REST API

박지호

REST API란 (일반적인 지식)

1. URL를 통해 자원을 지정
2. HTTP 메서드: 자원에 대한 행위 표현

CREATE	POST	/user
READ	GET	/user/1
UPDATE	PUT	/user/1
DELETE	DELETE	/user/1

Roy fielding

Roy fielding 의 블로그

Search my dissertation and you won't find any mention of CRUD or POST…
The main reason for my lack of specificity is because the methods defined by HTTP are part of the Web's architecture definition, not the REST architectural style. Specific method definitions ~~ simply don't matter to the REST architectural style, so it is difficult to have a style discussion about them…. As long as the method is being used according to its own definition, REST doesn't have much to say about it

- It is okay to use POST (2009.03.03)

Roy fielding 의 블로그

나의 논문에는 CRUD에 대한 내용은 없다.

HTTP 메서드는 REST가 아니라 웹의 아키텍쳐 스타일의 일부다.

HTTP에서 정의한 방식대로만 잘 쓴다면 REST에 추가적인 할 말은 없다.

Roy Fielding의 REST API란?

REST

아키텍쳐 스타일에 부합

하는 API

REST 아키텍쳐 스타일에 부합하는 API

1. Client - Server
2. Stateless
3. Cache
4. **Uniform Interface**
5. Layered System
6. Code - On - Demand

Uniform Interface

1. Identification of resources(자원에 대한 식별)

2. Manipulation of resources through representations(표현을 통한 자원에 대한 조작)

3. Self - descriptive message(자기 서술적 메세지)

4. HATEOAS (hypermedia as the engine of application state)

1.Identification of resources

자원에 대한 식별

<자원>

- 이름을 지닐 수 있는 모든 정보
- 개념적인 대상 (ex. 문서, 이미지, 자원들의 집합, 실존하는 대상 등)

1.Identification of resources

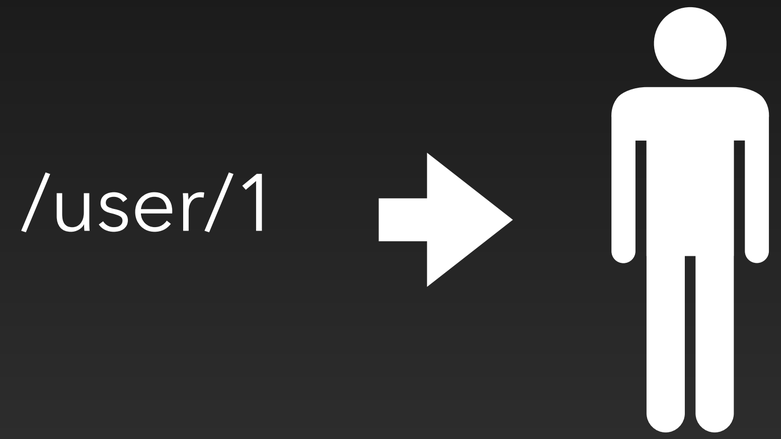
자원은 **객체**

- 상태가 변한다 -> 변하지 않는 **식별자**가 필요하다.

1.Identification of resources

"**URI을 통해서 자원을 식별**해야 한다."

