

간지나는 API 만들기

: 스프링부트 뉴비의 외부 API 연동 적용기

Server/cloud 이정민

목차

1. 왜 외부 API 연동이 필요했나면요
2. 제가 찾아본 스프링 부트에서 외부 API를 연동할 수 있는 n가지 방법
3. 성능 문제 로그
4. 미완의 해결책

들어가기 전에

뉴비에게 간지나는 API

== DB를 통해 단순히 CRUD 작업만을 하는 것이 아닌
그 이상의 작업을 하는 API

Chapter 1

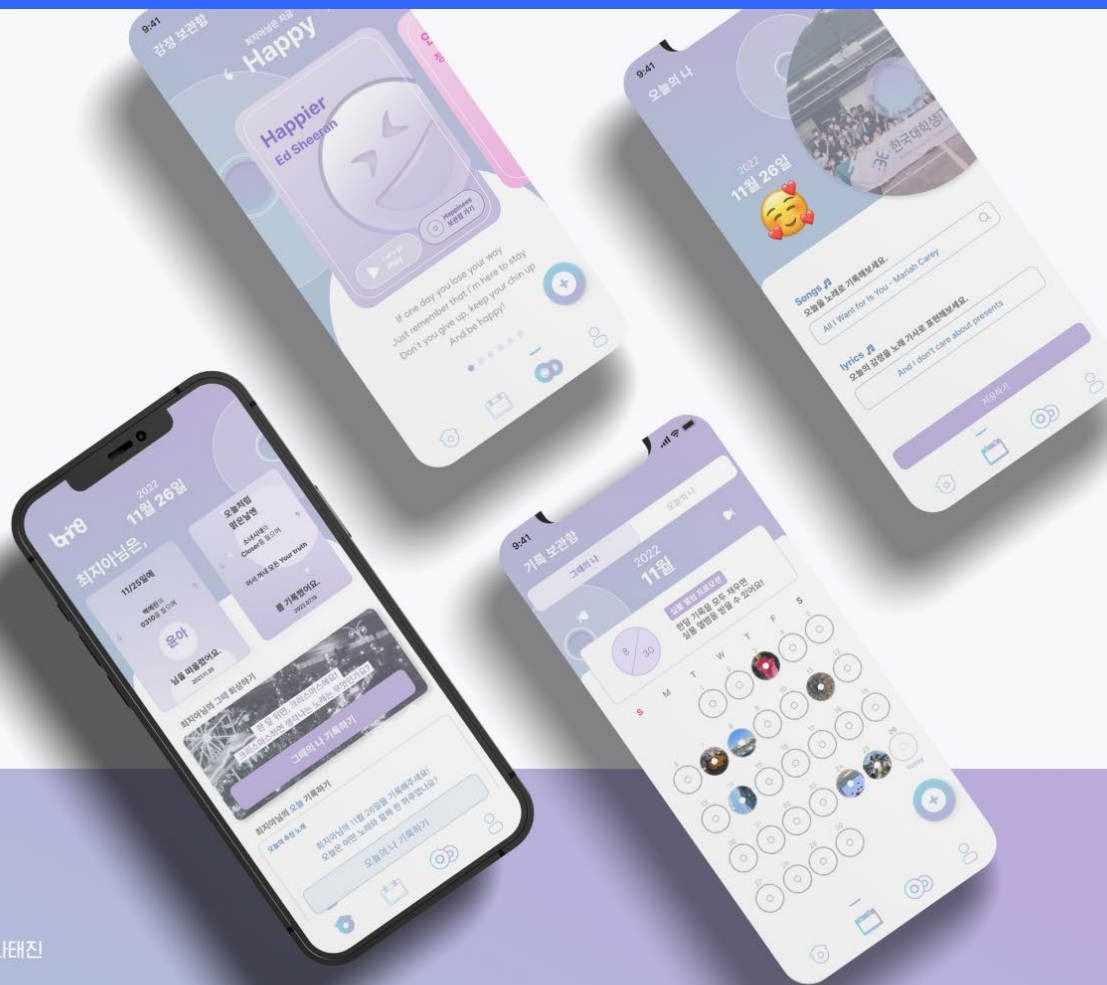
**왜 외부 API 연동이
필요했냐면요**

기억 속 음악을 통한 감정 기록 서비스

누구나 음악을 통해 자신의 감정을 쉽게 기록할 수 있는 공간이 될 수 있도록

brin9

음악이라는 매개체를 통해
사용자가 자신의 감정과 이야기를
조금 더 쉽게 기록할 수 있게 합니다.
사용자의 감정 이해를 도와줌으로써,
자신을 이해하고 이를 삶에 적용할 수 있도록 합니다.



프로젝트 기본 정보

팀	스물여섯, 스물하나
팀원 구성	오예진 윤지원 최윤아 이정민 박영민 김다형 나태진
구현 수준	iOS 개발



유저가 검색한 음악을
Spotify 서버에서 찾은
후 입력값을 정제해서

정제한 입력값을 유튜브 data
api에 파라미터로 넣어준다.

