Introduction To Restful Web Service

https://www.tutorialspoint.com/restful

1. What is REST?

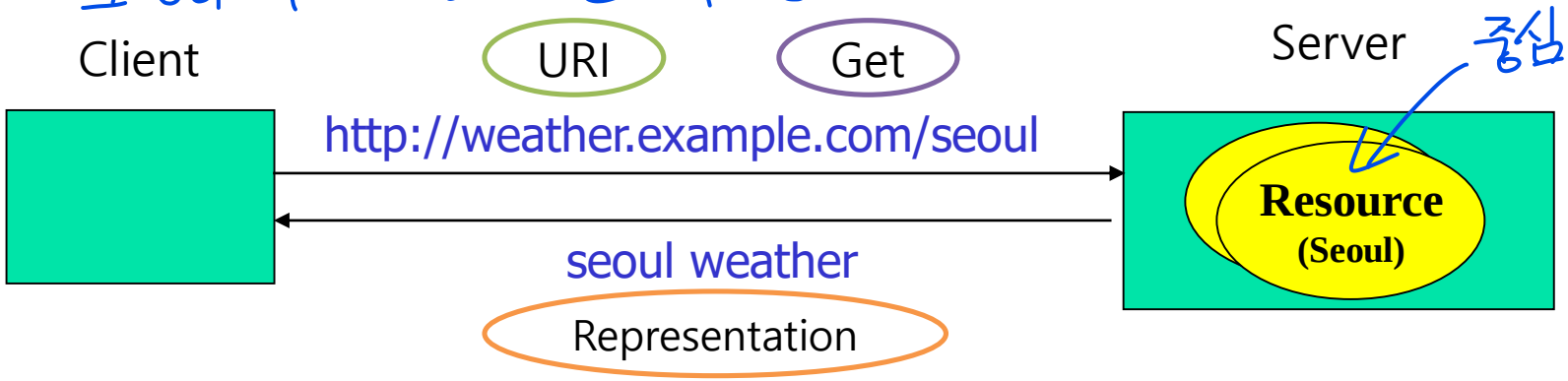
웹의 표준에 기반한 아키텍처, http 프로토콜 이용

- REST stands for REpresentational State Transfer
- An architectural style for developing web services
- REST is a web standards based architecture and uses HTTP Protocol for data communication

Resource 에 대해서 CRUD를 함

- It revolves around resources which are accessed by a common interface using HTTP standard methods

요청하여 Resource 에 접근

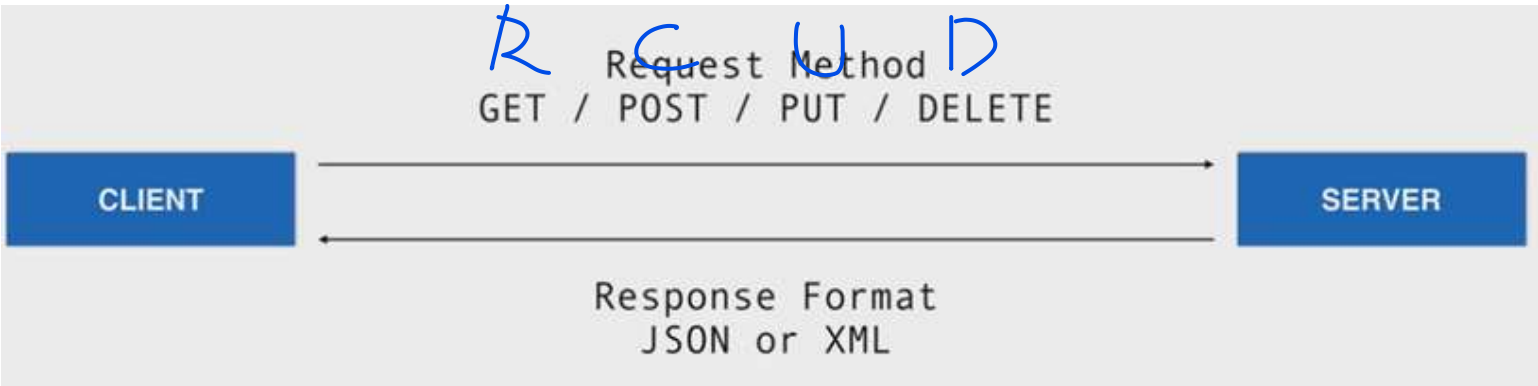


What is REST?

- In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources
  - Each resource is identified by URIs/ Global IDs
  - REST uses various representations to represent a resource like Text, JSON and XML
  - JSON is now the most popular format being used in Web Services

서버가 resource를 갖고있다
resource는 text나 JSON이나 XML로 표현

What is REST?