/user/1

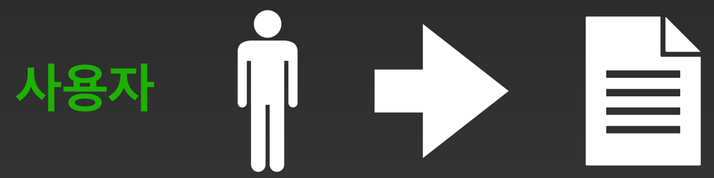


2.manipulation of resources through representations

표현을 통한 자원에 대한 조작

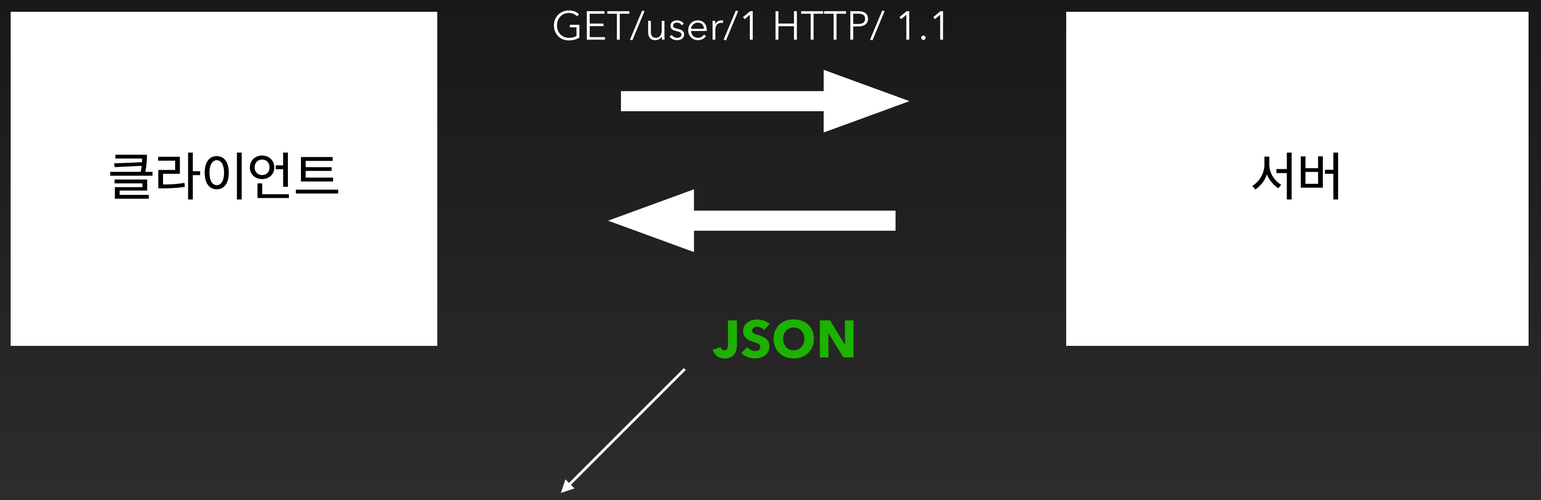
- 자원은 다양한 방식으로 표현 가능

ex) 문서, 파일, HTTP 메세지 엔티티 등등

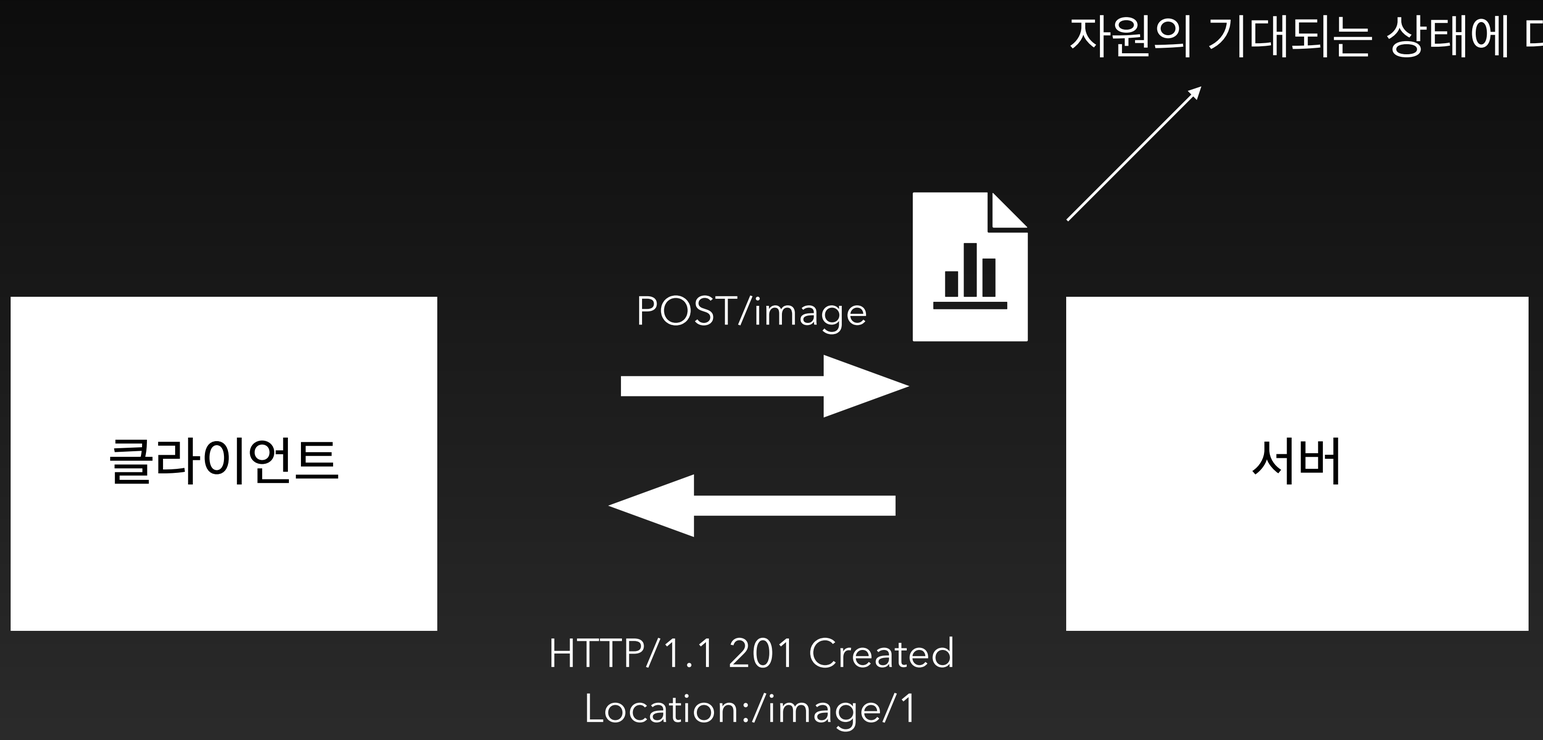


HTTP/1.1 200ok
Content-Type : text/plain ~~

2.manipulation of resources through representations



2.manipulation of resources through representations



REST : 자원에 대한 표현 전송

Representational State Transfer

: 표현된 (자원의) 상태

- 자원의 현재 상태

- 자원의 기대되는 상태

3. Self - descriptive

자기 서술적 메세지

메세지는 스스로에 대해 설명해야 한다.

3. Self - descriptive

