REST API

진짜 실무에서는 어떻게 개발할까?

상명대학교 멋쟁이 사자처럼 🦁 김현빈 사자 🦁

목표

- API
- Public API, Private API
- API 사용 과정
- DRF
- REST API 실습

API

Application Programming Interface

- 컴퓨터 프로그램 사이의 상호 작용을 하기 위한 인터페이스 사양을 의미
- API는 사용자가 아닌 프로그램을 위한 기능
- API의 종류
  - Public API
  - Private API

Public API

Public API를 사용하는 이유

- 직접 개발할 수 없거나 필요한 데이터를 사용하기 위해서

지도

윈도우

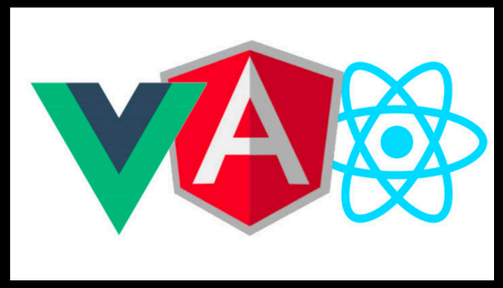
공공데이터

Private API

Private API를 사용하는 이유

- 모바일 앱, AJAX 등 기능 수행이나 데이터가 필요한 경우

좋아요 기능

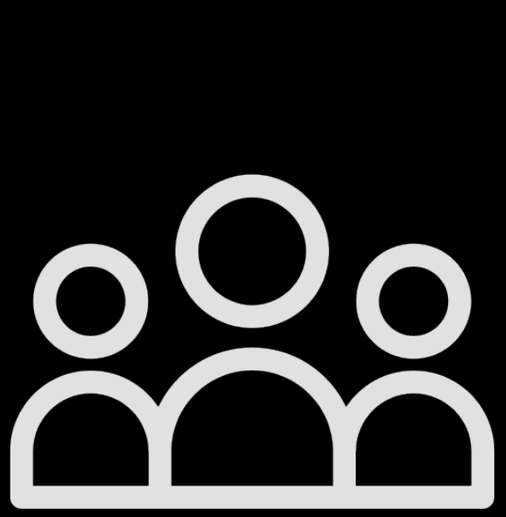


SPA 방식 웹앱

모바일 앱

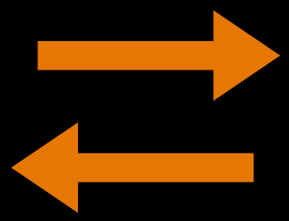
API 동작 과정

음식점 주문 과정



고객

주문



음식

요리사

API 동작 과정

음식점 주문 과정

API 동작 과정

API 사용 과정

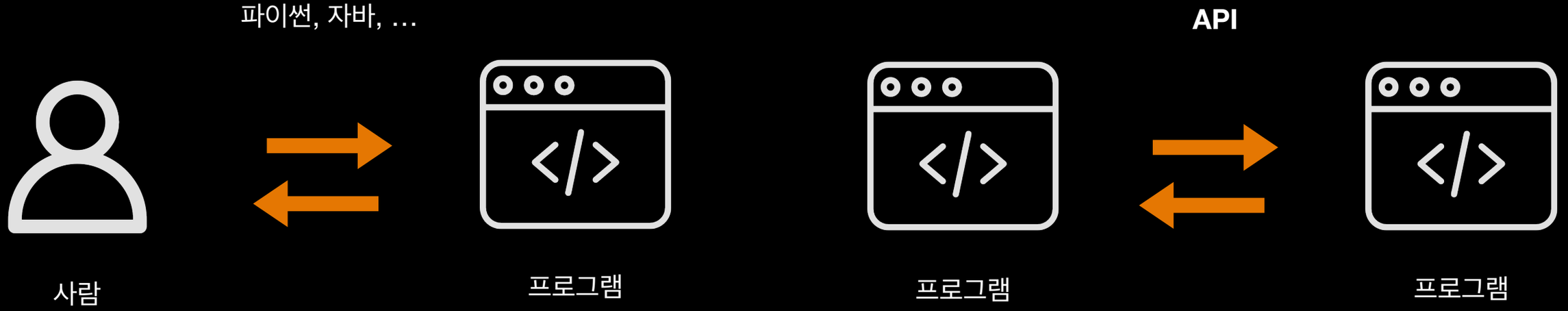
DRF

Django REST Framework

- **DRF = Django + API**
- DRF는 장고를 기반으로 웹 API를 구축할 수 있도록 기능들을 만들어 놓은 툴킷

API

API는 프로그램 간의 소통



API

API의 개발 기준

개발자 간의 협력과 유지보수 및 확장에 용이하도록

REST API

REpresentational State Transfer

- HTTP 통신에서 어떤 자원에 대한 CRUD 요청을 Resource(자원)와 Method(행위)로 표현하여 전달하는 방식
- REST 설계 규칙을 잘 지켜서 설계된 API를 ‘RESTful하다’고 표현

REST API

REST API 특징

특징	역할
