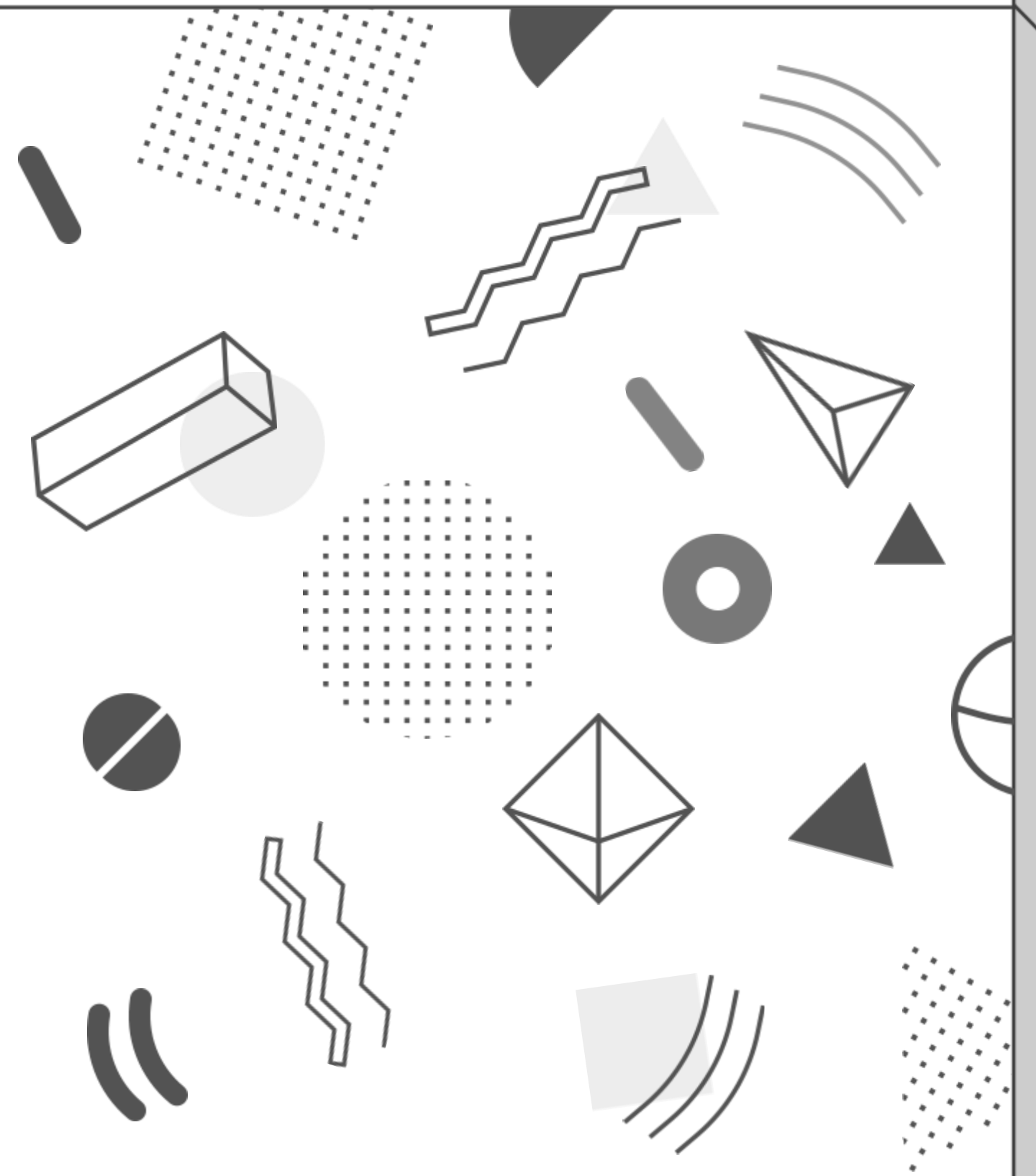


도대체 REST API가 뭐야?

SSAFY 7기 서울 17반 김희영



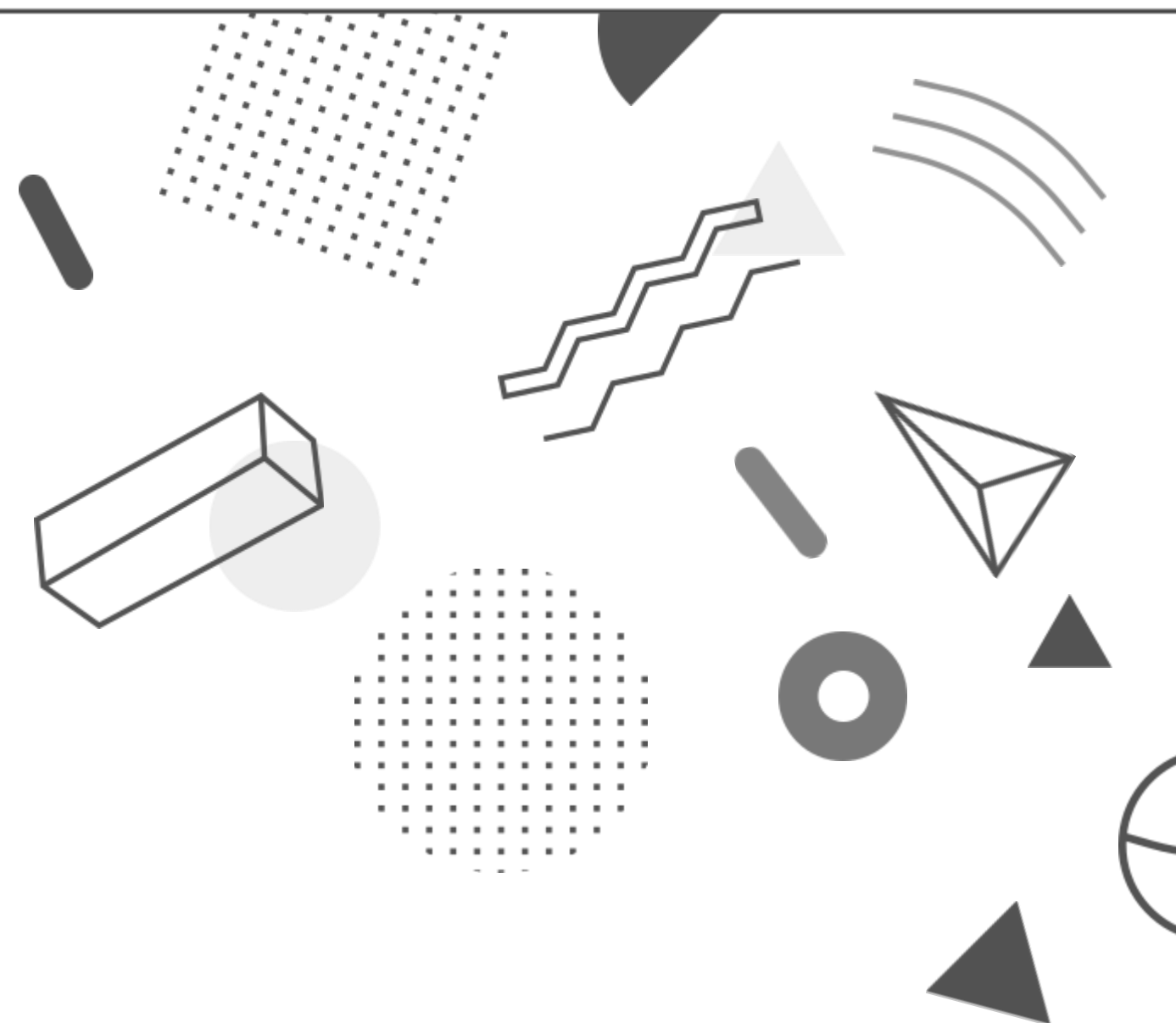
1. REST의 탄생
2. REST API의 구성 요소
3. REST API의 예시
4. REST 아키텍처의 6가지 제약 조건
5. Uniform Interface
6. 마무리



01

REST의 탄생

REST는 언제, 왜 만들어졌는가





로이 필딩 (Roy Fielding)



컴퓨터 과학자

영어에서 번역됨 - Roy Thomas Fielding은 미국 컴퓨터 과학자이며 HTTP 사양의 주요 저자 중 하나이며 Representational State Transfer 아키텍처 스타일의 창시자입니다. 위키백과(영어)

