# REST API

Kim Hye Kyung

topickim@naver.com

# REST API[Representational State Transfer]

REST란?

**웹에 존재하는 모든 자원(이미지, 동영상, DB 자원)에 고유한 URI를 부여해 활용하는 것**

**하나의 URI는 고유한 리소스와 연력되며 이 리소스는 GET/POST/PUT/DELETE등의**

**HTTP 메  소드로 제어하자는 개념**

Restful API란?

REST 특징을 지키면서 API를 제공하는 것을 의미

REST 형태로 사용자가 원하는 기능을 제공하는 것의 의미

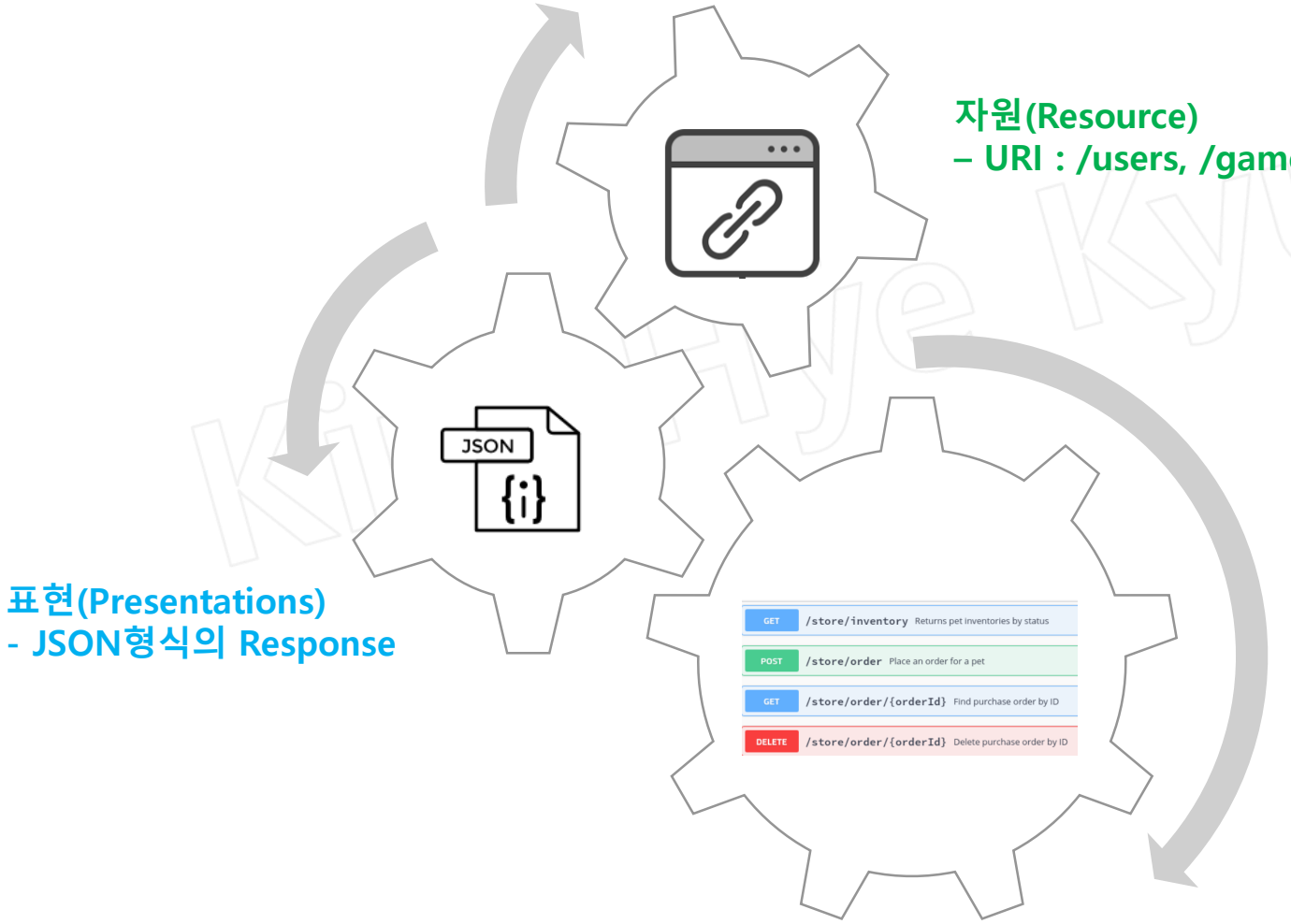
예시 : 많이 활용되는 공공 API들이 대표적

REST 방식으로 서비스를 제공하는 것을

**Restful 하다**라고 표현

# REST API[Representational State Transfer]

- 2000년도에 로이 필딩(Roy Fielding)의 박사학위 논문에서 최초로 소개




**자원(Resource)**
**– URI : /users, /games, /lists와 같은 URI**

**표현(Presentations)**
**- JSON형식의 Response**

행위(Verb)
– HTTP Method : GET, POST, PUT, DELETE와 같은 HTTP Method

# REST API 디자인 가이드

- REST API 설계 시 가장 중요한 항목

URI는 **정보의 자원을 표현해야 함**

자원에 대한 행위는 HTTP Method(GET, POST, PUT, DELETE)로 표현

# RESTful 이란?

- Representational State Transfer
  - 웹의 장점을 최대한 활용할 수 있는 아키텍처
  - 최근의 서버 프로그램은 다양한 브라우저와 안드로이폰, 아이폰과 같은 모바일 디바이스에서도 통신을 할 수 있어야 함
  - REST 아키텍처는 Hypermedia API의 기본을 충실히 지키면서 범용성을 보장



**RESTful이란 REST 원리를 따르는 시스템을 지칭하는 용어**

# HTTP Method 종류 및 특징

- Self-descriptiveness (자체 표현 구조)
