SEEDS SEASON2

Seeds
FIRST STEP
-2 week-

Seeds운영진 **김민성**

INDEX

01 REST

- REST(Representational State Transfer)의 약자로 자원을 이름(자원의 표현)으로 구분하여 해당 자원의 상태(정보)를 주고 받는 모든 것을 의미한다.

- 즉, 자원(Resource)의 표현 (Representation)에 의한 상태 전달

01 REST

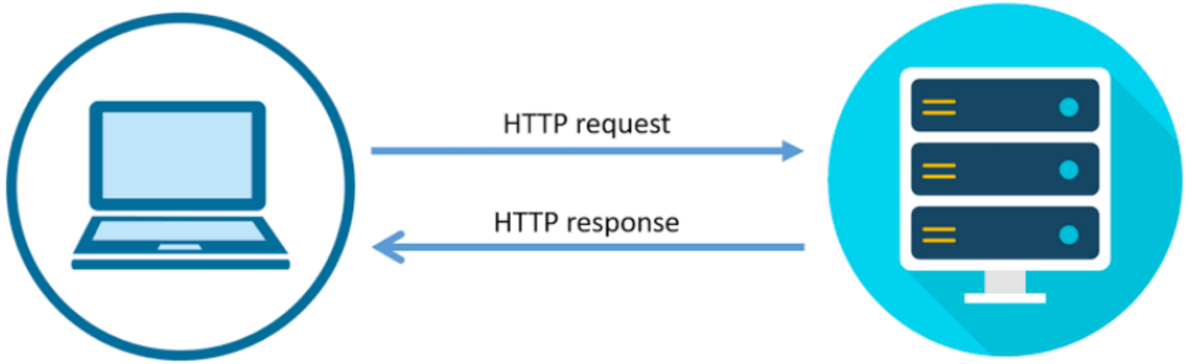
- **REST에서 얘기하는 자원은 무엇일까?**
  - 해당 소프트웨어가 관리하는 모든 것을 의미함.
    - ex) 문서, 그림 , 데이터, 해당 소프트웨어 자체 등

- **자원의 표현은 어떻게 할까?**
  - DB의 학생 정보가 자원일때, 'students'를 자원의 표현으로 정함

- **자원의 상태 전달은 어떻게 할까?**
  - 데이터가 요청되어 지는 시점에서 자원의 상태(정보)를 전달
    - 자원의 상태를 주고 받았다는 것은 결국 데이터를 주고 받았다는것과 같음
    - 일반적으로 JSON 혹은 XML을 통해 데이터를 주고받는다.

01 REST

- REST는 기본적으로 웹의 기존 기술과 HTTP 프로토콜을 그대로 활용하기 때문에 웹의 장점을 최대한 활용할 수 있는 아키텍처 스타일
- REST는 네트워크 상에서 Client와 server 사이의 통신 방식중 하나
- 월드 와이드 웹(WWW)과 같은 분산 하이퍼미디어 시스템을 위한 소프트웨어 개발 아키텍처의 한 형식



01 REST의 구체적인 개념

- HTTP URI(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고, HTTP Method(POST, GET , PUT , DELETE)를 통해 해당 자원에 대한 CRUD Operation을 적용하는 것을 의미