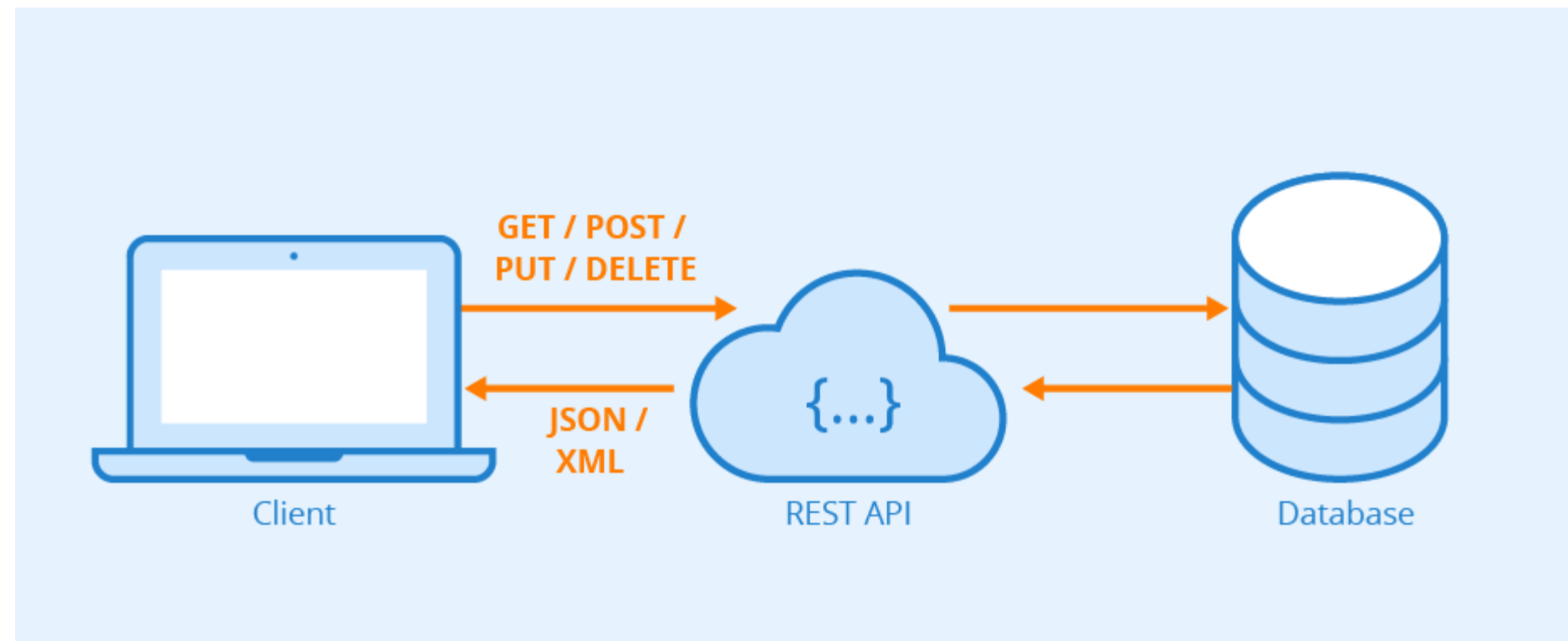
REST : 웹 그리고 HTTP의 장점을 최대한 활용할 수 있는 아키텍처

REST(Representational State Transfer)는 월드 와이드 웹과 같은 분산 하이퍼미디어 시스템을 위한 소프트웨어 아키텍처의 한 형식이다. 이 용어는 로이 필딩(Roy Fielding)의 2000년 박사학위 논문에서 소개되었다. 필딩은 **HTTP**의 주요 저자 중 한 사람이다. 이 개념은 네트워킹 문화에 널리 퍼졌다.