스포티파이 음악 id와 유튜브
영상 id를 모두 모아서 RDS에
넣는다!



Data API v3



Search for Item

Search for Item

API Reference [Search for Item](#)

Endpoint <https://api.spotify.com/v1/search>

HTTP Method GET

OAuth Required

q *

type *



publishedAfter

string

publishedBefore

string

q

NewJeans Attention

regionCode

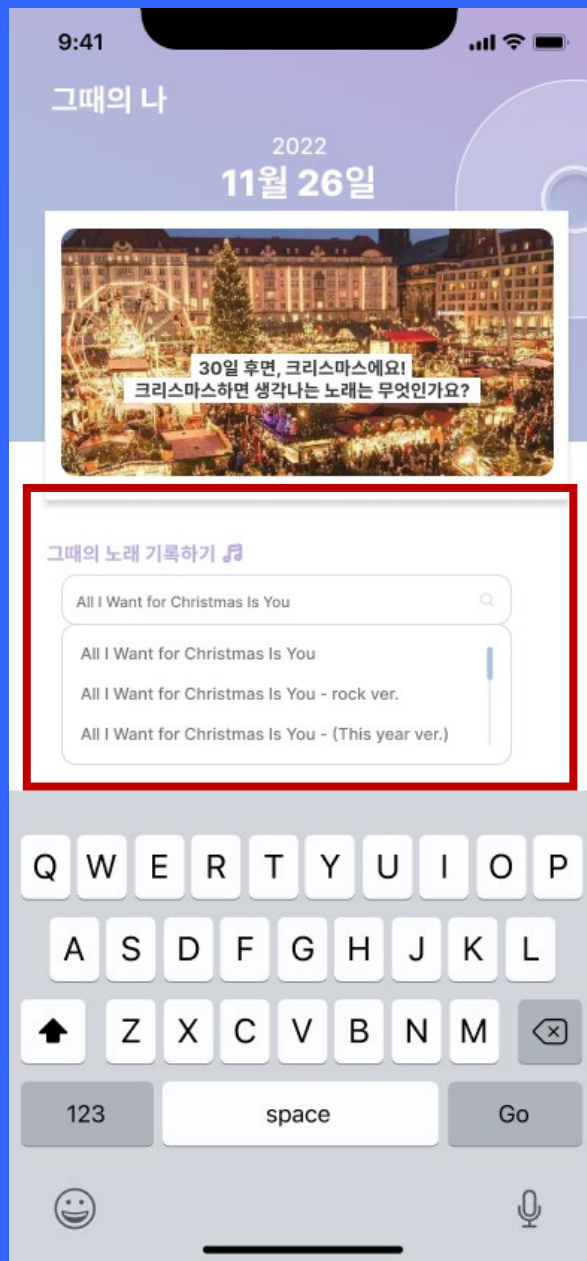
string

relatedToVideoId

200



```
},
"snippet": {
  "publishedAt": "2022-08-01T09:07:24Z",
  "channelId": "UCx0qS3cYg4FaHboblCo7nFQ",
  "title": "Attention",
  "description": "Provided to YouTube by 'ADOR' Attention · NewJeans NewJeans 1st EP 'New Jeans' Released on: 2022-08-01",
  "thumbnails": {
    "default": {
      "url": "https://i.ytimg.com/vi/kdOS94ljzzE/default.jpg",
      "width": 120,
      "height": 90
    },
    "medium": {
      "url": "https://i.ytimg.com/vi/kdOS94ljzzE/mqdefault.jpg",
      "width": 320,
      "height": 180
    },
    "high": {
```



유저가 Last Christmas를 입력했다면
입력값이 포함된 track을 스포티파이 서버에
서 조회하는 API

GET

http://localhost:8080/bring/spotify/track?track=LastChristmas

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

body

Cookies

Headers (11)

