

TCT-시스템&솔루션개발 실기형 문제지

[2022년 #차]

사번		성명	
유의 사항	1. 공정한 평가를 위해 동료들 도와주는 행위, 보여주는 행위를 금지하고 있습니다. 2. 부정행위 적발 시, 응시한 평가는 0점 처리됩니다. 3. 본 시험지는 응시장 외부로 유출할 수 없으며, 시험 종료 후 감독관에게 제출해야 합니다.		

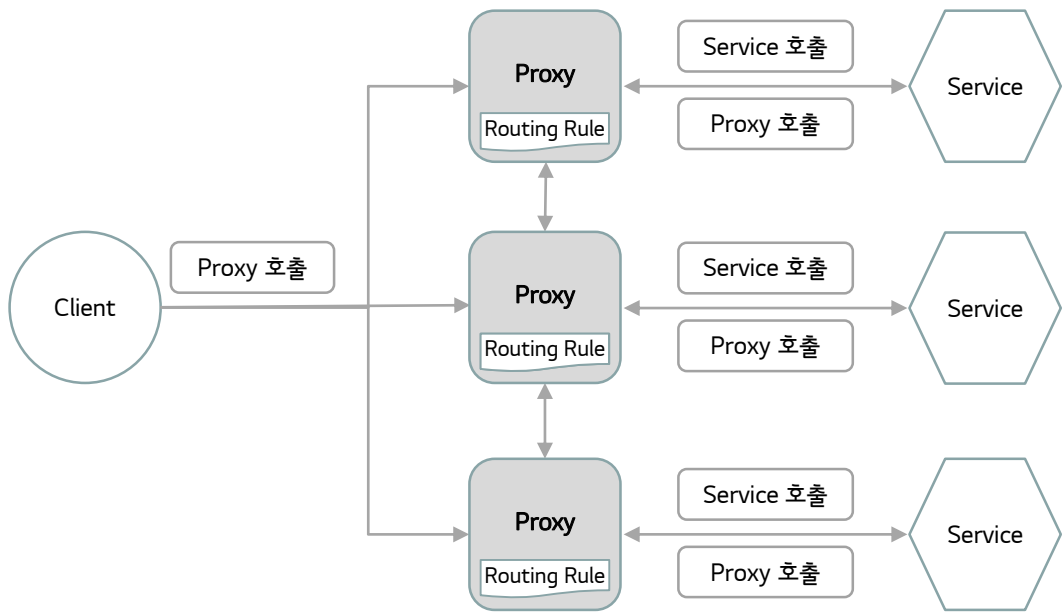
Service Proxy

개요

해당 시스템 구현을 통해 요구사항 분석, 파일처리, 데이터관리, HTTP Server/Client 구현 등의 기술역량 및 프로그램 구현 역량을 측정하기 위한 문제입니다.

설명

본 프로그램은 콘솔/HTTP 'Client'로 부터 'Proxy 호출'이 발생하면, 요청에 해당되는 'Service'를 검색하여 'Service 호출'을 하고 응답하는 'Service Proxy' 프로그램 입니다.
'Service Proxy'는 제공되는 Routing Rule에 따라 'Service 호출'을 수행하고 호출이력, 응답시간 등을 관리하여 'Service 추적', '회로 차단' 기능을 제공합니다.



[기능 요약]