균일한 인터페이스	아키텍처를 단순화하고 상호 작용의 가시성을 향상시킴
무상태	클라이언트에서 서버로의 각 요청에 요청을 이해하고 완료하는데 필요한 모든 정보가 포함되어야 함
계층화	- 서버는 다중계층으로 구성될 수 있으며 보안, 로드밸런싱, 암호화 등을 위한 계층을 추가하여 구조를 변경할 수 있음 - Proxy, Gateway와 같은 네트워크 기반의 중간매체를 사용할 수 있게 해주며 클라이언트는 서버와 직접 통신하는지, 중간 서버와 통신하는지 알 수 없음
캐시 처리 기능	클라이언트는 응답을 캐싱 할 수 있어야 함
클라이언트/서버 구조	아키텍처를 단순화시키고 작은 단위로 분리함으로써 클라이언트-서버의 각 파트가 독립적으로 구분하고 서로 간의 의존성을 줄임

REST API

REST API 설계 규칙

1. URI를 명사로 사용
2. 슬래시(/)로 계측 관계를 표현
3. 밑줄(_)을 사용하지 않고, 하이픈(-)을 사용
4. URI는 소문자로만 구성
5. HTTP 응답 상태 코드 사용
6. 파일확장자는 URI에 포함하지 않음

REST API

HTTP Method의 역할

HTTP Method	역할
POST	POST를 통해 해당 URI를 요청하면 리소스를 생성
GET	GET을 통해 해당 리소스를 조회
PUT	PUT을 통해 해당 리소스를 수정
PATCH	PATCH를 통해 해당 리소스를 부분 수정
DELETE	DELETE를 통해 해당 리소스를 삭제

REST API

RESTful API 예시

URI	HTTP Method	CRUD	비고
{Resource}/ 예) posts/	POST	Create	생성
{Resource}/ 예) posts/	GET	Read	조회
{Resource}/{identity}/ 예) posts/3/	GET	Read	조회
{Resource}/{identity}/ 예) posts/3/	PUT	Update	전체 수정
{Resource}/{identity}/ 예) posts/3/	PATCH	Update	부분 수정
{Resource}/{identity}/ 예) posts/3/	DELETE	Delete	삭제

URI, URL

URI(Uniform Resouce Identifier)

