드림어스컴퍼니(FLO) 백엔드(API) 과제

단계별 안내를 확인하고, 요구사항을 만족하는 프로젝트를 완성하세요.

정준용 jjundragon88@gmail.com 2월 10일 12:00 마감

STEP 1. 프로젝트 요구사항

때는 2014년 11월! FLO 서비스 기획이 끝나고 본격적으로 개발 단계에 착수해야 하는 상황이 되었습니다. 런칭 목표 일자는 2015년 1월로 모든 팀원들이 전력질주하는 단계죠. 얼마 전 팀에 합류한 당신은 자바와 스프링을 활용해 API 서버를 구성해야 하는 미션을 받았고, 아래와 같은 요구사항을 받았습니다. 내용을 잘 읽고, 실력을 발휘해 FLO 앱의 중요한 API를 완성해주세요.

서비스 초기에 필요한 API들

1. 앨범/곡 검색
2. 앨범 리스트 조회
3. 플레이리스트 생성
4. 플레이리스트에 곡/앨범 추가/삭제
5. 플레이리스트 목록 조회
6. 플레이리스트 삭제

제공 데이터

다음과 같은 데이터가 제공되며, 이 데이터를 기반으로 이후 설명하는 API를 구현하기에 적절한 테이블을 구성해야 합니다. 초기 데이터는 적은 양이 주어지지만, 앞으로 천만 건 내외의 row가 추가될 것이라고 가정하고 성능을 고려하여 테이블을 설계하는 것이 중요합니다.

[샘플 데이터] : https://gist.github.com/programmers-gitbot/8a8f20aa4bfcb6d76c58efca7d7c4c53

- 데이터에는 앨범 제목(album_title), 해당 앨범을 서비스 할 수 있는 지역(locales), 앨범에 포함된 곡 정보(songs)가 포함되어 있음
- 앨범에 포함된 곡 정보(songs)는 각 곡에 대해 제목(title), 길이(length), 트랙 번호(track)를 가지고 있음
- locales에 all이 들어있는 경우, 모든 국가에서 제공 가능한 앨범을 의미함

(제공 데이터 예시)

```
[
    {
        album_title: 'Thriller',
        locales: ['en', 'kr'],
        songs: [
            {
                title: "Wanna Be Startin' Somethin'",
                track: 1,
                length: 170,
            },
            {
                title: "Baby Be Mine",
                track: 2,
                length: 170,
            }
        ]
    },
    {
        album_title: 'The Bodyguard',
        locales: ['all'],
        songs: [
            {
                title: 'abc',
                track: 1,
                length: 170,
            }
        ]
    },
    ...
]
```

상세 요구사항

[1] Search

```
GET /search
```

문자열로 앨범/곡을 검색해서 제목이 검색어를 포함하는 앨범과 곡을 찾는 API입니다.

- 공백이나 특수문자를 포함하여 입력한 검색어를 그대로 포함하는 경우만 검색되도록 합니다.
- 예) `chicken noodle` 이라는 검색어를 입력한 경우 `chicken` 만을 포함하거나 `noodle` 만을 포함하는 경우는 표시하지 않습니다. `chicken noodle` 또는 `chicken noodle soup` 과 같이 입력한 검색어 전체를 그대로 포함하는 앨범, 노래들만 검색이 되어야 합니다.
- 사용자의 지역에 따라 검색된 앨범을 제공할 수 있는 저작권이 확보되어 있지 않을 수 있습니다. 따라서 해당 앨범, 노래를 이용할 수 없다면 해당 앨범, 노래가 검색되어서는 안 됩니다. (검색 결과에는 나오지만 재생은 되지 않는 것이 아니라, 아예 검색 결과에 노출되지 않아야 함)

Parameter

Parameter	Type	Description
title	String	(필수) 검색어
locale	String	(필수) 검색을 요청하는 사용자의 지역

Response Body

```
{
  "albums": [
     {
      "title": "chicken noodle soup",
      "id": 15,
      "songs": [
        {
          "title": "chicken noodle soup",
          "id": 74,
          "track": 1,
          "length": 450
        },
        {
          "title": "oksusu soup",
          "id": 75,
          "track": 2,
          "length": 400
        },
      ]
    },
    {
      "title": "chicken noodle donkatsu",
      "id": 2463,
      "songs": [
        {
          "title": "donkatsu",
          "id": 482,
          "track": 1,
          "length": 240
        }
      ]
    }
  ],
  "songs": [
    {
      "title": "chicken noodle soup",
      "id": 74,
      "track": 1,
      "length": 450
    },
    ...
  ]
}
```

[2] Album List

```
GET /albums
```

앨범을 10개 단위로 paging 해서 return 하는 API입니다.

