1. **HTTP에 대한 이해**

▶**HTTP(HyperText Transfer Protocal)이란?**

: 인터넷 상에서 데이터를 주고 받기 위해 사용하는 하이퍼텍스트기반 프로토콜,

최근에는 HTML뿐 아니라 모든 관련 API통신에 이용하고 있는 통신 프로토콜이다.

텍스트, 점수판이(가) 표시된 사진

자동 생성된 설명

네트워크 전송 시 데이터표준을 정리한 것이 OSI 7계층, 이를 인터넷 표준으로 정의한 것이 TCP/IP 4계층이다. HTTP프로토콜은 응용계층 레벨의 프로토콜로 TCP/IP 위에서 작동한다.

**▶ 특징**

1. 비연결성: 처음 연결을 맺은 후 요청(Request)과 한번의 응답(Response)이후 연결이 종료된다. 매 요청마다 다시 연결을 맺는다.
2. 무상태성: 프로토콜에서 Client의 상태를 기억하지 않는다. Client의 상태를 보관하기 위해 쿠키나 세션, JWT토큰 등을 이용하여 Client의 상태를 유지한다.

**▶ 구조**

텍스트이(가) 표시된 사진

자동 생성된 설명

**<요청(Requests) 메시지>**

start-line / header/  empty-line/  body로 구성

\* start-line

: HTTP METHOD와 URL, 그리고 protocol version 에 대해서 기술

(어떤 요청을 하는지에 대한 내용이 담겨져 있음)

1. HTTP METHOD: GET, POST, PATCH, PUT, DELETE 와 같은 http method를 적는 부분

(READ CREATE UPDATE(part) UPDATE(all) DELETE를 의미)

1. URL: 리소스를 요청하는 하는 주소를 의미
2. protocol version: http의 버전을 의미. HTTP/1.1을 가장 많이 사용

GET /images/logo.gif HTTP/1.1

1. HTTP Method: 해당 요청이 의도한 액션을 정의. GET, POST, DELETE가 주로 많이 쓰임.

2. Request Target: 해당 request가 전송되는 목표 url.

3. HTTP Version: 사용되는 HTTP 버전. 주로 1.1 버전 사용.

--> GET 메소드로 images/logo.gif 라는 요청 타겟에 HTTP 1.1버전으로 요청 보내겠다는 의미.

\*Header

: 요청에 대한 정보, 응답에 대한 요청, 인증 정보, 접속 정보등 요청에서 필요한 다양한 요소를 담을 수 있는 공간. key, value로 구성됨.

Key : Value 형태로 되어있습니다. (JavaScript의 객체, Python에서의 딕셔너리 형태)

주로 사용되는 Headers의 정보.

Headers: {

Host: 요청을 보내는 타겟의 주소. 요청 보내는 웹사이트의 기본 주소.

User-Agent: 요청을 보내는 클라이언트의 대한 정보 (chrome, safari, firefox, ...)

Content-Type: 해당 요청이 보내는 메시지 body의 타입 (ex.application/json)

Content-Length: body 내용의 길이

Authorization: 회원의 인증/인가를 처리하기 위한 로그인 토큰을 담는 곳.

}

\*empty-line

: Header 와 body를 구분

\*Body  
: 메시지 전문이 들어감. Content-Type와 꼭 Type을 맞춰야하며 Get과 Delete는 body를 쓰지 않음. (주로 Body를 사용하는 메소드는 POST)

Body: {

"user\_email" : "kho5420@gmail.com"

"user\_password" : "password"

}

**<응답(Response) 메시지>**

: 전체적인 구조는 요청과 똑같으나 내용에 있어서 차이가 존재

\*start-line

: protocol version, Http Status Code, Http Status 에 대해서 기술

-Http Status

|  |  |
| --- | --- |
| **1xx(정보)** | 요청을 받았으며 프로세스를 계속 진행합니다. |
| **2xx(성공)** | 요청을 성공적으로 받았으며 인식했고 수용하였습니다. |
| **3xx(리다이렉션)** | 요청 완료를 위해 추가 작업 조치가 필요합니다. |
| **4xx(클라이언트 오류)** | 요청의 문법이 잘못되었거나 요청을 처리할 수 없습니다. |
| **5xx(서버 오류)** | 서버가 명백히 유효한 요청에 대한 충족을 실패했습니다. |

HTTP/1.1 404 Not Found

1. HTTP Version: request와 동일하게 HTTP 버전.

2. Status Code: response 메시지의 상태 코드

3. Status Text: 메시지의 상태를 간략하게 설명해주는 텍스트

--> HTTP 1.1버전으로 응답하고 있으며, 예를 들어 프론트엔드에서 로그인 시도에 대한 요청을 보냈는데 유저 정보를 찾을 수 없기에 404 상태메시지를 보낸다.