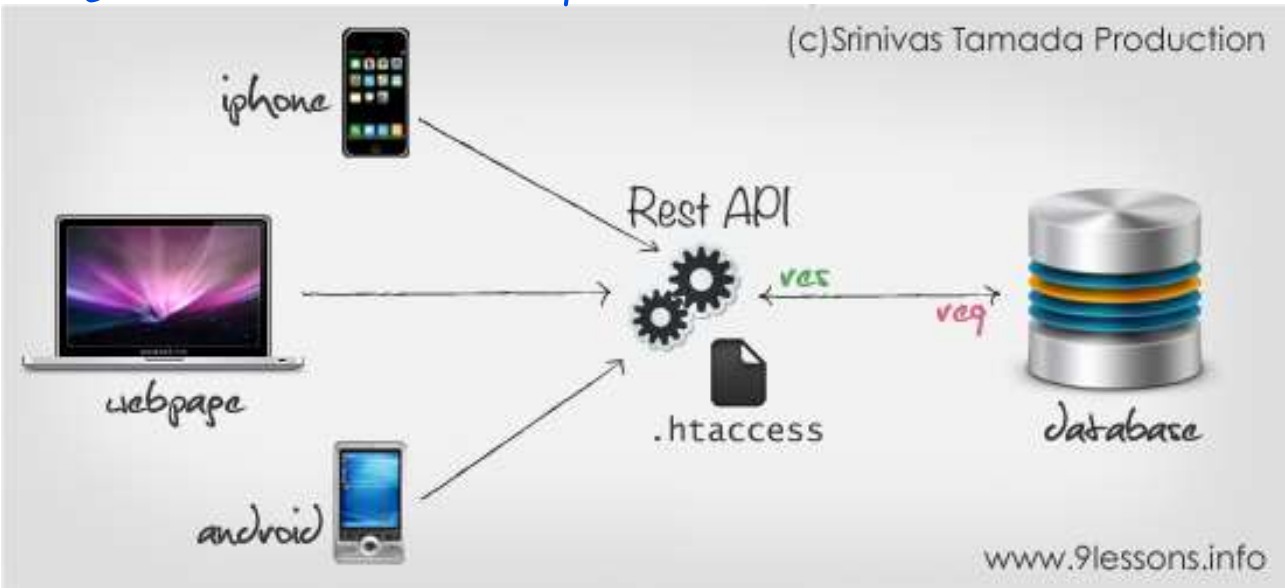
Example: Web service called User Management

HTTP Method	URI	Operation
GET	/api/users	returns a list of users
GET	/api/users/1	returns the user with ID 1
POST	/api/users	creates a new user
PUT	/api/users/3	updates the user with ID 3
DELETE	/api/users/4	deletes the user with ID 4
DELETE	/api/users	deletes all the users

2. RESTful Web Services

- Web services based on REST Architecture are known as RESTful Web Services

웹은 REST에 기반한다



Decoupling + Platform Agnostic

구조

RESTful Web Services

Notice that pattern :

A set of **commands (method)** performed on **things (resource)** generates **responses (message)**

This is the foundation of a REST API

3 Pillars of REST

Method
POST

Resource
Things

Message
201 CREATED

2.1 RESTful Web Services (Resources)

- Representation of Resources

태그

XML

```
<user>
    <id>1</id>
    <name>Mahesh</name>
    <profession>Teacher</profession>
</user>
```

속성

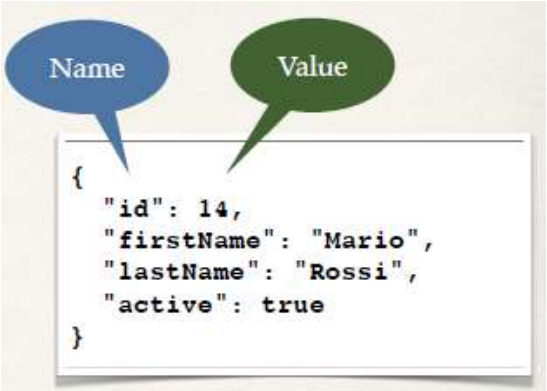
JSON

```
{
    "id":1,
    "name":"Mahesh",
    "profession":"Teacher"
}
```

Simple JSON Example

Curley braces가 객체를 정의

- Curley braces define objects Object members are name / value pairs
  - Delimited by colons
- Name is **always** in double-quotes
- JSON Values
  - Numbers: no quotes
  - String: in double quotes
  - Boolean: **true**, **false**
  - Nested JSON object
  - Array
  - null




```
{
"id": 14,
"lastName": "Rossi",
"active": true,
"languages" : ["Java", "Python", "Javascript"],
"address" : {
        "street" : "100 Main St",
        "city" : "Philadelphia",
        "state" : "Pennsylvania",
        "zip" : "19103",
        "country" : "USA"
    }
}
```