{ REST } REpresentational S tate T ransfer

(자원의) 표현에 의한 상태의 전달

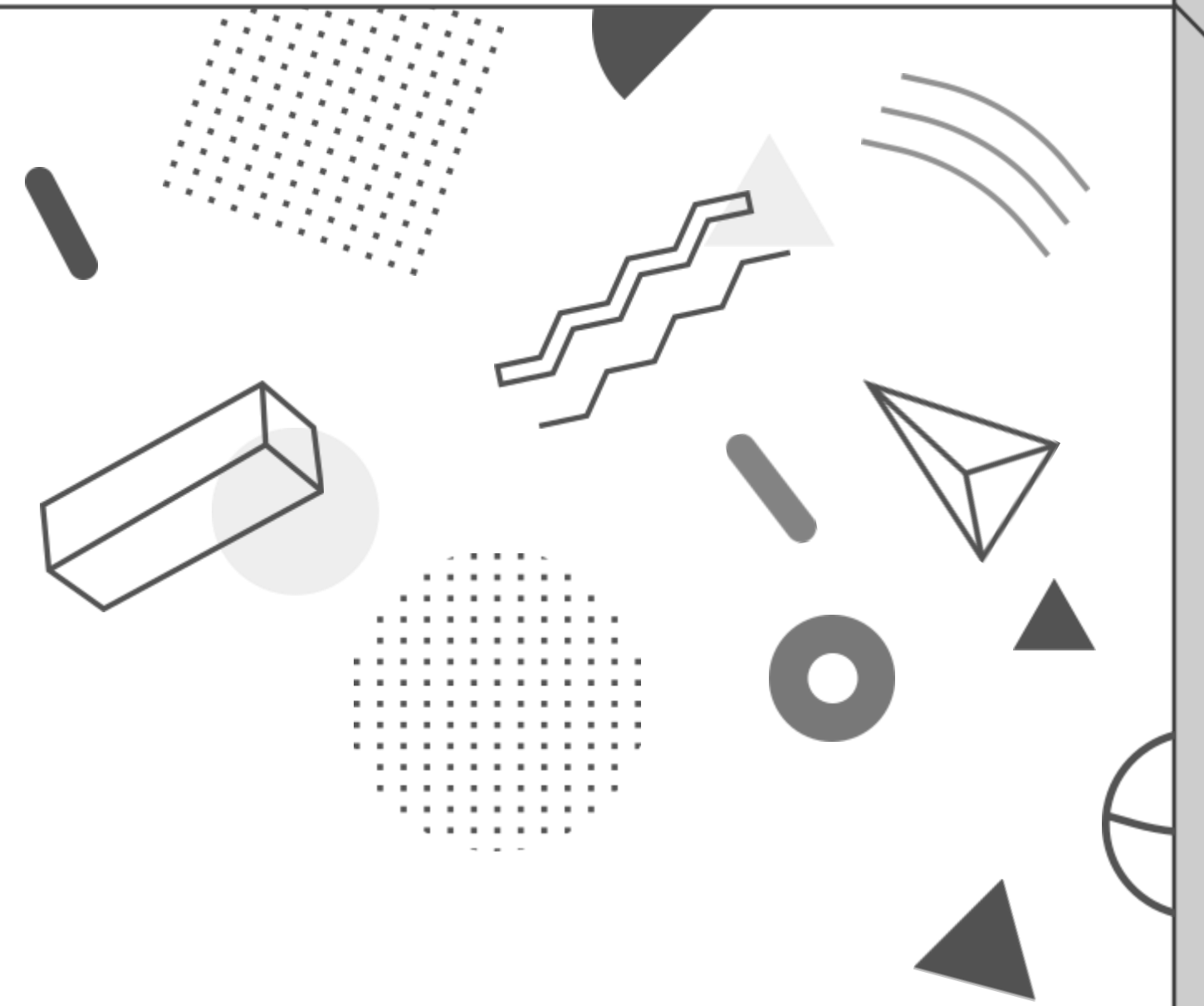
→ 데이터를 주고받는데 **자원의 이름과 표현을 통해** 주고받는 것





02 REST API의 구성 요소

REST API는 무엇으로 이루어져 있는가

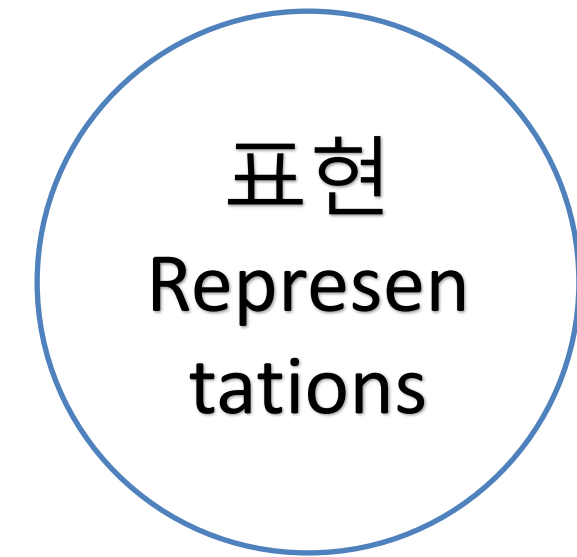




URI



HTTP Method

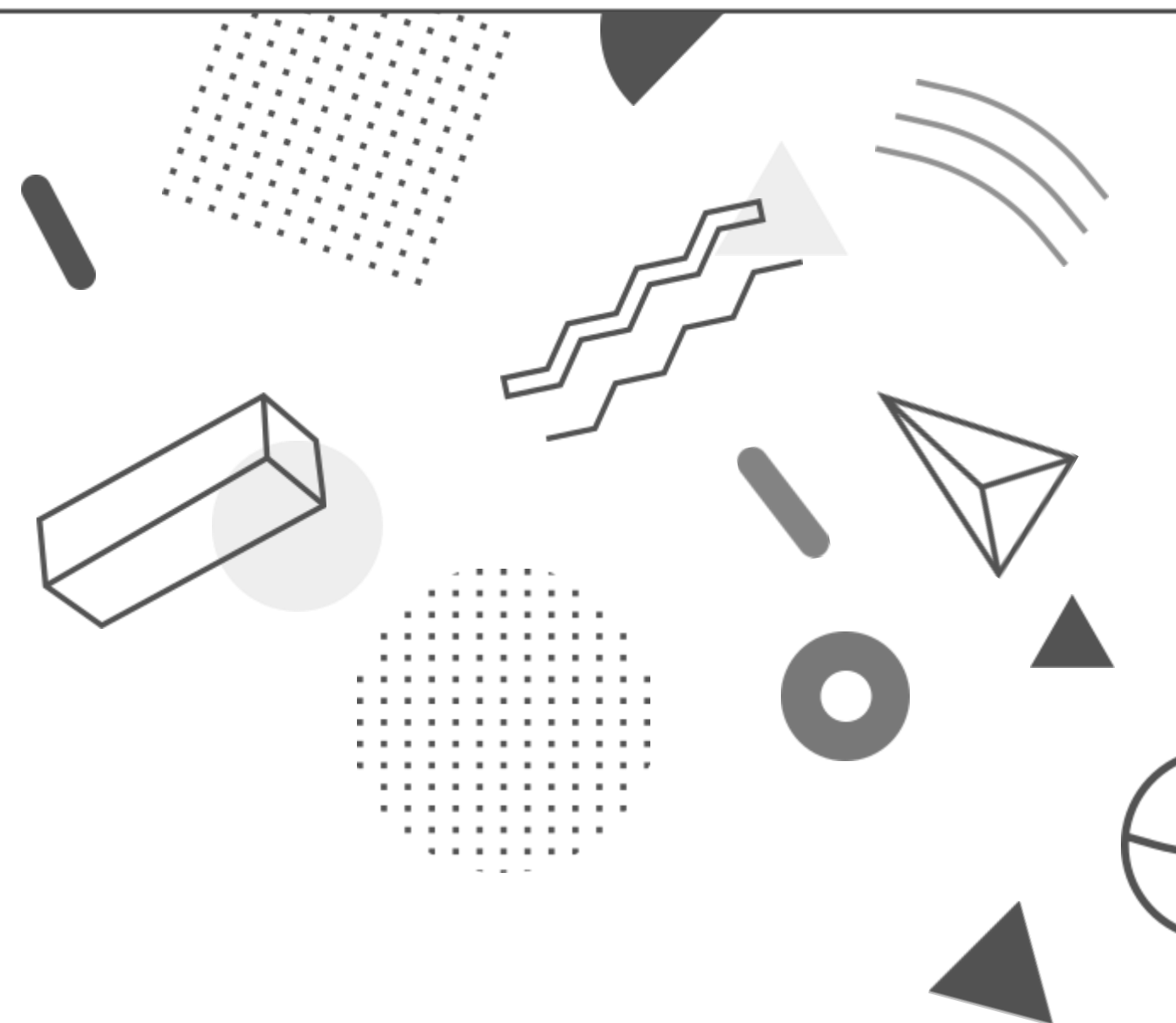


JSON, XML 등

03

REST API의 예시

그래서 REST API는 어떻게 사용하는가



<https://ssafy.com/classes>

```
{  
  "results": [  
    {"idx": 16, "track": "java", "professor": "홍길동"},  
    {"idx": 17, "track": "java", "professor": "서민규"},  
    {"idx": 18, "track": "java", "professor": "김길동"},  
    {"idx": 19, "track": "java", "professor": "이길동"}  
  ]  
}
```

<https://ssafy.com/classes>

<https://ssafy.com/classes/17>

```
{  
  "results": [  
    {"idx": 16, "track": "java", "professor": "홍길동"},  
    {"idx": 17, "track": "java", "professor": "서민규"},  
    {"idx": 18, "track": "java", "professor": "김길동"},  
    {"idx": 19, "track": "java", "professor": "이길동"}  
  ]  
}
```

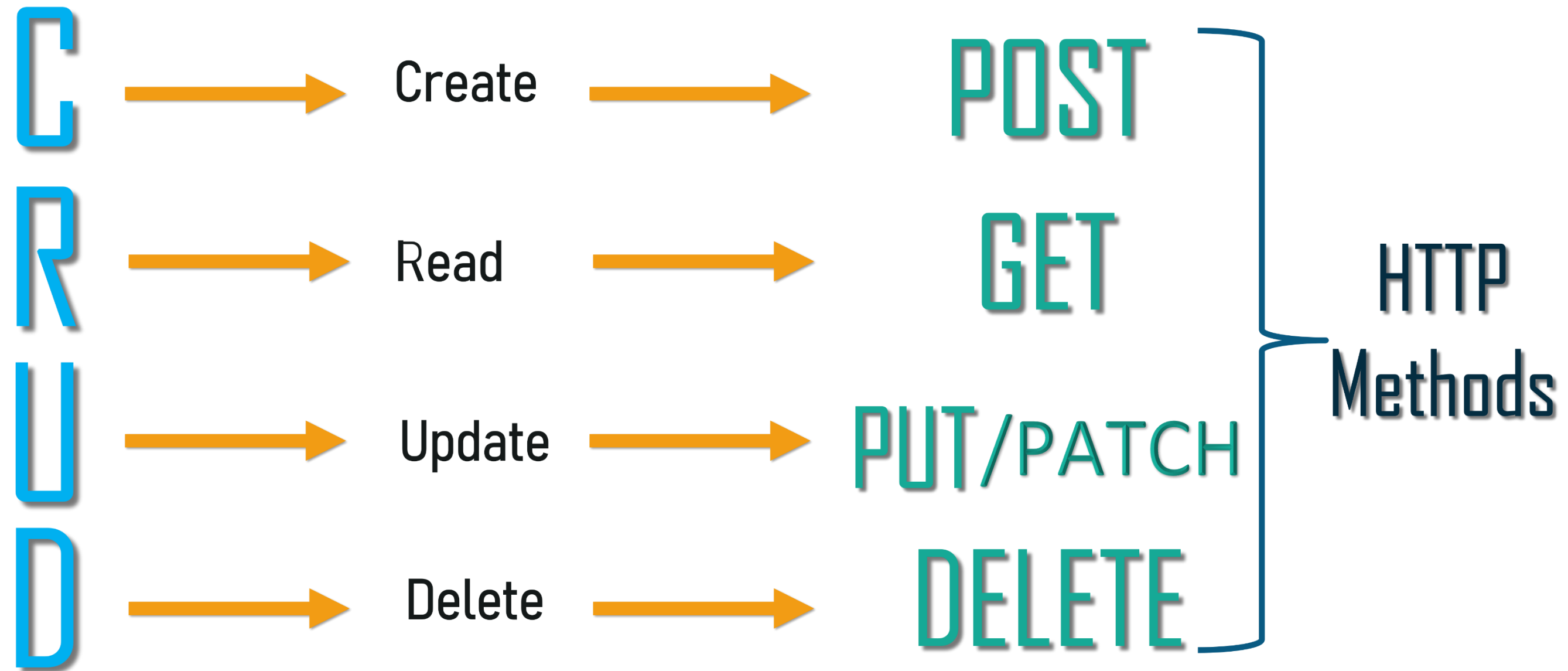
<https://ssafy.com/classes/17/students>

```
{
  "results": [
    {"idx": 6, "name": "김인태"},
    {"idx": 7, "name": "김현교"},
    {"idx": 10, "name": "김희영"},
    {"idx": 21, "name": "이진오"},
    {"idx": 25, "name": "추준성"}
  ]
}
```

<https://ssafy.com/classes/17/students>

<https://ssafy.com/classes/17/students/10>

```
{
  "results": [
    {"idx": 6, "name": "김인태"},
    {"idx": 7, "name": "김현교"},
    {"idx": 10, "name": "김희영"},
    {"idx": 21, "name": "이진오"},
    {"idx": 25, "name": "추준성"}
  ]
}
```



