5 주차 Web Client - Server

1. Client - Server

a. Client 와 Server

i. server
: client 에게 network 를 통해 서비스를 제공하는 시스템

ii. client
: 서비스를 사용하는 사용자
- web browser

b. Web Server 와 WAS 란?

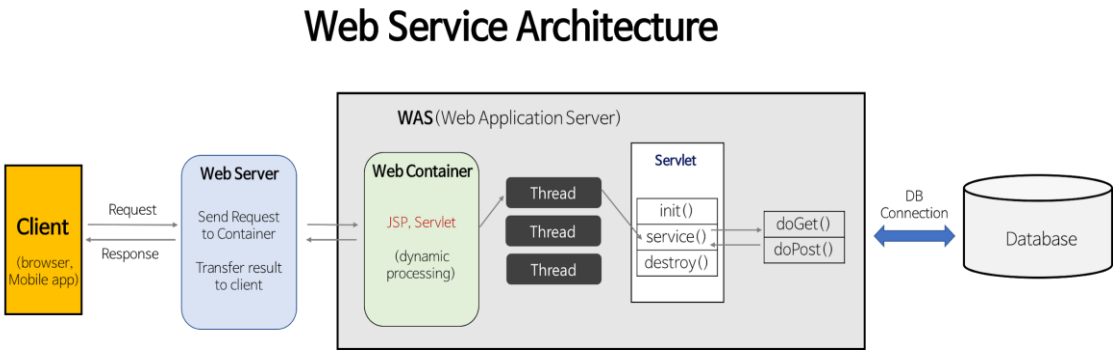
i. Web Server
: 정적인 컨텐츠를 제공하는 서버
ex) Apache, nginx

ii. WAS(Web Application Server)
: 정적뿐만 아니라 aplication program(PHP 등)으로 DB 조회, 로직처리 결정 등과 같은 동적인 컨텐츠를 제공하는 서버
- (= Web Server + CGI(Common Gateway Interface)

ex) Tomcat, Jeus

+ **일반적으로 web server 를 WAS 보다 앞단에 두어 web server 는 정적인 콘텐츠에 대한 요청을 바로 처리할 수 있게 하고 WAS 는 동적인 콘텐츠에 대한 요청을 처리할 수 있게 기능을 분리하여 서버의 부담을 줄임**
+ **또한 Client 와 WAS 의 포트가 직접적으로 연결되지 않게 함으로써 중요한 환경설정 파일들이 노출될 수 있는 가능성을 방지**

c. request message 를 주고받는 과정



i. 구조

Client -> Web Server -> DB
Client -> WAS -> DB
Client -> Web Server -> WAS -> DB

ii. 동작과정

Web Server 는 웹 브라우저 클라이언트로부터 HTTP 요청을 받는다.
Web Server 는 클라이언트의 요청을 WAS 에 보낸다.
WAS 는 관련된 Servlet 을 메모리에 올린다.
WAS 는 web.xml 을 참조하여 해당 Servlet 에 대한 Thread 를 생성한다.(Thread Pool 이용)
HttpServletRequest 와 HttpServletResponse 객체를 생성하여 Servlet 에 전달한다.

- Thread 는 Servlet 의 service() 메서드를 호출
- service() 메서드는 요청에 맞게 doGet() 또는 doPost() 메서드를 호출
- protected doGet(HttpServletRequest request, HttpServletResponse response)

doGet() 또는 doPost() 메서드는 인자에 맞게 생성된 적절한 동적 페이지를 Response 객체에 담아 WAS 에 전달한다.
WAS 는 Response 객체를 HttpResponse 형태로 바꾸어 Web Server 에 전달한다.
생성된 Thread 를 종료하고, HttpServletRequest 와 HttpServletResponse 객체를 제거한다.