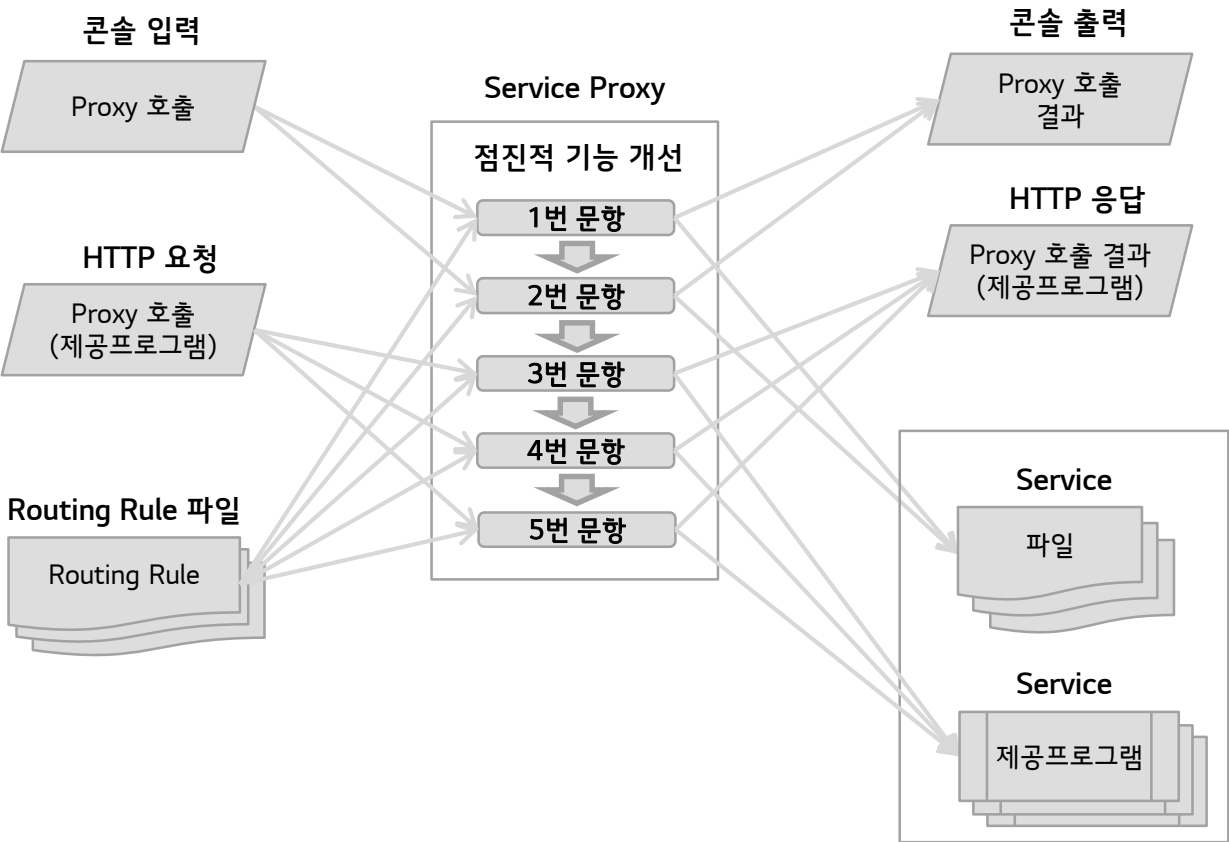
- 'Client'는 콘솔 입/출력 또는 HTTP 요청/응답 방식으로 'Service Proxy'를 통해 Service를 호출한다.
- Service Proxy는 Routing Rule에 따라 Service 또는 타 Service Proxy를 호출한다. ('Service 호출', 'Proxy 호출')
- 'Service Proxy'는 Service 또는 타 Service Proxy를 호출한 이력을 관리하여 Service간 호출 관계 정보를 제공한다. ('Service 추적')
- Service에서 비 정상적인 응답 상황이 감지되면 'Service 호출'을 일정 기간동안 차단한다. ('회로 차단')

주의사항

실행 결과로 평가하고 부분점수는 없으므로 아래사항을 필히 주의해야 함

- 구현된 프로그램은 실행 완결성 필수 (명확한 실행&종료 처리, 정확한 결과 출력, 통상의 실행 시간)
- 소 문항별 결과 검수 필수 (선행문항 오류 시, 후속문항 전체에 오류가 발생할 수 있음)
- 제시된 조건이 없는 한 선행요구사항 유지 필수 (소 문항별 입출력 관계도 참고)
- 프로그램 실행 위치 및 실행결과출력 (위치, 파일명, 데이터포맷)은 요구사항과 정확히 일치 필수
- 제시된 모든 위치는 상대경로 사용 필수 (프로그램 실행 위치 기준)
- 종료조건에 맞는 자동종료 처리 필수 (불필요한 종료방해처리(pause/입력대기 등)를 하면 안됨)
- 모든 문자는 대소문자 구분 필수