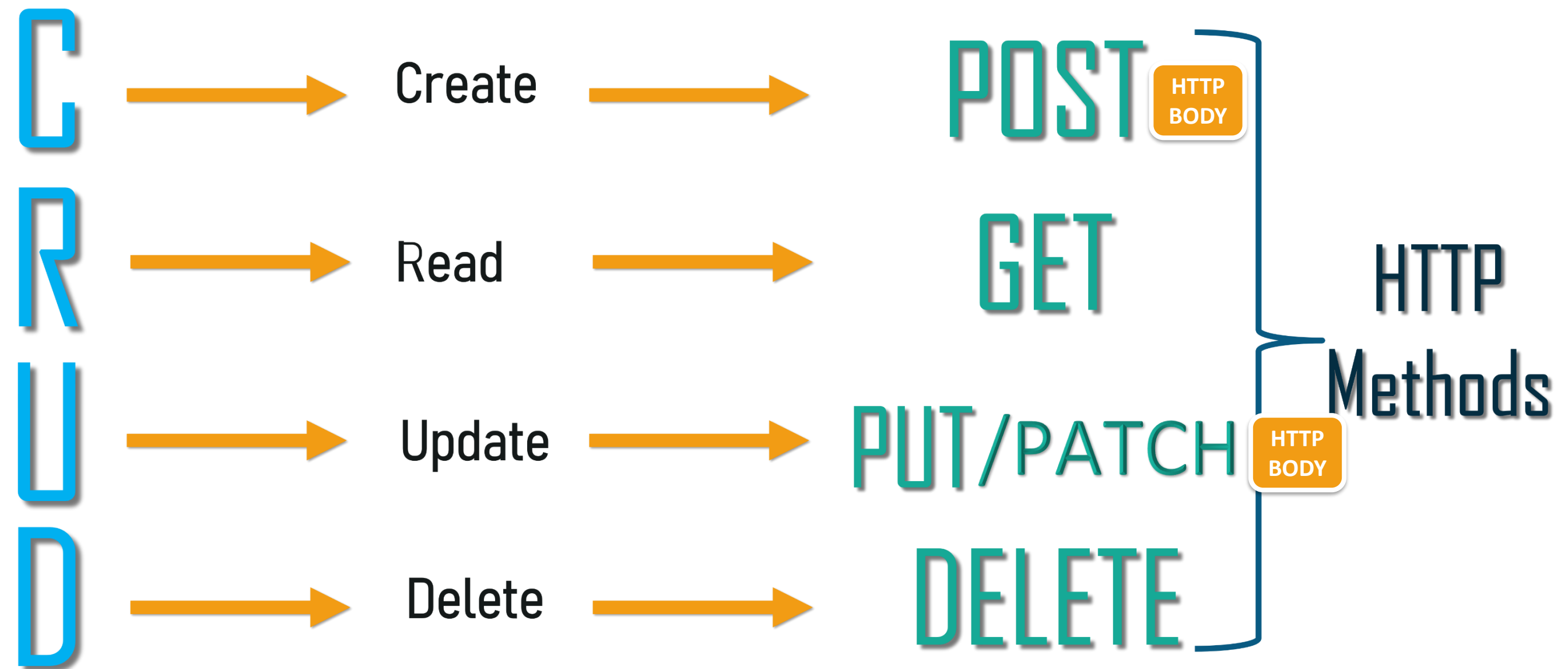


GET

<https://ssafy.com/classes/17/students/10>



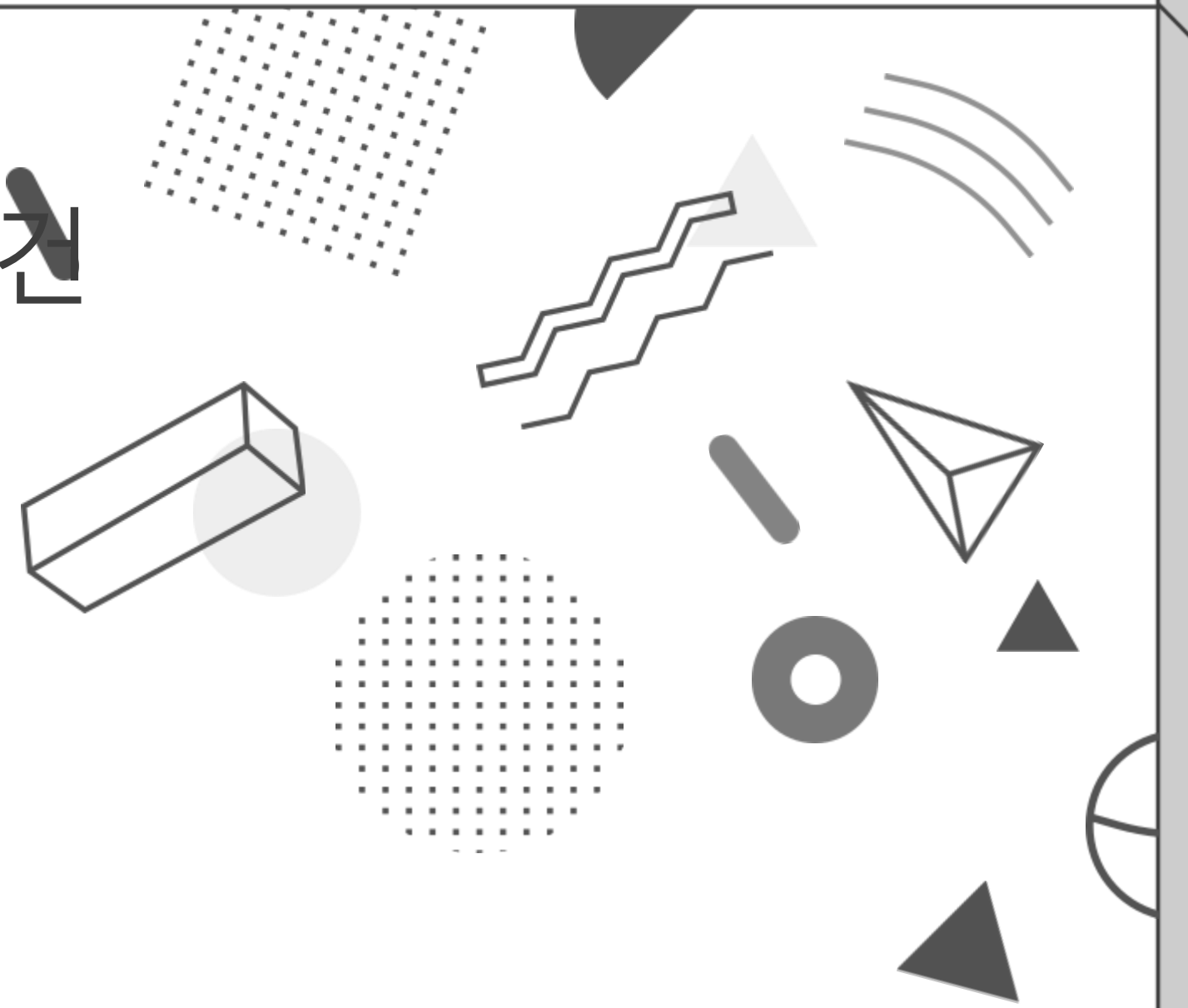
POST <https://ssafy.com/classes/17/students/10>





04 REST 아키텍처의 6가지 제약 조건

무엇이 REST 아키텍처인가



1. CLIENT – SERVER 구조

- ❖ REST 아키텍처는 클라이언트와 서버로 구분
- ❖ 클라이언트와 서버는 완전히 독립적

2. Uniform Interface 인터페이스 일관성

- ❖ REST API는 일관적인 인터페이스로 분리되어야 함
- ❖ 동일한 리소스에 대한 모든 API 요청은 오직 하나의 URI에 속함을 보장
- ❖ 클라이언트와 서버의 결합도를 낮출 수 있음 → Decoupling

3. Stateless 무상태성

- ❖ REST API 서버는 작업을 위한 **상태정보를 따로 저장하고 관리하지 않음**
- ❖ 서비스의 **자유도**가 높아지고 서버의 **구현이 단순**해짐

4. Cacheable 캐시 가능

- ❖ **대량의 요청을 효율적**으로 처리하기 위해 **캐시**가 요구됨
- ❖ 전체 응답시간, 성능, 서버의 자원 이용률을 향상

5. Layered System(계층화)

- ❖ 클라이언트나 서버가 서로 직접 연결되었는지, 또는 중간 서버를 통해 연결되었는지를 알 수 없도록 **계층적으로 설계**
- ❖ 중간 서버는 로드 밸런싱 기능이나 공유 캐시 기능을 제공함으로써 **시스템 규모 확장성을 향상**

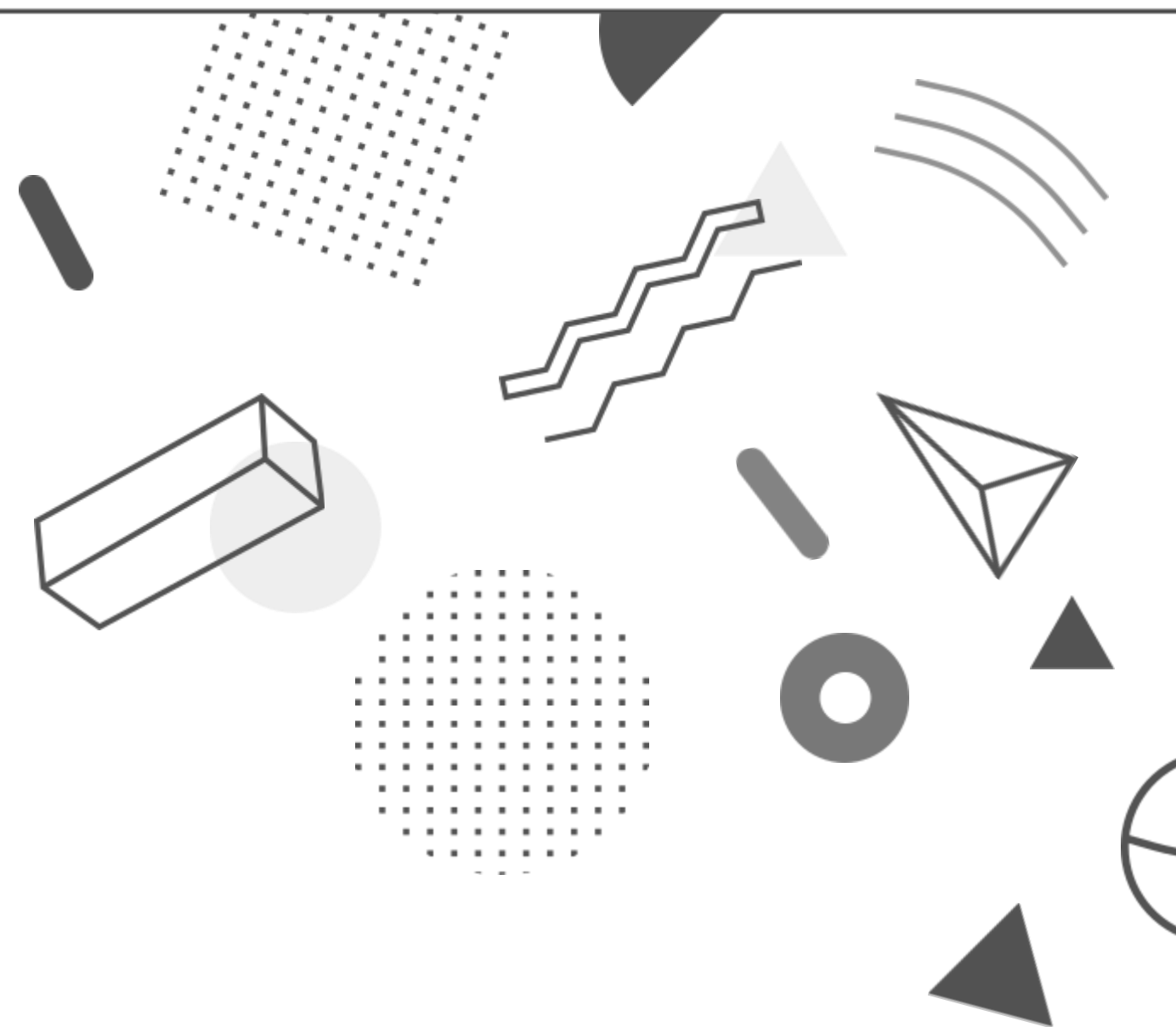
6. Code-On-Demand [선택적]