2. HTTP (Hyper Text Transfer Protocol)

a. HTTP 란?

i. 의미

: Web 에서 client 와 server 가 서로 데이터를 주고 받을 수 있게(통신할 수 있게) 주고 받을 데이터의 틀(form)을 정한 것

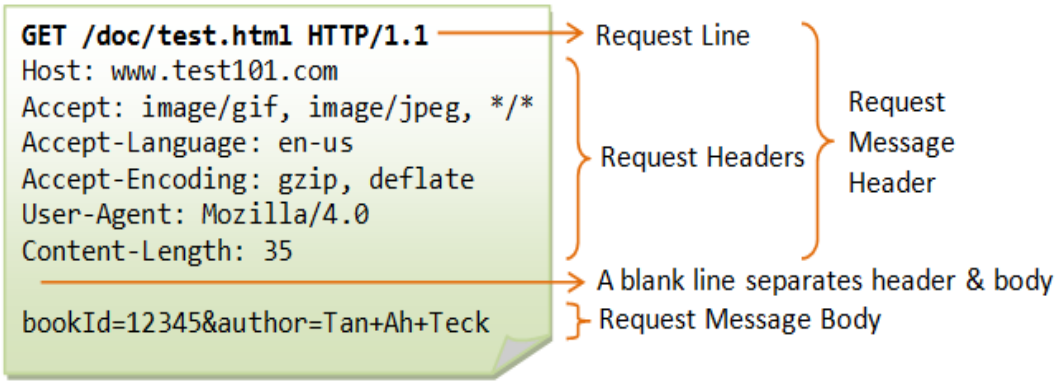
ii. 특징

: TCP/IP 를 이용하는 응용 프로토콜(application protocol)
: HTTP 는 연결 상태를 유지하지 않는 비연결성 프로토콜
(이러한 단점을 해결하기 위해 Cookie 와 Session 등장)
: HTTP 는 연결을 유지하지 않는 프로토콜이기 때문에 요청/응답(request/response) 방식으로 동작합니다.

iii. Message 구조

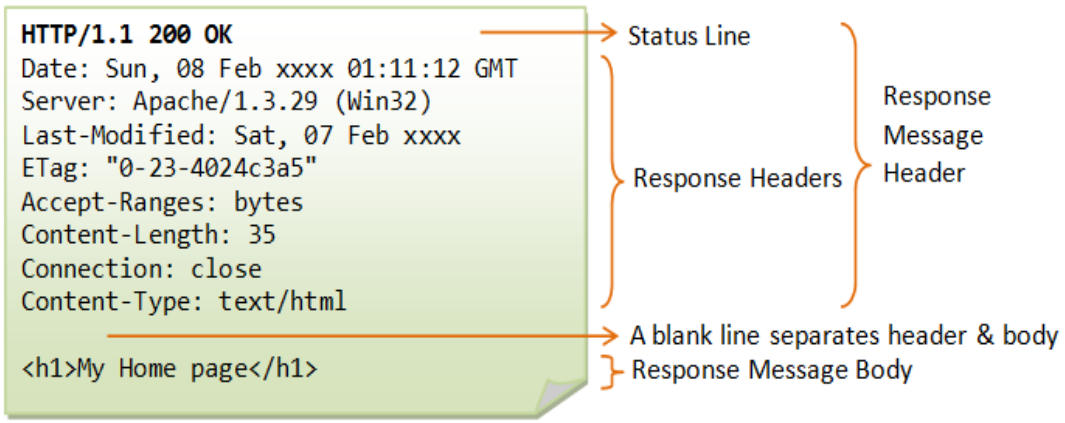
- **message 는 header 와 body 로 구성**

request message
: client 가 server 에 전송하는 message



response message

: server 가 client 에게 전송하는 message



header

: data 를 주고 받기 위해서 client, server 간에 필요 정보들

[request]

Request Line

1. **METHOD**

   : client 가 WAS 에 전달할 data 를 전송하는 방식

   ex) GET, POST, PUT, OPTIONS,HEAD, DELETE, TRACE, CONNECT

2. **Request - URI**

   : 원하는 web page 의 주소를 표시

   (GET 의 경우, URI 에 WAS 로 전달될 data 표시됨)