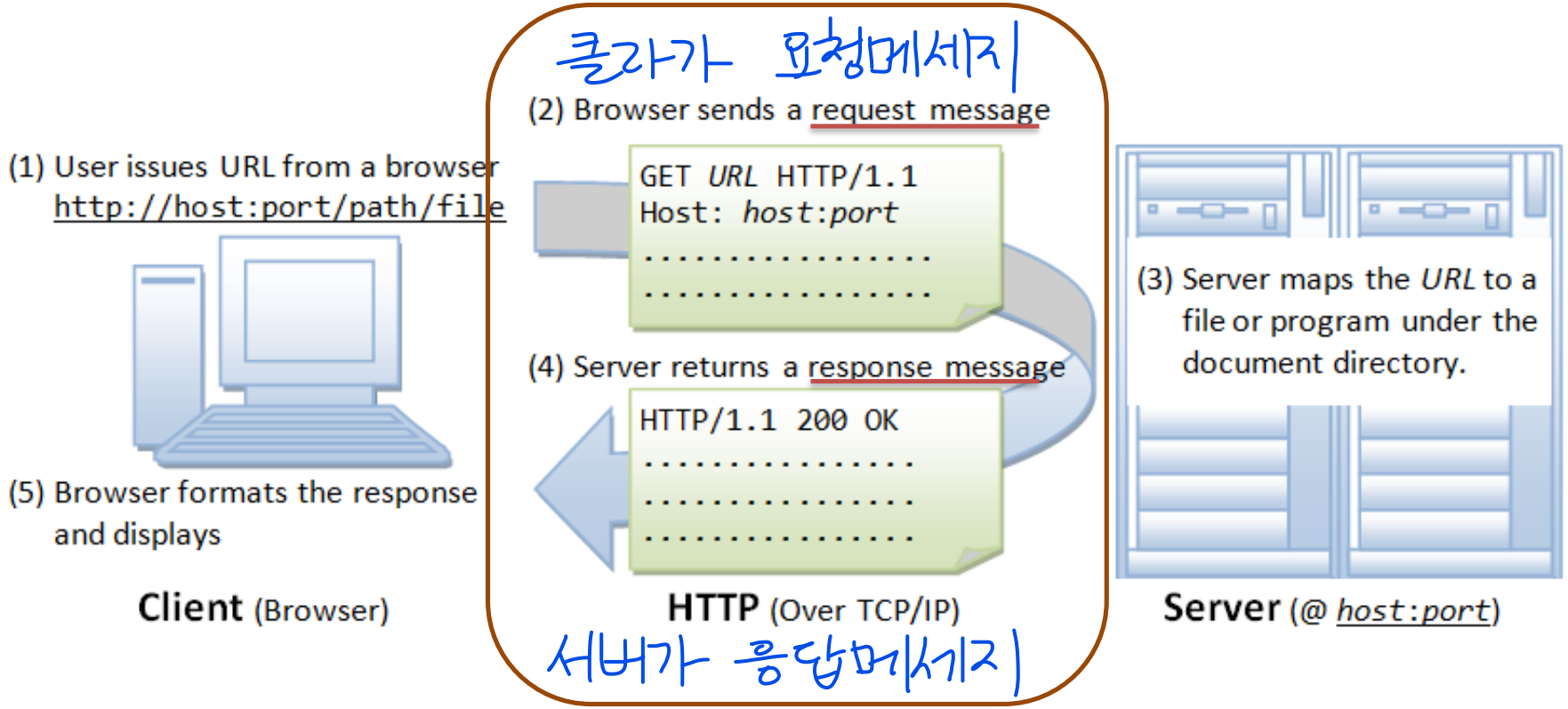
Array

배열도 가능

Nested

name value

2.2 RESTful Web Services (Messages)

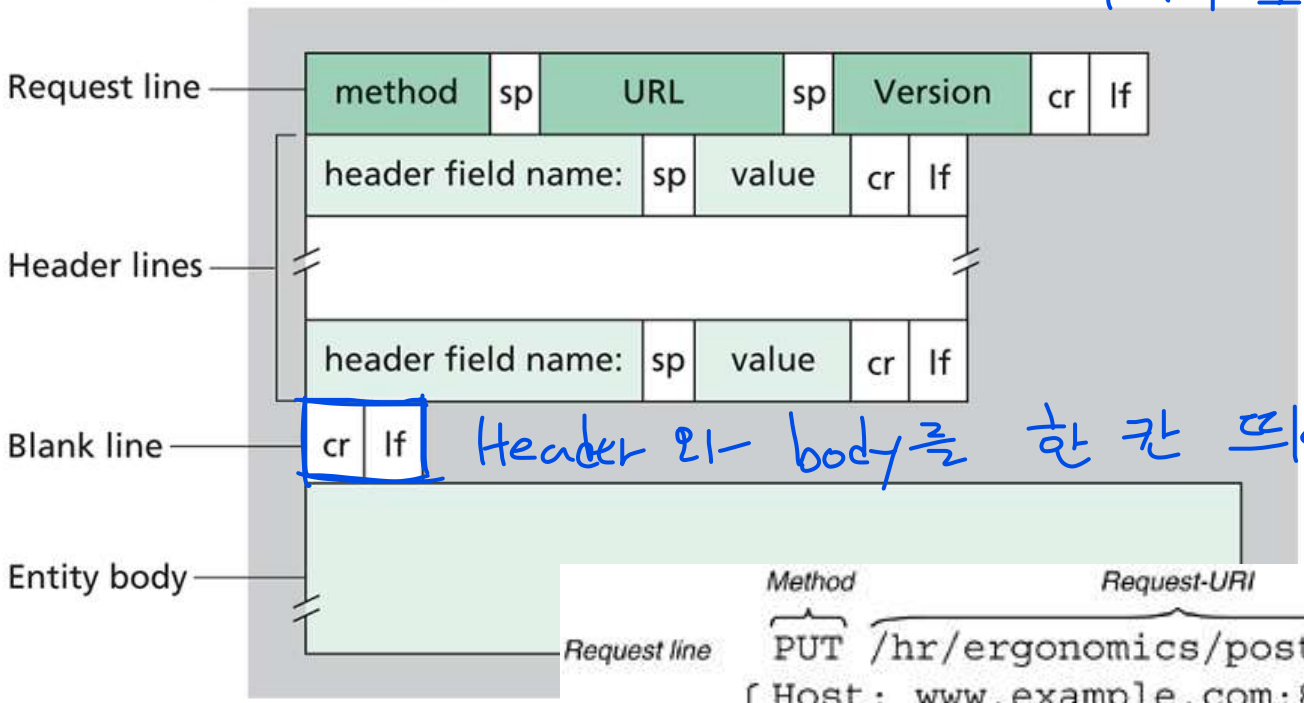


(Image: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html)