클라이언트와 서버 사이의 컴포넌트들은
메세지의 내용을 참고하여 적절한 작업 수행

3. Self - descriptive

```
HTTP/1.1 200 OK

[ { "op": "remove", "path": "/a/b/c" } ]
```

기본적인 Content-Type이 빠져 있어 Self - descriptive 하지 못한 상태

3. Self - descriptive

```
HTTP/1.1 200 OK
Content-Type: application/json

[ { "op": "remove", "path": "/a/b/c" } ]
```

기본적인 Content-Type이 있지만 아직까지 부족한 상태

3. Self - descriptive

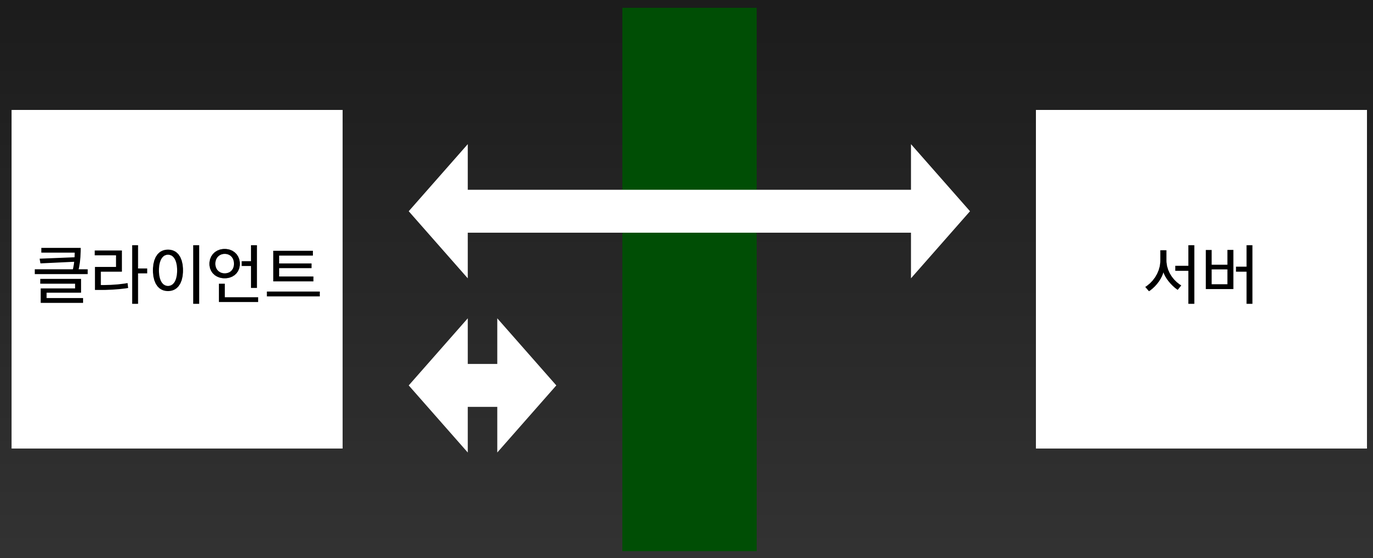
```
HTTP/1.1 200 OK
Content-Type: application/json-patch+json

[ { "op": "remove", "path": "/a/b/c" } ]
```

IANA에 정의된 application/json-patch+json 명세를 보고
op와 path의미 파악할 수 있어 Self - descriptive하다.