3. **HTTP - Version**

   : client 가 사용중인 http version

   ex) HTTP/1.1

Request Headers

: 요청을 한 서버의 Host 주소, client 가 가능한 decoding 방식(accept-encoding) ,client 의 web-

browser 환경(user-agent) 등을 전달해 server 가 해당 client 의 환경에 적합한 방식으로 데이터를 보낼 수 있게 해줌

+ general-header, request-header, entity-header

[response]

Status Line

1.HTTP - Version

2.Status-Code Phrase

: 요청에 대한 결과가 어떻게 되었는지 알려줌

ex) 200 OK (-> Success)

2xx : Success
3xx : Redirection
403 : Forbidden
404 : Not Found
5xx : Server Error

3.Response Headers

: general header 와 entity header 는 request 와 동일, Response header 는 accept-ranges,age,etag,location,proxy-authenticate,retry-after,server,vary,www-authenticate 가 있다.

body : 전달하고자 하는 Data

a.request

POST 인 경우

: request message 가 WAS 로 넘겨줄 파라미터의 값을 가짐

GET 인 경우: 내용 없음

b.response

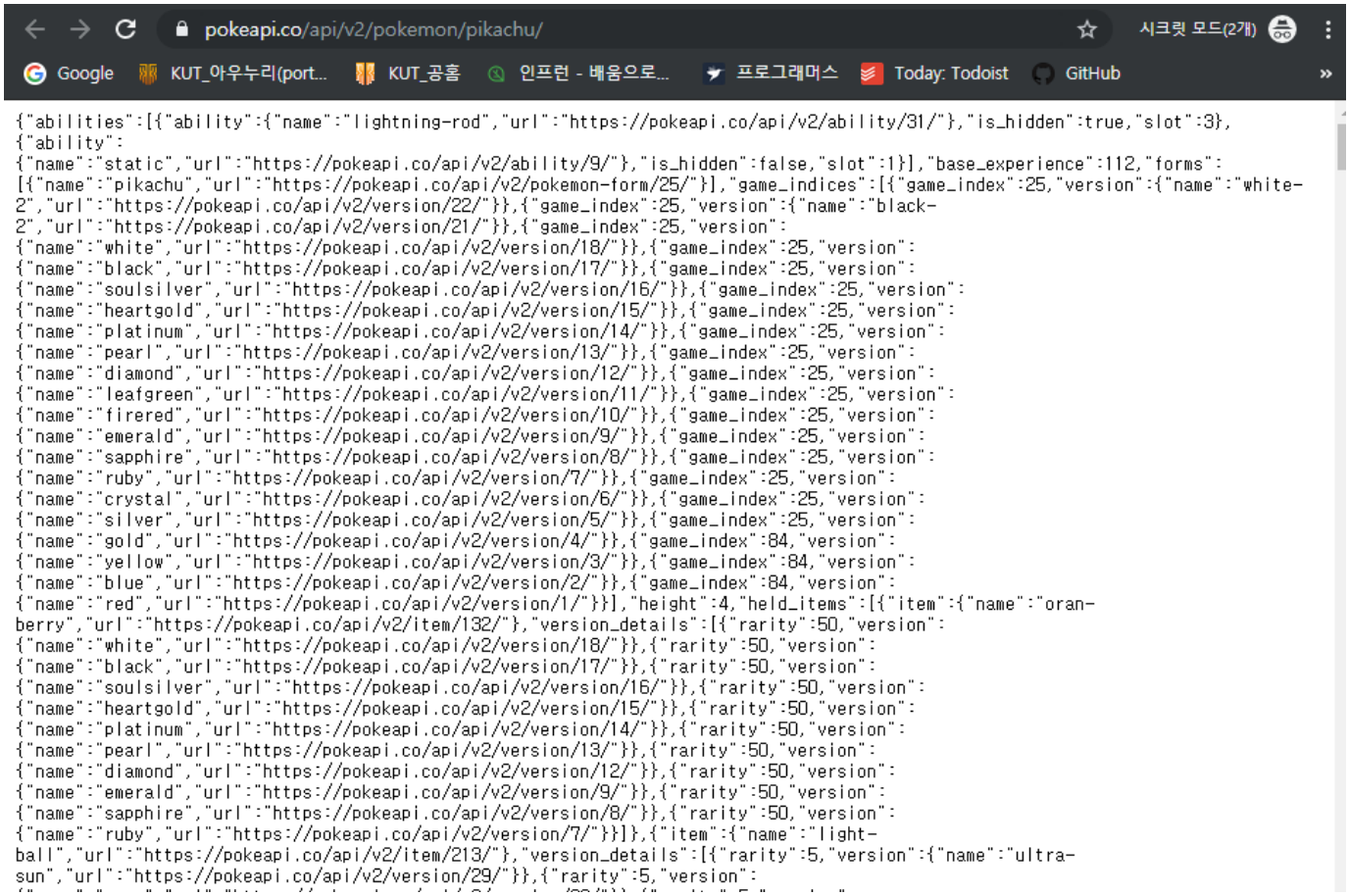
: 작성된 script 또는 binary 로 변경된 img 를 text 형태로 가지고 있음