Test Results

Status: 200 OK

Time: 615 ms

Size: 781 B

Save Response

Pretty

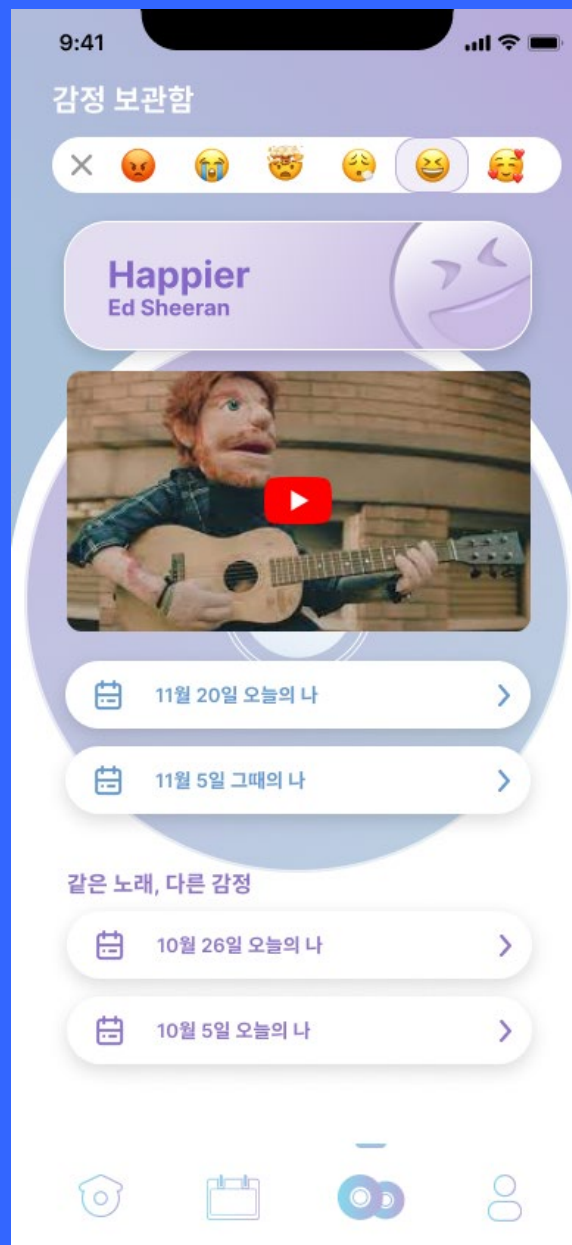
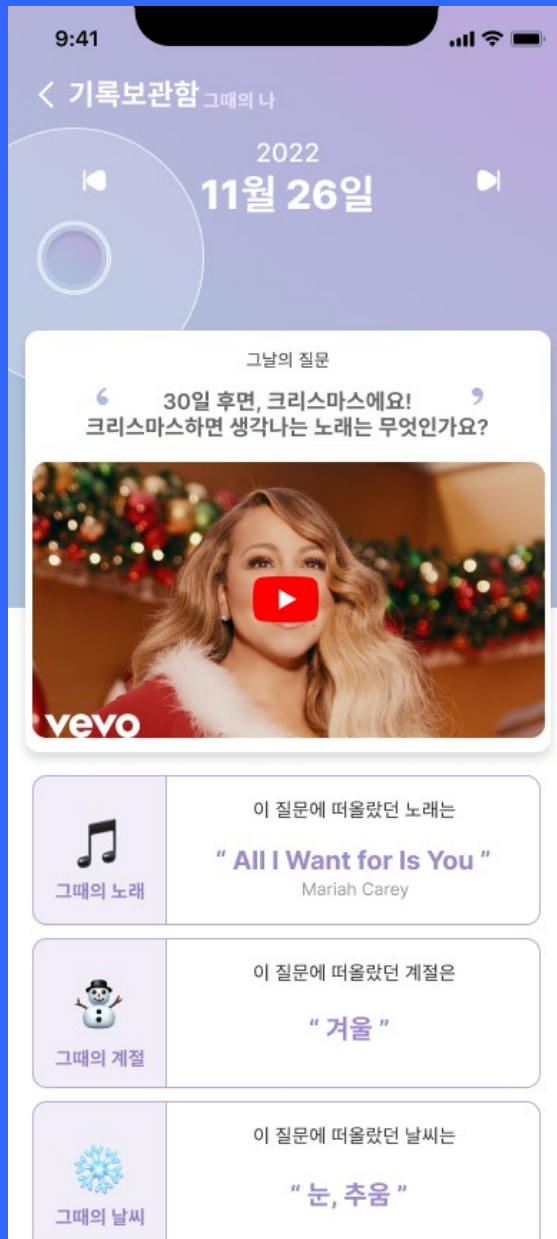
Raw

Preview

Visualize

JSON

```
4      "message": "요청에 성공하였습니다.",
5      "result": [
6        {
7          "track": "Last Christmas",
8          "trackIdx": "2FRnf9qhLbvW8fu4IBXx78",
9          "artist": "Wham!"
10       },
11       {
12         "track": "lastchristmas",
13         "trackIdx": "5aKhujPWrZjEUzkFFdTCMs",
14         "artist": "A.NAYA"
15       },
16       {
17         "track": "Last Christmas",
18         "trackIdx": "5xDx09DEDJGUQGfyoHvgDJ",
19         "artist": "Ariana Grande"
20       },
21       {
22         "track": "LastChristmas 2023",
23         "trackIdx": "1AFPHI6STqB2Jl75jQ40oG",
24         "artist": "FREEZONES"
```



음악 제목에 해당하는 유튜브 동영상을 유튜브 서버에서 검색하고 유튜브 videoldx를 DB에 저장하는 API

GET

http://localhost:8080/bring/youtube?track=IVE&artist=Love Dive&trackIdx=0Q5VnK2DYzRyfQQRJuUtvI

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

<input checked="" type="checkbox"/>	Accept	(i)	*/*	
<input checked="" type="checkbox"/>	Accept-Encoding	(i)	gzip, deflate, br	
<input checked="" type="checkbox"/>	Connection	(i)	keep-alive	
	Key		Value	Description