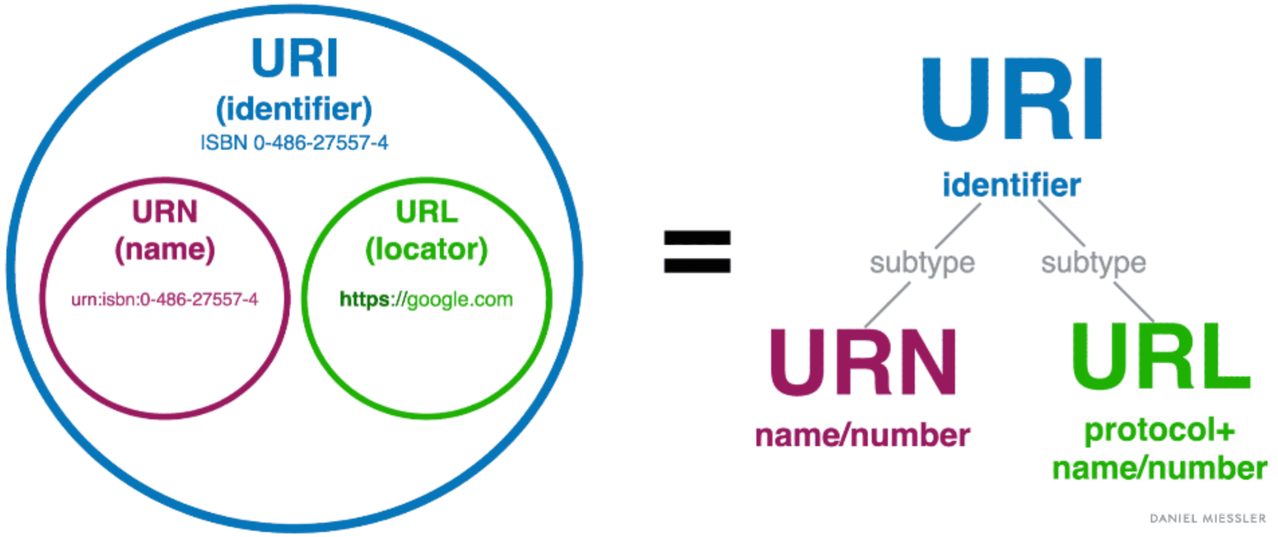
- CRUD Operation이란?
  - Create: 생성(POST)
  - Read : 조회(GET)
  - Update : 수정(PUT)
  - Delete : 삭제 (DELETE)
  - Head : 헤더 정보 조회(HEAD)

CRUD	HTTP	REST
Create	POST	/api/movie
Read	GET	/api/movie/{id}
Update	PUT	/api/movie
Delete	DELETE	/api/movie/{id}

01 REST

- 즉, REST는 자원 기반의 구조 (ROA, Resource Oriented Architecture) 설계의 중심에 Resource가 있고 HTTP Method를 통해 Resource를 처리하도록 설계된 아키텍쳐를 의미