- 특정 리소스를 식별하는 **통합 자원 식별자**를 의미
- 인터넷에 있는 자원을 나타내는 유일한 주소

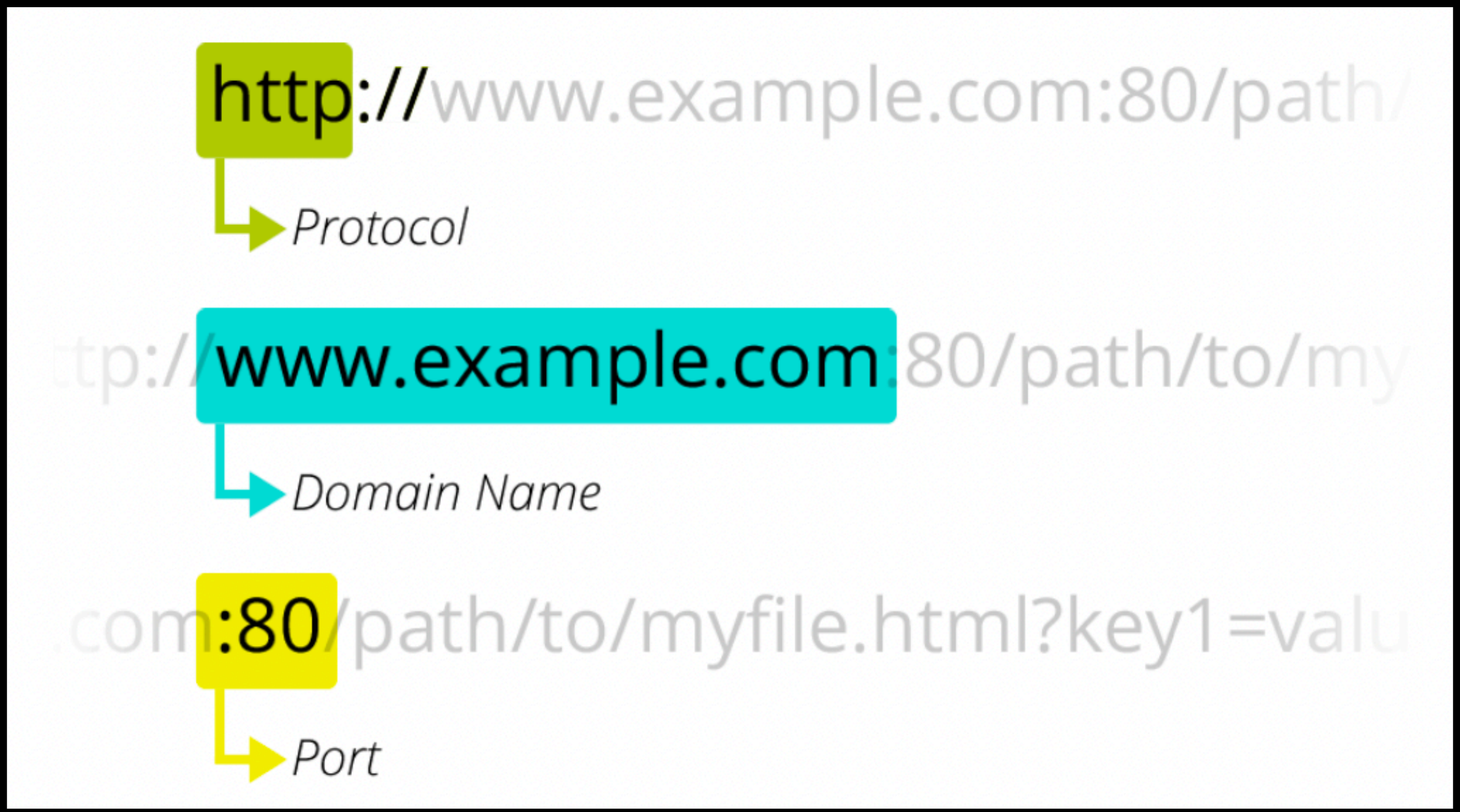
URI, URL

URL(Uniform Resouce Locator)

