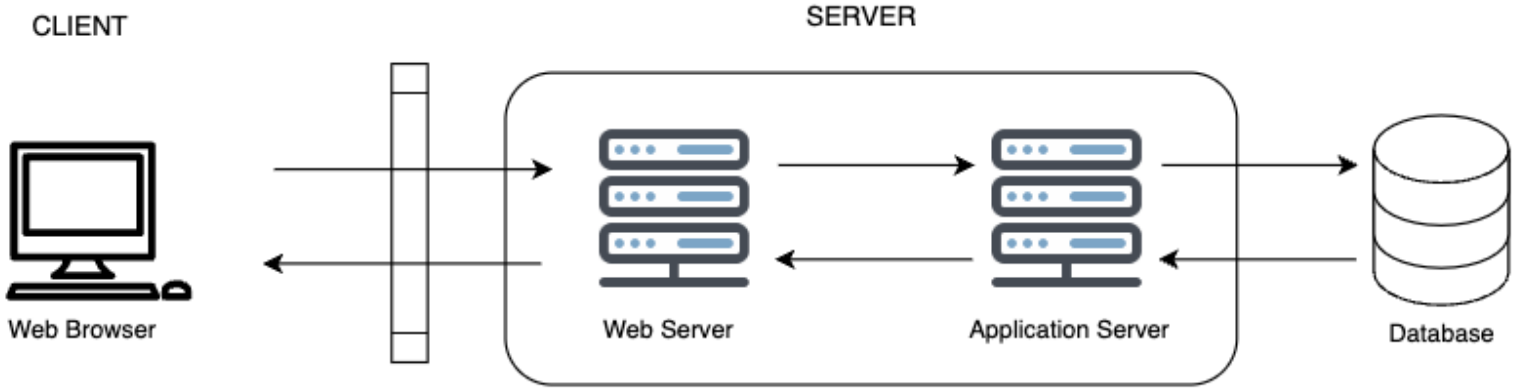
웹개발 스터디 1회차

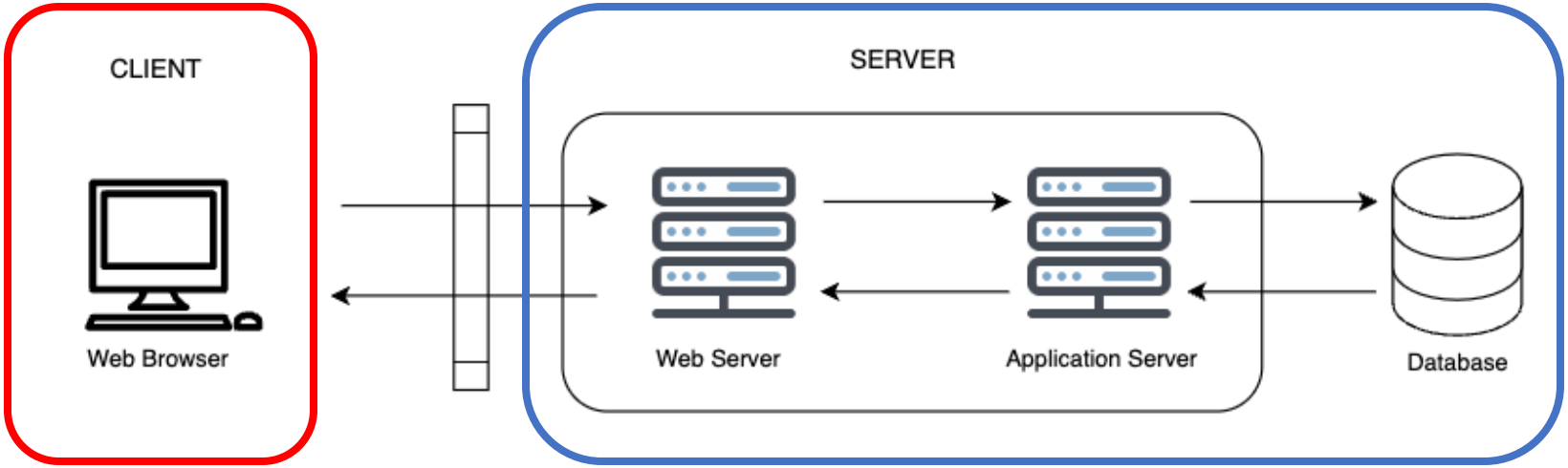
신동민

College of Art & Technology
Chung-Ang University

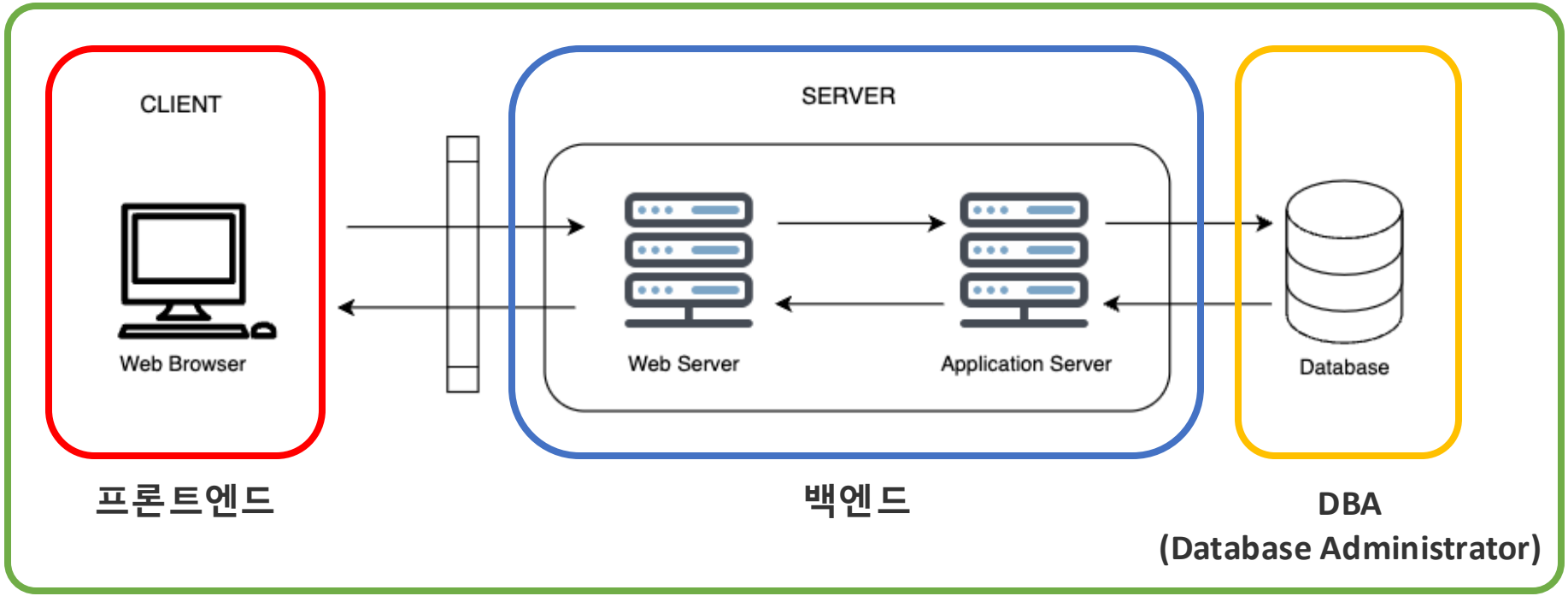
웹 아키텍처의 기본 구조


CLIENT
SERVER
Web Browser
Web Server
Application Server
Database

웹 아키텍처의 기본 구조


CLIENT
Web Browser
SERVER
Web Server
Application Server
Database
프론트엔드
백엔드

웹 아키텍처의 기본 구조



DevOps
(Development + Operations)