3. API

a. API Application Programming Interface

i. 한 프로그램의 기능을 다른 프로그램이 이용할 수 있게 하는 것
ii. 프로그램간 커뮤니케이션을 담당하는 기능
iii. human interface 와 다르게 데이터를 기계가 이해하기 쉽도록 만들어 이를 입출력으로 구성함

+ EX) 실제 api 로 받은 정보


pokeapi.co/api/v2/pokemon/pikachu/
시크릿 모드(2개)
Google
KUT_아우누리(port...
KUT_공홈
인프런 - 배움으로...
프로그래머스
Today: Todoist
GitHub
{"abilities":[{"ability":{"name":"lightning-rod","url":"https://pokeapi.co/api/v2/ability/31/"},"is_hidden":true,"slot":3},
{"ability":
{"name":"static","url":"https://pokeapi.co/api/v2/ability/9/"},"is_hidden":false,"slot":1}],"base_experience":112,"forms":
[{"name":"pikachu","url":"https://pokeapi.co/api/v2/pokemon-form/25/"}],"game_indices":[{"game_index":25,"version":{"name":"white-
2","url":"https://pokeapi.co/api/v2/version/22/"}},{"game_index":25,"version":{"name":"black-
2","url":"https://pokeapi.co/api/v2/version/21/"}},{"game_index":25,"version":
{"name":"white","url":"https://pokeapi.co/api/v2/version/18/"}},{"game_index":25,"version":
{"name":"black","url":"https://pokeapi.co/api/v2/version/17/"}},{"game_index":25,"version":
{"name":"soulsilver","url":"https://pokeapi.co/api/v2/version/16/"}},{"game_index":25,"version":
{"name":"heartgold","url":"https://pokeapi.co/api/v2/version/15/"}},{"game_index":25,"version":
{"name":"platinum","url":"https://pokeapi.co/api/v2/version/14/"}},{"game_index":25,"version":
{"name":"pearl","url":"https://pokeapi.co/api/v2/version/13/"}},{"game_index":25,"version":
{"name":"diamond","url":"https://pokeapi.co/api/v2/version/12/"}},{"game_index":25,"version":
{"name":"leafgreen","url":"https://pokeapi.co/api/v2/version/11/"}},{"game_index":25,"version":
{"name":"firered","url":"https://pokeapi.co/api/v2/version/10/"}},{"game_index":25,"version":
{"name":"emerald","url":"https://pokeapi.co/api/v2/version/9/"}},{"game_index":25,"version":
{"name":"sapphire","url":"https://pokeapi.co/api/v2/version/8/"}},{"game_index":25,"version":
{"name":"ruby","url":"https://pokeapi.co/api/v2/version/7/"}},{"game_index":25,"version":
{"name":"crystal","url":"https://pokeapi.co/api/v2/version/6/"}},{"game_index":25,"version":
{"name":"silver","url":"https://pokeapi.co/api/v2/version/5/"}},{"game_index":25,"version":
{"name":"gold","url":"https://pokeapi.co/api/v2/version/4/"}},{"game_index":84,"version":
{"name":"yellow","url":"https://pokeapi.co/api/v2/version/3/"}},{"game_index":84,"version":
{"name":"blue","url":"https://pokeapi.co/api/v2/version/2/"}},{"game_index":84,"version":
{"name":"red","url":"https://pokeapi.co/api/v2/version/1/"}}],"height":4,"held_items":[{"item":{"name":"oran-
berry","url":"https://pokeapi.co/api/v2/item/132/"},"version_details":[{"rarity":50,"version":
{"name":"white","url":"https://pokeapi.co/api/v2/version/18/"}},{"rarity":50,"version":
{"name":"black","url":"https://pokeapi.co/api/v2/version/17/"}},{"rarity":50,"version":
{"name":"soulsilver","url":"https://pokeapi.co/api/v2/version/16/"}},{"rarity":50,"version":
{"name":"heartgold","url":"https://pokeapi.co/api/v2/version/15/"}},{"rarity":50,"version":
{"name":"platinum","url":"https://pokeapi.co/api/v2/version/14/"}},{"rarity":50,"version":
{"name":"pearl","url":"https://pokeapi.co/api/v2/version/13/"}},{"rarity":50,"version":
{"name":"diamond","url":"https://pokeapi.co/api/v2/version/12/"}},{"rarity":50,"version":
{"name":"emerald","url":"https://pokeapi.co/api/v2/version/9/"}},{"rarity":50,"version":
{"name":"sapphire","url":"https://pokeapi.co/api/v2/version/8/"}},{"rarity":50,"version":
{"name":"ruby","url":"https://pokeapi.co/api/v2/version/7/"}}]},{"item":{"name":"light-
ball","url":"https://pokeapi.co/api/v2/item/213/"},"version_details":[{"rarity":5,"version":{"name":"ultra-
sun","url":"https://pokeapi.co/api/v2/version/29/"}},{"rarity":5,"version":