RESTful Web Services (Messages)

요청메시지는 ASCII 형식이라서 우리가 보고 읽을 수 있다

- HTTP request message: ASCII (human-readable format)



Header 와 body를 한 칸 띄어야 인식함

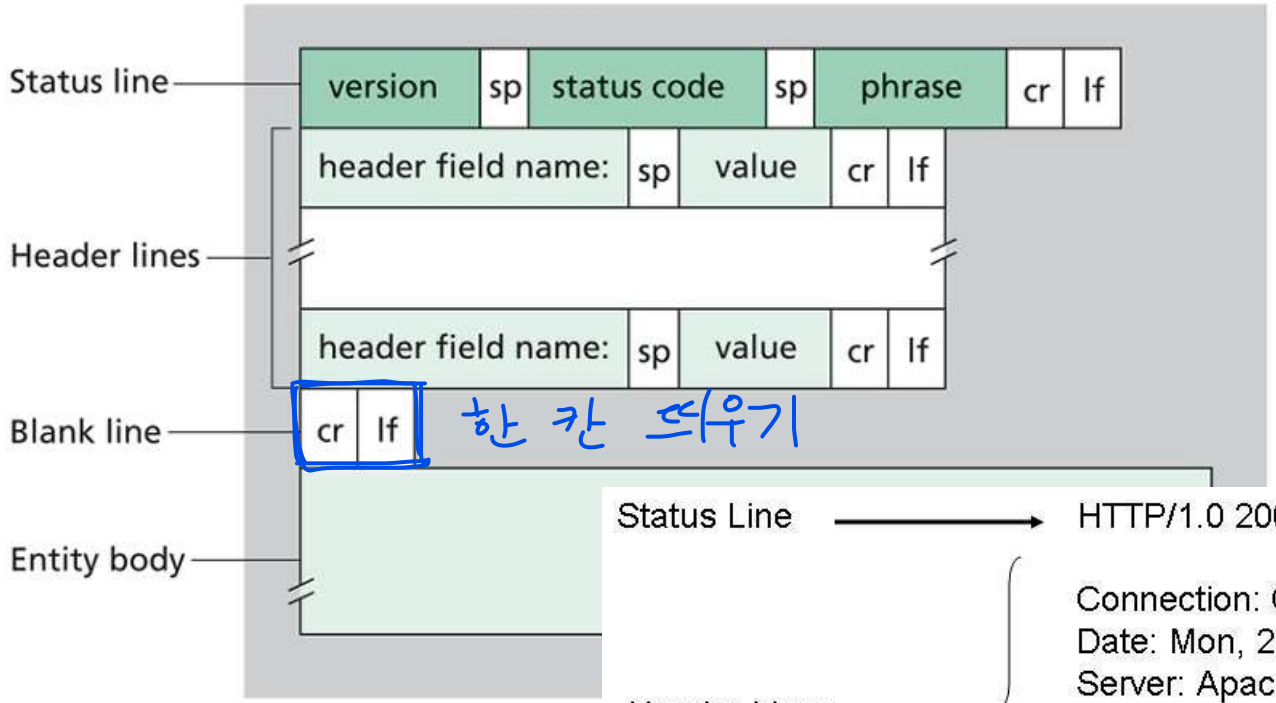
	Method	*Request-URI*	*Protocol version*
Request line	PUT	/hr/ergonomics/posture.doc	HTTP/1.1
Headers	Host: www.example.com:8080 Content-Length: 1234		
Empty line			
Body (optional)	Body must include the number of characters specified in the content length header...		

RESTful Web Services (Messages)

- An HTTP Request has five major parts:
  - Method
    - Indicates the HTTP methods such as GET, POST, DELETE, PUT, etc
  - URI
    - Uniform Resource Identifier (URI) to identify the resource on the server
  - HTTP Version
    - Indicates the HTTP version. For example, HTTP v1.1
  - Request Header
    - Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by the client, format of the message body, cache settings, etc
  - Request Body
    - Message content or Resource representation

RESTful Web Services (Messages) 응답메시지도 ASCII

- **HTTP response message:** ASCII (human-readable format)



Status Line → HTTP/1.0 200 OK

Header Lines
- Connection: Close
- Date: Mon, 20 Nov 2006 12:00:10 GMT
- Server: Apache/1.3.0 (Unix)
- Last-Modified: Mon 22 Jul 2006 .....
- Content-Length: 6821
- Content-Type: Text/html

Data → Data, data, data, data.....
i.e.: requested file

RESTful Web Services (Messages)

