**서킷브레이커를 통하여 장애 전파를 원천 차단**

**서킷브레이커 설정 전 호출**

* monolith 서비스와 reqres\_delivery 서비스를 실행한다.
* 부하 툴을 사용하여 주문을 넣어본다.
* **siege -c2 -t10S -v --content-type "application/json" 'http://localhost:8088/orders POST {"productId":2, "quantity":1}'**

**서킷브레이커 설정**

* monolith 서비스의 application.yaml 파일의 27 번째 라인의 주석을 해제한다.
* **feign:**
* **hystrix:**
* **enabled: true**
* **hystrix:**
* **command:**
* **# 전역설정**
* **default:**
* **execution.isolation.thread.timeoutInMilliseconds: 610**
* reqres\_delivery 서비스의 Delivery.java 에 강제 딜레이 발생 코드를 넣는다.
  + (49~53 line 주석해제)
* **try {**
* **Thread.currentThread().sleep((long) (400 + Math.random() \* 220));**
* **} catch (InterruptedException e) {**
* **e.printStackTrace();**
* **}**
* monolith 서비스와 reqres\_delivery 서비스를 종료하고 재실행한다.
* 부하 툴을 사용하여 주문을 넣어본다.
* **siege -c2 -t10S -v --content-type "application/json" 'http://localhost:8088/orders POST {"productId":2, "quantity":1}'**

**fallback 처리를 하여 유연하게 대처**

* reqres\_delivery 서비스가 중지된 상태로 주문을 넣어본다. ( 500 에러 )
  + http localhost:8088/orders productId=1 quantity=3
* monolith 서비스의 DeliveryService.java 의 FeignClient에 fallback 옵션을 준다.
* **@FeignClient(name ="delivery", url="${api.url.delivery}", fallback = DeliveryServiceImpl.class)**
* monolith 서비스를 재실행 후 주문을 넣어본다. ( 주문 가능 )
  + 이때 배송 서비스는 중지 상태 이어야 한다.
  + DeliveryServiceImpl 의 startDelivery 메서드가 실행되는 것을 확인 할 수 있다.

**checkpoint 체크 방법**

fallback 처리 여부를 확인 하기 위하여 monolith 서비스의 console 창을 선택하고, 상단메뉴의 labs > 결과제출 을 클릭하여 제출한다.

**다른 Circuit Breaker 들**

<https://dzone.com/articles/comparing-envoy-and-istio-circuit-breaking-with-ne?fbclid=IwAR0wYnXPiAZSVtluJ-17Ywb9dK3xrytAMo3ImIZv8KwoOo2WGGnyTKm6c04>

**Service Clear**

* 다음 Lab을 위해 기동된 모든 서비스 종료

**fuser -k 8088/tcp**

**fuser -k 8082/tcp**

**상세설명**

Checkpoints

1. fallback 설정후 장애시 default 결과를 송출 한다.