조새나라

What Is Rest?

REST Definition

UNIVERSITY OF COLIFORNIA, IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

in Information and Computer Science

by

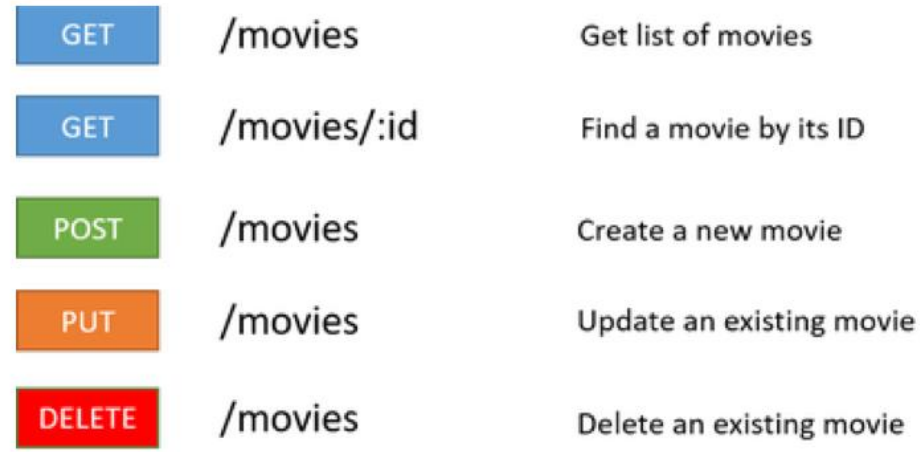
Roy Thomas Fielding

2000

REST(Representational State Transfer)는 대표적인 HTTP 아키텍쳐 스타일이다.

HTTP에 주요 기여자 중 한 명인 로이 필딩의 2000년에 발표된 박사학위 논문에서 소개되며 최초로 알려졌으며 현재의 아키텍쳐가 웹의 본래 설계된 우수성을 제대로 사용하지 못한다고 생각하였고, 때문에 웹의 장점을 활용한 네트워크 기반의 아키텍쳐 스타일인 REST를 소개하게 되었다.

“REST”

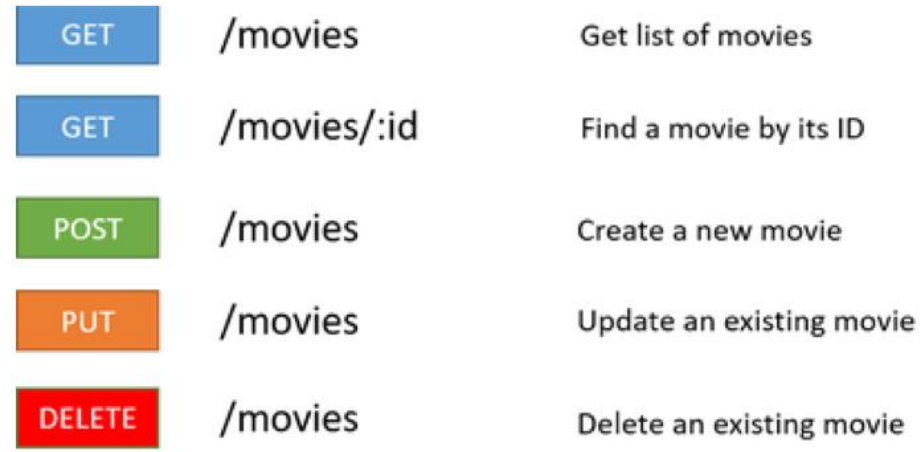


REST는 리소스 지향 아키텍쳐로 리소스를 이름으로 구분하여 해당 리소스의 상태를 주고받는 것을 의미한다.

리소스는 "명사" 로 표현하며 상태 정보를 HTTP의 CRUD 메소드를 통해 주고받게 되는 것이다.

이때 전송되는 데이터는 Json 혹은 XML로 주고 받는 것이 일반적이다.