- 웹 사이트의 이미지, 텍스트 , DB 내용 등의 모든 자원에 고유한 ID인 HTTP URI를 부여함



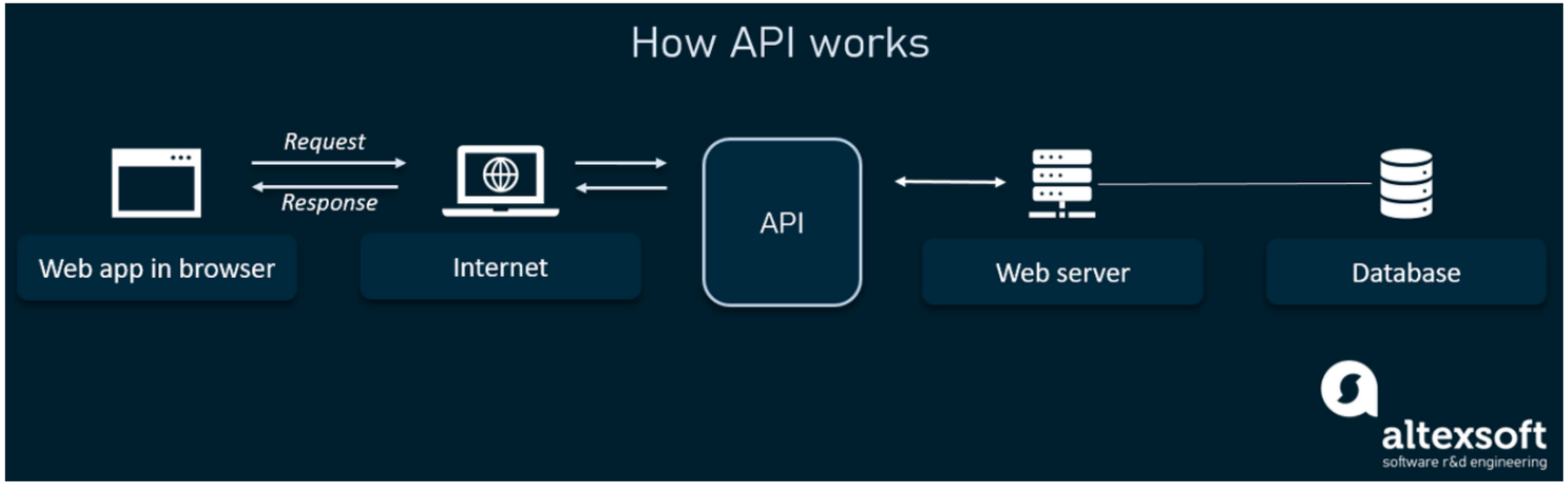
02 REST API

- **API(Application Programming Interface)**
  - 데이터와 기능의 집합을 제공하여 컴퓨터 프로그램간 상호작용을 촉진하며, 서로 정보를 교환 가능하도록 하는 것

- **REST API**
  - REST 기반으로 서비스 API를 구현하는 것

02 REST API의 특징

- REST 기반으로 시스템을 분산해 확장성과 재사용성을 높여 유지 보수 및 운영을 편리하게 할 수 있다.
- REST는 HTTP 표준을 기반으로 구현하므로, 클라이언트 – 서버구조를 구현할 수 있다.



02 REST API설계 원칙

- 자원에 대한 행위는 HTTP Method로 표현한다.
  - URI에 HTTP Method가 들어가면 안된다.
    - ex) GET /members/delete/1 -> DELETE /members/1

- URI에 행위에 대한 동사 표현이 들어가면 안된다. 즉, CRUD 기능을 나타내는 것은 URI에 사용하지 않는다.
  - ex) GET /members/show/1 -> GET /members/1
  - ex) GET /members/insert/2 -> POST /members/2

- URI 중 변하는 부분은 유일한 값으로 대체한다. 즉, id는 하나의 특정 resource를 나타내는 고유값이다.
  - student를 생성하는 경로: POST /students
  - id = 12인 student를 삭제하는 경로: DELETE /students/12

02 REST API설계 원칙

- 슬래시 구분자(/)는 계층 관계를 나타내는데 사용한다
  - ex) http://restapi.example.com/houses/apartments
  - http://restapi.example.com/members/students

- URI 마지막 문자로 슬래시(/)를 포함하지 않는다.

  - URI에 포함되는 모든 글자는 resource의 유일한 식별자로 사용되어야 하며 URI가 다르다는 것은 resource가 다르다는 것이고, 역으로 resource가 다르면 URI도 달라져야한다.

  - REST API는 분명한 URI를 만들어 통신을 해야 하기 때문에 혼동을 주지 않도록 URI경로의 마지막에는 슬래시(/)를 사용하지 않는다.
    - ex) http://restapi.example.com/houses/apartments/ (x)

02 REST API설계 원칙

- 하이픈(-)은 URI 가독성을 높이는데 사용한다.
  - 불가피하게 URI 경로를 사용하게 된다면 하이픈을 사용해 가독성을 높인다.
    - ex) http://api.example.com/blogs/guy-levin/posts/this-is-my-first-post
- 밑줄(_)은 URI에 사용하지 않는다.
- URI 경로에는 소문자가 적합하다.
- 파일 확장자는 URI에 포함하지 않는다.

02 REST API설계 원칙

- 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*

02 REST API설계 원칙

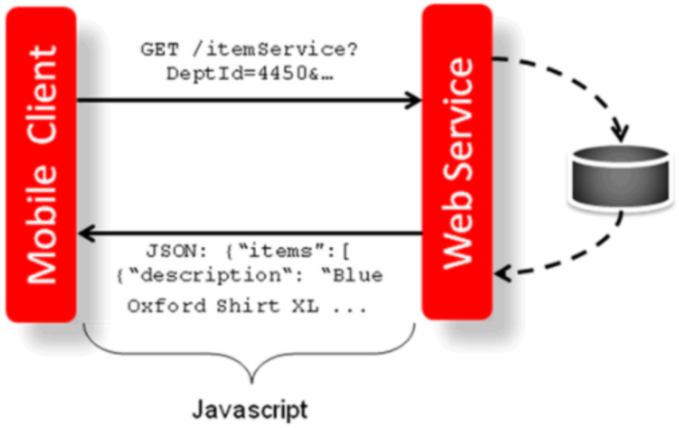
- 응답 상태 코드
  - 1xx: 전송 프로토콜 수준의 정보 교환
  - 2xx: 클라이언트 요청이 성공적으로 수행됨
  - 3xx: 클라이언트는 요청을 완료하기 위해 추가적인 행동을 취해야함
  - 4xx: 클라이언트의 잘못된 요청
  - 5xx: 서버쪽 오류