클라이언트-서버 상호작용 흐름

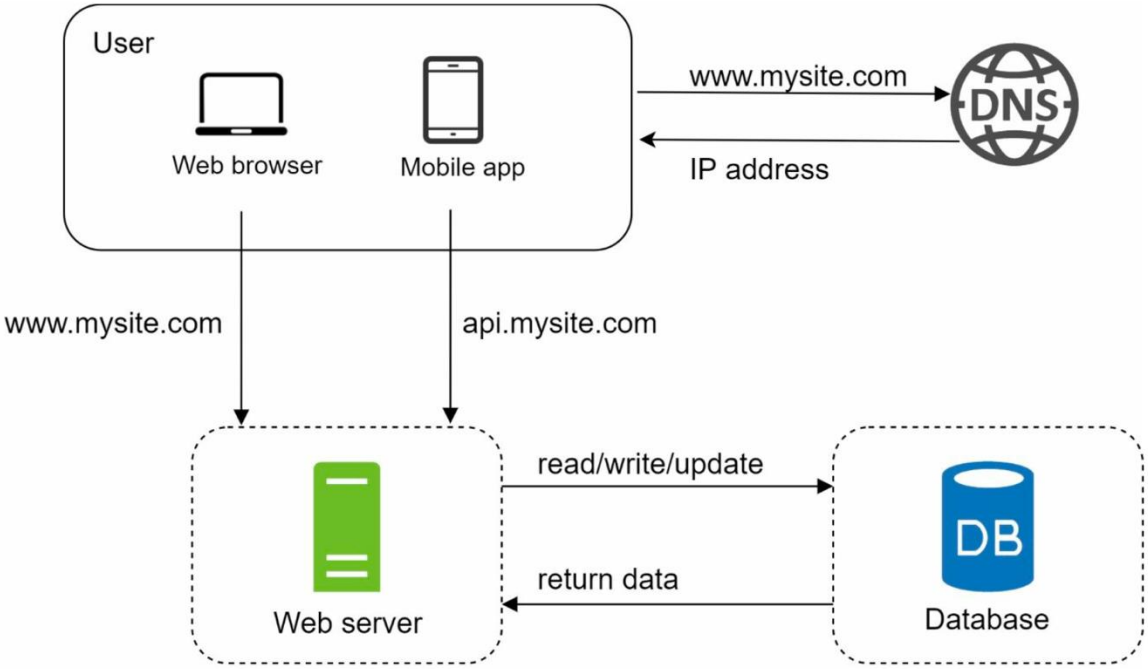


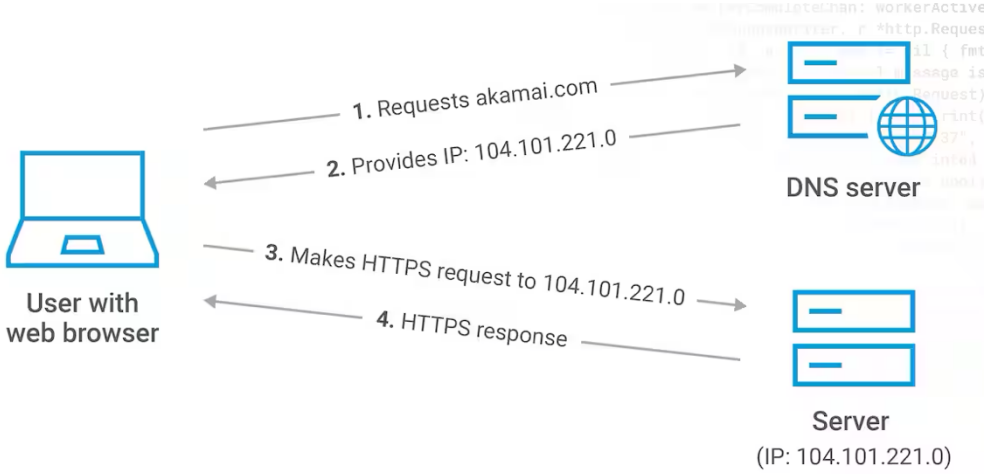
Figure 1-3

데이터베이스 서버 확장

사용자가 늘어남에 따라 서버를 확장

웹/모바일 트래픽 처리용, **데이터베이스** 용으로 **계층을 분리**하여 용도에 따라 독립적으로 분리

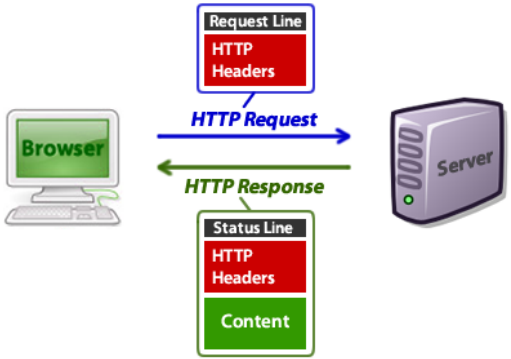
DNS (Domain Name Service)



읽을 수 있는 **도메인 이름**을 컴퓨터가 읽을 수 있는 **IP 주소**로 **변환**해주는 서비스

https://www.akamai.com/ko/glossary/what-are-dns-servers

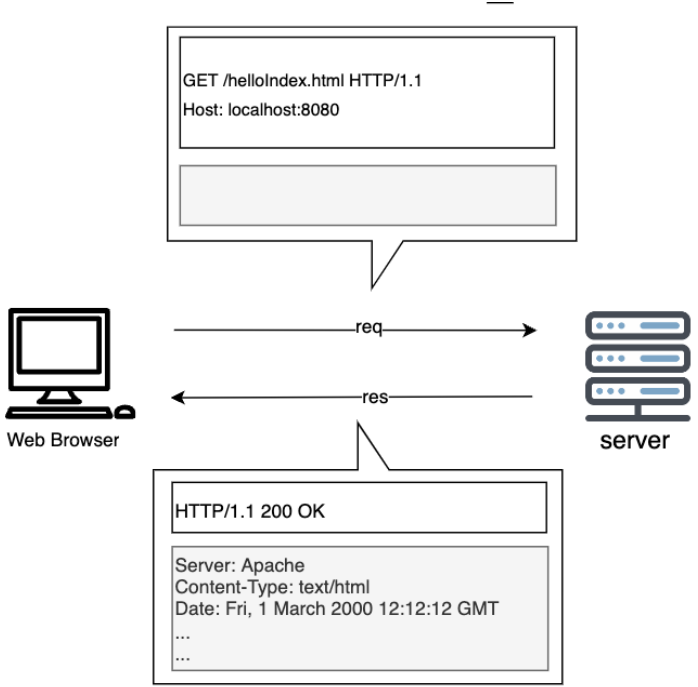
HTTP (HyperText Transfer Protocol)



웹 브라우저(클라이언트)와 **웹 서버** 간의 **통신 규칙**으로
주고 받는 형태가 **요청**과 **응답**으로 나뉘어진다

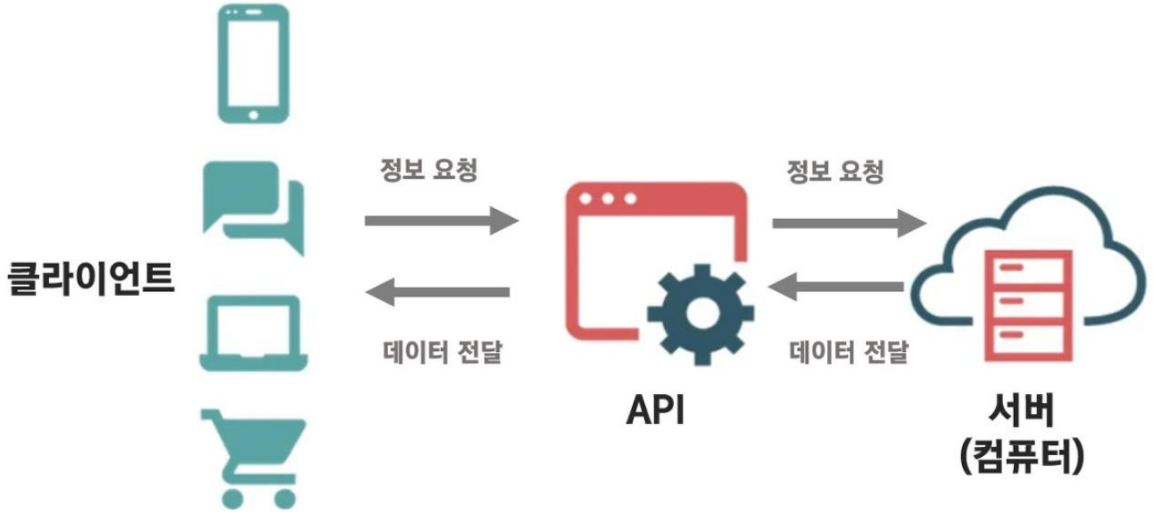
https://medium.com/@lunay0ung/protocol-http%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%BC%EA%B9%8C-84a896c5fc93

HTTP (HyperText Transfer Protocol)