b. RESTful API

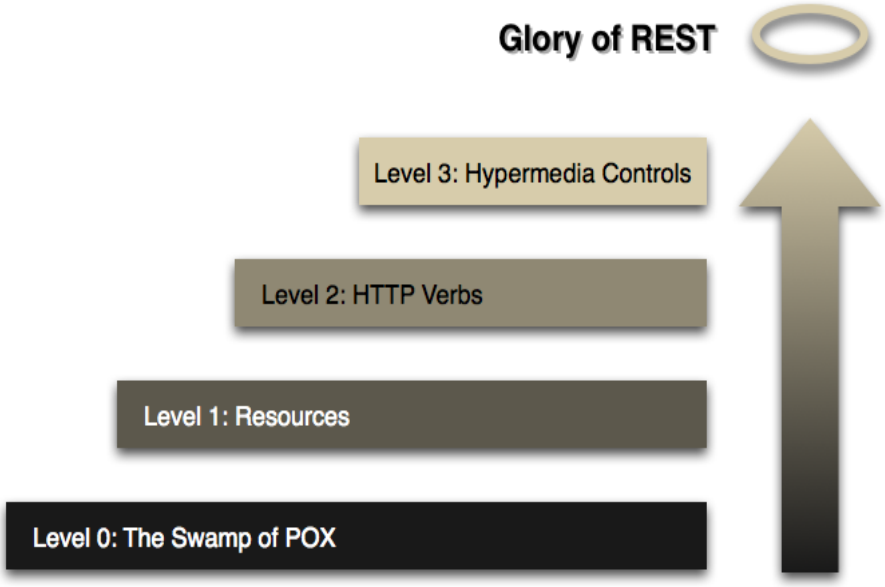
i. REST

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

CRUD	HTTP verbs	Route
resource 들의 목록을 표시	GET	*/resource*
resource 하나의 내용을 표시	GET	/resource/:id
resource 를 생성	POST	/resource
resource 를 수정	PUT	/resource/:id
resource 를 삭제	DELETE	/resource/:id

REST 구성 요소

1. 자원(Resource) : URI
    a. 모든 자원에 고유한 ID 존재
    b. 자원을 구별하는 ID 는 '/groups/:group_id'와 같은 HTTP URI
    c. Client 는 URI 를 이용해서 자원을 지정하고 해당 자원의 상태(정보)에 대한 조작(CRUD)를 Server 에 요청
2. 행위(Verb) : HTTP Method
    a. GET,POST,PUT, DELETE
3. 표현(Representation of Resource)
    a. Client 의 요청에 대한 Server 의 응답(Representation)
    b. REST 에서 하나의 자원은 JSON,XML,TEXT,RSS 등 여러 형태의 Representation 으로 나타내어 질 수 있다.
    c. 일반적으로 JSON 혹은 XML 를 통해 데이터를 주고 받음



- REST 특징

1. Server-Client 구조
2. Stateless(무상태)
    a. HTTP 를 이용하므로 REST 역시 마찬가지로 무상태성을 가짐
    b. Client 는 Context 를 Server 에 저장하지 않음
    c. Server 는 각각의 요청을 완전히 별개의 것으로 인식하고 처리
3. Cacheable(캐시 처리 가능)
4. Layered System(계층화)
    a. API Server 는 순수 비즈니스 로직을 수행하고 그 앞단에 보안, 로드밸런싱, 암호화, 사용자 인증 등을 추가하여 구조상의 유연성을 줄 수 있다.
    b. PROXY, 게이트웨이 같은 네트워크 기반의 중간 매체를 사용할 수 있다.
5. Code-On-Demand(optional)
6. Uniform Interface(인터페이스 일관성)
    a. URI 로 지정한 Resource 에 대한 조작을 통일되고 한정적인 인터페이스로 수행
    b. 특정 언어나 기술에 종속되지 않고 HTTP 표준을 따르는 모든 플랫폼에서 사용 가능

ii. REST API

- REST 기반으로 서비스 API 를 구현한 것

```
POST http://www.plusblog.co.kr/users Content-Type: application/json

{
  "username" : "newusers",
  "age" : "20"
}
```

iii. RESTful API 란?

- REST API 의 설계 의도를 정확하게 지켜주는 API 를 의미
- RPC(Remote Procedure Call-메서드 호출)이 목적이 아니라 시스템에서 제공하는 Resource 를 Bucket 화가 목적
- 설계 가이드

1. URL Rules
    - 1.1. 마지막에 / 포함하지 않는다.
    - 1.2. _(underbar) 대신 -(dash)를 사용한다.
    - 1.3. 소문자를 사용한다.
    - 1.4. 행위(method)는 URL 에 포함하지 않는다.
    - 1.5. 컨트롤 자원을 의미하는 URL 예외적으로 동사를 허용한다
    - 1.6 파일확장자는 URI 에 포함하지 않는다.
        - Accept header 사용
            - Ex) http://restapi.example.com/members/soccer/345/photo.jpg (X)
            - Ex) GET / members/soccer/345/photo HTTP/1.1 Host: restapi.example.com Accept: image/jpg (O)