- ❖ 클라이언트에 보내는 데이터에 **바로 실행 가능한 코드**를 보내서 이것을 클라이언트에서 실행하는 것 → 클라이언트를 단순화

05

Uniform Interface

모든 HTTP API가 REST API가 아닌 이유



1. Identification of Resources

- ❖ 리소스가 URI로 식별이 되는지

2. Manipulation of Resources through Representations

- ❖ 서버가 클라이언트에서 이해할 수 있는 형식으로 응답하는지

3. Self-Descriptive Messages

❖ 메시지가 스스로 설명되어야 한다.

→ 메시지의 모든 요소는 메시지만으로 그 뜻을 알 수 있어야 한다.

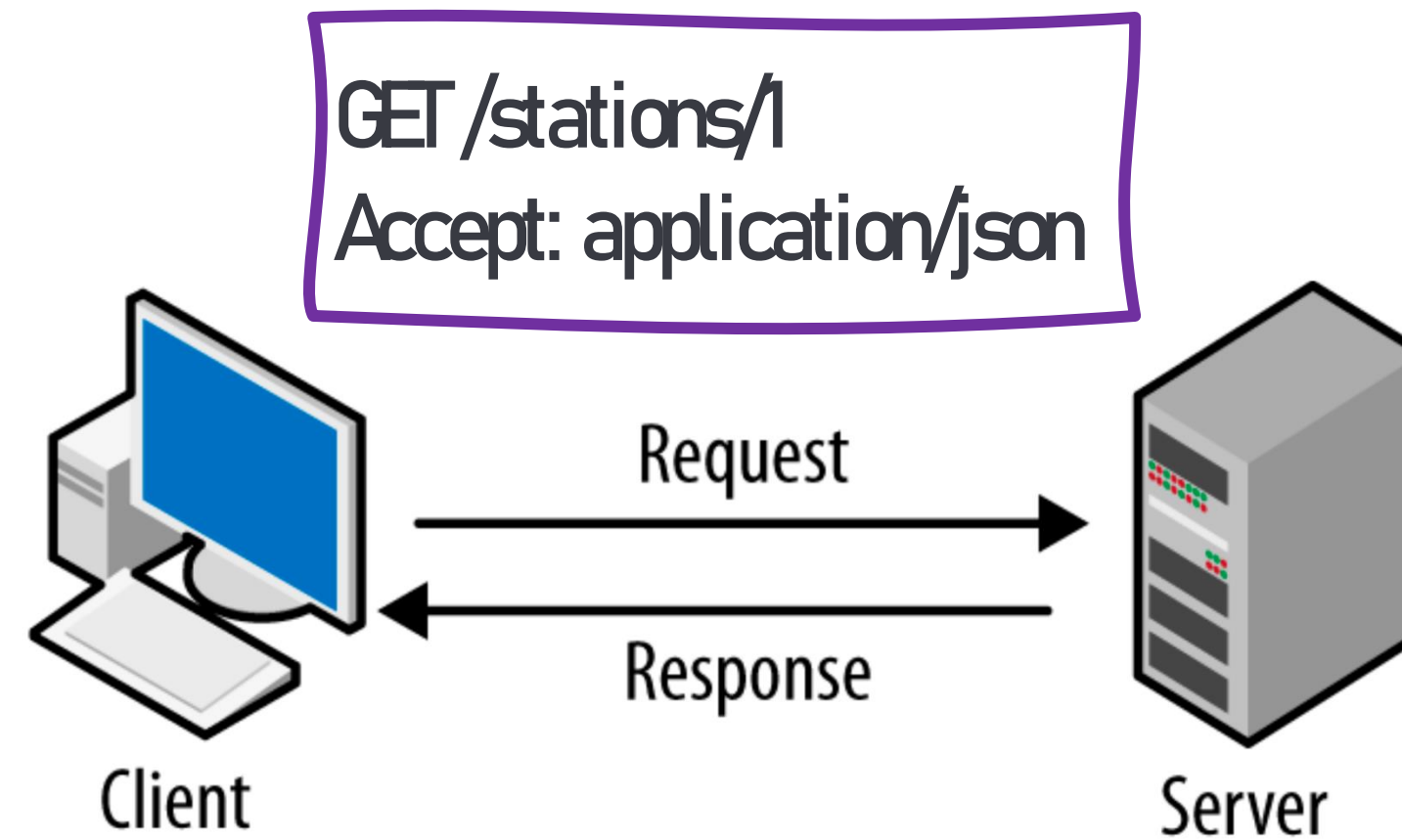
WHY?