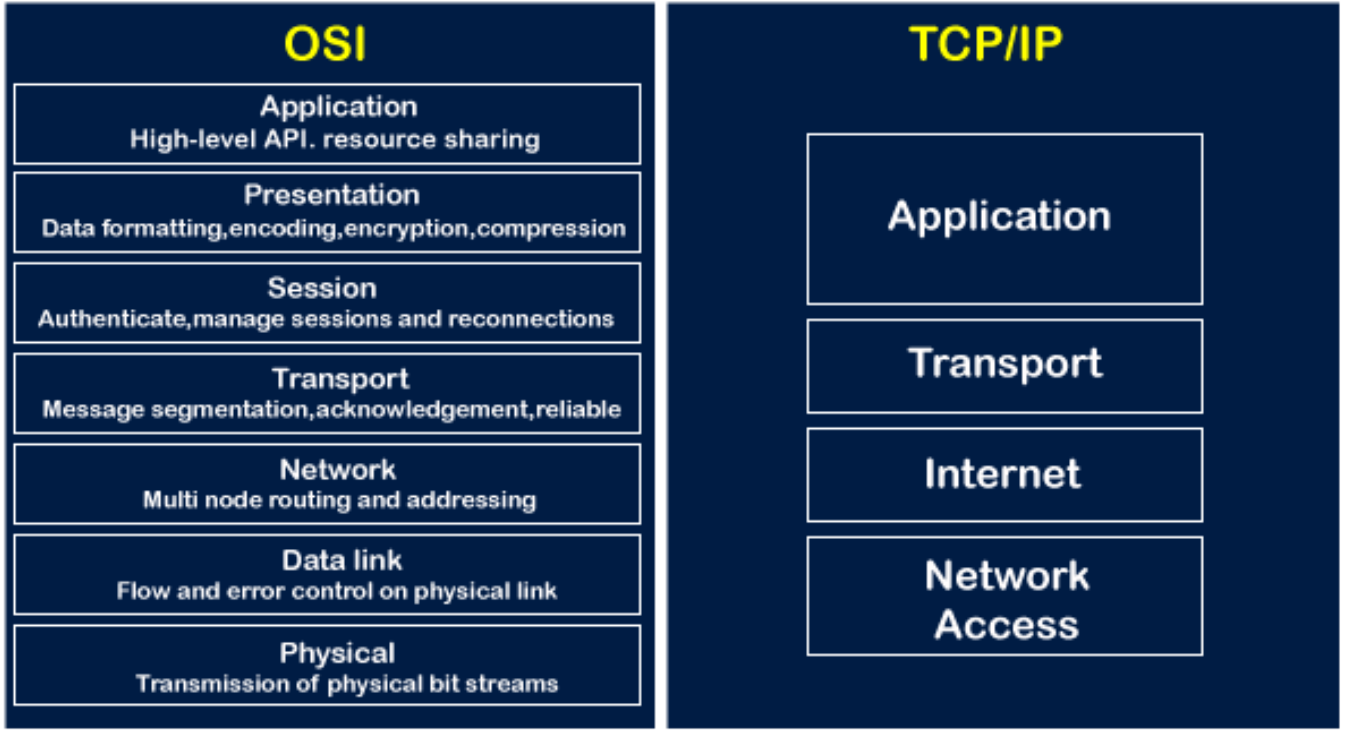


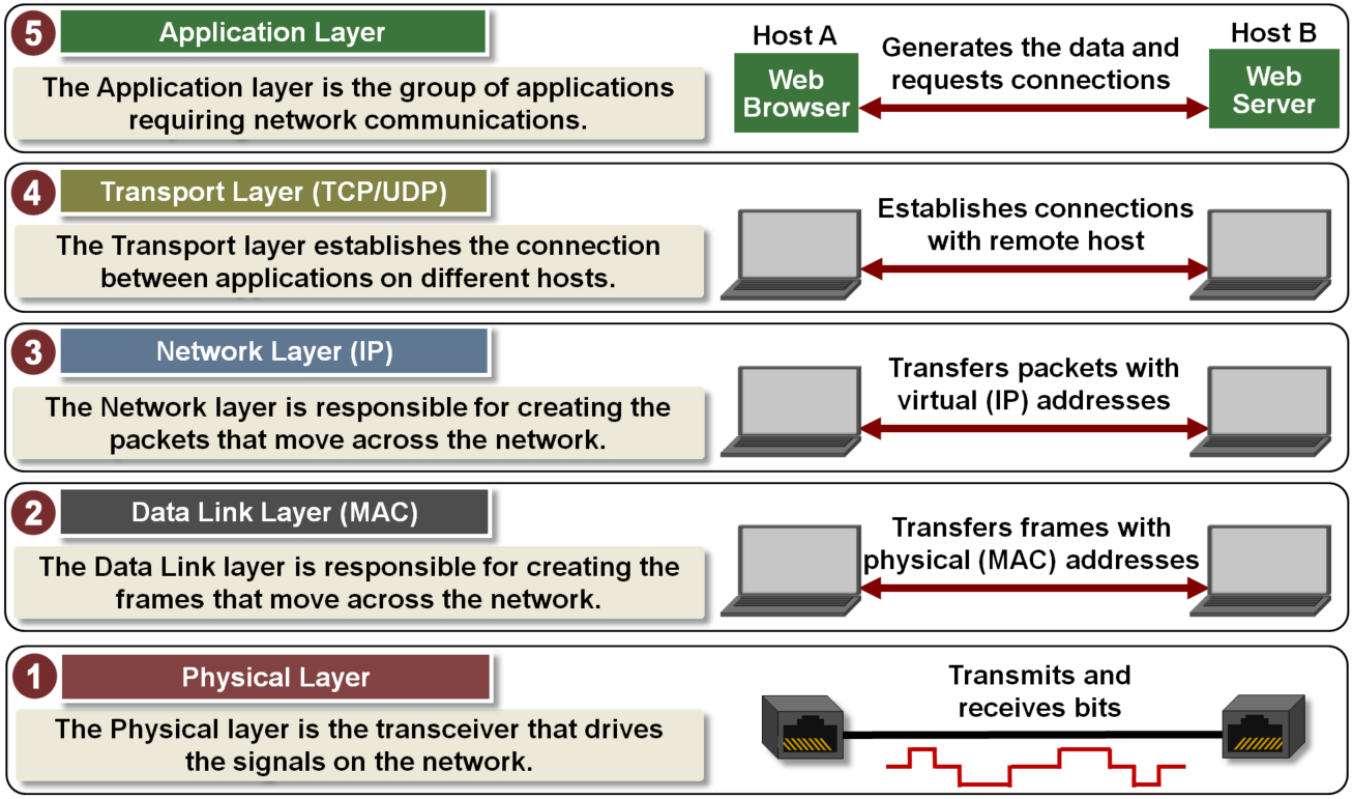
https://medium.com/@lunay0ung/protocol-http%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%BC%EA%B9%8C-84a896c5fc93

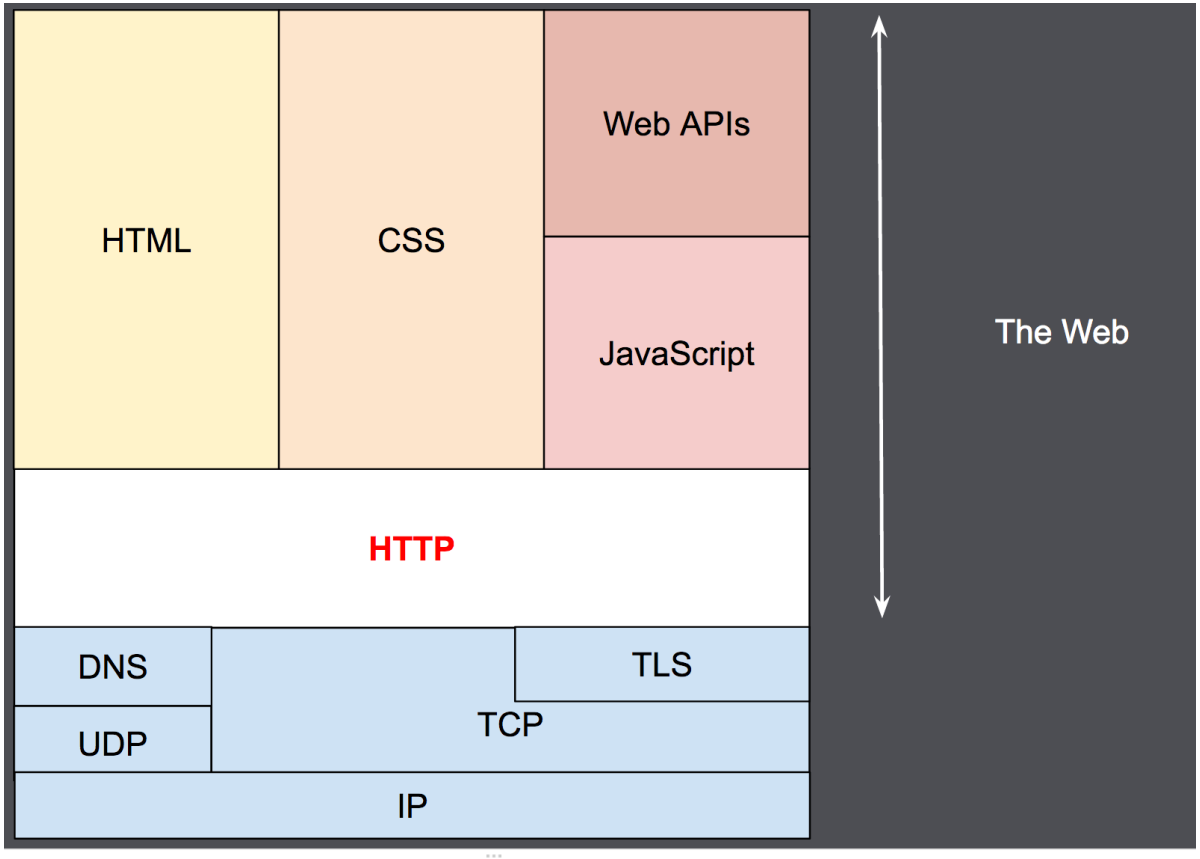
API (Application Programming Interface)