2. Set HTTP Headers
    - 2.1. Content-Location
        - POST 요청의 경우 반환되는 응답 리소스 결과가 동일하지 않기 때문에 요청의 응답 헤더에 새로 생성된 리소스를 식별할 수 있는 Content-Location 속성을 이용한다.
        - HATEOAS 로 대체 가능
    - 2.2. Content-Type
        - application/json 을 우선으로 제공하여 응답 포맷을 이원화 하지 않도록 한다.
    - 2.3. Retry-After
        - 2.3.1. Case 1. 인증
        - 2.3.2. Case 2. 자원 요청
    - 2.4. Link

3. Use HTTP methods
    - 3.1. POST, GET, PUT, DELETE 4 가지 methods 는 반드시 제공한다.
    - 3.2. OPTIONS, HEAD, PATCH 를 사용하여 완성도 높은 API 를 만든다.

        - 3.2.1. OPTIONS
        - 3.2.2. HEAD
        - 3.2.3. PATCH

4. Use HTTP status
    - 4.1. 의미에 맞는 HTTP status 를 리턴한다
    - 4.2. HTTP status 만으로 상태 에러를 나타낸다

5. Use the correct HTTP status code.
    - 5.1. 성공 응답은 `2XX` 로 응답한다.
    - 5.2. 실패 응답은 `4XX` 로 응답한다.

6. Use HATEOAS (Hypermedia As The Engine Of Application State)

    하이퍼미디어를 애플리케이션의 상태를 관리하기 위한 매커니즘으로 사용한다

    - 6.1. 구성 요소
        - 변경될 리소스의 상태 관계 `rel`
            - `self`: 현재 URL 자신, 예약어처럼 쓰임
        - 요청 URL `href`
        - 요청 Method `method`
        - …(그 외 추가사항)

    - 6.2. 응답 예제
        - HATEAOS 가 도입되어 자원에 대한 추가 정보가 제공되는 응답

```
201 Created
{
    "id": 1,
    "name": "hak",
    "createdAt": "2018-07-04 14:00:00"
    "links": [
        {
            "rel": "self",
            "href": "http://api.test.com/users/1",
            "method": "GET"
        },
        {
            "rel": "delete",
            "href": "http://api.test.com/users/1",
            "method": "DELETE"
        },
        {
            "rel": "update",
            "href": "http://api.test.com/users/1",
            "method": "PATCH",
            "more_info": "http://api.test.com/docs/user-update"
            "body": {
                "name": "{The value to be modified}"
            }
        },
        {
            "rel": "user.posts",
            "href": "http://api.test.com/users/1/posts",
            "method": "GET"
        }
    ]
}
```

+ 자원별로 API 를 추가할 필요없이, 자원의 상태에 따른 API 를 link 의 rel 키워드로 각 API 의 엔드포인트에 대한 유일한 이름을 할당해 API 엔드포인트를 제공함
+ Client 는 rel 이름만 알면 해당 기능의 실행 가능 여부 및 이를 실행하기 위해 필요한 데이터를 손쉽게 판단 가능
+ API 개발자들은 최초 진입을 위한 API 엔드포인트를 제외한 나머지 URI 들을 수정가능

4. Cookie, Session

a. Cookie

i. client(브라우저) 로컬에 저장되는 키와 값이 들어있는 작은 데이터 파일

ii. 특징

1. 사용자 인증이 유효한 시간을 명시 가능
2. 유효 시간이 정해지면 브라우저가 종료되어도 인증이 유지
3. 쿠키는 client 의 상태 정보를 로컬에 저장했다가 참조
4. 사용자가 따로 요청하지 않아도 브라우저가 request 시에 request header 에 넣어서 자동으로 쿠키를 서버에 전송

iii. 구성 요소

1. 이름 : 각각의 쿠키를 구별하는데 사용되는 이름
2. 값 : 쿠키의 이름과 관련된 값
3. 유효시간 : 쿠키의 유지시간
4. 도메인 : 쿠키를 전송할 도메인
5. 경로 : 쿠키를 전송할 요청 경로