Body

Cookies

Headers (11)

Test Results

⌚

Status: 200 OK

Time: 1034 ms

Size: 580 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

⌵

⌵

⌵


⌵

```
1  {
2    "isSuccess": true,
3    "code": 200,
4    "message": "요청에 성공하였습니다.",
5    "result": [
6      {
7        "videoId": "Y8JFxS1H1Do",
8        "videoTitle": "IVE 아이브 &#39;LOVE DIVE&#39; MV",
9        "videoURL": "https://www.youtube.com/watch?v=Y8JFxS1H1Do"
10     }
11   ]
12 }
```

musicIdx	artist	videoIdx	track	trackIdx
36	백예린	vnX0_lHypFk	I am not your ocean anymore	4ENVMTuK1ohqMjJYi1vEE4
37	C00L	oQjzTVCIK5A	아로하	5oX05zSKGE4uFZakzvg0B0
38	meenoi	TpoPuLwJ1do	Tea time	3JISGU3qHjPaBdWSh6ZJdq
39	DEAN	wKyMIrBClYw	Instagram	6z1kLsntE7FuzKZHZWrxYN
40	aespa	H69tJmsgd9I	Dreams Come True	6rVCUwfnuYTAsX4P9fIdIu
41	TWICE	k6jqx9kZgPM	Talk that Talk	646UF6MvpT5PbnLosjfx34
42	백예린	d4eiB-nmIoo	0310	6EWWZTKKhFh7sTYsm65zfV
46	Wham!	E8gmARGvPLI	Last Christmas	2FRnf9qhLbvW8fu4IBXx78
47	임창정	97VpSN_zjFc	소주 한 잔	1oHCyIQqR7whqRrF18Lm9a
48	Crush	PS0qk05qty0	Rush Hour	5aucVLKiumD89mxVCB4zvS
49	윤하	mnpQsM-tqQU	사건의 지평선	6RBziRcDeiho3iTPdtEeg9
50	BIBI	3ks5e9weKJQ	나쁜 X	0TQoxRybRBuKktu7VKA92w
51	Newjeans	11cta61wi0g	Hype boy	0a4MMYCrzT0En247IhqZbD
52	IVE	Y8JFxS1HlDo	Love Dive	0Q5VnK2DYzRyfQQRJuUtvI
53	Love Dive	Y8JFxS1HlDo	IVE	0Q5VnK2DYzRyfQQRJuUtvI

2022
11월 25일





Songs 🎵

오늘을 노래로 기록해보세요.

예뻐어

🔍

lyrics 🎵

오늘의 감정을 노래 가사로 표현해보세요.

지금 이 말이
우리가 다시
시작하자는 건 이나
그저 너의
남아있던 기억들이
떠올랐을 뿐이야
정말 하루도 빠짐없이 (너는)
사랑한다 말했었는지
잠들기 전에 또 눈 뜨자마자 말해주던 너
생각이 나 말해보는 거야
예뻐어
날 바라봐 주던 그 눈빛
날 불러주던 그 목소리
다 -
다 -
그 모든 게 내겐
예뻐어
더 바랄게 없는듯한 느낌
오직 너만이 주던 순간들
다 -
다 -
지났지만
넌 너무 예뻐어

너도 이제는
나와의 기억이
추억이 되었을 거야
너에게는
어떤 말을 해도 다
지나간 일일 거야
정말 한번도 빠짐없이 (너는)
.....

선택하기

저장하기

유저가 지정한 노래의
가사를 조회하는 API

GET

http://localhost:8080/bring/spotify/lyrics?trackid=0Q5VnK2DYzRyfQQRJuUtv

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Body

Cookies

Headers (11)

Test Results



Status: 200 OK

Time: 1005 ms

Size: 2.06 KB

Save Response

Pretty

Raw

Preview

Visualize

JSON



```
26 "직접 들어와 두 눈으로 확인해",
27 "내 맘 가장 깊은 데로 오면 돼",
28 "(Ooh-ooh-ooh-ooh) 망설일 시간은",
29 "(Ooh-ooh-ooh-ooh) 3초면 되는 걸",
30 "(Ooh-ooh-ooh-ooh) yeah",
31 "It's so bad (so bad), it's good (it's good)",
32 "원하면 감히 뛰어들어",
33 "Narcissistic, my God, I love it",
34 "서로를 비춘 밤 (ooh)",
35 "아름다운 짜만 눈빛 더 빠져 깊이",
36 "넌 내게로, 난 네게로 (숨 참고 love dive)",
37 "(Ooh-ooh-ooh-ooh) la-la-la-la-la-la-la",
38 "(Ooh-ooh-ooh-ooh) 어서, 와서 love dive",
39 "(Ooh-ooh-ooh-ooh) oh, perfect sacrifice",
40 "Yeah, 숨 참고 love dive",
41 "숨 참고 love dive (yeah)",
42 "숨 참고 love dive (ooh)",
43 "숨 참고 love dive (yeah)",
44 "숨 참고 love dive",
45 "(Ooh-ooh-ooh-ooh) la-la-la-la-la-la-la",
46 "(Ooh-ooh-ooh-ooh) 어서, 와서 love dive",
```



Chapter 2

스프링부트에서 외부 API를 연동하는 방법

스프링부트에서 외부 API 연동하는 방법

1. HttpURLConnection, Http Client

2. Rest Template

3. WebClient

4. 외부 라이브러리들을 dependency 추가 ★

1-1. HttpURLConnection

♥ 기본적으로 GET 메서드를 사용

♥ 순수 자바로만 HTTP통신 (wOw..)