- An HTTP Response has four major parts:
  - Status/Response Code
    - Indicates the Server status for the requested resource. For example, 404 means resource not found and 200 means response is OK
  - HTTP Version
    - Indicates the HTTP version. For example HTTP v1.1
  - Response Header
    - Contains metadata for the HTTP Response message as key-value pairs. For example, content length, content type, etc.
  - Response Body
    - Response message content or Resource representation

http 보다 https 가 암호화 좋다

RESTful Web Services (Messages)

- Content Negotiation 협상
  - Request (client)
    - Accept: Give me this kind of response. Here's a list in order of what I'm hoping you'll send

      Accept: text/html, application/xhtml+xml, application/xml

      나는 Accept에 해당하는 것만 받을 수 있어
  - Response (server)
    - Content-Type: This is the kind of response I'm sending you

      Content-Type: text/html; charset=UTF-8

      알겠어

RESTful Web Services (Messages)

- HTTP Status/Response Codes
  - HTTP is built in with a set of status codes for various types of scenarios:

404 같은 코드들

- 2xx Success (*200 OK, 201 Created…*)
- 3xx Redirection (*303 See other*)
- 4xx Client error (*404 Not Found*)
- 5xx Server error (*500 Internal Server Error*)

2.3 RESTful Web Services (Addressing)

- Each resource in REST architecture is identified by its URI (Uniform Resource Identifier)
- A URI is of the following format:

<protocol>://<service-name>/<ResourceType>/<ResourceID>

http://localhost:8080/estore/users/1

각 resource는 URI로 식별됨

RESTful Web Services (Addressing)

- Constructing a Standard URI URI의 표준
  - Use Plural Noun 복수형 사용
    - Use plural noun to define resources. For example, we've used *users* to identify users as a resource
  - Avoid using spaces 띄어쓰기 금지
    - Use underscore (_) or hyphen (-) when using a long resource name. For example, use *authorized_users* instead of authorized%20users
  - Use lowercase letters 소문자 사용
    - Although URI is case-insensitive, it is a good practice to keep the url in lower case letters only
  - Maintain Backward Compatibility 호환성 유지
    - As Web Service is a public service, a URI should always be available. In case URI gets updated, redirect the older URI to a new URI using the HTTP Status code, 300
  - Use HTTP Verb HTTP Verb를 사용금지
    - Always use HTTP Verb like GET, PUT and DELETE to do the operations on the resource. It is not good to use operations name in the URI

RESTful Web Services (Addressing)

- Example

Following is an example of a poor URI to fetch a user:

http://localhost:8080/.../getUser/1 나쁜 예

Following is an example of a good URI to fetch a user:

http://localhost:8080/.../users/1 좋은 예

2.4 RESTful Web Services (Methods)

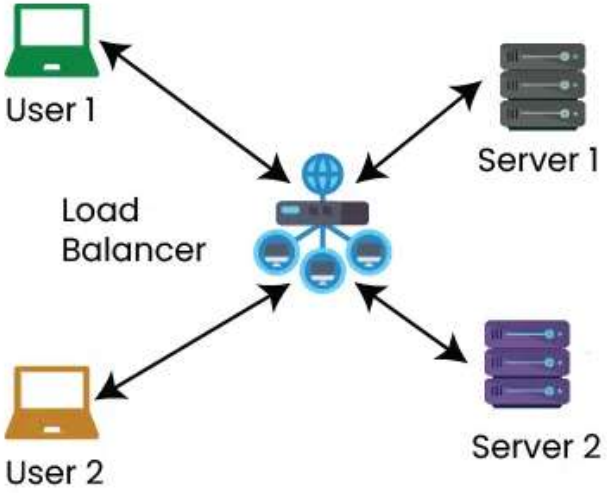
	HTTP Method, URI and Operation
1	GET http://localhost:8080/estore/users Gets the list of users 모든 사용자 조회
2	GET http://localhost:8080/estore/users/1 Gets the User of Id 1 특정 사용자 조회
3	POST http://localhost:8080/estore/users/2 Inserts User with Id 2
4	PUT http://localhost:8080/estore/users/2 Updates the User with Id 2
5	DELETE http://localhost:8080/estore/users/1 Deletes the User with Id 1
6	OPTIONS http://localhost:8080/estore/users 위 중에서, 어떤 Lists out the supported operations in a web service 오퍼레이션 가능?
7	HEAD http://localhost:8080/estore/users Returns the HTTP Header only, no Body 헤더 정보만 가져오기

2.5 RESTful Web Services (Statelessness)

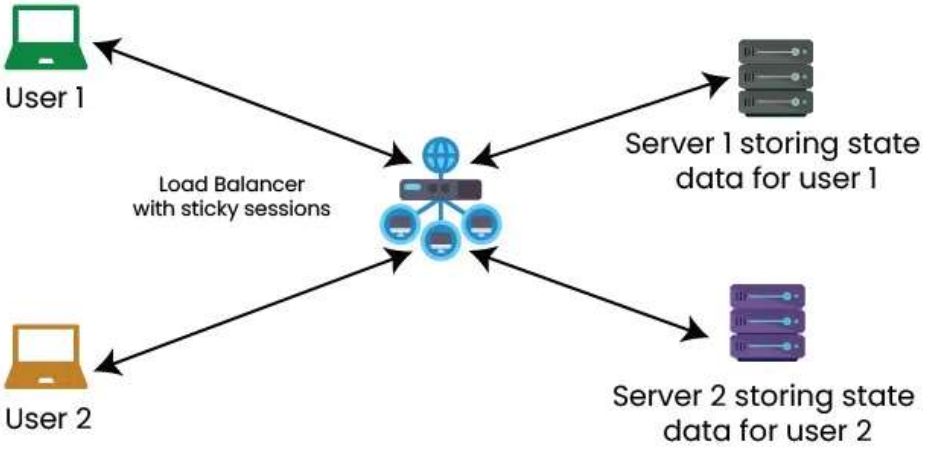
서버는 클라이언트의 상태 정보를 저장하면 안된다 : Statelessness