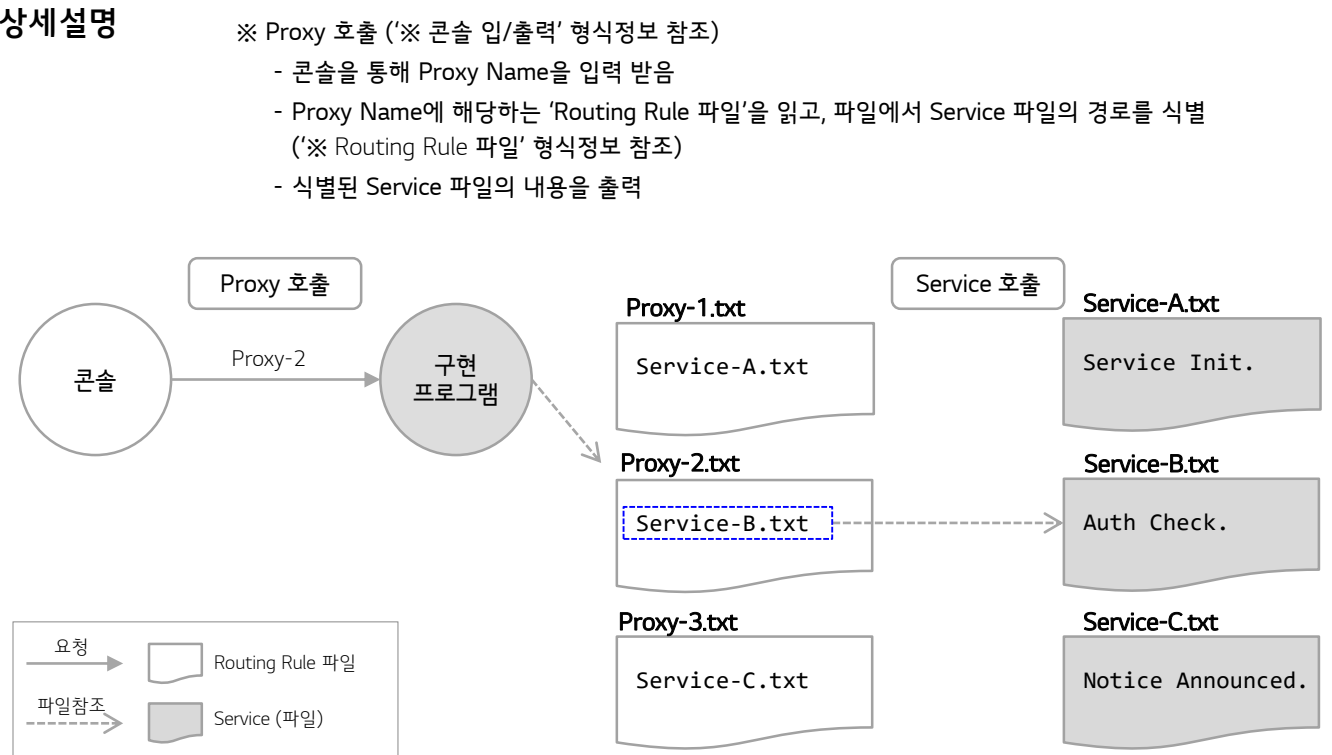
문항 관계



문제

아래 제시된 문항은 문항번호가 증가할 수록 점진적 개선을 요구하는 방식으로 구성되어 있으며, 제시된 문항번호 별로 각각 구현된 소스와 컴파일 된 실행파일을 제출하시오.
cf) 1번 구현 → 1번 소스복사 → 2번 구현 → 2번 소스복사 → ...

1. 콘솔 입력을 통해서 'Proxy Name'을 입력하면 해당 서비스 Proxy의 Routing Rule 파일을 읽어 식별된 Service 파일의 내용을 출력하는 프로그램을 구현하시오. (20점)



형식정보

※ 콘솔 입/출력

- 입력 포맷 : <Proxy Name>
- 출력 포맷 : <Service 파일 내용>

C:\>SP_TEST<엔터키>
Proxy-2<엔터키>
Auth Check.
C:\>

← 구현한 프로그램 실행 (Argument 없음)
← 콘솔 입력 (Proxy Name은 'Proxy-2')
← 콘솔 출력 ('Service' 파일 내용)

※ 'Routing Rule 파일' 형식정보

- 파일명 : <Proxy Name>.TXT (각 소문항 홈 아래)
- 데이터 포맷 : <Service 파일명>

Service-A.txt

※ Service (파일) 형식정보

- 파일위치 : 각 소문항 홈 아래

문제

2. 위 1번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 프로그램을 구현 하시오. (15점)

- Routing Rule 파일 형식이 변경됨 ('※ Routing Rule 파일 형식정보' 참조)

- 변경된 'Proxy호출' 구현 ('※ Proxy 호출 변경' 참조)

상세설명

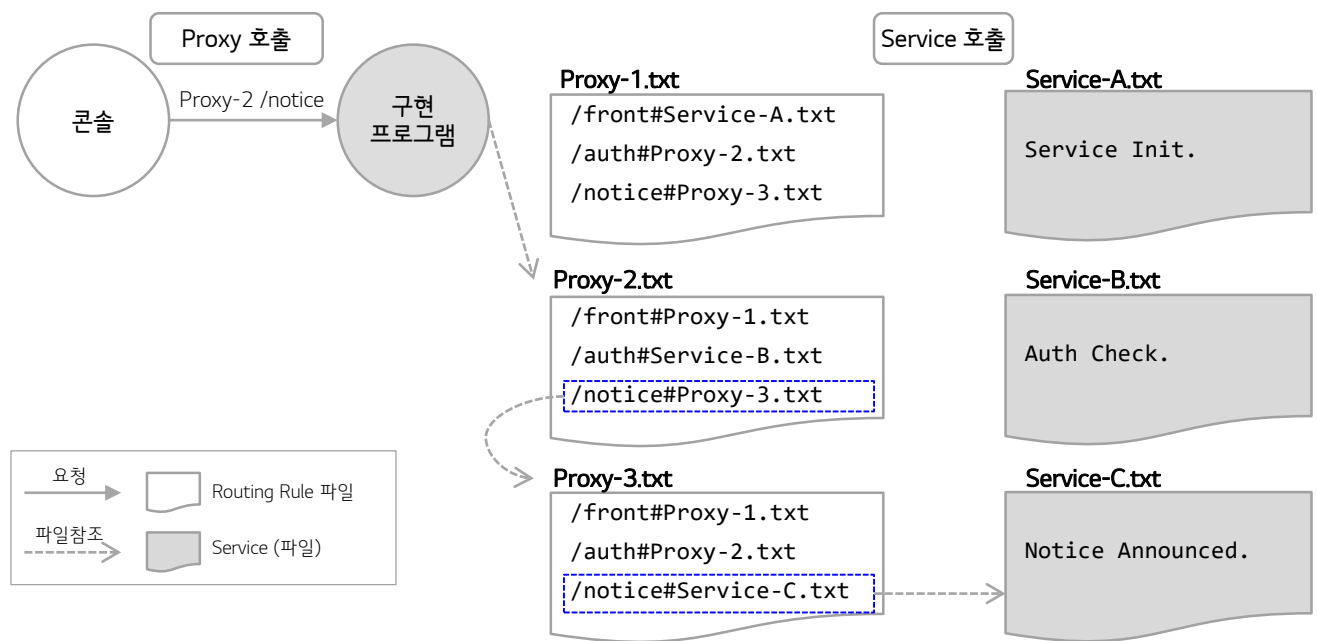
- ※ Proxy 호출 변경 ('※ 콘솔 입/출력' 형식정보 참조)
- 콘솔을 통해 'Proxy Name'과 'Path'를 입력 받는다.