  - REST의 또 다른 큰 특징 중 하나는 **REST API 메시지만 보고도 이를 쉽게 이해 할 수 있는 자체 표현 구조로 되어 있음**
- POST, GET, PUT, DELETE 4가지의 Method로 CRUD 수행

| METHOD | 기 능 |
|---|---|
| POST | POST를 통해 해당 URI를 요청하면 리소스를 **생성** or 수정 |
| GET | GET을 통해 해당 리소스 조회<br>리소스를 조회하고 해당 도큐먼트에 대한 정보를 가져옴 |
| PUT | 리소스 **수정** or 생성 |
| DELETE | 리소스 삭제 |

# REST API 중심 규칙

- URI는 정보의 자원을 표현해야 함
  - **리소스명**은 동사보다는 **명사**를 사용

GET /members/**delete**/1

REST를 제대로 적용하지 않은 URI

**GET 즉 정보 요청 방식과 delete 삭제 요청 logic 로직은 의미론 적으로 부적합**

**자원에 대한 행위는 HTTP Method(GET, POST, PUT, DELETE등)로 표현**

**GET** /members/1

# REST API 중심 규칙

- **회원정보를 가져올 때는 GET, 회원 추가 시의 행위를 표현하고자 할 때는 POST** METHOD를 사용하여 표현

GET 만으로 정보 요청
따라서 show 문구 불필요

GET /memers/**show**/1 - 부적합

**GET** /members/1 - 적합

**GET** /memers/**insert**/2 - 부적합

**POST** /members/2 - 적합

POST 만으로 정보 저장 요청
따라서 insert 문구 불필요

# URI 설계 시 주의할 점

슬래시 구분자(/)는 계층 관계를 나타내는 용도로 사용

URI 마지막 문자로 슬래시(/)를 포함하지 않음

하이픈(-)은 URI 가독성을 높이는데 사용

밑줄(_)은 URI에 사용하지 않음

URI 경로에는 소문자가 적합

파일 확장자는 URI에 포함시키지 않음

# URI 설계 시 주의할 점

슬래시 구분자(/)는 계층 관계를 나타내는 데 사용

- http://restapi.example.com/houses/apartments
- http://restapi.example.com/animals/mammals/whales