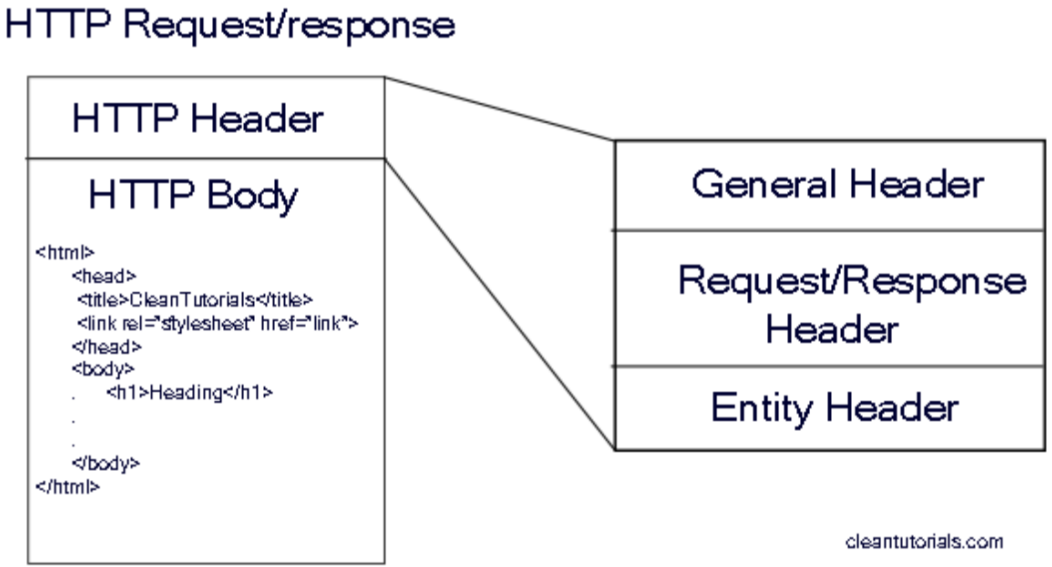


OSI Model & TCP/IP

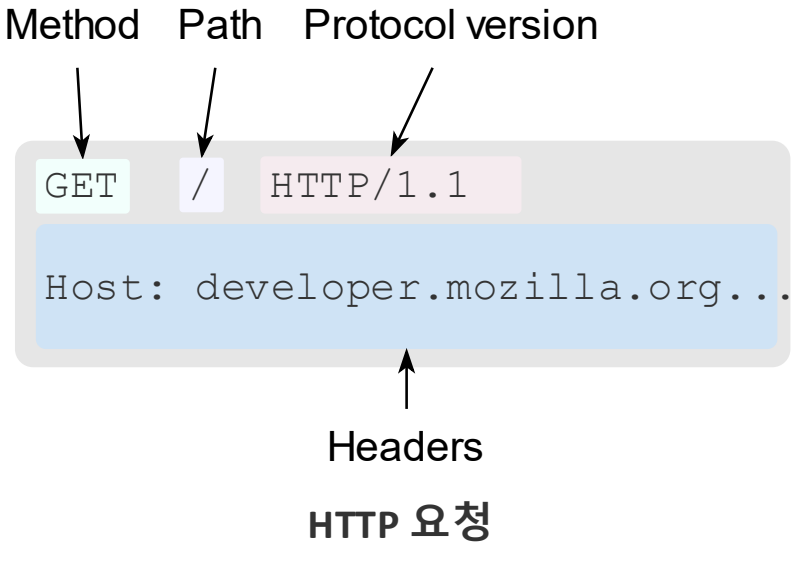



5
Application Layer
The Application layer is the group of applications requiring network communications.
Host A
Web Browser
Generates the data and requests connections
Host B
Web Server
4
Transport Layer (TCP/UDP)
The Transport layer establishes the connection between applications on different hosts.
Establishes connections with remote host
3
Network Layer (IP)
The Network layer is responsible for creating the packets that move across the network.
Transfers packets with virtual (IP) addresses
2
Data Link Layer (MAC)
The Data Link layer is responsible for creating the frames that move across the network.
Transfers frames with physical (MAC) addresses
1
Physical Layer
The Physical layer is the transceiver that drives the signals on the network.
Transmits and receives bits

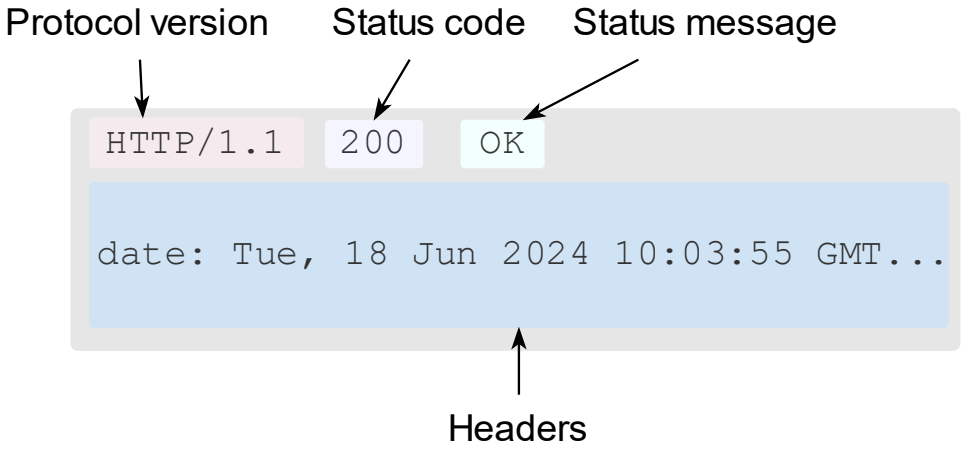

HTML
CSS
Web APIs
JavaScript
The Web
HTTP
DNS
TLS
UDP
TCP
IP


HTTP Request/response
HTTP Header
HTTP Body
<html>
<head>
<title>CleanTutorials</title>
<link rel="stylesheet" href="link">
</head>
<body>
<h1>Heading</h1>
</body>
</html>
General Header
Request/Response Header
Entity Header
cleantutorials.com

HTTP 구조

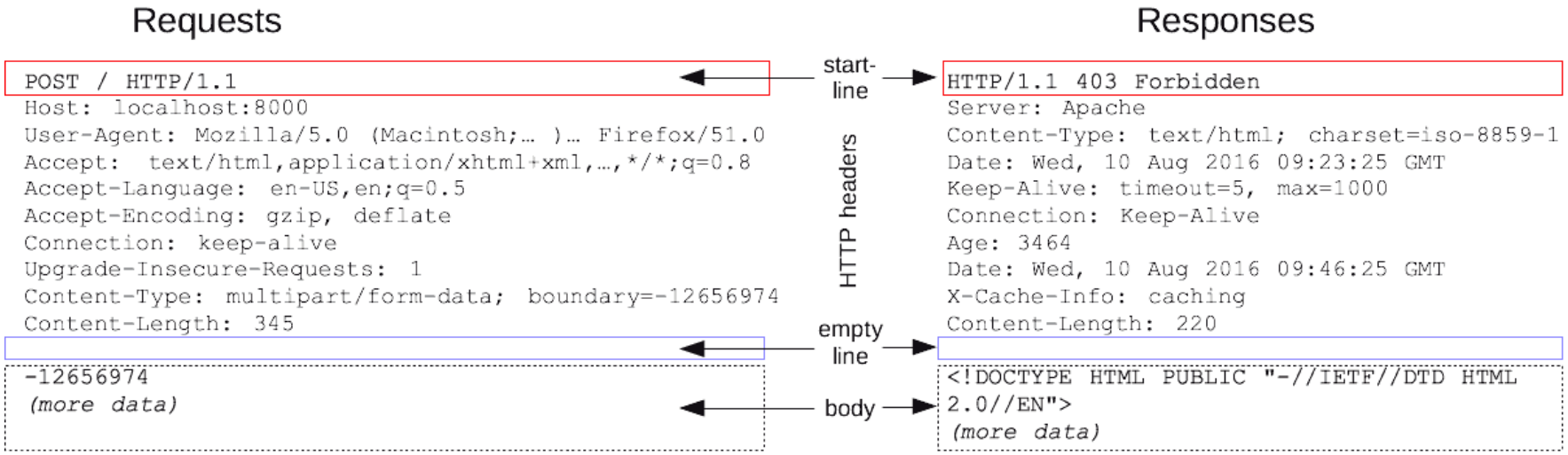

Method
Path
Protocol version
GET
/
HTTP/1.1
Host: developer.mozilla.org...
Headers