- 흔히 웹 주소라고 불림
- 컴퓨터 네트워크 상에서 리소스가 어디 있는지 알려주기 위한 규약

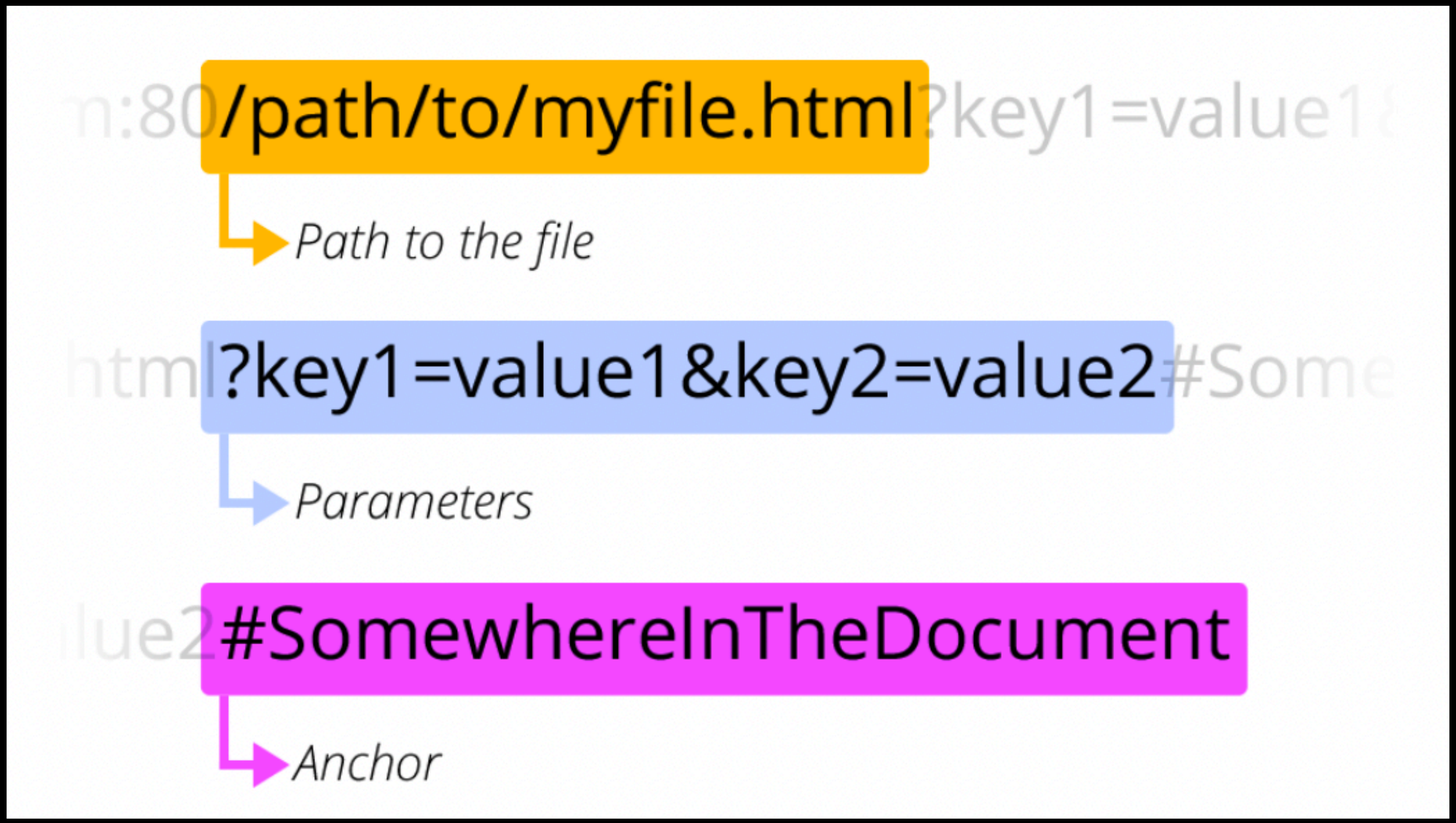
URI, URL

URI의 명칭



URI, URL

URI의 명칭



URI, URL

URI, URL 구분

구분	URI	URL
http://127.0.0.1:8000	O	O
http://127.0.0.1:8000/posts/	O	O
http://127.0.0.1:8000/posts/3/	O	X
http://127.0.0.1:8000/posts/?id=3	O	X
http://127.0.0.1:8000/static/image/	O	O
http://127.0.0.1:8000/static/image/logo.png	O	X