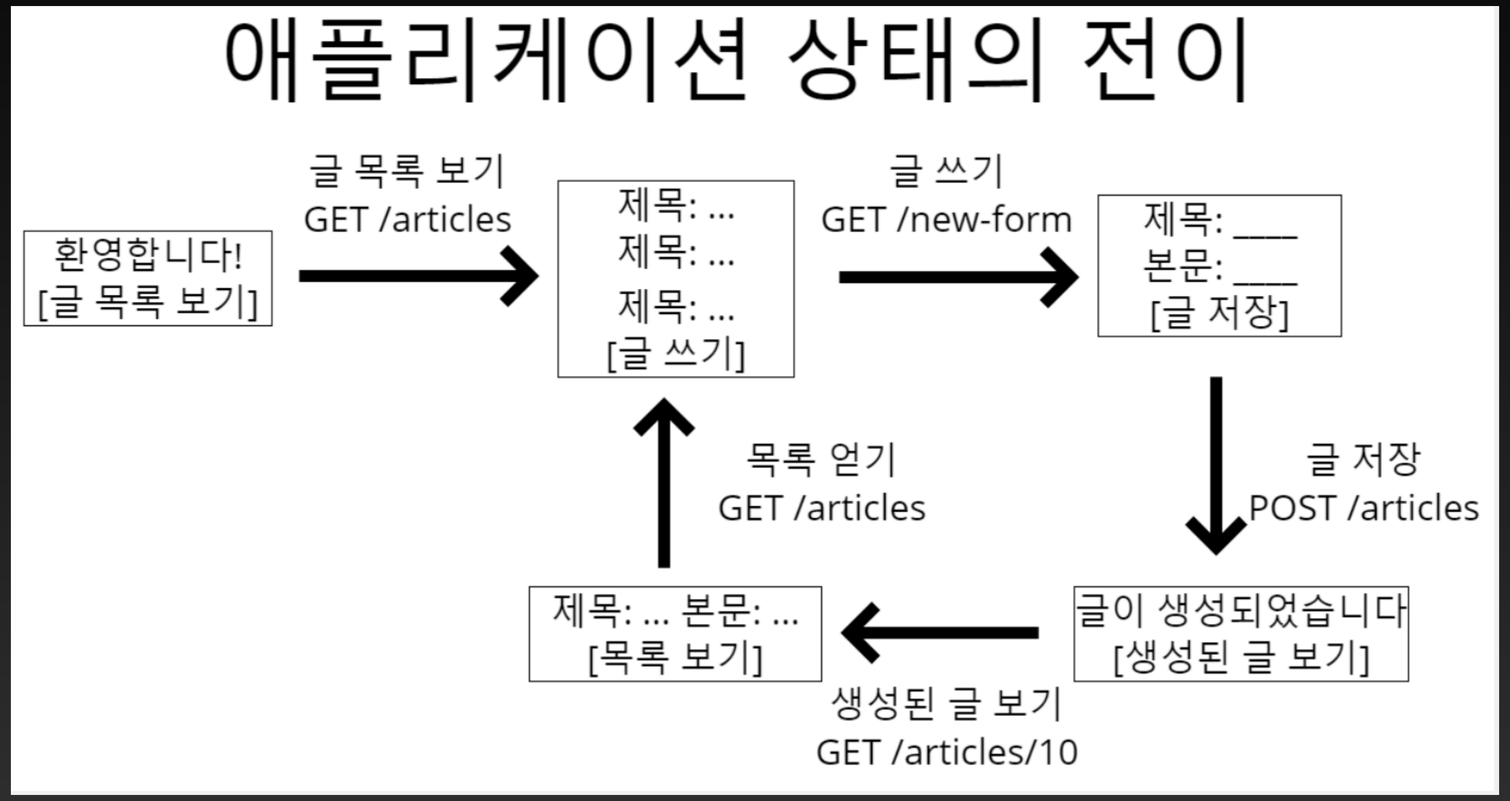
3. Self - descriptive

캐쉬 관련 헤더를 통한 캐쉬 전략 지정

- HTTP/1.1: Cache-Control, Age, Tag, Vary



4. HATEOS



애플리케이션의 상태(State)는 Hyperlink를 이용해 전이(Transfer)되어야 한다.