- Proxy Name에 해당하는 'Routing Rule 파일'에서 'Path'에 해당되는 파일명을 식별

- 식별된 파일명이 'Proxy-'로 시작되는 경우 해당 'Routing Rule 파일'을 다시 읽고, 'Path'에 해당되는 파일명이 'Service-'로 시작되는 'Service 파일'의 경로를 식별
(한번의 Proxy 호출에 Routing Rule파일을 읽는 횟수는 최대 2회만 발생함)

- 최종적으로 식별된 'Service 파일'의 내용을 출력
- ※ Routing Rule 파일 형식 변경
- 'Service' 파일명 외에 타 'Routing Rule 파일'명이 추가됨

- 각 파일명은 Service Path로 구분됨 ('※ Routing Rule 파일 형식정보' 참조)



형식정보

- ※ 콘솔 입/출력
- 입력 포맷 : <Proxy Name> + " "(공백) + <Path>

- 출력 포맷 : <Service 호출 결과>
- C:\>SP_TEST<엔터키>

Proxy-2 /notice<엔터키>

Notice Announced.

C:\>

← 구현한 프로그램 실행 (Argument 없음)

← 콘솔 입력 (Proxy Name은 'Proxy-2', Path는 '/notice')

← 콘솔 출력 ('Service' 파일 내용)
- ※ Routing Rule 파일 형식정보
- 파일명 : <Proxy Name>.TXT (각 소문항 아래)

- 데이터 포맷 : <Service Path> + "#" + <Service 파일명 또는 Routing Rule 파일명>
- /front#Service-A.txt

/auth#Proxy-2.txt

/notice#Proxy-3.txt

- 문제
3. 위 2번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 HTTP 기반의 Service Proxy를 구현하시오. (15점)

- 지정된 Routing Rule에 따라 HTTP 요청을 전달하고 수신된 응답을 반환하는 Service Proxy 추가 구현 ('※ HTTP 기반 Service Proxy' 참조)

- 'Routing Rule 파일' 형식이 변경됨 ('※ Routing Rule 파일' 참조)

상세설명

- ※ 용어 기준
- 문항에 사용되는 Service 또는 Service Proxy의 주소 관련 용어 기준은 다음과 같음

http://127.0.0.1:5002/auth/lgcns?id=apple&key=DFGE

첫번째 Path

URL

Path

Query

URI

※ HTTP 기반 Service Proxy

- Service Proxy는 기동 시 첫번째 Argument로 경로가 입력되는 'Routing Rule 파일'에 정의된 설정을 읽어서 기동 ('※ Routing Rule 파일 형식정보' 참조)

- Service Proxy는 요청된 URI의 Path에 따라 'Routing Rule 파일'에 정의된 Route 정보를 선택하여 해당 Service 또는 타 Service Proxy를 호출

- Service Proxy는 Service 또는 타 Proxy 호출 시 요청된 HTTP Method, Path, Query를 그대로 전달하고, 수신된 응답의 Status, Content-Type Header, Body를 변경없이 응답으로 전달 (HTTP Method는 GET, POST 만 사용됨)

- 제공프로그램인 Service는 필요시 타 Service를 직접 호출하지 않고, Service Proxy를 통해서만 호출 함

※ HTTP 기반 Service Proxy 및 제공프로그램 호출 관계 예시
- 제공프로그램 (MOCK.BAT)

Client

Proxy 호출

Service 호출

Proxy 호출

Service 호출

Proxy 1 (port: 5001)

Proxy-1.json

Proxy 호출

Service 호출

Proxy 2 (port: 5002)

Proxy-2.json

Service 호출

Service A (port: 8081)


Service B (port: 8082)

요청

Service Proxy (구현 프로그램)