- A RESTful Web Service should not keep a client state(session data, preferences) on the server
- This restriction is called *Statelessness*

Stateless



Stateful

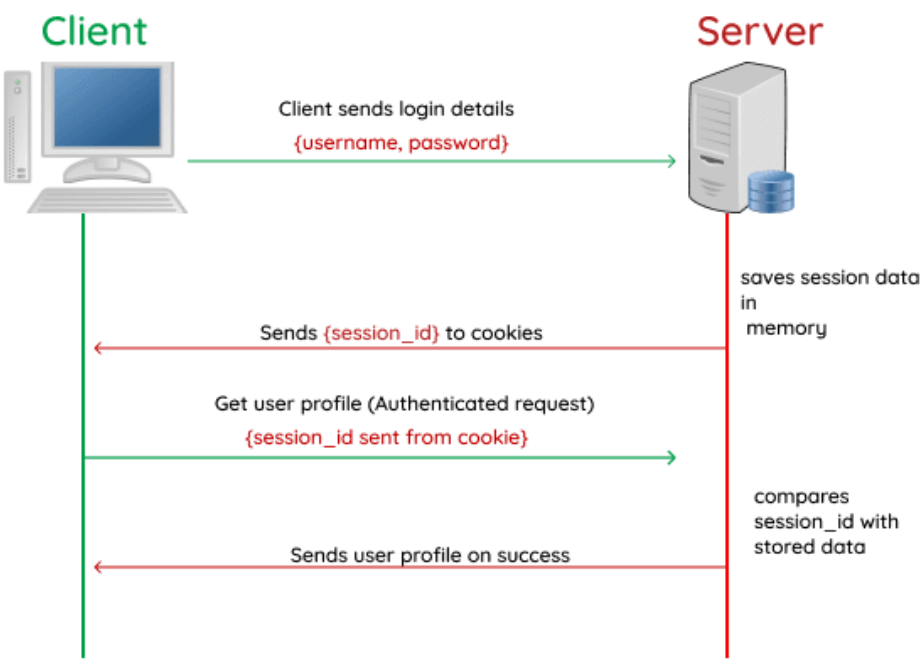


RESTful Web Services (Statelessness)

- Stateful components, such as authentication and authorization services
  - Session based Authentication 세션 인증
  - Token based Authentication 토큰 인증

Session-based authentication

- Stateful authentication technique
  - Use sessions to keep track of the authenticated user



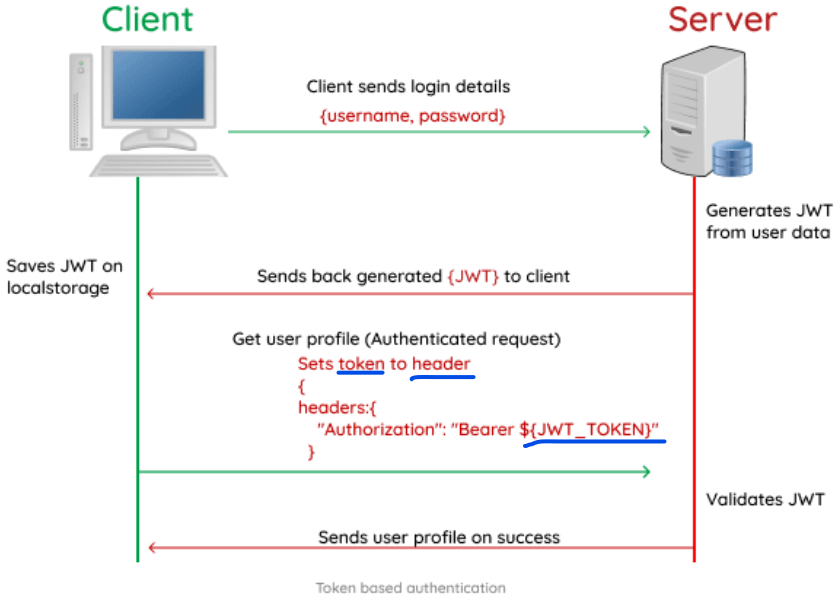
Session based authentication

Token based authentication

- Stateless authentication technique
  - servers use token authentication to check the identity of a user
- The preferred mode of authentication for RESTful APIs

사용자 정보를 담고있고, 암호화 됨

① The user data is encrypted into a JWT (JSON Web Token) with a secret and then sent back to the client
② The JWT is then stored on the client side and sent as a header for every subsequent request
③ The server receives and validates the JWT before proceeding to send a response to the client



RESTful Web Services (Statelessness)