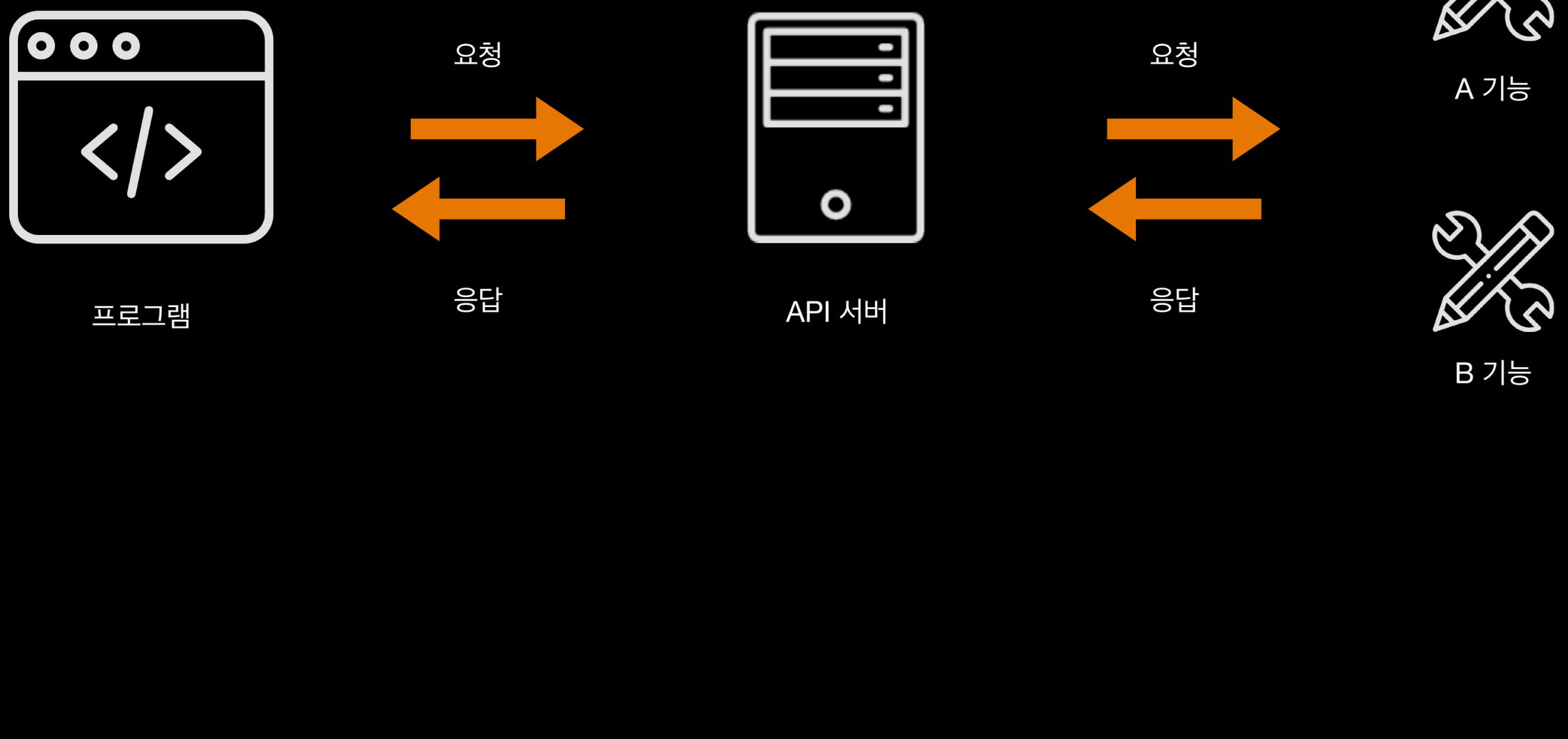
복습 | REST API

RESTful API 예시

URI	HTTP Method	CRUD	비고
{Resource}/ 예) posts/	POST	Create	생성
{Resource}/ 예) posts/	GET	Read	조회
{Resource}/{identity}/ 예) posts/3/	GET	Read	조회
{Resource}/{identity}/ 예) posts/3/	PUT	Update	전체 수정
{Resource}/{identity}/ 예) posts/3/	PATCH	Update	부분 수정
{Resource}/{identity}/ 예) posts/3/	DELETE	Delete	삭제

