Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

API 서버와 REST

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

API 서버란?

앱/웹 서비스를 만드는 **개발자**들이 이용하는 데이터 위주의 서비스

시간이 지나도 호환성을 유지해야 합니다.

- 앱 등의 유저가 사용하는 UI는 유저가 원할 때 업데이트가 됩니다.
- 유저층이 사용하는 앱의 버전이 다양하기에, API에도 버전 개념을 둡니다. ex) /api/v1/posts/, /api/v2/posts/
- 그에 반해, 웹 서비스를 이용하는 유저는 항상 최신버전을 사용합니다.

REST (Representational State Transfer)

아키텍처 스타일. 프로토콜에 독립적 → 일반적인 REST 구현에서 HTTP를 사용

RESTful API의 몇 가지 디자인 원칙

1. 리소스를 중심으로 디자인.
2. 클라이언트에서 액세스할 수 있는 모든 종류의 개체/서비스가 리소스에 포함
3. 리소스마다 해당 리소스를 고유하게 식별하는 식별자 → https://my-trips.com/trips/1/
4. 요청/응답 포맷으로 흔히 JSON을 사용 →
5. 균일한(uniform) 인터페이스를 적용

   리소스에 표준 HTTP 동사 (GET, POST, PUT, PATCH, DELETE)를 적용

```
{
    "pk": 1,
    "name": "프라하 여행",
    "user_id": 100,
    "place_set": [100, 102, 200]
}
```

[Azure 아키텍처] Web API 디자인 : https://docs.microsoft.com/ko-kr/azure/architecture/best-practices/api-design

리소스를 중심으로 API 구성

/orders/ 로의 POST 요청

~~/create-order/ 로의 POST 요청~~

/customers/ → 고객 컬렉션

/customers/5/ → pk가 5인 고객 ➔ 직관적인 웹 API

/customers/5/orders/ → 고객 5에 대한 모든 주문

/orders/99/customer/ → 주문 99의 고객

심플하게 URI을 구성하기

/customers/1/orders/99/products/ → 유연성이 떨어짐.

/customers/1/orders/ 를 통해 고객1의 모든 주문을 찾은 후에,

/orders/99/products/ 로 변경해서 동일한 처리

HTTP 메서드를 기준으로 기본 작업 정의

GET : 리소스의 표현. 응답 본문에 리소스의 세부 정보

POST : 새 리소스 생성 요청. 요청 본문에 새 리소스의 세부 정보를 제공. → 멱등성(idempotent) 미보장

PUT : 기존 리소스를 대체. 요청 본문에 갱신할 리소스 정보를 제공. → 반드시 멱등성 보장되어야.

PATCH : 기존 리소스를 부분 대체. 요청 본문에 갱신할 리소스 정보를 제공. → 멱등성 미보장.

DELETE : 지정 리소스를 제거.

리소스	POST	GET	PUT	DELETE
/posts	새 포스팅 만들기	모든 포스팅 목록	포스팅 대량 업데이트	모든 포스팅 삭제
/posts/1/	오류	포스팅 1에 대한 내용	포스팅 1의 정보 갱신	포스팅 1 삭제
/posts/1/comments/	포스팅 1의 새 댓글 만들기	포스팅 1에 대한 모든 댓글 목록	포스팅 1의 댓글 대량 업데이트	포스팅 1의 모든 댓글 삭제

요청/응답 형식 지정

요청 : Content-Type 헤더

ex) application/json, application/vnd.ms-excel, image/jpeg, application/pdf 등

요청 시에 처리를 원하는 형식 지정하면, 서버에서는 이 형식으로 응답

서버에서 해당 형식을 지원하지 않으면 HTTP 상태 코드 415 (지원하지 않는 미디어 유형) 반환

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types

HTTP METHOD별 다양한 상태 코드

GET

일반적으로 200 (OK) 응답, 리소스를 못 찾을 경우 404 (Not Found)응답

POST

201 (Created) 응답. 새 리소스의 URI는 응답의 Location 헤더에.

