



Network 통신(이론)

배 희호 교수
경북대학교
소프트웨어융합과과



Network Programming



■ 기초 지식

- Internet으로 연결되어 있는 Computer끼리 의사소통을 하기 위해서 TCP(Transmission Control Protocol) 또는 UDP(User Datagram Protocol)를 사용
- Program 작성은 Application 계층에서 하기 때문에 TCP와 UDP에 대해 자세히 알아둘 필요는 없지만, 그래도 TCP를 사용할 때 사용하는 API와 UDP를 사용할 때 사용하는 API도 다르고, 차이점 정도는 알아 두어야 함

Application (HTTP, ftp, telnet, ...)
Transport (TCP, UDP, ...)
Network (IP, ...)
Link (device driver, ...)



IP Address



- IP 주소(Internet Protocol Address, IP Address)는 Computer Network 상에서 각 장치들의 고유 식별 번호
- 쉽게 말하면, 장치들의 전화번호라고 생각하면 됨
- IP Address의 형식은 예를 들어,
2606:4700:4700:0000:0000:0000:0000:1111와 같음
- 이러한 번호들의 나열이 각 장치들의 고유한 주소인 것
- 각각의 장치들은 IP Address를 이용하여 Data를 전송하기도 전달받기도 함
- 하지만 IP Address 형식은 외우거나 구별하기 어렵기 때문에 , DNS를 통해 구별하기 쉬운 Domain Name으로 변환



Domain Name



- Domain Name는 외우거나 식별하기 어려운 IP Address(예 :240.10.20.1)를 example.com처럼 기억하기 쉽게 만들어주는 Network Host 이름을 의미함
- IP Address와 Domain Name을 비교하면 다음과 같음

IP Address	Domain Name
115.68.24.88	opentutorials.org
220.95.233.172	naver.com
114.108.157.19	daum.net



Port Address



- Port Address는 Software에서 Network Service나 특정 Process를 식별하는 논리 단위
- Port는 번호로 구별되며 이 번호를 Port Number라고 함
- Port Number는 IP Address와 함께 쓰여 해당하는 Protocol에 의해 사용
- 예) ftp://000.000.000.000:21라는 URL에서 맨 마지막에 있는 21이 Port Number
 - 20 : FTP(Data), 21 : FTP(Control)
 - 22 : SSH
 - 23 : Telnet
 - 53 : DNS
 - 80 : 월드 와이드 웹, HTTP
 - 119 : NNTP
 - 443 : TLS/SSL 방식의 HTTP



Sever/Client 통신

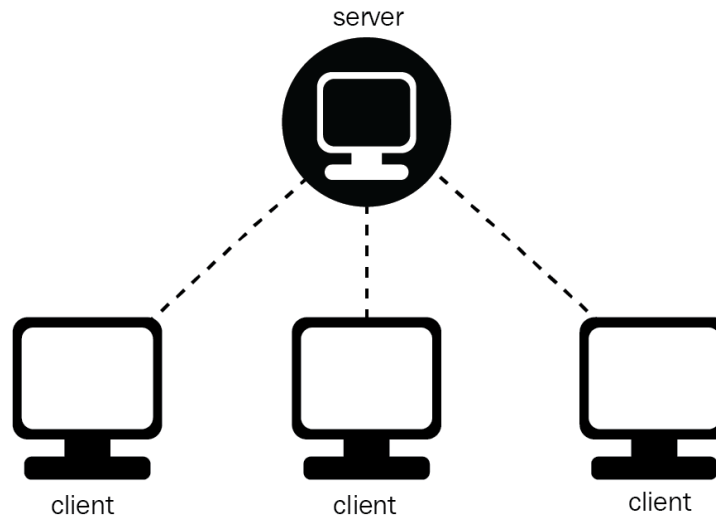


■ Client

- Server에게 Service를 요구하는 사용자나 Computer

■ Server

- Server Program이 실행되고 있는 Hardware
- 사용자의 Request로 Service를 수행하고 Network를 관리, 제어, 감시하며 File이나 Program, 혹은 Hardware Resource를 공유할 수 있도록 Service