- pages object에는 아래와 같은 link가 포함될 수 있습니다.
  - first: 결과의 첫 페이지 URI
  - prev: 현재 response의 이전 페이지 URI
  - next: 현재 response의 다음 페이지 URI
  - last: 결과의 마지막 페이지 URI
  - pages object에는 필요한 link들만 포함해야 합니다. 예를 들어 1page를 요청하였을 때, `first` 와 `prev` link는 포함되지 않아야 합니다.

Parameter

Parameter	Type	Description
locale	string	(필수) 검색을 요청하는 사용자의 지역
page	int	(필수) 요청할 page

Response Body

```
{
  "statusCode": 200,
  "pages": {
    "first": "https://SERVER_URL/api/albums?page=1",
    "prev": "https://SERVER_URL/api/albums?page=23",
    "last": "https://SERVER_URL/api/albums?page=25",
    "next": "https://SERVER_URL/api/albums?page=25",
  },
  "albums": [
    {
        "id": 1
        "title": "The Bodyguard",
        "songs": [
            {
                "id": 1
                "title": "abc",
                "track": 1
                "length": 170
            },
            {
                "id": 2
                "title": "efg",
                "track": 2
                "length": 240
            },
        ]
    },
    {
        "id": 2
        "title": "DJ-Kicks",
        "songs": [
            {
                "id": 3
                "title": "starry night (Edit)",
                "track": 1
                "length": 220
            },
        ]
    },
    ...
  ]
}
```

[3] Playlist 관련 API

아래 API들은 직접 Endpoint와 Parameter, Response body를 작성해야 합니다. 그리고 API 스펙 문서도 추가해 주세요(파일명: api_doc.md). 이 때, [1] Search, [2] Album list 요구사항에서 제시한 포맷이나 API의 형식을 그대로 따를 필요는 없습니다. 내가 생각하는 더 좋은 설계, 표현 방법이 있다면 자유롭게 작성해 주세요. 예외 상황들의 경우 적절한 처리와 함께 문서에 명시해 주세요.

- user를 저장하는 별도의 table은 없으며, user_id는 임의의 숫자를 사용합니다.
- request에는 요청을 보내는 사용자의 id (user_id)가 포함됩니다.
- request에 포함된 user_id는 모두 존재한다고 가정합니다.

아래 제시된 4개의 Endpoint를 구현해 주세요.

[3.1] Playlist 생성 API

- 해당 사용자의 playlist를 생성할 수 있습니다.
- 한 사용자는 여러 개의 playlist를 가질 수 있습니다.
- playlist 이름을 지정할 수 있습니다.

[3.2] Playlist 노래, 앨범 추가 API

- 노래를 playlist에 추가할 수 있습니다.

- 특정 앨범에 포함된 모든 노래를 playlist에 추가할 수 있습니다.

[3.3] Playlist 목록 API

- 특정 사용자의 playlist 목록을 조회할 수 있습니다.
- pagination은 하지 않습니다.

[3.4] Playlist 삭제 API

- 사용자의 특정 playlist 1개를 삭제합니다.

STEP 2. 개발 환경 설정 다운로드

테스트를 시작하기 전에 개발 환경 설정을 다운로드 받으세요.

Spring Boot
ZIP file format

STEP 3. 프로젝트 제출

제출할 프로젝트의 Github 저장소를 등록하세요.

- 저장소는 `username/repos` 형식으로 입력하시면 됩니다.
- 작업물이 `master branch` 에 있어야 합니다.
- 저장소는 비공개(Private) 상태여야하며, programmers-gitbot 을 `collaborator` 로 추가해 주세요.
- 저장소를 입력하고 아래의 [빌드] 버튼을 누르면 프로그래머스의 서버에서 코드를 빌드하여 실행합니다.
- [빌드]가 완료되면 버튼이 [실행하기]로 바뀌며, 이 버튼을 통해 프로젝트가 서버에서 잘 동작하는지 확인할 수 있습니다.

Github 저장소 입력

username/repos 빌드(Build)

빌드를 다시 하고 싶다면 우측의 [초기화] 버튼을 눌러주세요.

(선택) 프로젝트 빌드 대신 접속 할 수 있는 URL을 직접 입력

https://example.com

위의 [빌드]기능을 이용하지 않고 스스로 마련한 서버(Heroku, AWS 등)에 배포하여 제출하고 싶은 경우, 접속할 수 있는 URL을 입력하면 됩니다.

채점은 수동으로 진행되므로 채점자가 접속 시 URL에 접속되지 않으면 0점 처리 될 수 있습니다.

최종 제출 및 테스트 종료

STEP 3의 프로젝트 제출이 완료되면 최종 제출 버튼이 활성화 됩니다. 최종 제출 후 테스트가 종료되면 다시 시작할 수 없으며, 이후 업데이트한 커밋은 반영되지 않습니다.