HTTP 요청


Protocol version
Status code
Status message
HTTP/1.1
200
OK
date: Tue, 18 Jun 2024 10:03:55 GMT...
Headers

HTTP 응답

요청/응답 구조


Requests
Responses
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;…)… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=--12656974
Content-Length: 345
-12656974
(more data)
start-line
HTTP headers
empty line
body
HTTP/1.1 403 Forbidden
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 10 Aug 2016 09:23:25 GMT
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
Age: 3464
Date: Wed, 10 Aug 2016 09:46:25 GMT
X-Cache-Info: caching
Content-Length: 220
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
(more data)


{ REST }
DELETE
GET
POST
PATCH
PUT

REST(Representational State Transfer)

- DELETE → 자원을 삭제할 때 사용한다
- GET → 자원을 받아오기만 할때 사용한다
- POST → 새로운 자원을 추가할 때 사용한다
- PATCH → 특정 자원의 데이터를 부분적으로 변경할 때 사용한다
- PUT → 존재하는 자원을 변경 할 때 사용한다

주요 HTTP 메서드

HTTP 메소드	RFC	요청에 Body가 있음	응답에 Body가 있음	안전	멱등(Idempotent)	캐시 가능
GET	RFC 7231	아니요	예	예	예	예
HEAD	RFC 7231	아니요	아니요	예	예	예
POST	RFC 7231	예	예	아니요	아니요	예
PUT	RFC 7231	예	예	아니요	예	아니요
DELETE	RFC 7231	아니요	예	아니요	예	아니요
CONNECT	RFC 7231	예	예	아니요	아니요	아니요
OPTIONS	RFC 7231	선택 사항	예	예	예	아니요
TRACE	RFC 7231	아니요	예	예	예	아니요
PATCH	RFC 5789	예	예	아니요	아니요	예

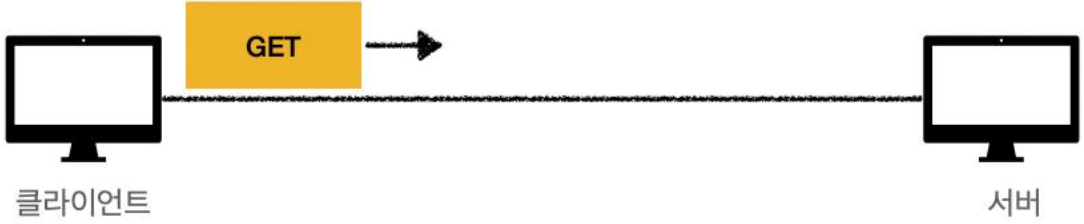
GET

리소스 조회1 - 메시지 전달

```
GET /members/100 HTTP/1.1
Host: localhost:8080
```

/members/100

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



GET

리소스 조회2 - 서버도착

```
GET /members/100 HTTP/1.1
Host: localhost:8080
```

/members/100

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

GET

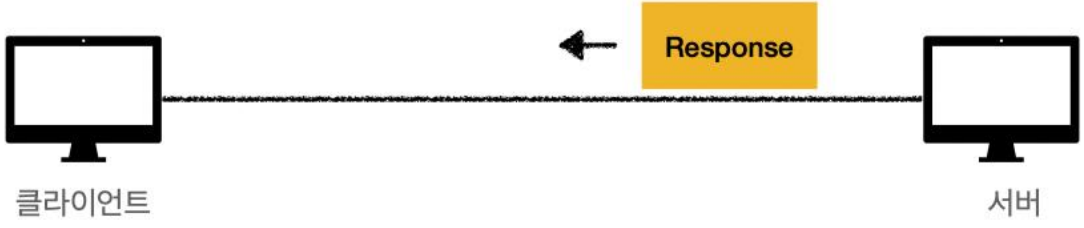
리소스 조회3 - 응답 데이터

응답 데이터

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 34

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

/members/100

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

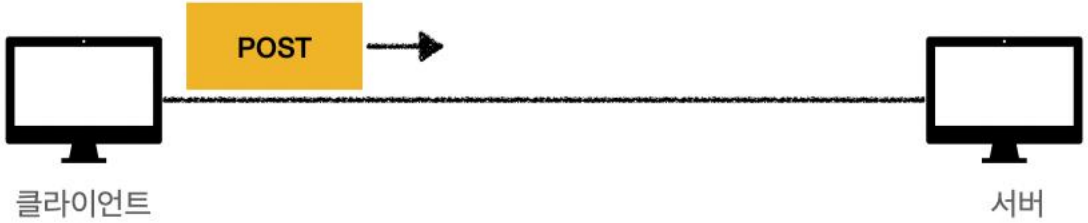


POST

리소스 등록1 - 메시지 전달

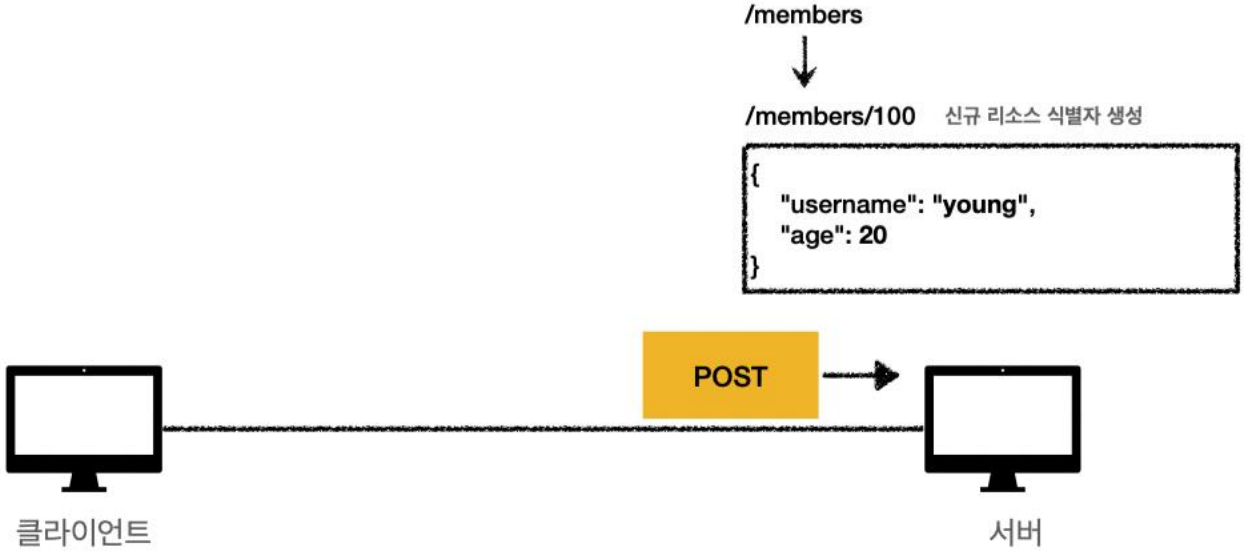
```
POST /members HTTP/1.1
Content-Type: application/json

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

/members



POST

리소스 등록2 - 신규 리소스 생성



POST

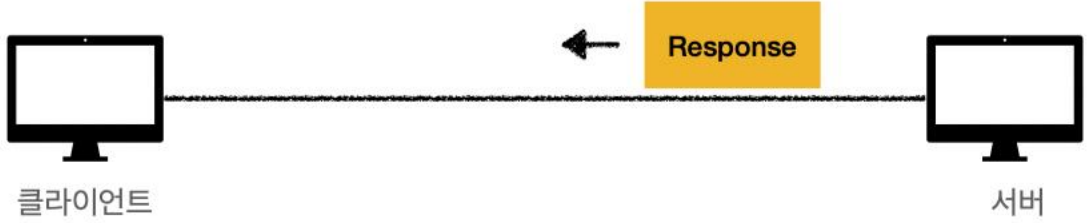
리소스 등록3 - 응답 데이터

응답 데이터

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 34
Location: /members/100

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

/members/100

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



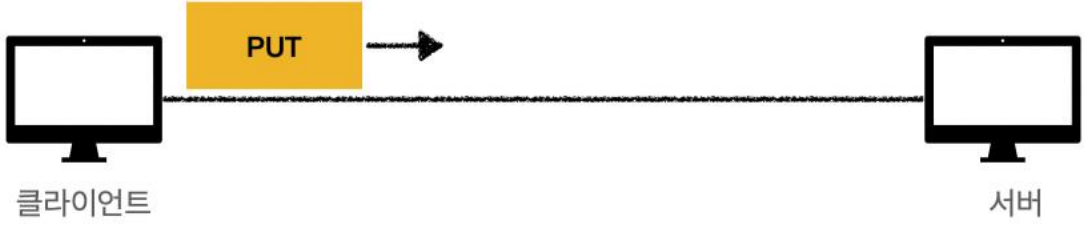
PUT

리소스가 있는 경우1