Sever/Client 통신



- Server/Client 통신은 Server에 있는 풍부한 Resource들과 Service를 통합된 방식으로 제공받기 위한 통신
- Client에서는 Service를 Request하고 Server에서는 Service에 Response하는 형태를 가지는 Network Model, 혹은 방식을 가리켜 **Server/Client Model**이라고 함
- 하나의 Server Program이 다수의 Client의 응용 Service를 제공하는 Server/Client Model은 Internet 응용 환경에서 가장 보편화된 연결 설정 방식
- 일반적으로 Network Service를 받기 위하여 Client가 통신을 시작
 - Client는 Server에 접속을 시도하고 그 연결 결과를 기다리든지, 어떤 Service를 Request하고 Response를 기다림
 - Client의 이와 같은 Request에 대하여 Server가 Response를 보내는 방식으로 동작이 이루어짐



Sever/Client 통신



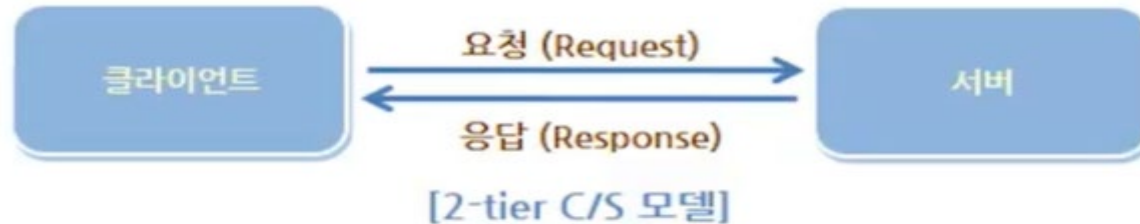
- Client/Server 설계
 - Server의 Port는 고정적
 - Server의 Port가 바뀌면 Client가 해당 Server를 찾기 어려움
 - Web Server는 Port 80, FTP Server는 Port 21로 고정
- 사용자가 Server Application 개발 시 1024 값보다 큰 Port를 선정하고, 이 Port를 Socket으로 열어 Client가 연결할 수 있도록 대기
- Client는 의미 없는 Port(현재 System에서 사용하지 않는 Port)를 Socket으로 연결하고, Web Application이 있는 Server의 IP와 해당 Server Application의 Server의 Port에 연결
- 상호 통신



Server/Client 통신

■ 2 - Tier C/S Model

- Client와 Server가 일대일로 연결하는 방식
- Client가 요청을 하고 Server가 응답을 하는 것이 기본적인 Networking 방식



■ 3 - Tier Model

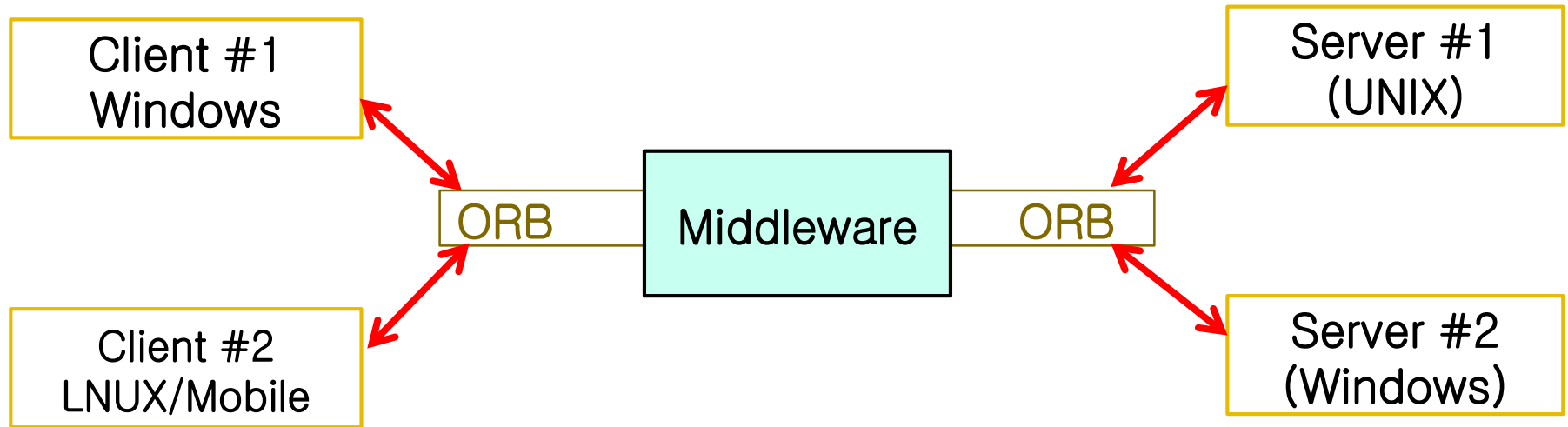
- Server를 좀 더 유연하게 구성
- 응용 Server와 Data Server를 구성하는 경우, DataBase를 분리시킴