\*Header

: 요청과 유사하나 서버에 대한 정보를 건내줌. 사용자의 Cookie나 Session 초기화에도 이용됨.

\*Body  
: 메시지 전문이 들어감. Content-Type와 꼭 Type을 맞춰야하며 Get과 Delete는 body를 쓰지 않음. (가장 많이 사용되는 Body의 데이터 타입은 'JSON')

1. **REST-API에 대한 학습**

**▶ API(**Application Programming Interface**)란?**

**:** 응용 프로그램에서 사용할 수 있도록, 운영 체제나 프로그래밍 언어 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻한다.

**▶ REST (**REpresentational State Transfer**)란?**

**:**  웹상에서 사용되는 여러 리소스를 HTTP URI로 표현하고, 해당 리소스에 대한 행위를 HTTP Method로 정의하는 방식을 말한다. REST의 궁극적인 목표는 성능, 확장성, 단순성, 수정 가능성, 가시성, 이식성 및 안정성을 높이는 것이다.

**▶ REST API 설계 원칙**

1. **균일한 인터페이스(Uniform Interface)**

**:** 동일한 리소스에 대한 모든 API 요청은 출처에 관계없이 동일하게 표시되어야 한다.

클라이언트가 필요로 하는 모든 정보를 포함해야 하지만 리소스는 너무 크지 말아야한다.

1. **클라이언트와 서버 분리**

: REST API에서 서버와 클라이언트는 완전히 독립된 상태여야한다. 클라이언트(응용 프로그램)이 알아야 하는 유일한 정보는 요청된 리소스의 URI여야한다.

다른 방식으로는 서버와 상호작용 할 수 없어야하며, 서버는 HTTP를 통해 요청된 데이터에 전달하는 것 외에는 클라이언트를 수정해서는 안된다.

1. **무상태성(Stateless)**

**:** 서버와 클라이언트 요청 간 클라이언트에 정보가 저장되지 않으며, 그 반대로도 마찬가지이다. 각 요청이 분리되어 서로 연결되어있지 않는다.

1. **주문 형 코드 (Code on demand)**

**:** REST API는 일반적으로 정적인 리소스만 보내지만, 특정 경우 응답으로 실행코드가 포함될 수 있다. 이런 경우 코드는 요청시에만 실행되어야 한다.

1. **캐시 가능성**

**:** 가능한 경우 리소스는 클라이언트 또는 서버 측에서 캐시가 가능해야한다.

1. **계층화 된 시스템 아키텍쳐**

**:** REST API에서의 요청과 응답은 서로 다른 계층을 거친다.

## **REST API 설계 가이드**

**1. URL Rules**

리소스에 대한 행위는 HTTP Method(POST, GET, PUT, DELETE)등을 이용한다.

/(슬래시)는 계층 관계를 나타낸다.

URI에 \_(underbar)를 사용하지 않는다. 영어 대문자보다는 소문자를 사용한다.

URI에 동사는 사용하지 않는다. 명사를 사용한다

URI에 파일의 확장자(.json, jpeg)등을 포함하지 않는다.

**2. Set HTTP Headers**

① Content-Location

GET, PUT등 요청의 응답값은 idempotent(input값이 같다면 output도 무조건 같다.)

POST 요청의 응답값은 매번 다르다. 그래서 새로 생성한 리소스를 식별하기 위해 Content-Location을 사용한다.

② Content-Type

application/json 을 사용한다! 되도록이면 포맷을 통일시킨다. 응답 포맷이 여러개면 요청 포맷도 나눠야 하기 때문이다.

③Retry-After

비정상적인 방법으로 API에게 많은 요청을 보내는 경우 429 Too Many Requests 라는 오류 응답으로 일정 시간 뒤에 요청하도록 한다

1) 인증일 경우(로그인등)

- n시간 동안 n회만 요청 가능

⇒ Retry-After를 사용하여 429 에러를 내뱉는다.