Service

Routing Rule 파일

※ SP_TEST.BAT를 실행하면, 동일 소스코드를 이용하여 복수 개의 Service Proxy가 실행됨
-  LG CNS

형식정보

- ※ Routing Rule 파일 형식정보
- 파일명 : <Proxy Name>.json (각 소문항 아래)
 - Service Proxy의 HTTP Port ("port")와 Service Proxy로 요청된 URI의 Path가 "pathPrefix"로 시작될 때 요청을 전달해야 할 대상의 URL ("url")이 설정되어 있음
 - Routing Rule 파일은 JSON 포맷으로 예시는 아래와 같음

개별 Route 정보
(반복)

```
{
  "port": 5001,
  "routes": [
    {
      "pathPrefix": "/front",
      "url": "http://127.0.0.1:8081",
    },
    {
      "pathPrefix": "/auth",
      "url": "http://127.0.0.1:5002"
    }
  ]
}
```

Service Proxy의 HTTP 포트

수신된 요청의 Path가 '/front'로 시작
되는 경우 http://127.0.0.1:8081로
요청 전달

- 제공되는 Routing Rule 파일은 아래와 같음

Proxy-1.json

```
{
  "port": 5001,
  "routes": [
    {
      "pathPrefix": "/front",
      "url": "http://127.0.0.1:8081"
    },
    {
      "pathPrefix": "/auth",
      "url": "http://127.0.0.1:5002"
    }
  ]
}
```

Proxy-2.json

```
{
  "port": 5002,
  "routes": [
    {
      "pathPrefix": "/auth",
      "url": "http://127.0.0.1:8082"
    }
  ]
}
```

- ※ 제공프로그램(MOCK.BAT)
- MOCK.BAT를 콘솔에서 실행하면 Client, Service가 기동 되며, 테스트 시나리오에 따라 Service Proxy의 기능을 순차적으로 테스트함
 - MOCK.BAT는 자가 검수의 기능도 수행하며, 모든 테스트 시나리오 성공 시 다음의 문구가 콘솔에 출력

C :>MOCK.BAT<엔터키>
...
테스트에 성공했습니다!

← 제공프로그램 실행

← 테스트 시나리오에 따른 테스트 실행

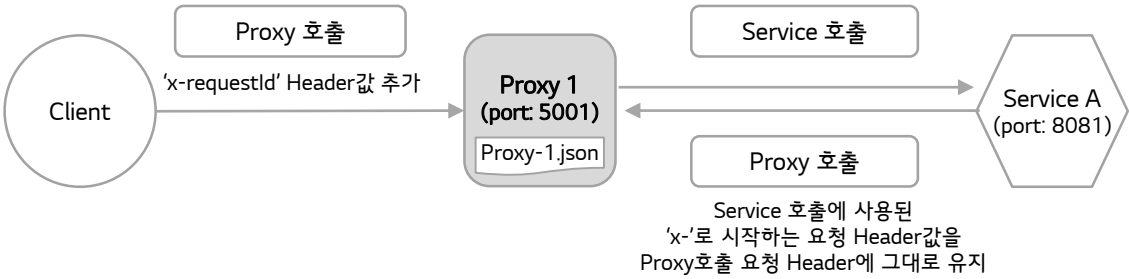
← 소문항의 모든 테스트시나리오 성공

- 문제
4. 위 3번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 Service Proxy를 구현하시오. (20점)
- Service 추적 API 추가 구현 ('※ Service 추적 API' 참조)

- 제공프로그램에 Header 관리가 추가됨 ('※ 제공프로그램 Header 관리' 참조)