```
PUT /members/100 HTTP/1.1
Content-Type: application/json

{
    "username": "old",
    "age": 50
}
```

/members/100

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



PUT

리소스가 있는 경우2

리소스 대체

/members/100

```
{
    "username": "old",
    "age": 50
}
```

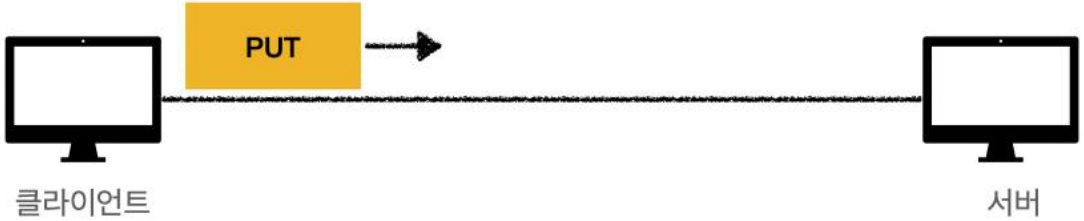
PUT

리소스가 없는 경우1

```
PUT /members/100 HTTP/1.1
Content-Type: application/json

{
   "username": "old",
   "age": 50
}
```

이런 리소스는 없음

/members/100



PUT

리소스가 없는 경우2

신규 리소스 생성

/members/100

```
{
    "username": "old",
    "age": 50
}
```

PUT

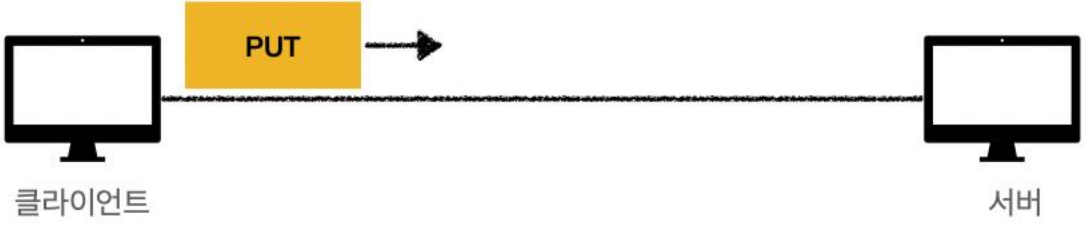
주의! - 리소스를 완전히 대체한다1

```
PUT /members/100 HTTP/1.1
Content-Type: application/json

{
   "age": 50
}
```

username 필드 없음

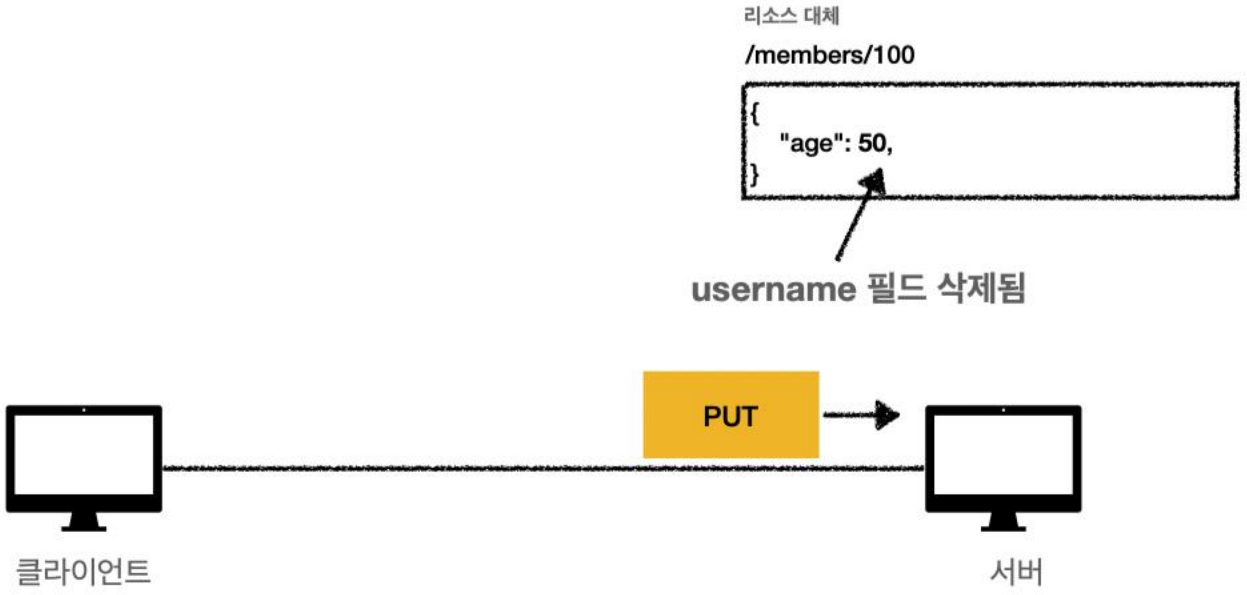
/members/100

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



PUT

주의! - 리소스를 완전히 대체한다2



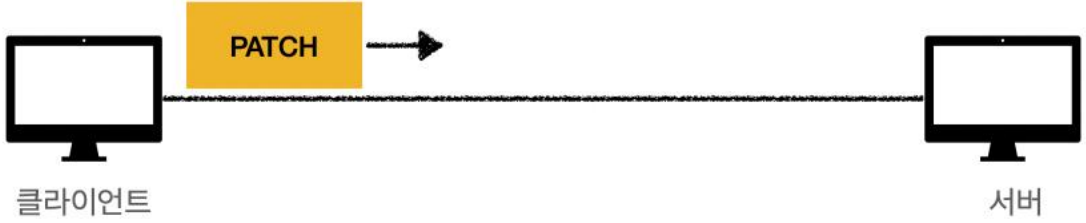
PATCH

리소스 부분 변경1