4. HATEOS

```
HTTP/1.1 200 OK
Content-Type: application/json
Link: </articles/1>; rel="previous",
      </articles/3>; rel="next;

{
    "title": "The second article",
    "contents": "blah blah..."
}
```

현 시점 사회에서는 !?

Microsoft REST API Guidelines(2016)

- uri는 https://{serviceRoot}/{collection}/{id} 형식어야 한다.
- GET, PUT, DELETE, POST, HEAD, PATCH, OPTIONS를 지원해야한다.
- API 버저닝은 Major.minor로 하고 uri에 버전 정보를 포함시킨다.
- 등 ~

아뉘
Self - descriptive
HATOES
없자나 REST 아님

왜 Uniform Interface?

독립적 진화

- 서버와 클라이언트가 각각 독립적으로 진화한다.
- 서버의 기능이 변경되어도 클라이언트를 업데이트할 필요가 없다.
- REST를 만들게 된 계기 “how do I improve HTTP without breaking the Web”

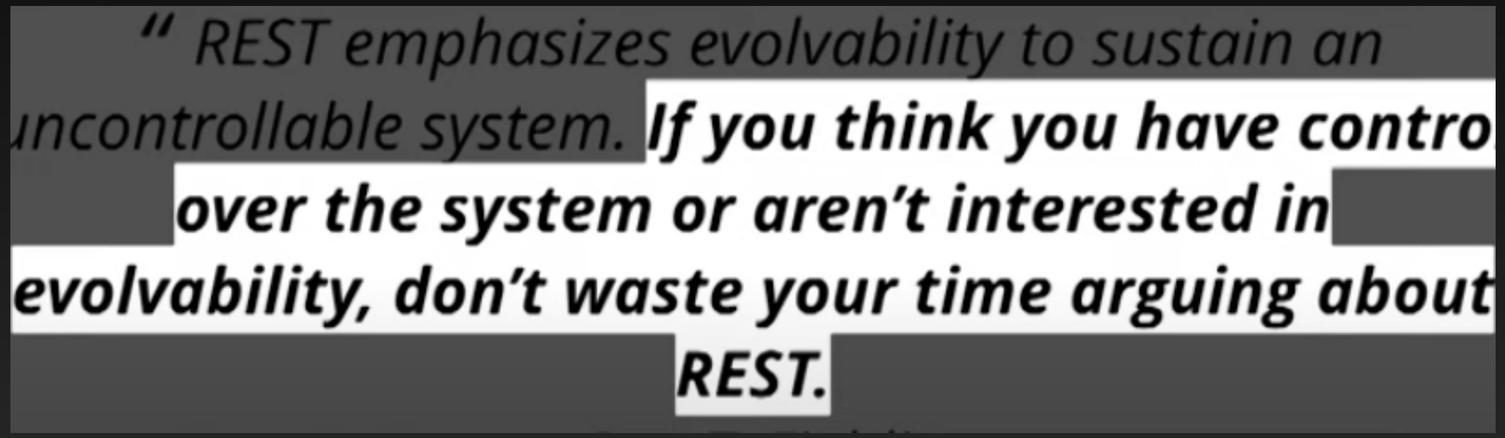
REST API의 제약조건들을 다지켜야 하나?

응 당연 ~

“An API that prvides network-based access to resources via a uniform interface of self-descriptive messages containing hypertext to indicate potential state transitions might be part of an overall system that is a RESTful application.

-Roy fielding-

원격 API가 꼭 REST API 여야 하는가?



시스템 전체를 통제할 수 있다고 생각하거나, 진화에 관심이 없다면, REST에 대해 따지느라 시간을 낭비하지마라

그럼 어떻게 해야할까

1. REST API를 구현하고 REST API라고 부른다.
2. REST API 구현을 포기하고 HTTP API라고 부른다.
3. REST API가 아니지만 REST API라고 부른다(현재 상태)

제발 ㅠㅠ
지구 끝까지 찾아가서 댓글 달거임!!

REST API의 최종 목적

REST API의 최종 목적은

웹 서비스와 애플리케이션 간의 상호 작용을 단순화하고 효율화하여, 개발자가 더 나은 사용자 경험을 제공하고, 다양한 플랫폼과 기기에서 원활하게 작동하는 애플리케이션을 쉽게 개발할 수 있도록 하는 것입니다.

THANKS

Q/A