URI 마지막 문자로 슬래시(/)를 포함하지 않음

- http://restapi.example.com/houses/apartments**/** (X)
- http://restapi.example.com/houses/apartments (0)
  - URI에 포함되는 모든 글자는 리소스의 유일한 식별자로 사용되어야 하며 URI가 다르다는 것은 리소스가 다름을 의미
  - 역으로 리소스가 다를 경우 반드시 URI도 달라져야 함
  - REST API는 분명한 URI를 만들어 통신을 해야 하기 때문에 혼동을 주지 않도록 URI 경로의 마지막에는 슬래시(/)를 사용하지 않음

# URI 설계 시 주의할 점

하이픈(-)은 URI 가독성을 높이는데 사용

- URI를 쉽게 읽고 해석하기 위해, 불가피하게 긴 URI경로를 사용하게 된다면 하이픈을 사용해 가독성을 높일 수 있음

밑줄(_)은 URI에 사용하지 않음

- 글꼴에 따라 다르긴 하지만 밑줄은 보기 어렵거나 밑줄 때문에 문자가 가려지는 현상 발생
- 밑줄 대신 하이픈(-)을 사용하는 권장(가독성)

# URI 설계 시 주의할 점

URI 경로에는 소문자가 적합

- URI 경로에 대문자 사용은 비권장
- 대소문자에 따라 다른 리소스로 인식하게 되기 때문
- RFC 3986(URI 문법 형식)은 URI 스키마와 호스트를 제외하고는 대소문자를 구별하도록 규정하기 때문

# URI 설계 시 주의할 점

파일 확장자는 URI에 포함시키지 않음

- REST API에서는 메시지 body 내용의 포맷을 나타내기 위한 파일 확장자를 URI 안에 포함시키지 않고, Accept header 사용하여 표현
  - Access header란?
    - 클라이언트가 자신이 선호하는 media type을 서버에 보내는 것
  - http://restapi.example.com/members/soccer/345/photo.jpg **(X)**
  - GET / members/soccer/345/photo HTTP/1.1 Host: restapi.example.com Accept: image/jpg **(0)**

# 리소스 간의 관계를 표현하는 방법

- REST 리소스 간에는 연관 관계가 있을 수 있고, 이런 경우 다음과 같은 표현 방법으로 사용

- /리소스명/리소스 ID/관계가 있는 다른 리소스명
  - GET : /users/**{userid}/devices**
    - 일반적으로 소유 'has'의 관계를 표현할 때

- 만약에 관계명이 복잡하다면 이를 서브 리소스에 명시적으로 표현
  - 예를 들어 사용자가 '**좋아하는' 디바이스 목록을** 표현해야 할 경우 다음과 같은 형태로 사용
  - GET : /users**/{userid}/likes/devices**
    - 관계명이 애매하거나 구체적 표현이 필요할 때 권장

# 실습을 위한 필요사항

- JSON 데이터 제공하는 온라인 사이트
  - https://jsonplaceholder.typicode.com/posts/1/comments
  - https://jsonplaceholder.typicode.com/albums/1/photos
  - https://jsonplaceholder.typicode.com/users/1/albums
  - https://jsonplaceholder.typicode.com/users/1/todos
  - https://jsonplaceholder.typicode.com/users/1/posts

  - https://api.androidhive.info/contacts/

- JSON viewer
  - https://jsonformatter.org/json-viewer

- Test tool
  - Post Man

# 참조

- https://meetup.toast.com/posts/92
- https://blog.remotty.com/blog/2014/01/28/lets-study-rest/