1-2. HttpClient

♥ Apache HttpComponents로 불림
(HttpComponents 프로젝트는 HttpCore와 HttpClient로 구성되어 있음!)

♥ HttpURLConnection을 사용할 때보다 조금 더 짧고, 직관적으로 코드를 짤 수 있음

스포티파이의 노래 가사를 검색해주는 API가 있었으나!

가사를 파싱해달라는
프론트의 요구로
HttpConnection으로
API를 자체적으로 만들면서
시간 ++;

```
"error": false,  
"syncType": "LINE_SYNCED",  
"lines": [  
  {  
    "startTimeMs": "3640",  
    "words": "Ooh-ooh, yeah",  
    "syllables": [  
  
    ],  
    "endTimeMs": "0"  
  },  
  {  
    "startTimeMs": "8400",  
    "words": "네가 참 궁금해, 그건 너도 마찬가지 (ooh-ooh)",  
    "syllables": [  
  
    ],  
    "endTimeMs": "0"  
  },  
  {  
    "startTimeMs": "12680",  
    "words": "이거면 충분해 쫓고 쫓는 이런 놀이",  
    "syllables": [  
  
    ],  
  },  
],
```

4 usages JjungminLee

```
public List<String> searchLyrics(String trackIdx) throws IOException, BaseException {  
    String lyricURL=String.format("https://spotify-lyric-api.herokuapp.com/?trackid=%s",trackIdx);  
    URL url = new URL(lyricURL);  
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
  
    connection.setRequestMethod(GET);  
    connection.setRequestProperty("User-Agent", USER_AGENT);  
  
    int responseCode = connection.getResponseCode();  
    System.out.println(responseCode);  
    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
    StringBuffer stringBuffer = new StringBuffer();  
    String inputLine;  
  
    while ((inputLine = bufferedReader.readLine()) != null) {  
        stringBuffer.append(inputLine);  
    }  
    bufferedReader.close();  
  
    String response = stringBuffer.toString();  
}
```

URL 인코딩

헤더설정

2. Rest Template

- ♥ 스프링에서 제공하는 http 통신에 유용하게 쓸 수 있는 템플릿
- ♥ 통신을 단순화하고 RESTful 원칙을 지킨다
- ♥ 멀티쓰레드 방식을 사용
- ♥ Blocking 방식을 사용
- ♥ 앞서 본 두 기술을 기반으로 돌아가는게 Rest Template

2. Rest Template

```
RestTemplate restTemplate = new  
RestTemplate();
```

```
String fooResourceUrl =  
"http://localhost:8080/spring-rest/foos";
```

```
ResponseEntity<String> response =  
restTemplate.getForEntity(fooResourceUrl +  
"/1", String.class);  
assertThat(response.getStatusCode(),  
equalTo(HttpStatus.OK));
```

응답 결과 ResponseEntity 객체를 반환한다. 즉 응답 헤더와 같은 더 상세한 응답 콘텐츠를 포함한다.

3. WebClient

```
//web client  
implementation 'org.springframework.boot:spring-boot-starter-webflux'
```

- ♡ 웹으로 API를 호출하기 위해 사용되는 Http Client 모듈 중 하나
- ♡ 스프링 측에서는 RestTemplate 보다 WebClient 사용을 권고함
- ♡ 싱글 스레드 방식 사용
- ♡ Non-Blocking 방식 사용
- ♡ Reactor기반
(* Reactor => 비동기/논블록을 이용해서 더 적은 자원으로 더 많은 트래픽을 처리 하기 위해 사용)

어이 잠깐 build.gradle이 뭔데?

프로젝트의 라이브러리 의존성, 플러그인, 라이브러리 저장소 등을 설정할 수 있는 빌드 스크립트 파일
리액트로 치면 package.json 같은거(?) 라고 생각하시면 될 것 같아요!