```
PATCH /members/100 HTTP/1.1
Content-Type: application/json

{
   "age": 50
}
```

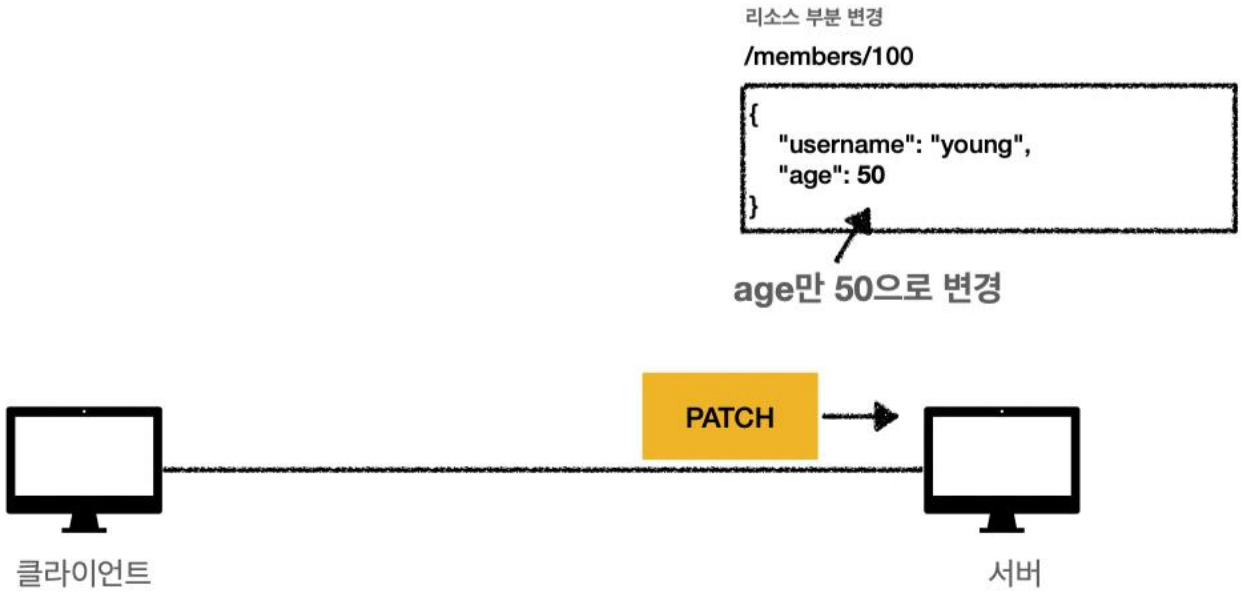
username 필드 없음

```
/members/100

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



PATCH

리소스 부분 변경2



DELETE

리소스 제거1

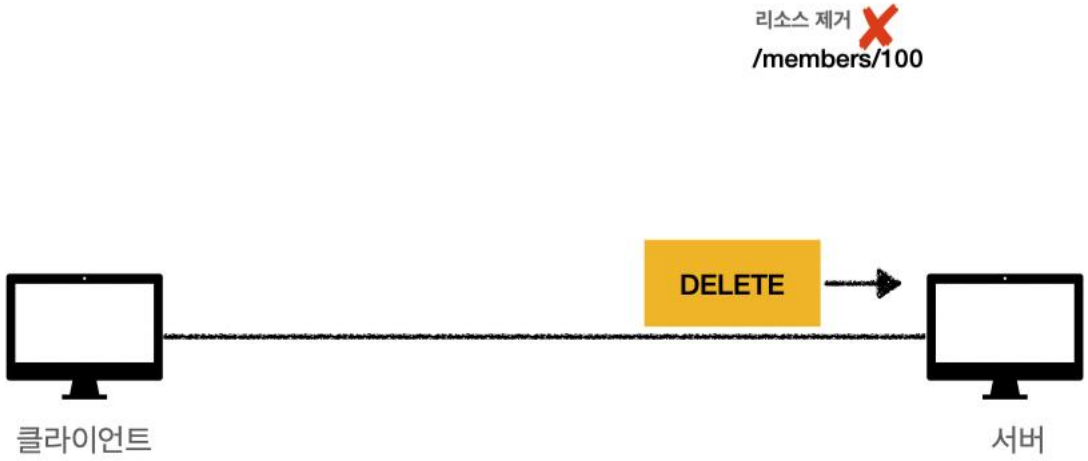
```
DELETE /members/100 HTTP/1.1
Host: localhost:8080
```

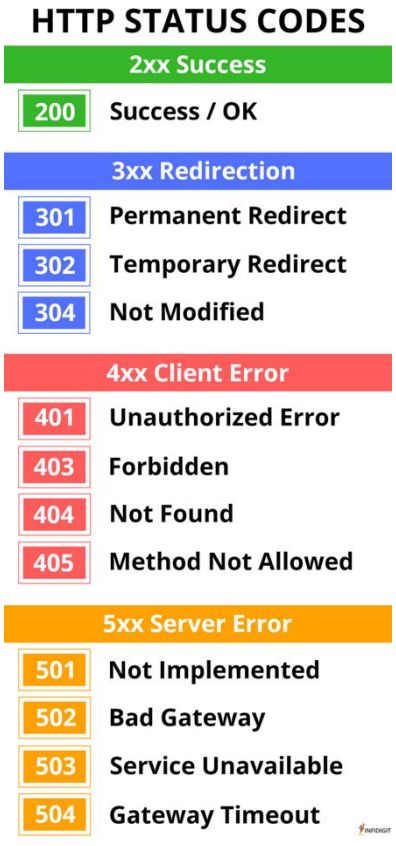
/members/100

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

DELETE

리소스 제거2




HTTP STATUS CODES
2xx Success
200 Success / OK
3xx Redirection
301 Permanent Redirect
302 Temporary Redirect
304 Not Modified
4xx Client Error
401 Unauthorized Error
403 Forbidden
404 Not Found
405 Method Not Allowed
5xx Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout

1. 2xx: 성공(Success)

200 OK
요청이 성공적으로 처리되어, 요청한 데이터가 정상적으로 반환됨. 가장 일반적으로 볼 수 있는 성공 코드

2. 4xx: 클라이언트 오류(Client Error)

400 Bad Request
요청이 잘못되어 서버가 이해할 수 없음(문법 오류, 잘못된 파라미터 등)

401 Unauthorized
인증이 필요하지만 제공되지 않았거나, 인증 정보가 잘못됨. 로그인 필요

403 Forbidden
서버가 요청을 이해했으나, 권한이 없어 거부함(접근 제한)

404 Not Found
요청한 리소스를 찾을 수 없음. 잘못된 URL, 삭제된 페이지 등에서 발생

405 Method Not Allowed
요청에 사용된 HTTP 메서드(GET, POST 등)가 허용되지 않음

429 Too Many Requests
너무 많은 요청을 짧은 시간 내에 보내서, 서버가 일시적으로 차단함(주로 rate limiting)

3. 5xx: 서버 오류(Server Error)

500 Internal Server Error
서버 내부에서 예기치 못한 오류가 발생함. 원인을 서버가 명확히 설명하지 못함

프론트엔드와 백엔드 역할

구분	프론트엔드(Frontend)	백엔드(Backend)
역할	사용자와 직접 상호작용, 화면(UI/UX) 구현	데이터 처리, 서버 로직, 데이터베이스 관리, API 설계
기술 스택	HTML, CSS, JavaScript, React, Vue, Angular 등	Java, Python, Node.js, Spring, FastAPI, RDBMS(MySQL 등)
주요 업무	웹페이지 레이아웃, 입력/출력, 사용자 경험 개선	서버 구축, 데이터 저장/조회, 인증/보안, 성능 최적화, API 개발
특징	클라이언트(브라우저)에서 실행, 사용자 경험에 중점	서버에서 실행, 보안/성능/확장성에 중점

기능명세서

기능 명세서

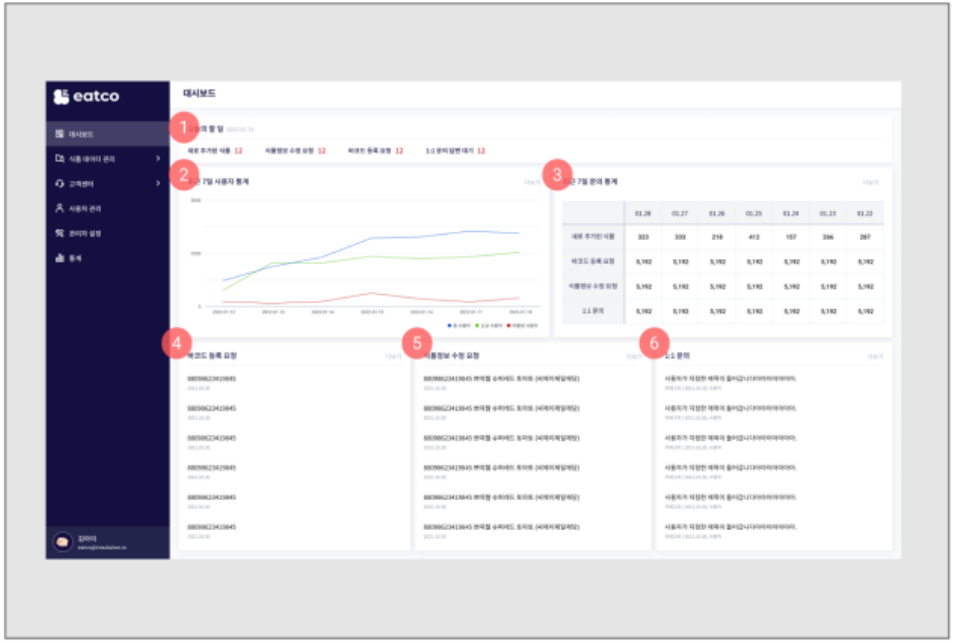
Table + Filter Sort New

기능 명세서

Aa 구분	주 기능	상세 기능	설명	가능여부
1. 회원가입	1.1 SNS 회원가입	1.1.1 카카오톡, 구글, Apple	Apple 로그인은 iOS에서만	
	1.2 프로필 설정	1.2.1 프로필 사진 / 캐릭터 선택	- 카카오톡, 구글 프로필 사진 가져오기 - 캐릭터 선택하기 (동숲 동물 중)	
		1.2.2 주민 대표 이름 작성		
OPEN		1.2.3 섬 이름 작성		
		1.2.4 자기 소개 작성		
2. 로그인	2.1 SNS 로그인	2.1.1 카카오톡, 구글, Apple		
3. 홈	3.1 동숲 뉴스 미리보기	3.1.1 동숲, 닌텐도 관련 뉴스 보기	좌우 슬라이드 형식	확인 필요
	3.2 일상 게시글 미리보기	3.2.1 lastest/ best 일상 게시글 항목 보기	좌우 슬라이드 형식 lastest 게시글 : 최신순 best 게시글 : 추천순	
		3.2.2 lastest/ best 일상 게시글 모두 보기		
	3.3 팔아요 item 미리보기	3.3.1 lastest “팔아요” 항목 보기	좌우 슬라이드 형식	
		3.3.2 팔아요 item 모두 보기	버튼 클릭	
	3.4 구해요 item 미리보기	3.4.1 lastest “구해요” 항목 보기	좌우 슬라이드 형식	