❖ 확장성

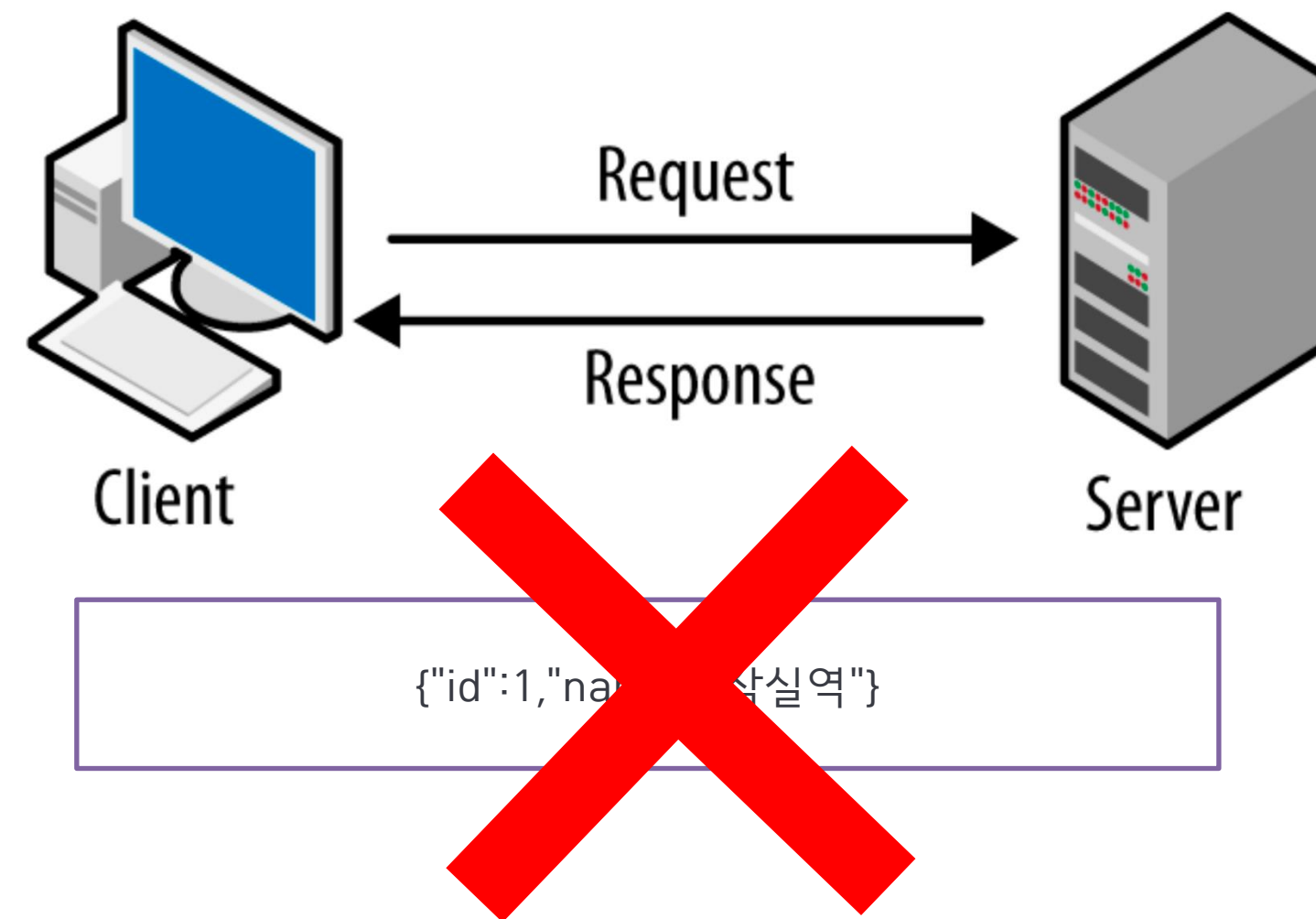
서버와 클라이언트가 바뀌어도 항상 해석이 가능하도록

API만으로 의미를 알 수 있게 작성하는 것

3. Self-Descriptive Messages

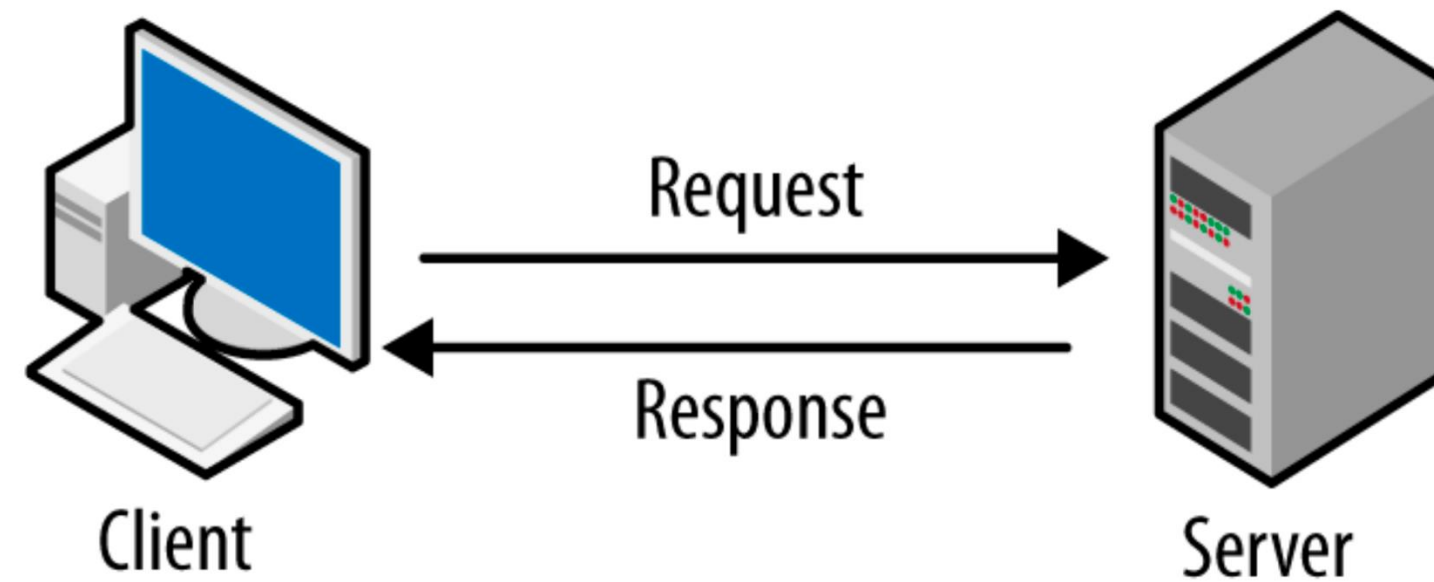


3. Self-Descriptive Messages



3. Self-Descriptive Messages

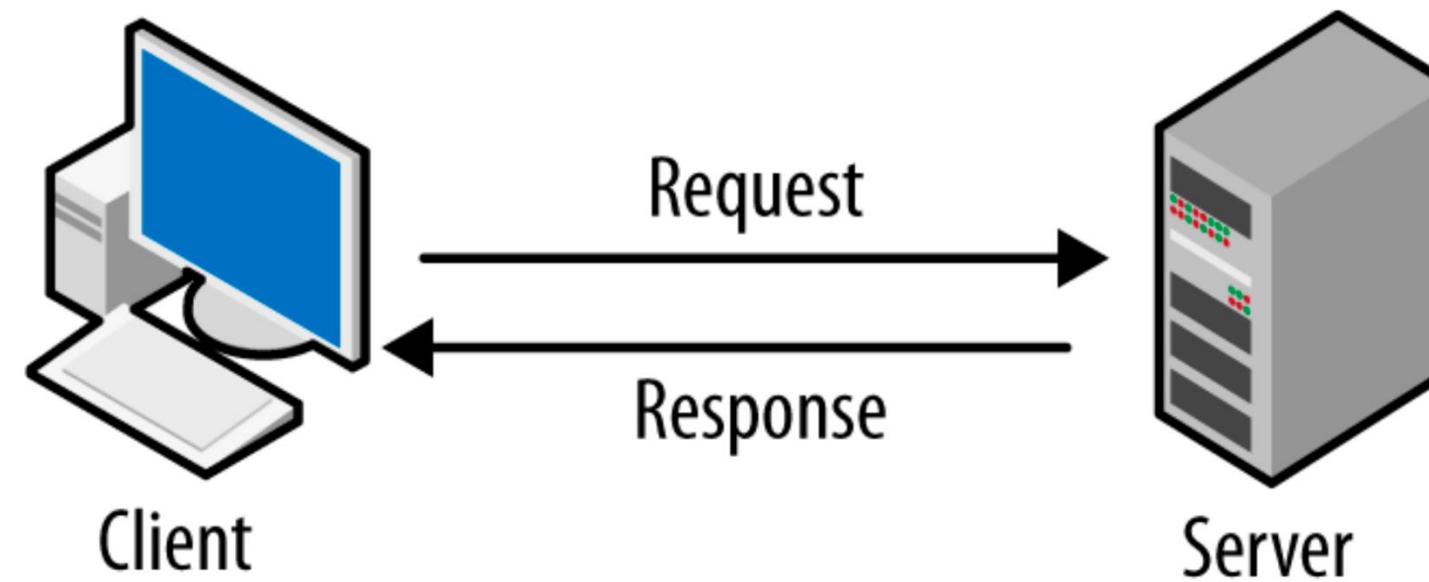
❖ 미디어 타입을 정의하여 콘텐츠 타입으로 지정



```
HTTP/1.1 200 OK  
Content-Type: application/ecsimsw.subways+json  
{ "id": 1, "name": "잠실역" }
```

3. Self-Descriptive Messages

- ❖ Link Header에 명세를 확인할 수 있는 링크를 넣기



```
HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://ecsimsw.com/docs/subway>; rel="profile"
{"id":1,"name":"잠실역"}
```

4. Hypermedia As The Engine Of Application State

HATEOAS - 헤이티오스

- ❖ 애플리케이션 상태는 Hyperlink를 이용해서 전이(이동)되어야 한다
 - 서버가 응답에 다음 요청이 될만한 URI를 포함시켜 반환하는 것

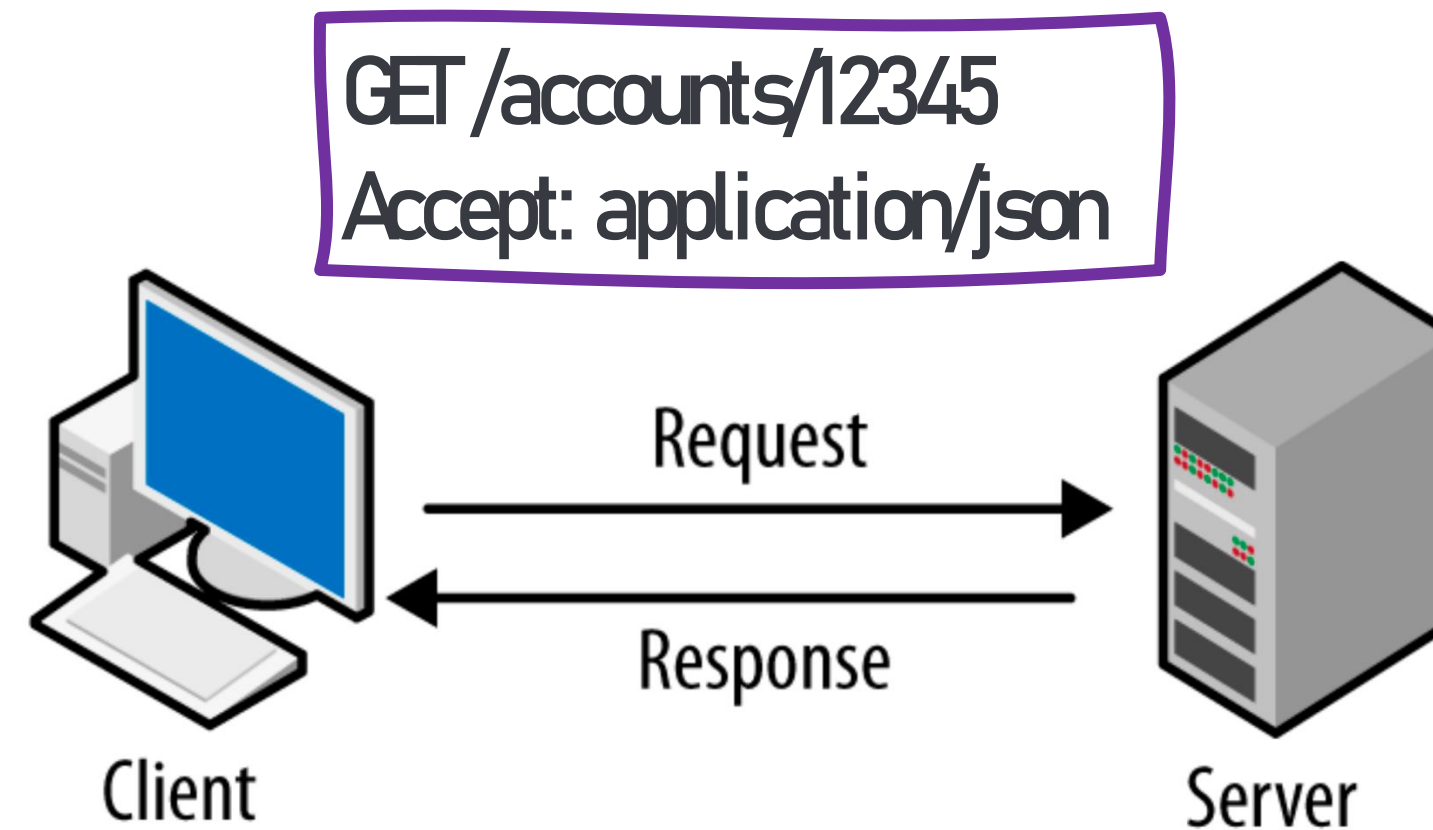
WHY?