3. WebClient

```
String BASE_URL="https://api.spotify.com/v1/search";
```

 볼러올 url 설정

```
String q= URLEncoder.encode(track, "UTF-8");
```

```
String token= spotifyToken.getAccessToken();
```

```
String type="track";
```

```
DefaultUriBuilderFactory factory=new DefaultUriBuilderFactory(BASE_URL);
```

```
factory.setEncodingMode(DefaultUriBuilderFactory.EncodingMode.VALUES_ONLY);
```

```
WebClient webClient= WebClient.builder()
```

 WebClient 생성

```
.uriBuilderFactory(factory)
```

```
.baseUrl(BASE_URL)
```

```
.defaultHeader(HttpHeaders.CONTENT_TYPE,MediaType.APPLICATION_JSON_VALUE)
```

```
.defaultHeader(HttpHeaders.ACCEPT,MediaType.APPLICATION_JSON_VALUE)
```

```
.defaultHeader(HttpHeaders.AUTHORIZATION,"Bearer "+token)
```

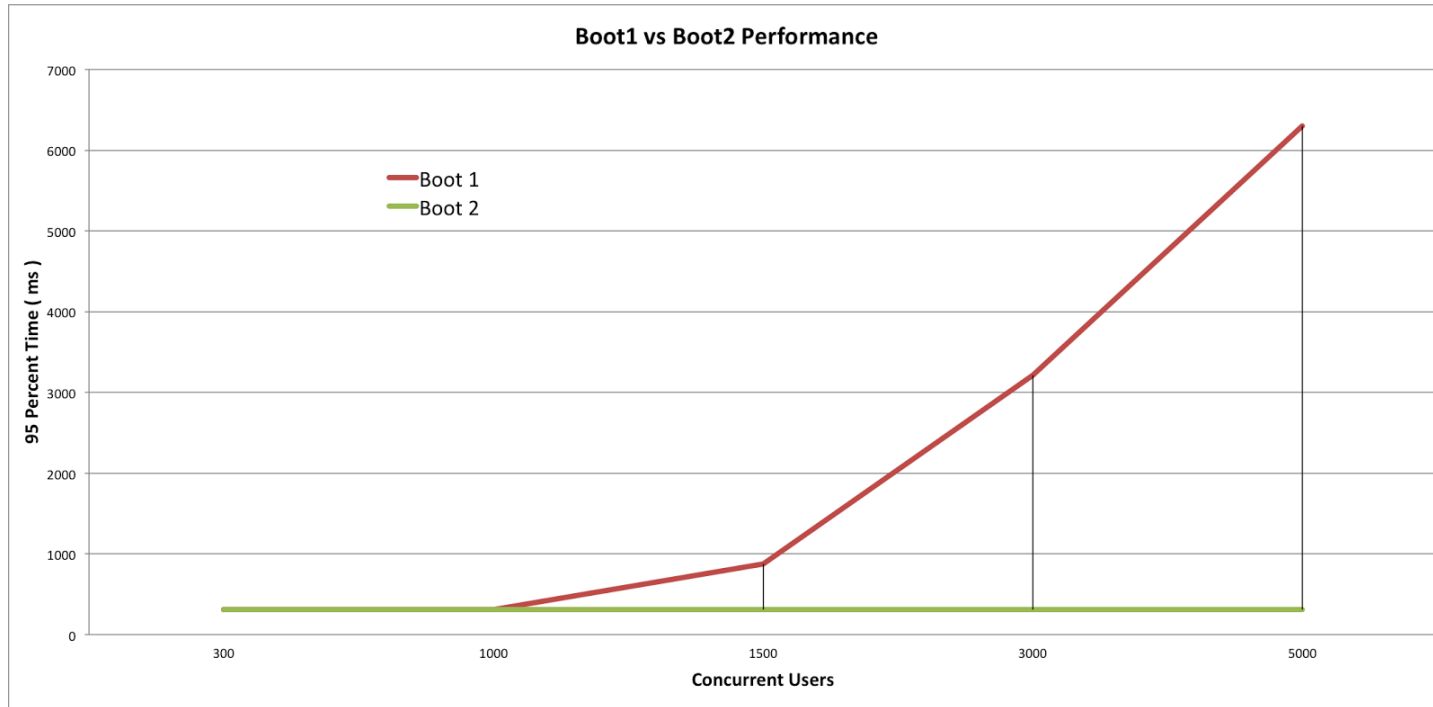
```
.build();
```

헤더 설정

3. WebClient

```
String response= webClient.get()      Get 메서드로 받아오기
    .uri(uriBuilder -> uriBuilder
        .queryParams("q",q)           스포티파이에서 하라는대로
        .queryParams("type",type)     파라미터 설정
        .build())
    .retrieve()                       retrieve\(\): body를 받아 디코딩하는 간단한 메소드
    .bodyToMono(String.class)         Mono: 0-1개의 결과만을 처리하기 위한
    .block();                         Reactor 객체
```

WebClient와 RestTemplate의 성능비교



Boot1 은 RestTemplate을 사용
Boot2 는 WebClient 를 사용

동시 사용자가 1,000명까지는 처리속도가 거의 비슷하지만 그 이후에서는 RestTemplate(Boot1) 이 급격하게 느려짐.

동시 사용자의 규모가 별로 없다면 RestTemplate을 사용하는 것은 별문제 없지만 어느정도의 규모가 있다면 WebClient를 선택하는 것이 바람직!