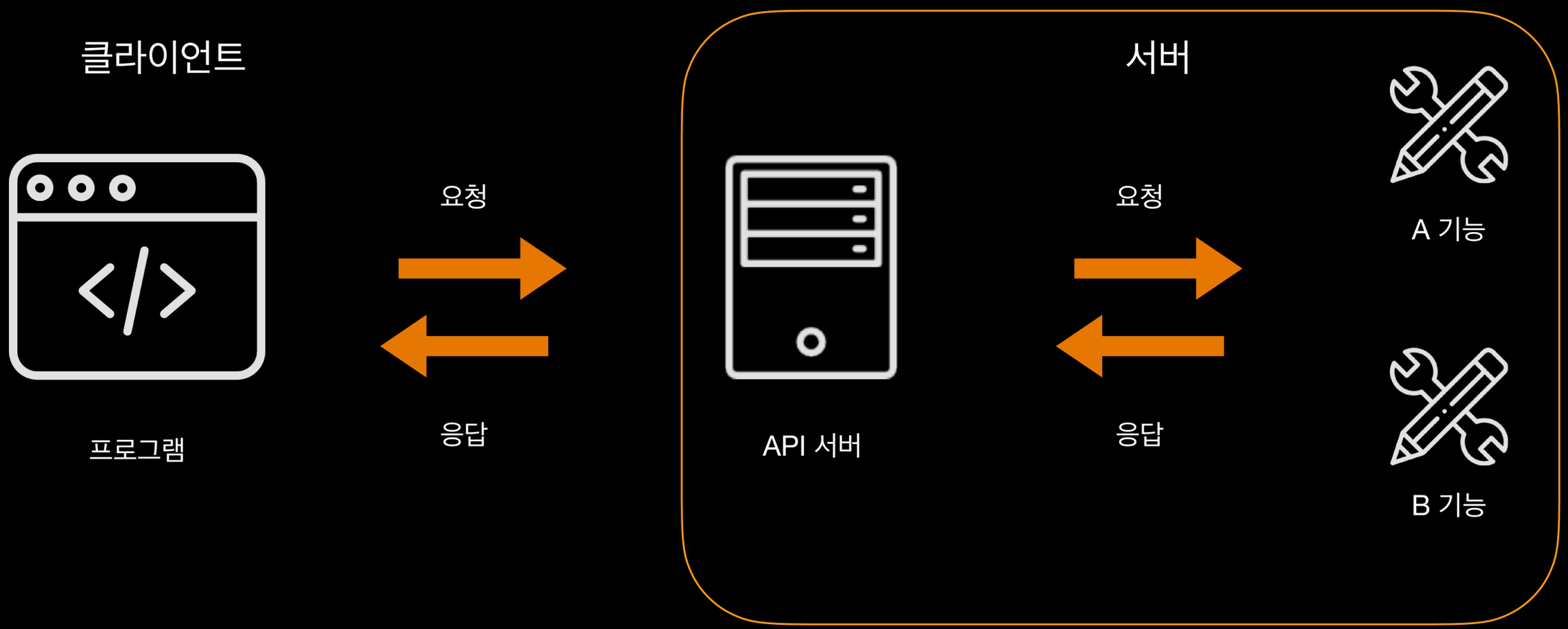
복습 | API

API 사용 과정



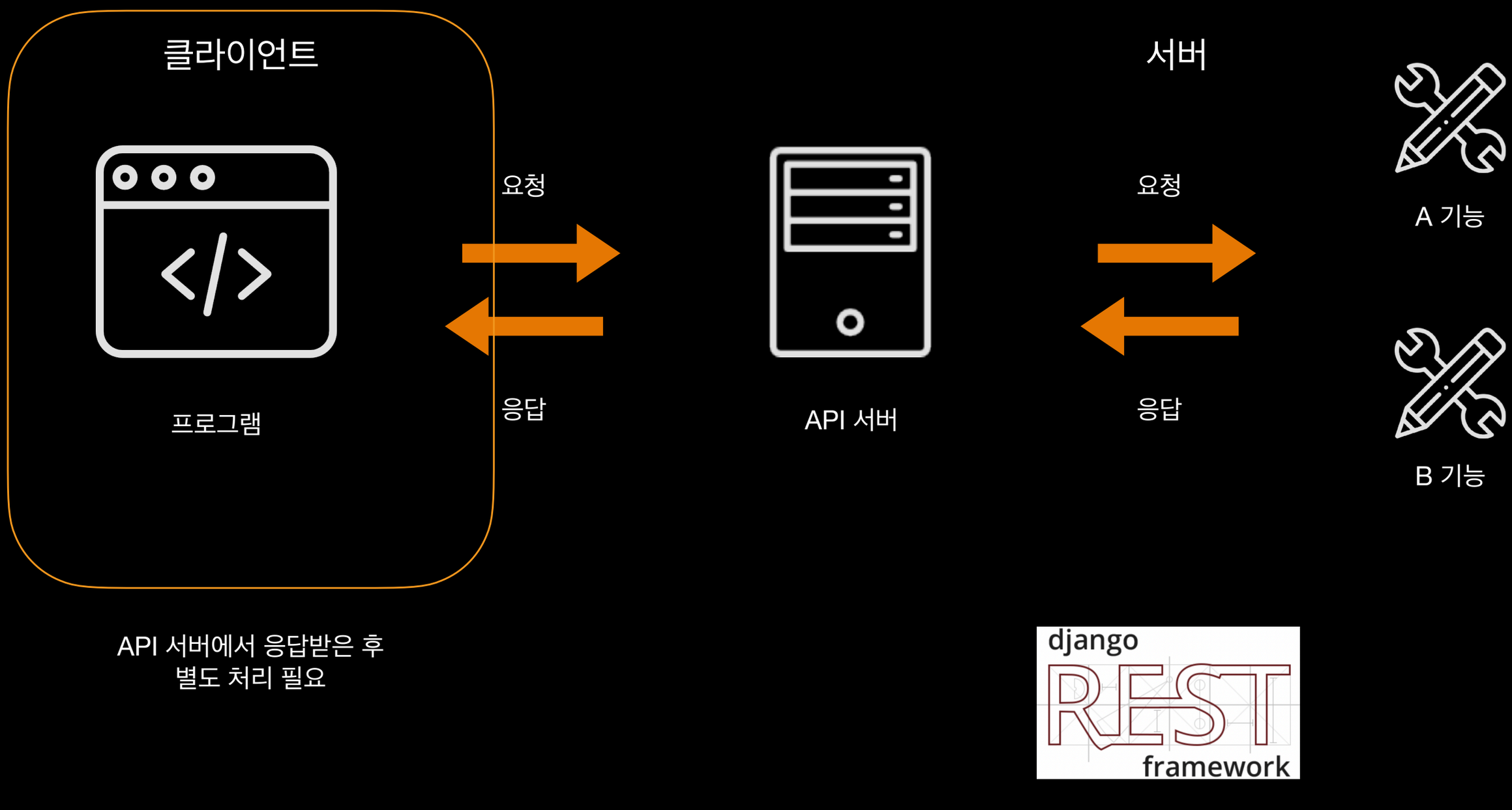
DRF

Django REST Framework



DRF

Django REST Framework



DRF

Django REST Framework

- DRF는 장고를 기반으로 웹 API를 구축할 수 있도록 기능 등을 만들어 놓은 툴킷
- RESTful 한 API 형태의 기능을 제공

https://www.django-rest-framework.org/

DRF

DRF의 장점

- API 개발을 쉽게 만들어 줌
- 인증 정책 OAuth1, OAuth2 사용 가능
- **Serializer(직렬화) 기능을 제공** (Model > JSON, JSON > Model)
- 문서화, 커뮤니티 지원

https://www.django-rest-framework.org/

DRF

DRF의 역할

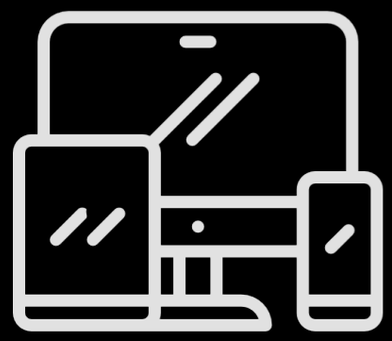
- 클라이언트에게 받은 요청을 처리, 응답