- Advantages of Statelessness
  - Web services can treat each method request independently 각 요청을 독립적으로 처리
  - Web services need not maintain the client's previous interactions. It simplifies the application design 이전 요청은 기억 안함
  - As HTTP is itself a statelessness protocol, RESTful Web Services work seamlessly with the HTTP protocols statelessness 여도 HTTP는 잘 동작

2.6 RESTful Web Services (Caching)

서버에서 온 응답을 클라이언트에 저장 ; 반복 요청 방지

- Caching refers to storing the server response in the client itself, so that a client need not make a server request for the same resource again and again
- A server response should have information about how caching is to be done, so that a client caches the response for a time-period or never caches the server response 정해진 시간 동안만 저장하거나, 캐싱이 금지되는 경우도 있음

▼ Response Headers

```
accept-ranges: bytes
age: 316770
alt-svc: quic=":443"; ma=2592000; v="39,38,37,36,35"
cache-control: public, max-age=31536000
content-encoding: gzip
content-length: 10718
```

RESTful Web Services (Caching)

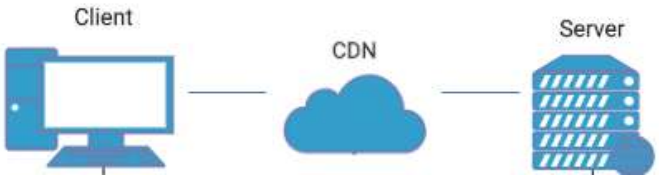
server response

	Header & Description
1	Date Date and Time of the resource when it was created
2	Last Modified Date and Time of the resource when it was last modified
3	Cache-Control Primary header to control caching
4	Expires Expiration date and time of caching

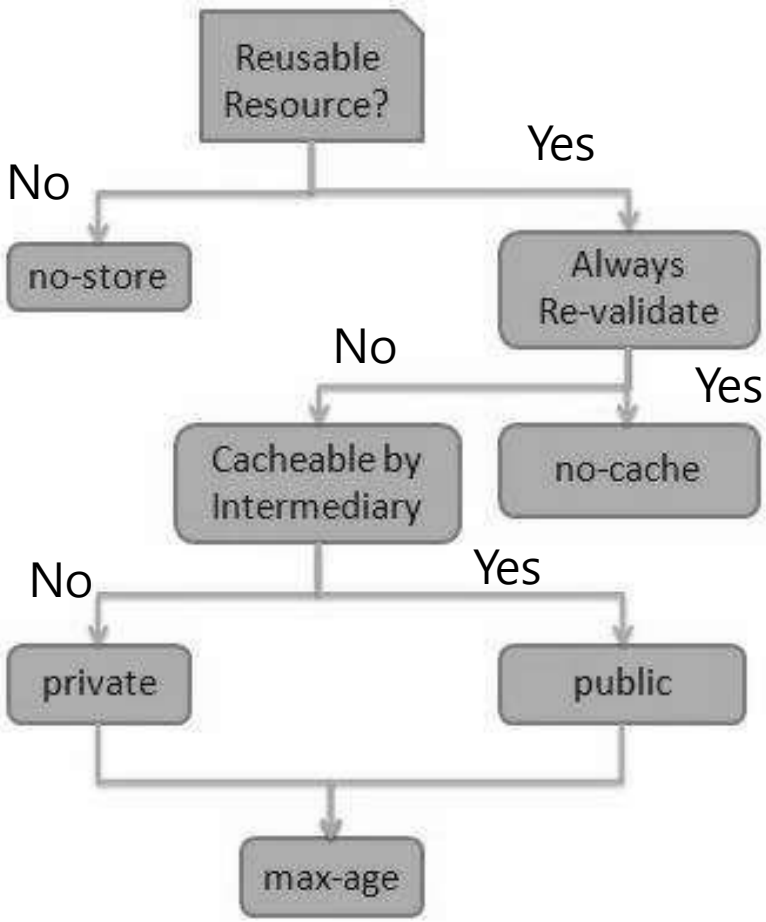
클라와 CDN이

	Header & Description
1	Public 컴포넌트 상관없이 캐싱 Indicates that resource is cacheable by any component
2	Private 클라이언트만 캐싱 Indicates that resource is cacheable only by the client and the server, no intermediary can cache the resource
3	no-store 캐싱 불가 Indicates that a resource is not cacheable
4	max-age 캐싱 유효시간 있음 Indicates the caching is valid up to max-age in seconds
5	must-revalidate Indicates that the response can be reused while fresh. If the response becomes stale, it must be validated with the origin server before reuse

유효시간 지난 캐싱은 변경 여부 검증



RESTful Web Services (Caching)



"no-store"
it disallows browsers and all intermediate caches from storing any versions of returned responses.
used for sensitive data,
such as personal banking details.