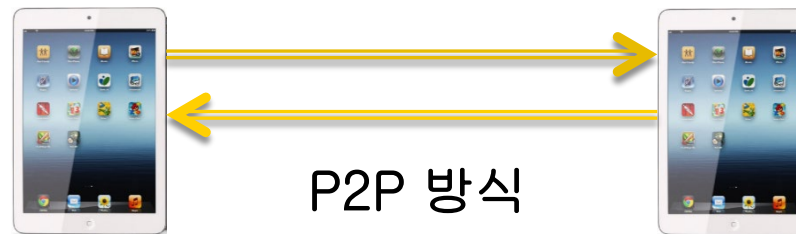
Sever/Client 통신

■ Middleware를 이용한 통신 방식





P2P 방식



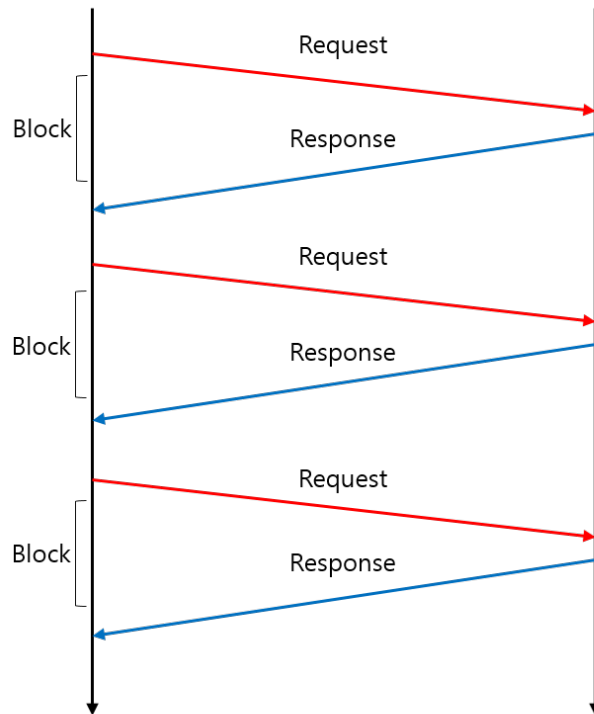


통신 방식



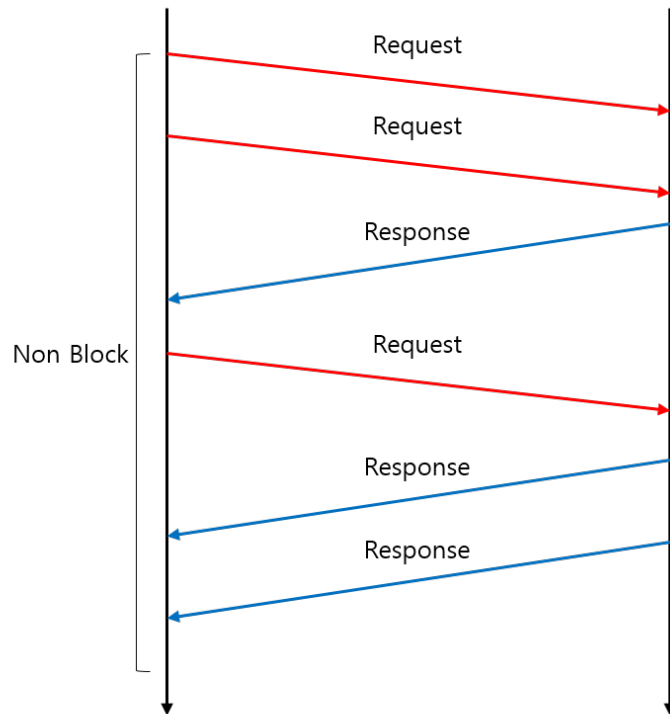
■ Synchronous과 Asynchronous

Synchronous (동기식)



※ 요청과 응답의 순서가 보장된다

Asynchronous (비동기식)



※ 요청과 응답의 순서가 보장되지 않는다



통신 방식



■ Asynchronous(비동기식) 통신

■ 비동기식 통신 및 비동기식 프로그래밍이란

ASynchronous란 뜻대로 동시에 수행하지 않는다는 뜻

■ 동기식과는 반대로 Request를 보내더라도 Response를 언제 받아도 상관이 없다는 말로 즉, Request를 보내고 Response를 상관하지 않는 상태가 되는 것