https://www.django-rest-framework.org/

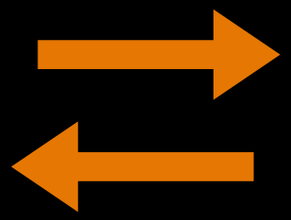
DRF

DRF의 역할

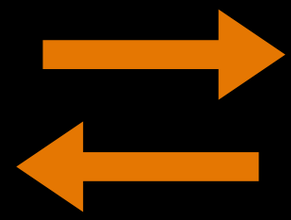
GET http://127.0.0.1:8000/posts/

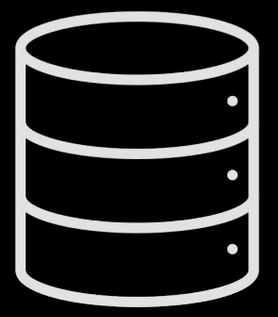


HTTP 요청



HTTP 응답





클라이언트

API 서버

DB 서버

```
HTTP/1.1 200 OK
{
        “id”:”1”,
        “title”: “첫 번째 게시글!”,
        “content”: “글쓰기”,
        “created_at”: “2022-03-01”
}
```

https://www.django-rest-framework.org/

DRF

Django REST Framework 핵심 요소

- 요청 / 응답
- View와 ViewSet
- 라우터
- 직렬화
- 인증 / 권한
- 페이징 / 필터

https://www.django-rest-framework.org/

감사합니다

이제 실습 같이 하러 가실까요? 🤓