상세설명

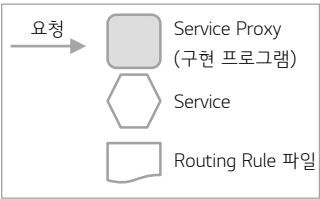
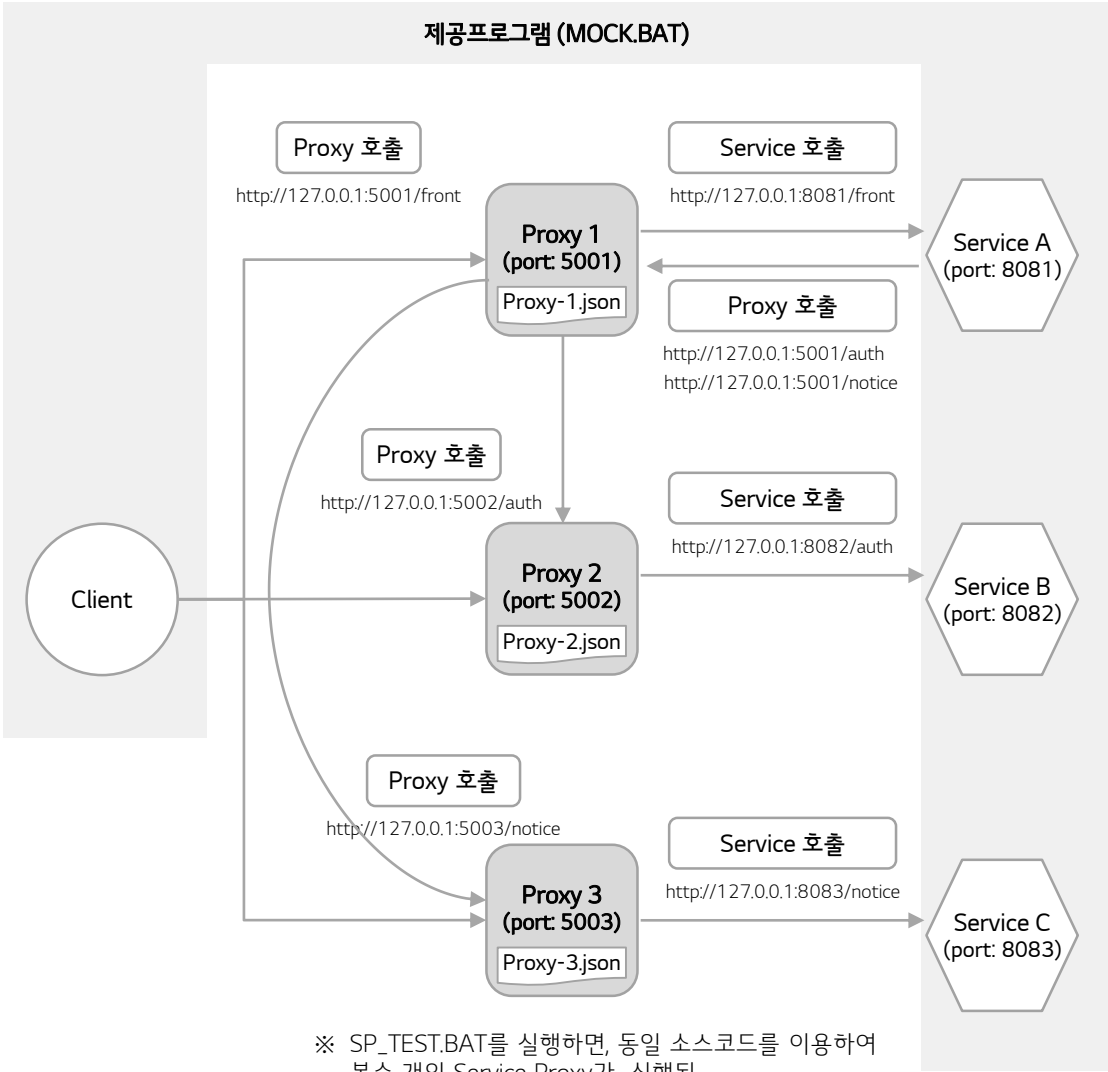
- ※ 제공프로그램 Header 관리 (구현 대상 아님)
- 제공프로그램인 Client는 매 'Proxy 호출' 시 고유의 '요청ID'를 생성하여 'x-requestId' HTTP Header값으로 설정하고 Proxy를 호출함
 - 제공프로그램인 Service는 'Proxy 호출' 시 수신된 HTTP 요청 Header 중 'x-'로 시작하는 Header 값을 그대로 유지하여 Proxy를 호출함



- ※ Service 추적 API
- Service Proxy에 '요청ID'별로 Proxy 및 Service 호출관계인 Service 추적정보를 응답하는 API를 추가
 - 제공프로그램 Header 관리를 이용하여 다양한 방식으로 구현 가능 (예 : 호출 이력 로그 파일, Service Proxy간 호출 이력 수집 API, 호출이력 수집 Server 등)
 - 요청 URI : GET <Service Proxy(URL)>/trace/<요청ID>
 - 응답 Body : 호출순서에 따라 계층구조로 표현된 JSON 형식 ('※ Service 추적 API 응답 형식' 참조)
 - 최초 Client에서 실행한 Service Proxy호출을 포함한 모든 Service Proxy 및 Service 호출에 대한 요청 주소, 응답 Status, 하위호출 정보를 출력
 - Service Proxy를 2번 이상 실행 해도 정상적으로 동작하도록 구현

상세설명
(계속)

※ HTTP 기반 Service Proxy 및 제공프로그램 호출 관계 예시



형식정보

- ※ Routing Rule 파일 형식정보
- 파일명 : <Proxy Name>.json (각 소문항 아래)
 - 제공되는 Routing Rule 파일은 아래와 같음

Proxy-1.json

```
{
  "port": 5001,
  "routes": [
    {
      "pathPrefix": "/front",
      "url": "http://127.0.0.1:8081"
    },
    {
      "pathPrefix": "/auth",
      "url": "http://127.0.0.1:5002"
    },
    {
      "pathPrefix": "/notice",
      "url": "http://127.0.0.1:5003"
    }
  ]
}
```

Proxy-2.json

```
{
  "port": 5002,
  "routes": [
    {
      "pathPrefix": "/auth",
      "url": "http://127.0.0.1:8082"
    }
  ]
}
```

Proxy-3.json

```
{
  "port": 5003,
  "routes": [
    {
      "pathPrefix": "/notice",
      "url": "http://127.0.0.1:8083"
    }
  ]
}
```

형식정보
(계속)