- ❖ 느슨한 연결을 위해

클라이언트와 서버가 독립적으로 진화할 수 있도록
서버와 서버, 서버와 클라이언트를 분리

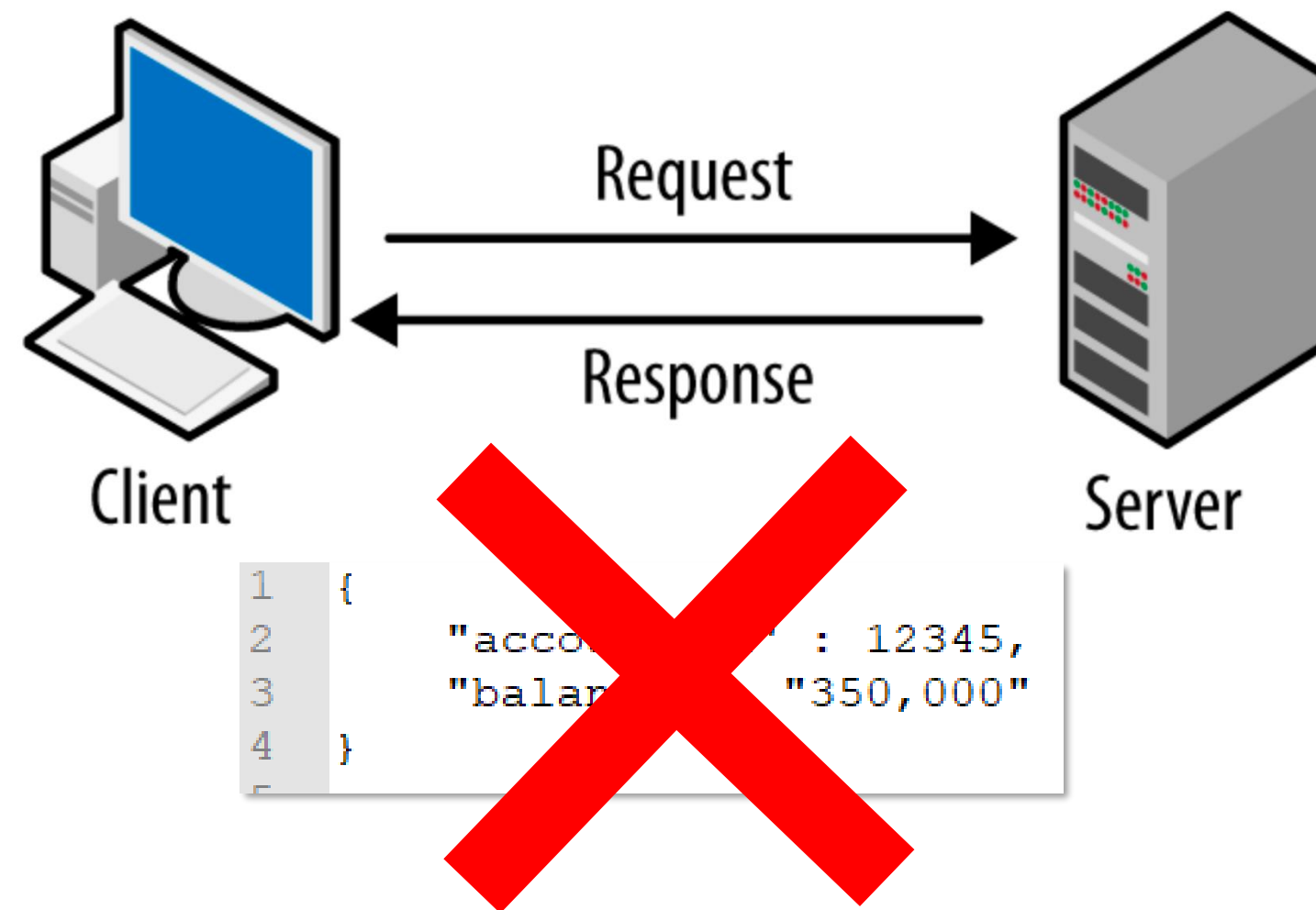
4. Hypermedia As The Engine Of Application State

HATEOAS - 헤이티오스



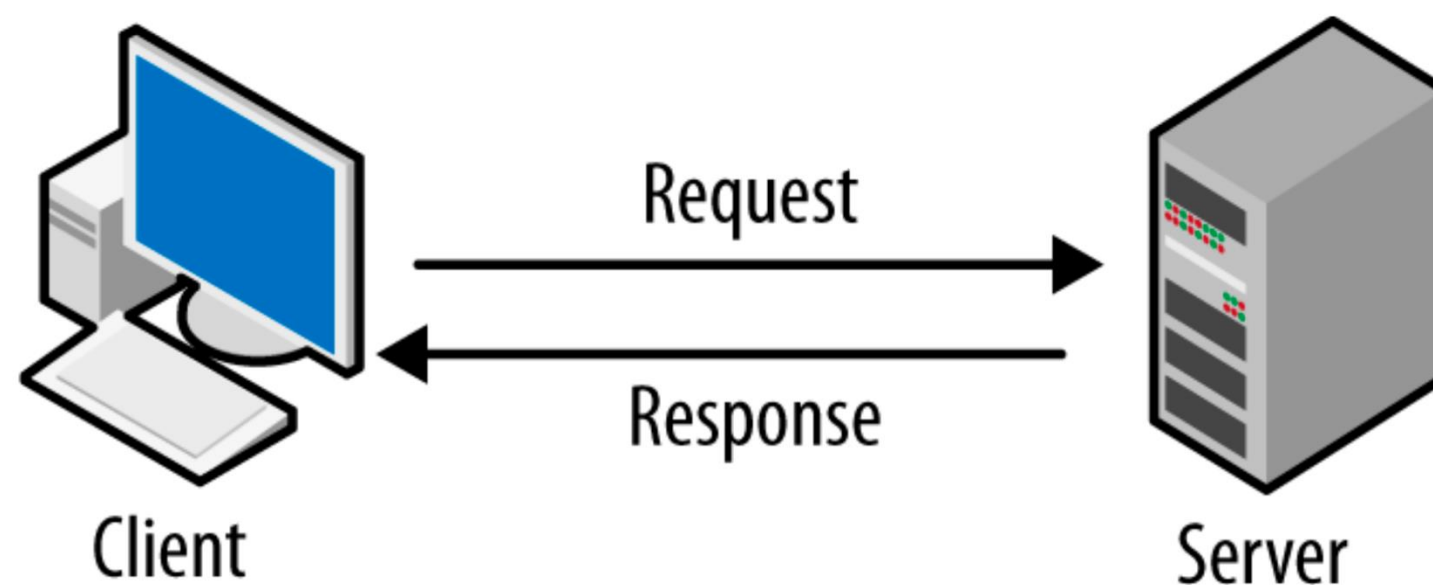
4. Hypermedia As The Engine Of Application State