4. 외부 라이브러리들을 dependency 추가해서 사용



Go

- [zmb3/spotify](#)

Java/Android

- [kaaes/spotify-web-api-android](#)
- [thelinmichael/spotify-web-api-java](#)
- [SiegenthalerSolutions/spotify-web-api-android](#)

Javascript (client-side)

- [jmperez/spotify-web-api-js](#)
- [eddiemoore/angular-spotify](#) (AngularJS service)
- [xinranxiao/meteor-spotify-web-api](#) (Meteor – Isomorphic/Universal apps)
- [loverajoel/spotify-sdk](#)

Node.JS

- [thelinmichael/spotify-web-api-node](#) (universal/isomorphic)

You Tube

Data API v3

`client_secret.json`.

3. Open the `build.gradle` file in your working directory and replace its contents with the following:

```
apply plugin: 'java'
apply plugin: 'application'

mainClassName = 'ApiExample'
sourceCompatibility = 1.7
targetCompatibility = 1.7
version = '1.0'

repositories {
    mavenCentral()
}

dependencies {
    compile 'com.google.api-client:google-api-client:1.23.0'
    compile 'com.google.oauth-client:google-oauth-client-jetty:1.23.0'
    compile 'com.google.apis:google-api-services-youtube:v3-revREVISION -CL_VERSION'
}
```

```
//spotify
```

```
implementation 'se.michaelthelin.spotify:spotify-web-api-java:7.2.2'
```

```
//youtube
```

```
implementation 'com.google.api-client:google-api-client:1.23.0'
```

```
implementation 'com.google.oauth-client:google-oauth-client-jetty:1.23.0'
```

```
implementation 'com.google.apis:google-api-services-youtube:v3-rev222-1.25.0'
```


음악을 검색하는 메서드가 필요하다?!

```
Instances of the SpotifyApi class provide access to the Spotify Web API.

public class SpotifyApi {

    | The default authentication host of Spotify API calls.
    6 usages
    public static final String DEFAULT_AUTHENTICATION_HOST = "accounts.spotify.com"

    | The default authentication port of Spotify API calls.
    6 usages
    public static final int DEFAULT_AUTHENTICATION_PORT = 443;

    | The default authentication http scheme of Spotify API calls.
    6 usages
    public static final String DEFAULT_AUTHENTICATION_SCHEME = "https";

    | The default host of Spotify API calls.
```

이미 남들이 만들어 놓은 Spotify API 클래스 안에

```
1 usage
public SearchTracksRequest.Builder searchTracks(String q) {
    return new SearchTracksRequest.Builder(accessToken)
        .setDefaults(httpManager, scheme, host, port)
        .q(q);
}
```

searchTracks 메서드를 가져다 쓰면된다!

```
1 usage  JungminLee *
public static Paging<Track> searchTracks(String accessToken, String track) throws BaseException {
    SpotifyApi spotifyApi = new SpotifyApi.Builder()
        .setAccessToken(accessToken)
        .build();

    SearchTracksRequest searchTracksRequest = spotifyApi.searchTracks(track)
        .limit(5)
        .market(CountryCode.KR)
        .build();
    Paging<Track> trackPaging = null;
    try {
        trackPaging = searchTracksRequest.execute();
        return trackPaging;
    } catch (IOException | SpotifyWebApiException | org.apache.http.core5.http.ParseException e) {
        System.out.println("Error: " + e.getMessage());
    }
    return trackPaging;
}
```



```
3 usages
public class SpotifyHttpManager implements IHttpManager {

1 usage
private static final int DEFAULT_CACHE_MAX_ENTRIES = 1000;

1 usage
private static final int DEFAULT_CACHE_MAX_OBJECT_SIZE = 8192;
```



Ctrl + click!

```
import org.apache.http.core5.http.Header;
import org.apache.http.core5.http.HttpEntity;
import org.apache.http.core5.http.ParseException;
```

Apache.http.core5 라는 패키지가 나옴!