- ※ Service 추적 API 응답 형식
- 응답 Status : 200
 - 응답 Body : JSON 포맷으로 "target" 속성에 호출된 <Service 또는 Service Proxy URL> + <첫번째 Path>, "status" 속성에 호출 결과의 <응답 Status>, "services" 속성에 <하위 Service 또는 Service Proxy 호출 정보> 배열
 - 응답 Body 예시



문제

5. 위 4번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 Service Proxy를 구현하시오. (10점)

- 회로차단 기능 추가 구현 (※ 회로차단 기능' 참조)

- 'Routing Rule 파일' 형식이 변경됨 (※ Routing Rule 파일' 참조)

상세설명

- ※ 회로차단 기능
- 개별 Route 정보에 회로차단 기능 설정 추가 (※ Routing Rule 파일 형식정보' 참조)
 - “responseTimeLimit”, “maxDelayCount”, “retryAfter”, “healthCheck” 속성이 추가되었으며, 해당 속성이 존재하는 경우에만 회로차단 기능 작동

- 응답시간이 “responseTimeLimit” 밀리초를 초과하는 Service 호출 횟수가 “maxDelayCount”회 발생한 후에 요청되는 동일 Service 호출을 차단(회로차단)하고 오류 응답을 회신 (‘오류 응답 형식’ 참조)

- 오류 응답 형식
 - 응답 Status : 503
 - 응답 Body : JSON 문자열 형식, {“result”: “service unavailable”}

- 회로차단이 되지 않은 상태에서는 “responseTimeLimit”을 초과하더라도 정상 요청/응답을 수행

- 회로차단 후 “retryAfter” 밀리초 후에 차단된 Service에 “healthCheck” 속성의 Path를 호출하여 응답이 정상이면 회로차단을 해제하고, 비정상이면 동일 과정을 반복

- Service가 제공하는 “healthCheck” 속성의 Path를 호출한 응답은 다음과 같음
 - 정상 응답 : 응답 Status 200
 - 비정상 응답 : 응답 Status 503

형식정보

- ※ Routing Rule 파일 형식정보
- 파일명 : <Proxy Name>.json (각 소문항 아래)

- 개별 Route 정보에 “responseTimeLimit”, “maxDelayCount”, “retryAfter”, “healthCheck” 속성이 개별 Route 정보에 선택적으로 추가됨

- 제공되는 Routing Rule 파일은 아래와 같음 (Proxy-1.json, Proxy-2.json은 4번 문항과 동일)

Proxy-3json

```
{
  "port": 5003,
  "routes": [
    {
      "pathPrefix": "/notice",
      "url": "http://127.0.0.1:8083",
      "responseTimeLimit": 500,
      "maxDelayCount": 3,
      "retryAfter": 2000,
      "healthCheck": "/health"
    }
  ]
}
```

개별 Route 정보

회로차단 기능 설정

평가대상

프로그램 정상 실행, HTTP 응답 결과 (응답 Body, 응답 Status)

폴더 정보

- ※ 프로그램 및 파일 위치 정보 (실행위치 기반 상대경로 사용 필수)
- 구현할 프로그램 위치 및 실행 위치 : 각 소문항 홈 (SUB1/SUB2/SUB3/SUB4/SUB5)
 - Routing Rule 파일명 : 각 소문항 홈 아래 Proxy-*.TXT (SUB1/SUB2/SUB3/SUB4/SUB5)
 - Service 파일 명 : 각 소문항 홈 아래 Service-*.TXT (SUB1/SUB2)
 - 자가 검수용 참고 파일명 : COMPARE 폴더 내 CMP_CONSOLE.TXT (SUB1/SUB2)
 - 제공되는 MOCK 프로그램 파일명 : 각 소문항 홈 아래 MOCK.BAT (SUB3/SUB4/SUB5)
- * 제공되는 파일들은 문항에 따라 다를 수 있음

실행 방식

- ※ 구현할 프로그램 형식
- 프로그램 형태 : 콘솔(Console) 프로그램
 - 프로그램 파일명 : SP_TEST
 - 실행 방식(문항1~2) : 콘솔 실행→결과처리→자동종료

C:\>SP_TEST<엔터키>
Proxy-2 /auth<엔터키>
Auth Check.
C:\>

← 구현한 프로그램 실행 (Argument 없음)
← 콘솔 입력 (Proxy Name은 'Proxy-2', Service Path는 '/auth')
← 콘솔 출력 ('Service 호출' 결과)

- 실행방식(문항3~5) : 콘솔 실행→실시간 HTTP 요청 수신 (종료 없음)

C:\>SP_TEST.BAT<엔터키>
...