HATEOAS - 헤이티오스



4. Hypermedia As The Engine Of Application State

HATEOAS - 헤이티오스

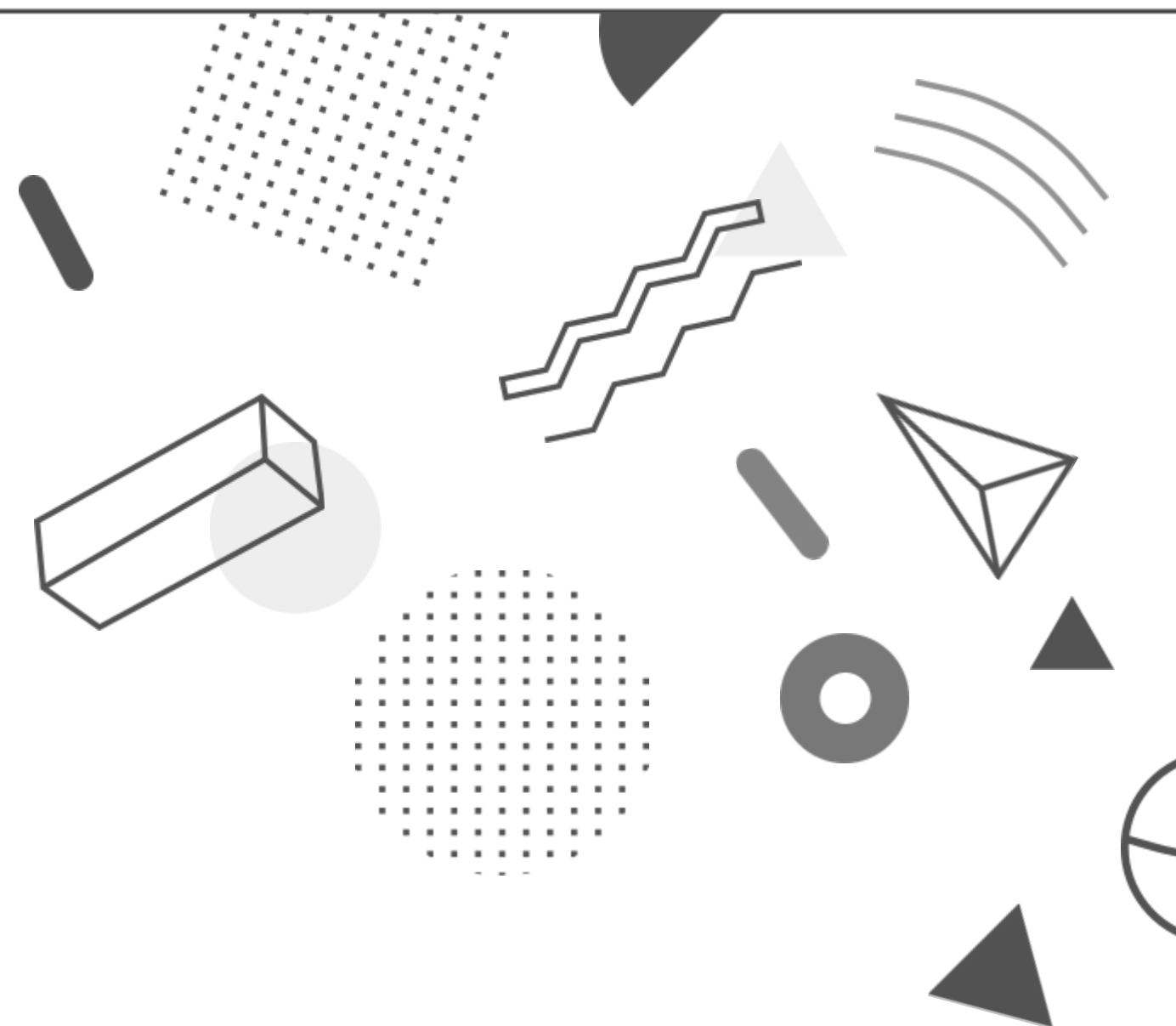


```
1 {  
2   "account_id" : 12345,  
3   "balance" : "350,000",  
4   "links" : [  
5     {  
6       "rel" : "self",  
7       "href" : "http://localhost:8080/accounts/12345"  
8     }, {  
9       "rel" : "withdraw",  
10      "href" : "http://localhost:8080/accounts/12345/withdraw"  
11     }, {  
12      "rel" : "transfer",  
13      "href" : "http://localhost:8080/accounts/12345/transfer"  
14     }  
15   ]  
16 }
```

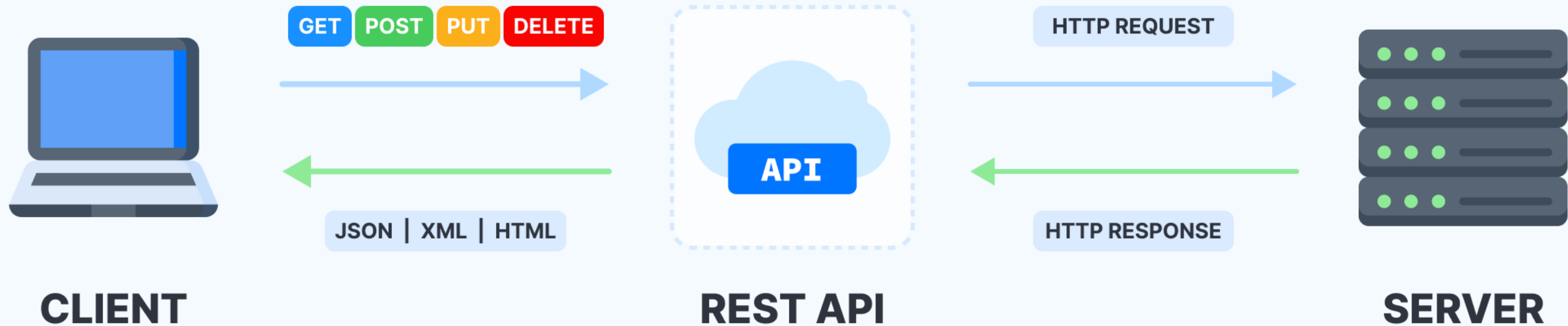


06 마무리

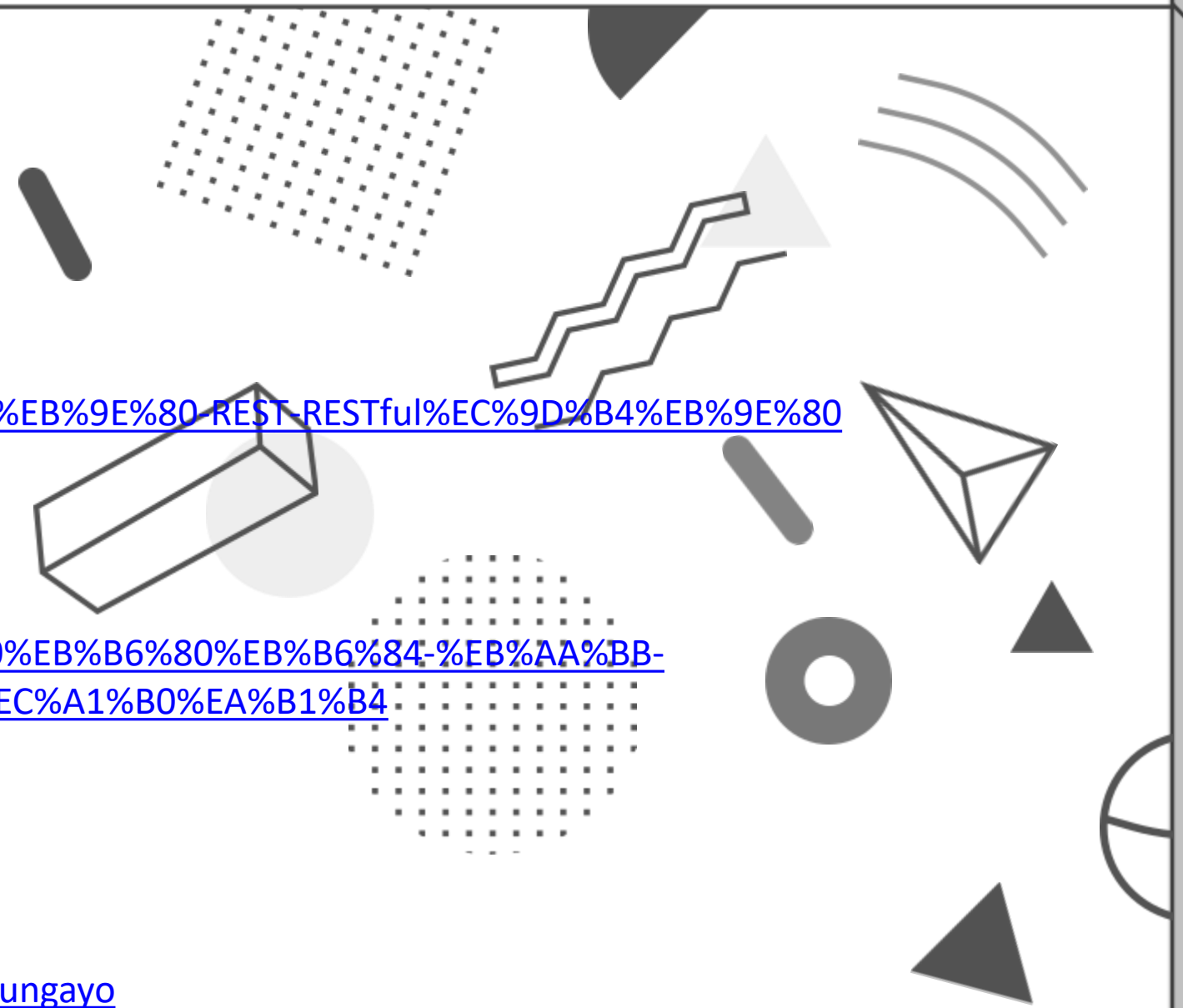
REST API



REST API Model



○○○



[%EB%9E%80 REST RESTful%EC%9D%B4%EB%9E%80](#)

[%EB%B6%80%EB%B6%84-%EB%AA%BB-EC%A1%B0%EA%B1%B4](#)

[ungayo](#)

<https://appmaster.io/ko/blog/rest-apiran-mueosimyeo-dareun-yuhyeonggwaeoddeohge-dareungayo>



THANK YOU

SSAFY 7기 서울 17반 김희영