- n회만 요청 가능

⇒ 401 응답과 함께 다시 요청하려면 특수한 절차가 필요하다고 한다.

(Retry-After X)

2) 의도적으로 서버 과부하를 목적으로 반복 요청하는 경우

- n시간 동안 n회 이상 요청한 경우

⇒ Retry-After를 사용하여 429 에러를 내뱉는다.

**3. HTTP Methods**

POST, GET, PUT, DELETE(CRUD) 는 반드시!! 제공해야한다.

OPTIONS, HEAD, PATCH를 사용해 완성도 높은 API를 만든다.

**4. HTTP Status**

200 으로 status를 응답하면서 body값은 실패했다는 메시지를 응답하는건 안 좋은 방식이다.

400 으로 status를 응답하면서 실패한 메시지를 보내는게 좋은 방식이다.

세부 에러사항을 body에 담아 해당 에러를 확인할 수 있는 link를 표시한다.

**5. Correct Status Code**

1.2XX (성공)

200 ⇒ OK

201 ⇒ Created (PUT, POST에 사용된다.)

202 ⇒ Accepted (비동기 작업에 사용된다.)

204 ⇒ No Content (응답 body가 없는 경우 ex) DELETE)

2.4XX (실패)

400 ⇒ Bad Request

클라이언트의 요청이 정의된 요구사항을 위반한 경우

파라미터의 위치, 사용자 입력 값, 에러 이유등을 반드시 알린다.

401 ⇒ Unauthorized

403 ⇒ Forbidden (요청은 유효하나, 접근이 허용되지 않는 경우)

404 ⇒ Not Found

405 ⇒ Method Not Allowed

404와 헷갈리지 말 것. 룰을 잘 정해야 한다.

ex) 해당 URL로 데이터는 찾았으니, Method가 허용되지 않는 경우이다.

409 ⇒ Conflict

해당 요청을 처리하는데 모순이 생긴 경우

ex) DELETE하는데 만약 사용자 테이블에 아무것도 없는 경우이다.

429 ⇒ Too Many Requests

Retry-After을 사용한다.

3. 5XX (서비스 장애)

해당 경우는 API Server Level에서 나타나는 경우이다. 그래서 절대 사용자에게 나타내면 안된다.

즉, API서버는 모든 발생 가능한 에러를 핸들링해야 한다는 말이 된다.

API 서버를 서빙하는 웹서버(apache, nginx)가 오류일 때는 가능하다.

**6. Use HATEOAS**

요청 - 응답으로 이루어지는데 응답은 되게 간단하게 응답한다.

여기서 문제는, 이 간단한 응답만으로는 사용자 리소스의 상태가 전이되기에는 정보가 부족하다.

HTML환경에서 보이는 환경이 있기 때문에 추후에 사용자의 상태가 전이될 수 있는 link를 화면에서 제공해줄 수 있다.

ex) 게시글을 만들고 해당 게시글을 바로 보려고 하려면, 클라이언트는 POST를 보내고 그에 대한 응답으로 게시글의 정보를 가져올 수 있는 link를 서버는 응답 본문에 같이 보내준다. 그러면 클라이언트는 URI를 하드코딩 하지 않고도 해당 link로 바로 서버에 요청을 보낼 수 있다.

자세한 내용은 https://pjh3749.tistory.com/260 참조하면 HATEOAS에 대한 내용을 알 수 있다.

**7. Sort, Filter**

7-1 Ordering

Collections(리스트)에 대한 GET 요청의 경우 order 라는 key를 이용해 정렬해서 응답한다.

7-2. Filtering

Collections(리스트)에 대한 GET 요청의 경우 AND OR = > < 등등.. 의 조건을 달 수 있다.

7-3. Field-Selecting

Collections(리스트)에 대한 GET 요청의 경우 일부분만 선택해서 응답받을 수 있다.

ex)

include : ?fields=id, name

exclude : ?-fields=level

만약에 Field에서 요구하는 Key가 없다면 존재하는 Key 로만 응답해서 보내준다.

**8. Versioning**

URI Versioning(이것을 사용한다.)

Version 정보는 host레벨이 아닌 path레벨에 사용한다.

ex) http://api.test.com/v1

Accept Header

ex) Accept: application/vnd.example.v1+json

ex) Accept: application/vnd.example+json;version=1.0

개발 코드에서는 관리하지 않는다!!

웹 서버의 reverse-proxy 기능을 활용한다.