← 구현한 프로그램 실행 (Argument 없음)

제공 및 제출

- ✓ 각 언어별 제공파일 압축 해제 후 자동 생성된 폴더 사용 필수
 - ✓ 제공되는 주요 내용
 - 샘플 파일
 - 제공 프로그램 실행파일 (MOCK.BAT)
 - 구현 프로그램 실행파일 (SP_TEST.BAT, SUB3/SUB4/SUB5)
 - 제출시 사용할 문항별 폴더 구조
 - ✓ 제출 파일 및 폴더 상세 내용 (각 언어별 실기 가이드 참고)
- <주의사항>
- 제출 파일 관련 내용 (폴더위치, 파일명, 프로그램명 등) 이 틀린 경우 및 상대경로를 사용하지 않은 경우에는 평가 시 불이익이 발생할 수 있으므로 반드시 요구되는 내용과 일치시켜 제출해야 함.

테스트 방법

※ 자가 검수를 위해 제공되는 샘플은 검수용 데이터와 다를 수 있음

검수를 위한 샘플 결과 파일은 각 문항 출력 폴더(COMPARE)에 사전 제공됨

[문항1]

- SP_TEST를 실행한 후 콘솔 입/출력 결과를 샘플 결과 파일(CMP_CONSOLE.TXT)와 동일한지 비교

C:\>SP_TEST<엔터키>	← 구현한 프로그램 실행 (Argument 없음)
Proxy-2<엔터키>	← 콘솔 입력 (Proxy Name은 'Proxy-2')
Auth Check.	← 콘솔 출력 ('Service 호출' 결과)
C:\>	

[문항2]

- SP_TEST를 실행한 후 콘솔 입/출력 결과를 샘플 결과 파일(CMP_CONSOLE.TXT)와 동일한지 비교

C:\>SP_TEST<엔터키>	← 구현한 프로그램 실행 (Argument 없음)
Proxy-2 /notice<엔터키>	← 콘솔 입력 (Proxy Name은 'Proxy-2', Path는 '/notice')
Notice Announced.	← 콘솔 출력 ('Service 호출' 결과)
C:\>	

테스트 방법
(계속)

- [문항3]
- SP_TEST.BAT를 실행한 후, MOCK.BAT를 실행

- MOCK.BAT의 콘솔에 출력되는 테스트 결과 확인
- (SP_TEST.BAT가 문항의 요건을 만족하지 못하는 경우, MOCK.BAT의 콘솔에 테스트 Fail의 사유 또는 Stack trace가 출력되므로 자가 검수에 참고 요망)

- 테스트 케이스 설명
- Client 요청 주소
- Service 수신 요청 주소
- Service 응답 정보
- 테스트 케이스 실행 결과

```
C:\>MOCK.BAT<엔터키>
...

[CASE01] 기본 Proxy <-> Service 호출, HTTP Method(GET) 전달 테스트
[CASE01][Client]<REQ> (GET) http://127.0.0.1:5001/front
[CASE01][Service A]<REQ> (GET) http://127.0.0.1:8081/front
[CASE01][Service B]<REQ> (GET) http://127.0.0.1:8082/auth
[CASE01][Service B]<RES> (Status) 200
[CASE01][Service B]<RES> (Header) Content-Type: application/json
[CASE01][Service B]<RES> (Body) {"result":"Auth ...
...
[CASE01][Client][성공]

[CASE02] 기본 Proxy <-> Service 호출, HTTP Method(POST) 전달 테스트
...
[CASE03] GET Parameter 전달 테스트
...
[CASE04] Path 전달 테스트
...

테스트에 성공했습니다!
```

테스트 방법
(계속)

[문항4]

- SP_TEST.BAT를 실행한 후, MOCK.BAT를 실행
 - MOCK.BAT의 콘솔에 출력되는 테스트 결과 확인
- (SP_TEST.BAT가 문항의 요건을 만족하지 못하는 경우, MOCK.BAT의 콘솔에 테스트 Fail의 사유 또는 Stack trace가 출력되므로 자가 검수에 참고 요망)

```
C:\>MOCK.BAT<엔터키>
...
[CASE01] 정상 Proxy <-> Service 호출 테스트
...
[CASE02] Service 추적 API 테스트
...
[CASE03] 500 오류 발생 호출 테스트
...
[CASE04] Service 추적 API 테스트
...

테스트에 성공했습니다!
```

[문항5]

- SP_TEST.BAT를 실행한 후, MOCK.BAT를 실행
 - MOCK.BAT의 콘솔에 출력되는 테스트 결과 확인
- (SP_TEST.BAT가 문항의 요건을 만족하지 못하는 경우, MOCK.BAT의 콘솔에 테스트 Fail의 사유 또는 Stack trace가 출력되므로 자가 검수에 참고 요망)

```
C:\>MOCK.BAT<엔터키>
...
[CASE01] 회로차단 테스트 - 600ms 1회
...
[CASE02] 회로차단 테스트 - 600ms 2회
...
[CASE03] 회로차단 테스트 - 600ms 3회
...
[CASE04] 회로차단 테스트 - 3회 responseTimeLimit(500ms) 초과로 회로차단
...
[CASE05] 회로차단 테스트 - healthCheck 503 응답이후 회로차단 유지 확인
...
[CASE06] 회로차단 테스트 - healthCheck 200 응답이후 정상 확인
...

테스트에 성공했습니다!
```