


REST API

배정	 Lisa Yeon@이건우
상태	8월
속성	@2021년 8월 15일 오후 4:51
언어	

- 발표 당시의 상황

HTTP의 장점을 최대한 활용 할 수 있는 아키텍처로 REST를 소개했고 이는 HTTP 프로토콜을 의도에 맞게 디자인 하도록 유도하고 있음. REST의 기본 원칙을 성실히 지킨 서비스 디자인을 'RESTful'이라 표현했다.

▼ 보충설명

"어떻게 인터넷에서 정보를 공유할 것인가?"

이에 대한 해답으로 "웹"이 출범하게 된다. 이에 팀 버너스 리의 답은 아래와 같다

정보들을 하이퍼 텍스트로 연결한다.

- 표현 형식 : HTML
- 식별자 : URI
- 전송방법 : HTTP

그래서 이제 HTTP라는 프로토콜을 여러 사람들이 설계를 하게 되었다. 그 중에 1명, 대학원생이었던 로이 필딩(Roy T. Fielding)이라는 사람이 이 프로토콜 작업에 참여하게 된다.

그 와중에 고민이 생긴다. 이미 94년도에 로이는 http 1.0 작업에 참여했다. 이 명세가 나오기 전에 이미 http는 당연히 www의 전송 프로토콜로서 이용이 되고 있었다. 그리고 또한 웹은 이미 급속도로 성장하는 도중이었다.

이 시점에서 로이는 http를 정립하고 이 명세에 기능을 더하고 기존의 기능을 고쳐야하는 상황에 놓이게 된다. 그러나 무작정 http 프로토콜을 고치게 된다면, 기존 구축된 웹하고 호환이 안되는 가능성이 존재 했다. 이에 로이는 고민을 한다.

"How do I improve HTTP without breaking the Web?"

"웹을 망가뜨리지 않고 어떻게 http 기능을 증가시킬 수 있을까?"

로이는 고민 끝에 HTTP Object Model이라는 것을 만든다. 아직 REST는 아니다. 이는 4년 후 "Representational State Transfer: An Architectural Style for Distributed Hypermedia interaction"에서 REST를 최초로 공개한다. 이후 2년 후, "Architectural Styles and the Design of Network-based Software Architectures"을 박사 논문으로 발표하게 된다. 이 박사 논문이 바로 그 REST라는 것을 정의한 논문이다.

출처 : <https://velog.io/@kjh03160/그런-REST-API로-괜찮은가>

▼ REST VS REST API

REST는 REpresentational State Transfer의 약자이다. 'interoperability' 는 상호 운용성이라는 뜻으로, 컴퓨터 시스템과 인터넷 사이에 상호 운용성을 제공하는 방법이라고 한다.

HTTP를 기반으로 클라이언트가 서버의 리소스에 접근하는 방식을 규정한 아키텍처, REST API는 REST를 기반으로 서비스 API를 구현한것을 의미한다.

▼ REST API의 구성(더 자세한 설명)

자원(resource), 행위(verb), 표현(representations) 3가지 요소로 구성됨.

REST는 자체 표현 구조로 구성되어있다. → REST API만으로 HTTP 요청 내용을 이해할 수 있다.

Aa 구성요소	≡ 내용	≡ 표현 방법
<u>자원</u>	자원	URI(엔드포인트)
<u>행위</u>	자원에 대한 행위	HTTP 요청메서드
<u>표현</u>	자원에 대한 행위의 구체적내용	페이로드

▼ REST API 설계원칙

1. URI는 리소스를 표현해야한다.

리소스를 표현하는데 중점을 뒀다. 리소스를 식별 할 수 있는 이름은 '동사' 보다 '명사'를 사용한다. 이름에 get과 같은 행위에 대한 표현이 들어가서는 안된다.

```
# BAD
GET /getTodos/1
GET /todos/show/1

# Good
GET /todos/1
```

2. 리소스에 대한 행위는 HTTP 요청 메서드로 표현한다.

HTTP 요청 메서드는 클라이언트가 서버에 요청 종류와 목적(리소스에 대한 행위)을 알리는 방법이다. 주로 5가지 요청메서드(GET, POST, PUT, PATCH, DELETE)를 사용하여 CRUD를 구현한다.

메서드 표현 정리표

Aa HTTP 요청 메서드	≡ 종류	≡ 목적	≡ 페이로드
<u>GET</u>	index/retrieve	모든/특정 리소스 취득	X
<u>POST</u>	create	리소스 생성	O
<u>PUT</u>	replace	리소스의 전체교체	O
<u>PATCH</u>	modify	리소스의 일부수정	O
<u>DELETE</u>	delete	모든/특정 리소스 삭제	X

▼ 참고, REST API의 설계기준 (출처, 레드햇)

API가 RESTful로 간주되려면 다음 기준을 따라야 합니다.

- 클라이언트, 서버 및 리소스로 구성되었으며 요청이 HTTP를 통해 관리되는 클라이언트-서버 아키텍처
- 스테이트리스(stateless). 클라이언트-서버 커뮤니케이션: 요청 간에 클라이언트 정보가 저장되지 않으며, 각 요청이 분리되어 있고 서로 연결되어 있지 않음
- 클라이언트-서버 상호 작용을 간소화하는 캐시 가능 데이터
- 정보가 표준 형식으로 전송되도록 하기 위한 구성 요소 간 통합 인터페이스. 여기에 필요한 것은 다음과 같습니다.
 - 요청된 리소스가 식별 가능하며 클라이언트에 전송된 표현과 분리되어야 합니다.
 - 수신한 표현을 통해 클라이언트가 리소스를 조작할 수 있어야 합니다(이렇게 할 수 있는 충분한 정보가 표현에 포함되어 있기 때문).

- 클라이언트에 반환되는 자기 기술적(self-descriptive) 메시지에 클라이언트가 정보를 어떻게 처리해야 할지 설명하는 정보가 충분히 포함되어야 합니다.
- 하이퍼미디어: 클라이언트가 리소스에 액세스한 후 하이퍼링크를 사용해 현재 수행 가능한 기타 모든 작업을 찾을 수 있어야 합니다.
- 요청된 정보를 검색하는 데 관련된 서버(보안, 로드 밸런싱 등을 담당)의 각 유형을 클라이언트가 볼 수 없는 계층 구조로 체계화하는 계층화된 시스템.
- 코드 온디맨드(선택 사항): 요청을 받으면 서버에서 클라이언트로 실행 가능한 코드를 전송하여 클라이언트 기능을 확장할 수 있는 기능.

그외 참고하면 좋을 자료

<https://meetup.toast.com/posts/92>