nGrinder에 적용한 HttpCore 5와 HttpClient 5 살펴보기

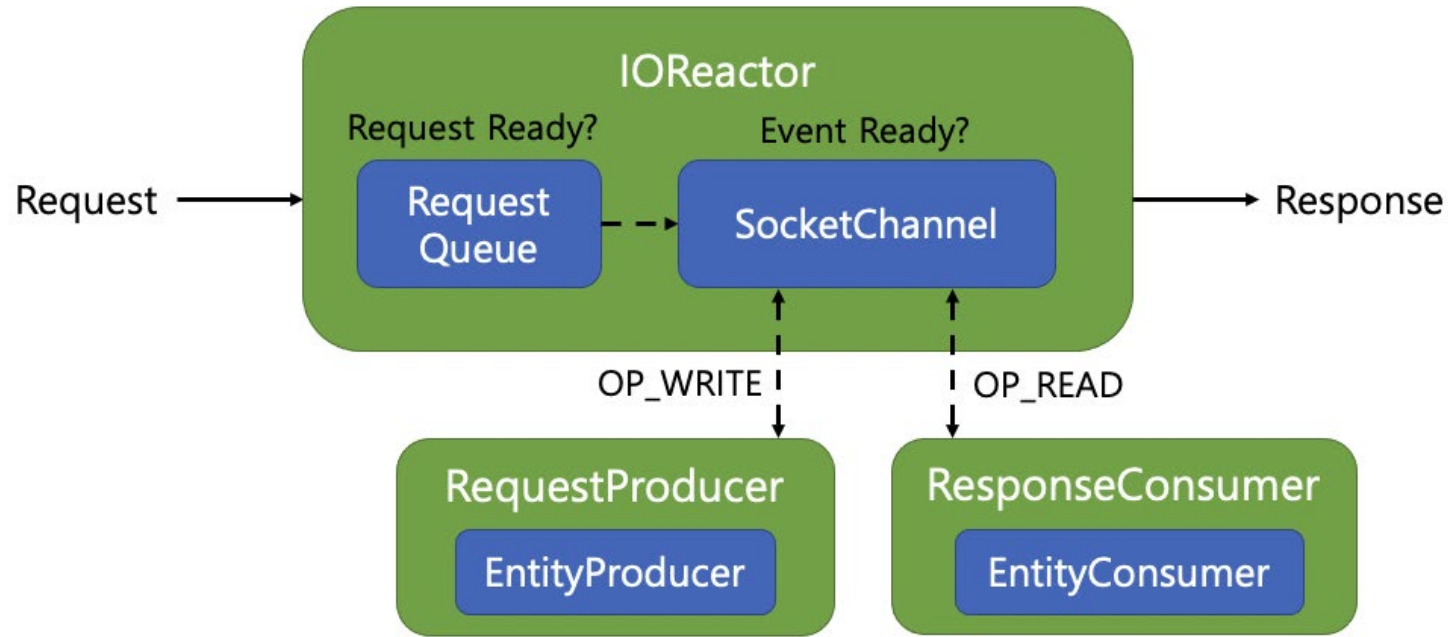
🕒 2021.07.26 | 👁 12287

<https://d2.naver.com/helloworld/0881672>

nGrinder는 네이버에서 개발해 오픈소스 소프트웨어로 공개한 부하 테스트 플랫폼입니다. 가상의 사용자를 모방해 타깃 서버에 부하를 가하는 부하 테스트 플랫폼의 특성상 상당히 많은 스레드를 사용해 HTTP 요청을 발생시킵니다. nGrinder에서 사용하던 HTTP 클라이언트에는 스레드 수가 많아질수록 부하 테스트 양의 증가가 미미해진다는 한계가 있었습니다. I/O를 처리하는 스레드가 많아짐에 따라 컨텍스트 스위칭 비용이 증가해 부하를 생성하는 작업에 컴퓨팅 리소스를 원활하게 제공하지 못하는 것이 원인이라 생각했고, 이런 문제를 해결하고자 [HttpClient 5](#)를 적용했습니다.

이 글에서는 nGrinder에 HttpClient 5를 적용하면서 분석한 [HttpCore 5](#)와 HttpClient 5의 특징과 작동 방식을 소개하겠습니다.

HTTP Core 5



1. 요청이 오면 큐에 쌓는다.
2. 요청을 처리할 준비가 되면 연결을 맺고, 해당 소켓 채널에 OP_WRITE와 OP_READ 이벤트에 대한 감지를 시작한다.
3. 데이터를 쓸 준비가 되면 producer를 통해 요청을 만들어서 전달한다.
4. 데이터를 읽을 준비가 되면 consumer를 통해 특정 형식으로 응답을 읽는다.

한줄정리 : IOReactor가 가장 중요한 컴포넌트이고 애는 논블록킹을 기반으로 동작한다.



Data API v3

```
4 usages
public final class ApacheHttpTransport extends HttpTransport {

    Apache HTTP client.
    4 usages
    private final HttpClient httpClient;

    Constructor that uses newDefaultHttpClient() for the Apache HTTP client.
    Use ApacheHttpTransport.Builder to modify HTTP client options.
```



Ctrl +
click!

```
package com.google.api.client.http;
```

Google.api.client.http 라는 패키지가 나옴!

Chapter3

성능문제

로그



Chapter4

미완의 해결책

미완의 해결책

1. 로직의 문제다
2. 슬기로운 남 탓 1 : ec2 프리티어 가 문제다
3. 슬기로운 남 탓 2 : 프론트가 알아서 파싱해서 쓰자 백엔드가 갑이다!
4. GraphQL?
5. 디자인한테 넘기자! > 로딩화면을 기깔나게 만들어서 UI,UX적으로 해결하자

미완의 해결책

1. 로직의 문제다
2. 슬기로운 남 탓 1 : ec2 프리티어 가 문제다
3. 슬기로운 남 탓 2 : 프론트가 알아서 파싱해서 쓰자 백엔드가 갑이다!
4. GraphQL?
5. 디자인한테 넘기자! > 로딩화면을 기깔나게 만들어서 UI,UX적으로 해결하자

들어주셔서 감사합니다!