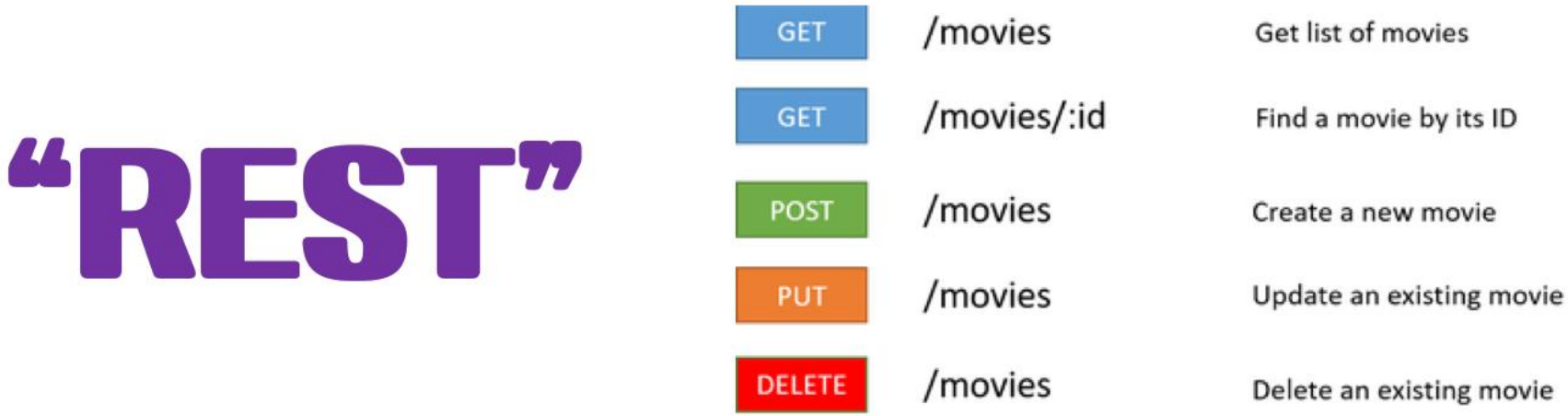
“REST”



REST는 "리소스" , "메서드" , "메시지" 3가지 요소로 구성된다.
"제목이 Red인 영화를 생성한다." 라는 호출이 있을 때 이는 아래와 같다.

HTTP POST , http://myweb/movies/

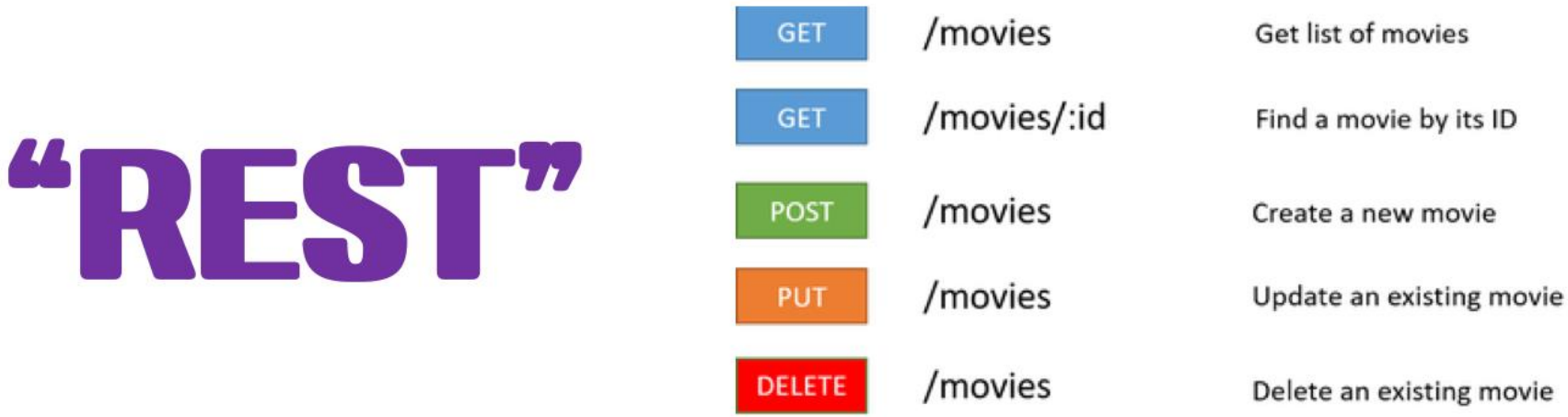
```
{
   “movies":{
      “title":“Red"
   }
}
```



"제목이 Red인 영화의 개봉 년도를 2010으로 수정한다." 라는 호출이 있을 때 이는 아래와 같다.

HTTP PUT, http://myweb/movies/red

```
{
   "title":"Red",
   "year":"2010"
}
```



"제목이 Red인 영화를 조회한다." 라는 호출이 있을 때 이는 아래와 같다.

