
  }
}
```