iv. 동작 방식

1. 클라이언트가 페이지를 요청
2. 서버에서 쿠키를 생성
3. HTTP header 에 쿠키를 포함시켜 응답
4. 브라우저가 종료되어도 쿠키 만료 기간이 있다면 클라이언트에서 보관
5. 같은 요청을 할 경우 HTTP header 에 쿠키를 함께 보냄
6. 서버에서 쿠키를 읽어 이전 상태 정보를 변경할 필요가 있을 때 쿠키를 업데이트하여 변경된 쿠키를 HTTP header 에 포함시켜 응답

b. Session

i. 일정 시간동안 같은 사용자(브라우저)로부터 들어오는 일련의 요구를 하나의 상태로 보고 그 상태를 일정하게 유지시키는 기술

ii. 방문자가 웹서버에 접속해 있는 상태를 하나의 단위로 보는 것

iii. 특징

1. 각 client 에게 고유 ID 부여
2. Session ID 로 client 를 구분해서 client 의 요구에 맞는 서비스를 제공
3. 보안 면에서 쿠키보다 우수
4. 사용자가 많아질수록 서버 메모리를 많이 차지하게 됨
5. 로그인과 같이 보안상 중요한 작업을 수행할 때 사용

iv. 동작 방식

1. client 가 서버에 접속 시 Session ID 를 발급
2. client 는 Session ID 에 대해 쿠키를 사용해서 저장

3. client 가 서버에 다시 접속 시 이 쿠키를 이용해서 Session ID 값을 서버에 전달

c. Cookie 와 Session 의 차이

i. 사용자의 기록 정보가 저장되는 위치

1. 쿠키 : 사용자 Local 저장소
2. Session : 서버 내 메모리에 저장

ii. 보안 : 쿠키 < Session (SessionID 만으로 구분해서 서버에서 처리하기 때문)

iii. 요청 속도 : 쿠키 > Session (쿠키는 바로 Local 에서 가져올 수 있기 때문)

5. 웹 프레임워크

a. 프레임워크란?

i. 프레임워크란, 소프트웨어의 구체적인 부분에 해당하는 설계와 구현을 재사용이 가능하게끔 일련의 협업화된 형태로 클래스들을 제공하는 것

ii. 필요성

- enterprise 규모 프로그램 개발을 진행하면서 투입되는 개발자의 수도 점점 늘어남에 따라 개발자 개인마다 코드를 짜는 형태가 일관되지 못하게 되어 시스템의 통합성, 일관성이 부족하게 되었음
- 이 문제를 해결하기 위해 개발자에게 가이드(클래스)를 제공하여 개발에 대한 방법론을 강제함

b. 종류

구분	종류
자바 프레임워크	Struts, Spring, 전자정부 프레임워크
QRM 프레임워크	myBatis(iBatis), Hibernate
자바스크립트 프레임워크	AngularJS, React, Polymer, Ember
프론트엔드 프레임워크	Bootstrap, Foundation, MDL

c. 장단점

i. 장점

1. 효율성 향상
   - 무에서 코드를 짜는 것보다 시간과 비용이 절약, 생산성 증가
2. Quality 향상
   - 버그 발생 가능성을 처리해줌으로써 개발자가 반복 작업에서 실수하기 쉬운 부분을 커버해줌
3. 유지 보수하기 유용
   - 프레임워크를 사용하면 개발 방법이 강제되서 코드가 보다 체계적이므로 담당자가 바뀌더라도 위험부담을 줄일 수 있음

ii. 단점

1. 학습시간이 김
   - 프레임워크에서 제공되는 코드는 본인이 짠 것이 아니므로 초기에 프레임워크에 있는 코드를 습득하고 이해하는데 오랜 시간이 걸림

2. 제작자의 의도된 제약 사항
    - 제작자가 설계한 구조를 유지한 상태로 코드에 살을 붙여나가야하므로 개발자가 자유롭고 유연하게 개발하는데 한계가 존재