REST

REST 개요

- REST 정의
  - REST는 Representational State Transfer의 약자이다.
  - 자원을 이름(자원의 표현)으로 구분하여 해당 자원의 상태(정보)를 주고받는 모든 것을 의미한다.
    - ✓ 자원(resource) : 해당 소프트웨어가 관리하는 모든 것 (문서, 그림, 데이터 등)
    - ✓ 표현(representation) : 자원을 표현하기 위한 이름 (DB의 학생정보를 students로 표현)
    - ✓ 상태(State) : 자원이 요청될 때 자원의 상태(정보)를 전달한다. (JSON, XML 등으로 전달)
  - REST는 네트워크 상에서 Client – Server 구조의 통신 방식 중 하나다.
  - REST는 기본적으로 웹의 기존 기술과 HTTP 프로토콜을 그대로 활용한다.

- REST 구체적인 개념
  - HTTP URI(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고, HTTP Method(GET, POST, PUT 등)을 통해 해당 자원에 대한 CRUD Operation을 적용하는 것을 의미한다.
  - CRUD Operation
    - ✓ Create : 생성 (POST)
    - ✓ Read : 조회 (GET)
    - ✓ Update : 수정 (PUT)
    - ✓ Delete : 삭제 (DELETE)
    - ✓ HEAD : header 정보 조회 (HEAD)
  - 즉, REST는 자원 기반 구조(ROA, Resource Oriented Architecture) 설계의 중심에 Resource가 있고 HTTP Method를 통해 Resource를 처리하도록 설계된 아키텍처를 의미한다.
  - 웹 사이트의 이미지, 텍스트, DB내용 등의 모든 자원에 고유한 ID인 HTTP URI를 부여한다.

REST 장단점

- 장점
  - HTTP 프로토콜 인프라를 그대로 사용하므로 REST API 사용을 위한 인프라를 구축할 필요가 없다.
  - HTTP 프로토콜을 따르는 모든 플랫폼에서 사용이 가능하다.
  - Hypermedia API의 기본을 충실히 지키면서 범용성을 보장한다.
  - REST API 메시지가 의도하는 바를 명확하게 나타내므로 의도하는 바를 쉽게 파악할 수 있다. (URI)
  - Server와 Client의 역할을 명확하게 분리한다.

- 단점
  - REST API에 대한 표준이 존재하지 않는다.
  - 구형 브라우저가 아직 제대로 지원해주지 못하는 부분이 존재한다.

REST 구성 요소

- 자원(Resource) : URI
  - 모든 자원에 고유한 ID가 존재하고, 이 자원은 Server에 존재한다.
  - 자원을 구별하는 ID는 '/groups/people/:people_num'과 같은 HTTP URI이다.
  - Client는 URI를 이용해서 자원을 지정하고 해당 자원의 상태(정보)에 대한 조작을 Server에 요청한다.
  - URL은 'http://host_ip:port/grouts/people...'과 같이 서버의 프로토콜, 주소, URI 등 모두를 의미한다.

- 행위(Verb) : HTTP Method
  - HTTP 프로토콜의 Method를 사용한다.
  - HTTP 프로토콜은 GET, POST, PUT, DELETE와 같은 Method를 기본 제공한다.

- 표현(Representation of Resource)
  - Client가 자원의 상태(정보)에 대한 조작을 요청하면 Server는 이에 적절한 Representation(표현)을 보낸다.
  - REST에서 하나의 자원은 JSON, XML, TEXT, HTML 등 여러 형태의 표현으로 나타내어 질 수 있다.

REST API

- API
  - API는 Application Programming Interface의 약자이다.
  - 데이터와 기능의 집합을 제공하여 컴퓨터 프로그램간 상호작용을 촉진하여, 서로 정보를 교환 가능하도록 하는 것

- REST API
  - REST 기반의 서비스 API
  - 최근 OpenAPI, 마이크로 서비스 등을 제공하는 업체 대부분은 REST API를 제공한다.

- **REST API 설계 규칙**
  - 슬래시 구분자(/)는 계층 관계를 나타내는데 사용한다. (/houses/apartments)
  - URI 마지막 문자로 슬래시(/)를 포함하지 않는다. (/houses/apartments/ <= x)
  - 하이픈(-)은 URI 가독성을 높이는데 사용한다.
  - 언더바(_)는 URI에 사용하지 않는다.
  - URI 경로에는 소문자가 적합하다.
  - 파일확장자는 URI에 포함하지 않는다.
  - REST API 설계 예시

CRUD	HTTP verbs	Route
resource들의 목록을 표시	GET	/resource
resource 하나의 내용을 표시	GET	/resource/:id
resource를 생성	POST	/resource
resource를 수정	PUT	/resource/:id
resource를 삭제	DELETE	/resource/:id

- HTTP 응답상태 코드
  - 1xx : 전송 프로토콜 수준의 정보 교환
  - 2xx : 클라이언트 요청이 성공적으로 수행 됨
  - 3xx : 클라이언트 요청을 완료하기 위해 추가적인 행동을 취해야 함
  - 4xx : 클라이언트의 잘못된 요청
  - 5xx : 서버 오류로 인한 상태코드

RESTful

- RESTful
  - RESTful은 일반적으로 REST라는 아키텍처를 구현하는 웹 서비스를 나타내기 위해 사용되는 용어이다.
  - 즉, REST API를 제공하는 웹 서비스를 "RESTful"하다고 할 수 있다.
  - 다시 말해 REST 원리를 따르는 시스템은 RESTful이란 용어로 지칭된다.

- RESTful 목적
  - 이해하기 쉽고 사용하기 쉬운 REST API를 만드는 것
  - RESTful한 API의 근본적인 목적은 일관적인 컨벤션을 통한 API의 이해도 및 호환성을 높이는 것이 주 동기이다.
  - 성능이 중요한 소프트웨어에서는 굳이 RESTful한 API를 구현할 필요가 없다.

- RESTful 하지 못한 경우
  - CRUD 기능을 모두 POST로만 처리하는 API
  - route에 resource, id외의 정보가 들어가는 경우 (/students/updateName)

참고문헌

- heejeong Kwon (2018). https://gmlwjd9405.github.io/2018/09/21/rest-and-restful.html