HTTP Get, http://myweb/movies/Red

"제목이 Red인 영화를 삭제한다." 라는 호출이 있을 때 이는 아래와 같다.

HTTP DELETE, http://myweb/movies/Red

{REST}

HTTP CRUD 메서드

CRUD	HTTP 메서드	기능	Idempotent
C	Post	Create	No
R	GET	Select	Yes
U	PUT	Update	Yes
D	DELETE	Delete	Yes

*Idempotent : 멱등성, 여러 번 수행해도 결과가 같은 경우.
멱등적이지 않은 경우 트랜잭션에 대한 처리가 필요하다.

{REST}

REST API 정의는 상당히 간단하다.
단순하게 리소스를 URI로 정해준 후에,
거기에 HTTP 메서드를 이용해서 CRUD를 구현하고,
메시지를 JSON으로 표현하여 HTTP Body에 실어 보내면 된다.

{REST} 특성

유니폼 인터페이스(Uniform Interface)

: HTTP 표준에만 따른다면 어떠한 기술이든 백엔드 플랫폼에서 사용할 수 있다.
이는 프론트와 백간의 느슨한 연결을 지원하기에 기술 다양성이 제공되는 것이다.

무상태성(Stateless)

: HTTP의 특성과 같이 상태를 지니고 있지 않다. 즉 들어오는 요청에 대해서만
처리하고 HTTP Session과 같은 컨텍스트 저장소에 상태를 저장하지 않아도 된다.

캐쉬(Cacheable)

: HTTP 기존의 웹 표준을 그대로 사용하기 때문에 캐쉬 또한 사용이 가능하다.
일반적으로 60% ~ 80%의 트랜잭션이 Select이기 때문에 웹 캐쉬 서버에
캐쉬하는 것은 용량 및 성능의 향상에 큰 도움이 된다.

{REST} 특성

자기 서술성(Self-descriptiveness)

: REST API의 구조가 간단 명료하기 때문에 API 메시지만 보고도 행위의 이해가 가능한 자기 서술성 구조를 지니고 있다.

클라이언트-서버 구조(Client-Server)

: REST 서버는 API를 제공하고 제공된 API를 통해 비즈니스 로직을 처리한다. 클라이언트는 사용자 인증이나 컨택스트등을 관리만 함으로서 각자의 역할이 구분되어지고 개발 관점에서 서로의 의존성이 줄어 분업화가 일어나게 된다.

계층형 구조(Layered System)

: 클라이언트 입장에선 REST API만 호출하며 REST 서버 입장에서 다중 계층이 구성된다. 비즈니스 로직 계층, 사용자 인증, 암호화, 로드 밸런싱 등을 지원하는 다양한 계층 등이 추가되어 유연한 계층형 구조를 이루게 된다.