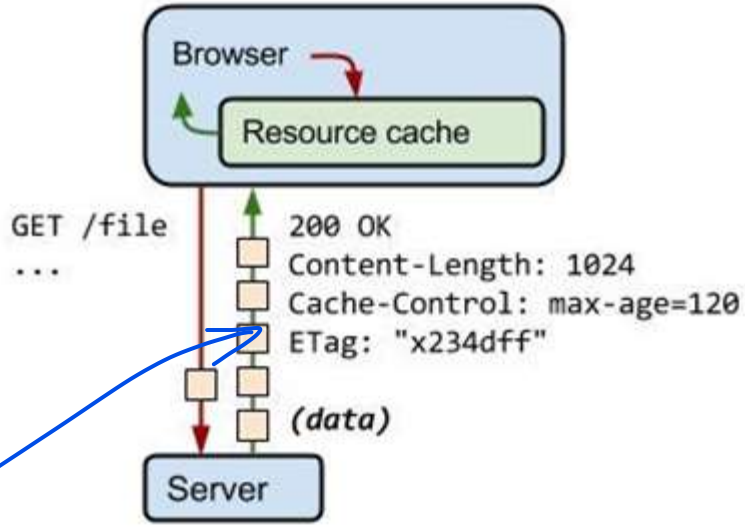
"no-cache"
web browsers might cache the resources but they have to check on every request if the resources have changed

"max-age"

- max-age=120 means that the returned resource is valid for 120 seconds, after which the browser has to request a newer version
- Always keep static contents like images, CSS, JavaScript cacheable, with expiration date of 2 to 3 days
- Dynamic content should be cached for a few hours only

RESTful Web Services (Caching)

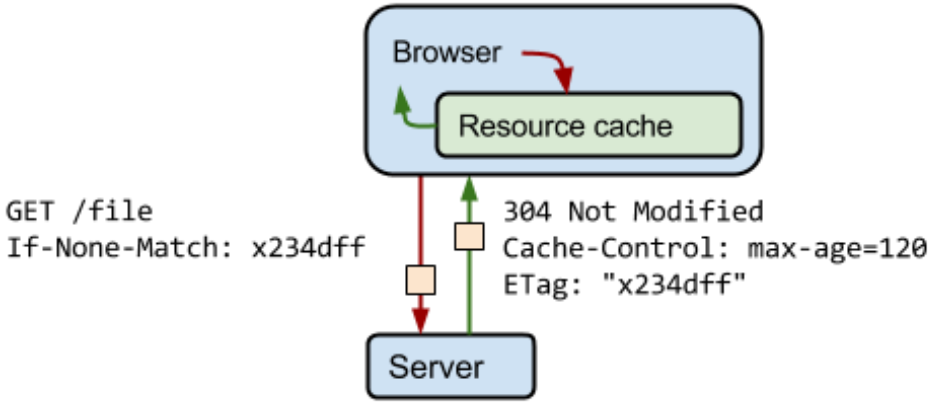
When using the "Cache-Control: no-cache" header, clients must always request revalidation, typically using the "If-None-Match" header (ETag value)



ETag : 자원의 버전

The web server will return the resource's current representation along with its corresponding ETag value(validation token).

An ETag is an identifier assigned by a web server to a specific <u>version</u> of a resource found at a URL.
If the resource representation at that URL ever changes, a new and different ETag is assigned.



If the client wants to retrieve the same URL resource again,
it will first determine whether the local cached version of the URL has expired

요청이 오면 가장 먼저, 캐시의 유효기간이 지났나 확인

- If the URL has not expired, it will retrieve the local cached resource.
- If it is determined that the URL has expired (stale), then the client will contact the server and send its previously saved copy of the ETag along with the request in a "If-None-Match" field

지났으면 변경 여부 검증

변경되었으면 새로 가져옴

- The server may now compare the client's ETag with the ETag for the current version of the resource
- If the ETag values match, meaning that the resource has not changed, then the server may send back a very short response with a **HTTP 304 Not Modified** status. (The 304 status tells the client that its cached version is still good and that it should use that)

2.7 RESTful Web Services (Security)

The best practices to be adhered to while designing a RESTful Web Service:

- Validation 검증
  - Validate all inputs on the server. Protect your server against SQL injection attacks
- No Sensitive Data in the URL URL 속 정보를 암호화
  - Never use username, password or session token in a URL, these values should be passed to Web Service via the POST method
- Restriction on Method Execution 메서드 제한
  - limiting specific HTTP methods for certain URLs. It is possible to disallow or restrict PUT and DELETE methods for specific URLs
- Throw generic Error Messages 에러 메시지 사용
  - A web service method should use HTTP error messages like 403 to show access forbidden, etc.

Sr.No.	HTTP Code & Description
1	**200** **OK** – shows success.
2	**201** **CREATED** – when a resource is successfully created using POST or PUT request. Returns link to the newly created resource using the location header.
3	**204** **NO CONTENT** – when response body is empty. For example, a DELETE request.
4	**304** **NOT MODIFIED** – used to reduce network bandwidth usage in case of conditional GET requests. Response body should be empty. Headers should have date, location, etc.
5	**400** **BAD REQUEST** – states that an invalid input is provided. For example, validation error, missing data.
6	**401** **UNAUTHORIZED** – states that user is using invalid or wrong authentication token.
7	**403** **FORBIDDEN** – states that the user is not having access to the method being used. For example, Delete access without admin rights.
8	**404** **NOT FOUND** – states that the method is not available.
9	**409** **CONFLICT** – states conflict situation while executing the method. For example, adding duplicate entry.
10	**500** **INTERNAL SERVER ERROR** – states that the server has thrown some exception while executing the method.