새 리소스를 만들지 않은 경우, 200 응답하고 응답 본문에 포함할 수도. 반환할 결과가 없으면 204 (내용없음) 반환할 수도.

잘못된 데이터로 요청하면 400 (잘못된 요청) 응답하고 응답 본문에 오류정보 또는 자세한 정보를 제공하는 URI 링크 포함.

PUT

기존 리소스를 업데이트할 경우 200 (OK) 또는 204 (내용없음) 반환. 상황에 따라 업데이트할 수 없는 경우 409 (충돌) 반환.

DELETE

성공하면, 응답 본문에 추가정보가 포함되지 않았음의 의미로 204 응답. 리소스가 없는 경우 404 응답.

비동기 작업

작업 완료에 시간이 오래 걸릴 경우, 다른 Task Queue를 통해 비동기 처리를 할 수 있습니다. (예: Celery 등). 이때 요청은 수락되었지만 아직 완료되지 않았음을 나타내는 202 (수락됨) 응답

클라이언트가 이 작업을 Polling을 통해 모니터링할 수 있도록, 비동기 요청의 상태를 반환하는 URI을 Location 헤더로 반환

보통의 API는 HTTP API라고 불러야 하지 않을까요?

REST API라고 부르는 것들은 단순히 HTTP프로토콜을 통한 API이기에,
HTTP API라고 불러야 적절할 듯 싶습니다. 혹은 **Web API**.

대부분의 REST API 라는 API들은 REST 아키텍처 스타일 X

django-rest-framework

설계의 영역에 대해서는 다루지 않습니다.

장고 API에서 널리 쓰여지는 DRF에 대해서 자세히 살펴보는 시간

장고의 패러다임 하에 빠르고 관리하기 쉬운 API를 만들 수 있습니다.

DRF는 아래 REST API 컨셉을 쉽게 만들 수 있도록 도와줍니다.

이것이 REST API의 전부는 아닙니다.

- URI는 `https://{serviceRoot}/{collection}/{id}` 형식이어야 한다.
- GET, PUT, DELETE, POST, HEAD, PATCH, OPTIONS를 지원해야한다.
- API 버저닝은 Major.minor로 하고, URI에 버전정보를 포함시킨다.

django-rest-framework의 주요 기능들

Serializer/ModelSerializer를 통한 데이터 유효성 검증 및 데이터 직렬화

각종 Parser를 통한 데이터 처리

APIView/Generic/ViewSet/ModelViewSets를 통한 요청 처리

각종 Renderer를 통한 다양한 응답 포맷 지원

인증(Authentication)/권한(Permission)체계 - 써드파티를 통해 JWT 지원

Throttling (최대 호출 횟수 제한)

CRUD

위키백과 : https://ko.wikipedia.org/wiki/CRUD

모든 데이터는 기본적으로 "추가/조회/수정/삭제" 액션으로 관리될 수 있습니다.

- C : Create (생성) : 새 레코드 생성
- R : Read, Retrieve (조회) : 레코드 목록 조회, 특정 레코드 조회
- U : Update (수정) : 특정 레코드 수정
- D : Delete(삭제) : 특정 레코드 삭제

주의) CRUD는 리소스에 대한 대표적인 동작일 뿐, API의 전부는 아닙니다.

프레임워크를 쓴다는 것은 ...

DRF에 대해 보다 심도있는 이해를 하기 위해서는 장고의 Model/Form에 대한 이해가 필요합니다.

무작정 타이핑만 치는 것만이 개발이 아닙니다.

탄탄한 이해와 설계 아래 탄탄한 애플리케이션이 갖춰집니다.

프레임워크를 쓴다는 것은, 그 프레임워크가 제시하는 길을
명확히 이해하고 존중하는 것에서 시작합니다.

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company