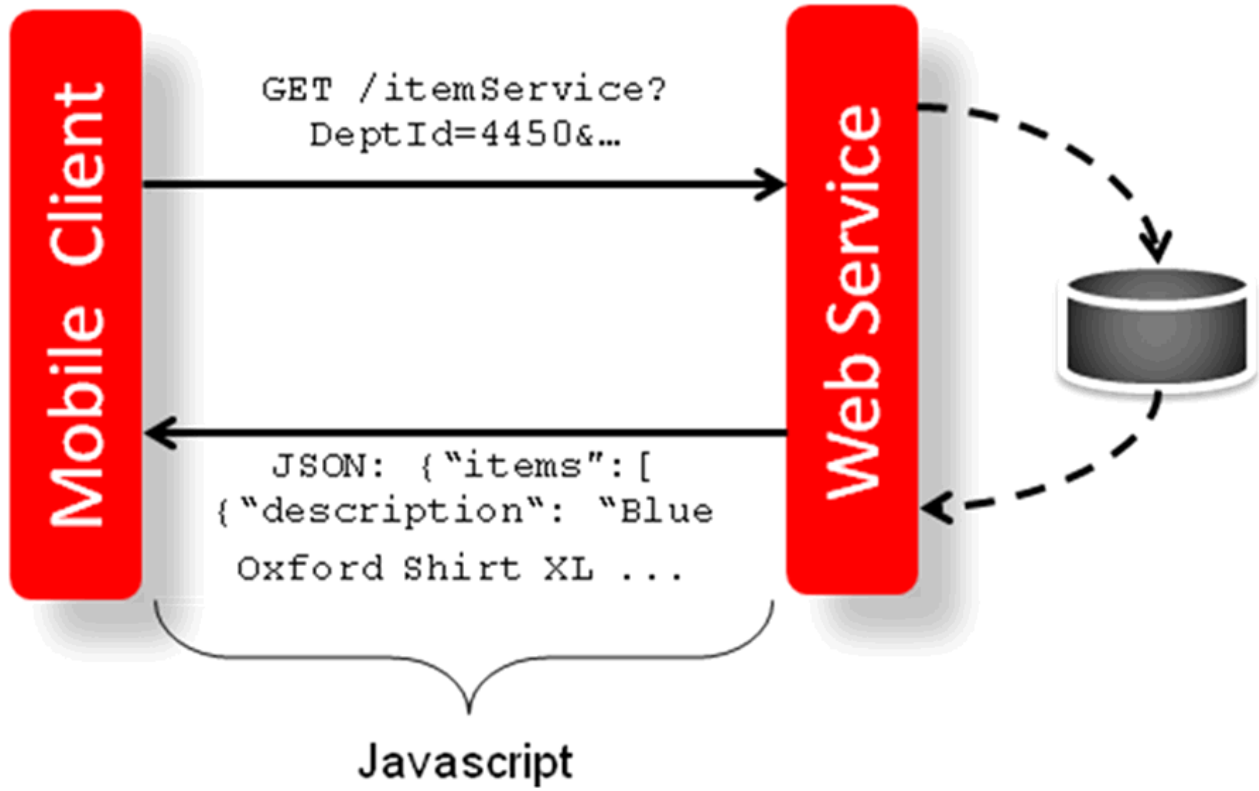
{REST} 단점

표준이 존재하지 않음.
: 기본적인 형식은 정해져 있으나 명확한 표준이 없다.
이 때문에 RESTful이란 용어가 나왔다고 볼 수 있다.

사용 가능한 메소드가 4개밖에 없다.
: CRUD에 해당하는 4개의 메소드만 지원함으로 더 다양한 기능을 표현하고 싶을 때 다소 무리가 있을 수 있다. 이를 해결하기 위해 SOAP를 사용하게 된다.

구형 브라우저의 메소드 지원 문제.
: 구형 브라우저에선 PUT, DELETE 메소드와 pushState를 지원하지 않는다.
이는 REST API의 원활한 작동을 방해한다.
* PUT, DELETE의 지원은 Overloaded POST를 통해 해결이 가능하다.

RESTful?



REST 아키텍쳐 구조에 따라 만들어진 어플리케이션이 RESTful한 어플리케이션이다.

만약 REST 아키텍쳐 구조가 제대로 지켜지지 않는다면 RESTful하다고 할 수 없다.

{REST:API}

REST 기반으로 서로 정보를 교환가능 하도록 API를 구성한 것.

많은 Open API들이 REST API로 개발되어있으며
MSA(마이크로 서비스 아키텍쳐)를 구성 시에도 대부분 REST의
특징을 활용하여 REST API로 개발한다.

REST API는 설계 시 REST보다 많은 설계 규칙이 있다.

{REST:API} 특징

REST 기반의 API 분산으로 확장성과 재사용성 증대.

: 기존의 시스템들 또한 REST API로 기능을 분산함으로서 확장성 및 재사용성을 증대시킬 수 있고 유지보수 및 운용의 편리성도 제공이 된다.

유니폼 인터페이스(Uniform Interface)에 따라 다양한 사용 가능.

: HTTP 메소드 표준을 기반하여 구현하므로 HTTP 표준만 지원한다면 어떠한 언어로든 클라이언트 및 서버를 구현할 수 있다. 즉 REST API를 만들어놓으면 웹 뿐만 아니라 델파이, C#으로도 클라이언트 개발이 가능하다.

REST API 기본 설계 규칙 존재.

: REST의 기본 규칙에 더불어 추가적인 설계 규칙이 존재한다.
또한 도큐먼트, 컬렉션, 스토어라는 추가적인 개념이 등장하여 사용된다.

RESTful Quiz

로그인, 로그아웃 시 RESTful한 방식의 메소드는 무엇이 좋을까?

감사합니다.

– 출처

[Network] REST란? REST API란? RESTful이란?
https://gmlwjd9405.github.io/2018/09/21/rest-and-restful.html

REST API의 이해와 설계#1-개념 소개
https://bcho.tistory.com/953#recentEntries

[REST API] LogIn GET vs POST
https://velog.io/@jch9537/REST-API-LogIn-GET-vs-POST