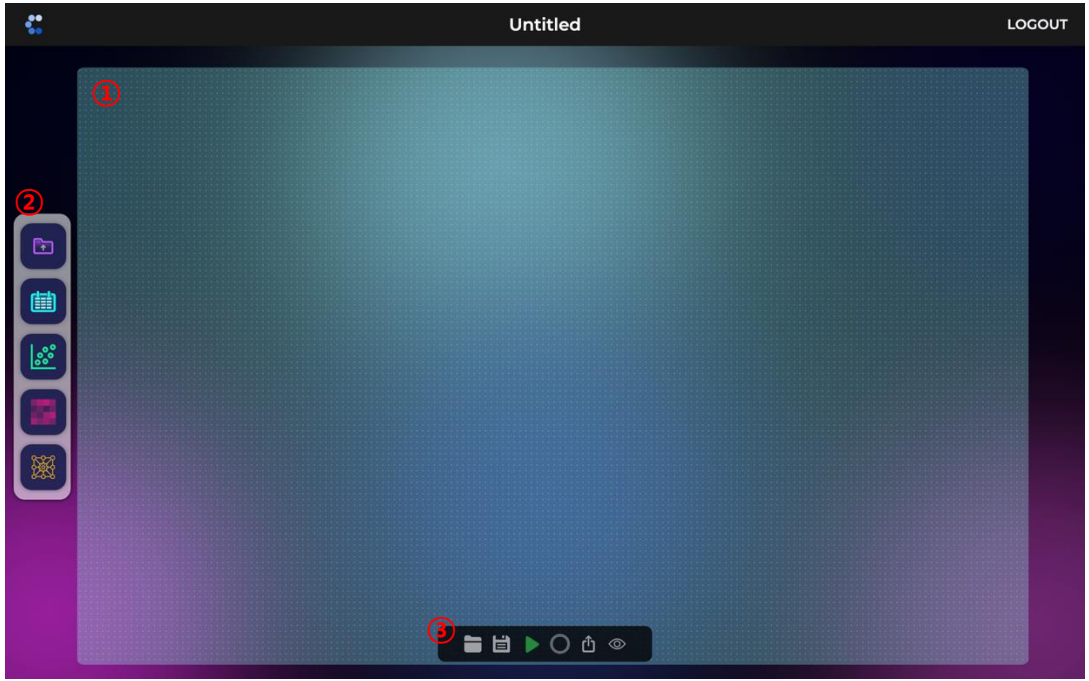
화면설계서

Page Title	대시보드 기본	Screen ID	UI-I-	Author	최은솔	Date	2022.03.18
Screen Path	대시보드						



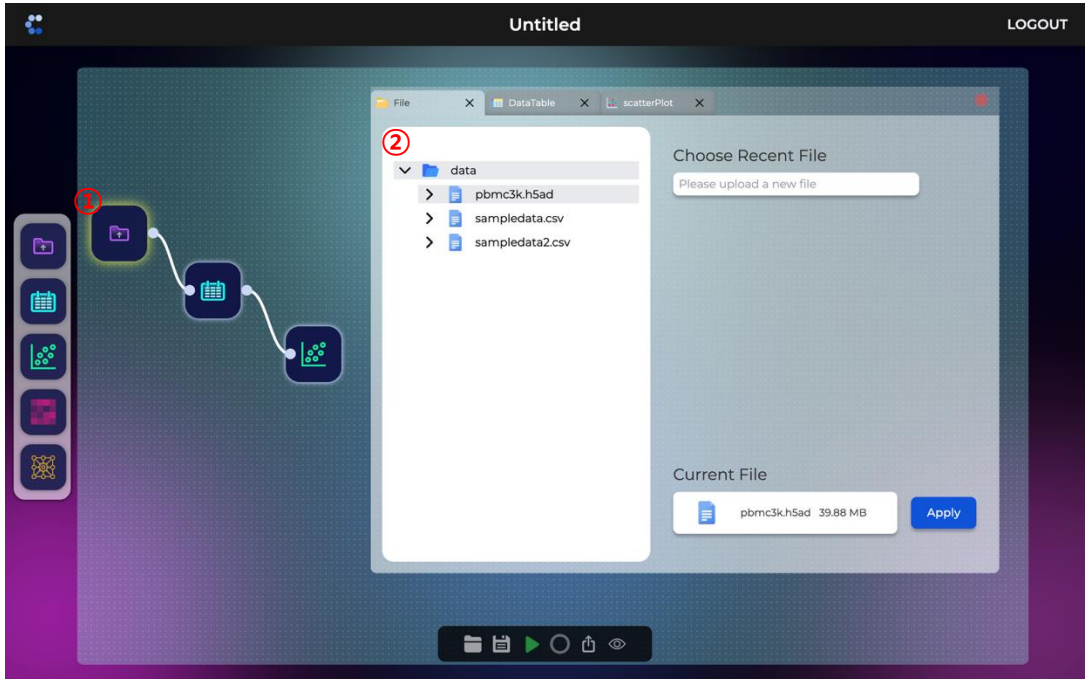
Description	
1	처리되지 않은 문의 내역 갯수를 보여줌 숫자를 누르면 해당 문의 페이지로 이동
2	오늘 기준 7일 동안의 사용자 통계 그래프
3	오늘 기준 7일 동안의 문의 통계 그래프
4	답변되지 않은 문의를 최근 순서대로 6개 표시
5	답변되지 않은 문의를 최근 순서대로 6개 표시
6	답변되지 않은 문의를 최근 순서대로 6개 표시
Check Point	
더보기 누르면 각 해당 페이지로 이동함	

Workflow Page



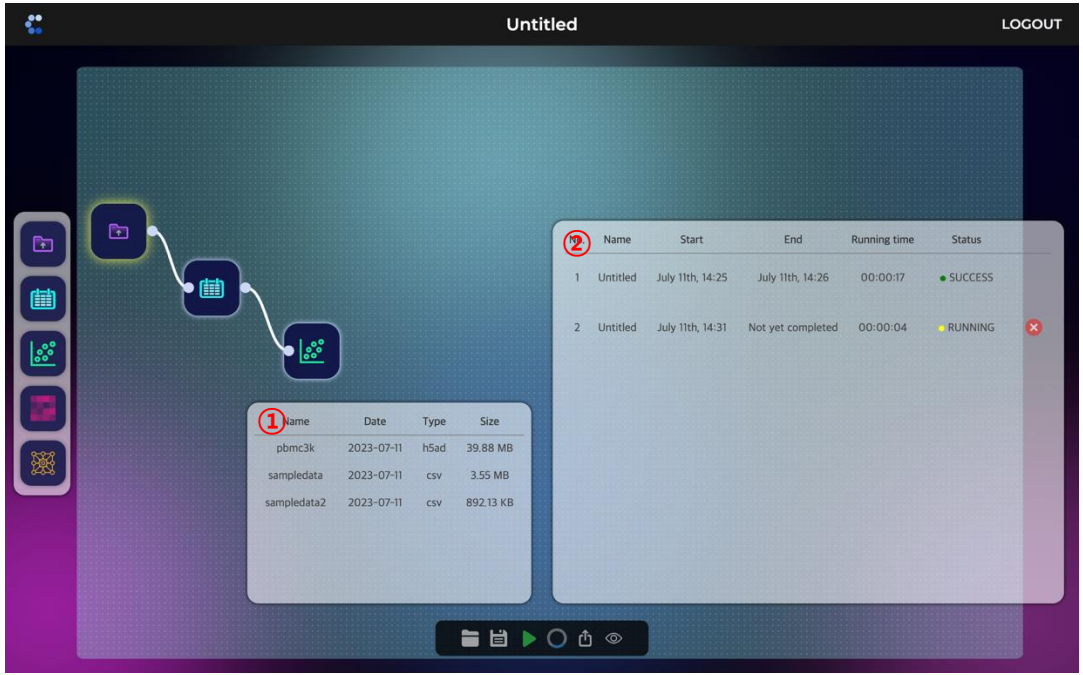
설명	
①	**비주얼 프로그래밍** 노드 간의 연결을 통해 데이터 분석을 진행
②	**노드 리스트** 각 노드를 생성할 수 있는 리스트
③	**컨트롤 바** 저장, 실행 등 기능들을 포함하고 있는 바

Workflow Page : File Node



설명
① **File 노드** File 컴포넌트와 대응되는 노드
② **File 컴포넌트** 업로드 한 File을 데이터 분석에 사용하는 기능을 수행

Workflow Page : File / Job Management



설명	
①	**File** 사용자가 업로드 한 File을 볼 수 있는 리스트
②	**Job Management** 사용자가 실행한 Job의 정보를 확인 및 취소 기능

API 명세서

auth	URL	Method	URL Params	Data Params	Success Response	Error Response
login	/auth/auth/login	post		object: `{ oAuth: int, email: string }`	code:200 Content: `{ loginSuccess: true }`	code:500 Content: `{ loginSuccess: false }`
logout	/auth/auth/logout	get			code:200 Content: `{ logoutSuccess : true }`	code:500 Content: `{ logoutSuccess: false }`
deleteuser	/auth/auth/deleteuser	get			code:200 Content: `{ deleteSuccess : true }`	code:500 Content: `{ deleteSuccess: false }`