03 RESTful

- RESTful은 일반적으로 REST라는 아키텍쳐를 구현하는 웹서비스를 나타내기 위해 사용되는 용어이다.
  - 'REST API' 를 제공하는 웹 서비스를 'RESTful' 하다고 할 수 있다.

- RESTful은 REST를 REST답게 쓰기 위한 방법으로 누군가가 공식적으로 발표한 것이 아니다.
  - 즉, REST 원리를 따르는 시스템은 RESTful이란 용어로 지칭된다.

RESTful의 개념



03 RESTful의 목적

- 이해하기 쉽고 사용하기 쉬운 REST API를 만드는 것.

- RESTful한 API를 구현하는 근본적인 목적이 성능 향상에 있는 것이 아니라 일관적인 컨벤션을 통한 API의 이해도 및 호환성을 높이는 것이 주 동기이니, 성능이 중요한 상황에서는 굳이 RESTful한 API를 구현 할 필요는 없다.

03 RESTful하지 못한 경우

- ex) CRUD 기능을 모두 POST로만 처리하는 API
- ex) 경로에 resource, id 이외의 정보가 들어가는 경우
  - /students/update-name

03 RESTful 장점

- 정해진 규칙대로 routing 주소를 만들기 때문에 route 이름을 짓는 수고를 덜 수 있고, 통일성이 있다
- API의 확장이 쉽다.
- 즉, 통일성과 확장성이 RESTful API의 장점이다. 정해진 규칙에 따라 API주소, request 구조 return 구조를 만들기 때문에 개발팀이 바뀌거나 하는 경우에도 혼란을 줄일 수 있다.

04 REST API 사용

ex) GET예제

```
import requests

def get_user_info(user_id):
    response = requests.get(f'https://jsonplaceholder.typicode.com/users/{user_id}')
    #1번 사용자의 정보를 조회하기 때문에 requests.get 사용
    if response.status_code == 200:
        user_data = response.json() #user 정보를 딕셔너리 형태로 응답 받음
        print(f"User ID: {user_data['id']}")
        print(f"User Name: {user_data['name']}")
        print(f"User Email: {user_data['email']}")
        print(f"User address : {user_data['address']}")
    else:
        print("유저 정보를 조회하는 도중 오류가 발생하였습니다.")
get_user_info(1) # 1번 사용자의 정보를 가져옴
```

04 REST API 사용

ex) POST 예제

```
import requests
import json #json 데이터를 다루기 위한 모듈

def create_post(title, body, user_id):
    url = 'https://jsonplaceholder.typicode.com/posts'
    headers = {'Content-type': 'application/json; charset=UTF-8'} #header 정의
    data = {
        'title': title,
        'body': body,
        'userId': user_id
    }
```

*content-type이란 간단히 말해 보내는 자원의 형식을 명시하기 위해 헤더에 실리는 정보 이다.

04 REST API 사용

ex) POST 예제

```
response = requests.post(url , headers=headers , data=json.dumps(data))
if response.status_code == 201:
        post_data = response.json()
        print(f"Post ID: {post_data['id']}") # post_data에서 id를 가져옴
        print(f"Post Title: {post_data['title']}") # post_data에서 title을 가져옴
        print(f"Post body: {post_data['body']}") # post_data에서 body를 가져옴
    else:
        print("게시물을 생성할 수 없습니다.")

create_post('Test2  Title', 'Test  Body2', 1) # 새 게시물 생성
```

*딕셔너리 자료형을 JSON 문자열로 만들려면 json.dumps() 함수를 사용하면 된다.

과제

PUT 과 DELETE 부분을 구현해 오세요.