■ Asynchronous에서는 Synchronous와는 반대로 Response를 기다리지 않고 있으니 Request를 보낸 Thread는 다른 일을 할 수 있음

■ Request를 보내고 Response를 기다리지 않고 다른 일을 하는 이러한 상태를 Non Blocking 상태라고 함

■ Thread가 Response를 받지 않고 여러가지 Request 보냈을 때 뒤에 보낸 Request가 먼저 처리가 되었다면 뒤에 Request 값에 대한 Response 값이 먼저 올 수도 있음

■ 이러한 특징으로 Async(비동기식) 통신은 순서를 보장하지 않음



통신 방식



■ Asynchronous(비동기식) 통신

- 이런 식으로 Response를 기다리지 않고 Non Blocking 상태로 계속 자기 일을 하는 Async(비동기식) 방식은 Sync(동기식) 방식에 비해 성능적으로 좋을 수 밖에 없음
- 하지만 Sync(동기식)과 반대로 Response에 대한 처리 결과를 보장받고 처리해야 되는 Service에는 적합하지 않음
- 이렇게 Sync(동기식)과 Async(비동기식)은 명확한 장단점을 가지고 있으며 Service 특징에 따라 처리 결과에 의해 처리하여 Service의 질을 높일 수 있는 Synchronous(동기식) 방법과 처리 결과에 의존하지 않고 성능적으로 빠른 처리가 가능한 ASynchronous(비동기식) 방법을 잘 적용 하는게 중요함



통신방식



■ Synchronous(동기식) 통신이란?

- 동기식 통신 및 동기식 프로그래밍이란 Synchronous란 뜻대로 동시에 일어난 다는 뜻
- 동시에 일어난 다는 것은 Request를 보내게 된다면 얼마나 시간이 걸리든 그 자리에서 Response를 받는다는 말로 즉, 두 System 사이의 Transaction을 맞추겠다는 뜻

■ 장단점

- Request를 보낸 Thread는 Response가 도착하기 전까지는 아무것도 하지 못하는 Blocking 상태가 됨을 의미
- 그렇게 되면 해당 Thread는 Request를 보내고 Response를 받고 Request를 다시 보내는 작업을 수행하게 됨
- 이는 Request와 Response 값의 순서를 보장하게 됨. 또한 보낸 Request에 대한 처리 결과 값을 보장 받을 수 있음



통신방식



■ Synchronous(동기식) 통신이란?

- 이는 Request 값에 대해 성공, 실패 및 처리 결과에 대해 변경되는 사항이 있는 경우엔 굉장히 중요한 요소
- 동기식 처리 방식은 철수가 영희에게 돈을 송금하였을 때 영희가 돈을 받았는지 확인하는 상태라고 생각 하면 됨
- ex) Request : 철수가 돈을 보냈다. Response : 영희는 돈을 받았다. or 영희는 돈을 받지 못했다.-> 이후 Data 처리
- 만약 철수가 돈을 보내고 영희의 확인을 하지 않고 Data 처리를 한다면 말도 안되는 상황이 만들어질 수도 있음
- 이러한 특징의 단점으로는 Response가 지연되게 된다면 Request를 보낸 Thread는 해당 Response를 무작정 기다리는 상태가 된다는 것. 순차적으로 Response를 받고 Request를 받는 구조로 Response가 계속 지연되게 된다면 뒤에 들어오는 Request들은 Connection 가능한 Thread가 없어 여격을 맴지 못하는 선느적이 이스가 발생



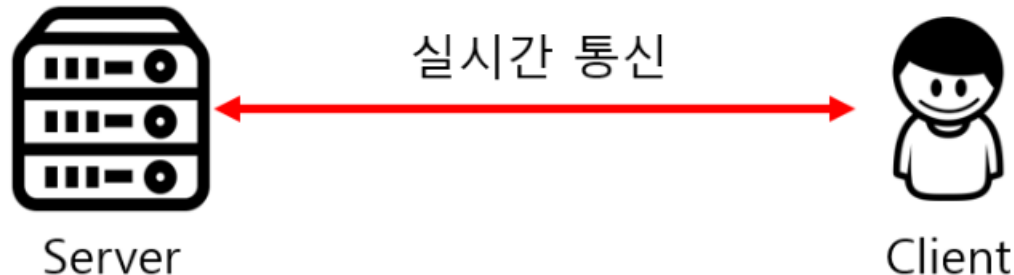
Android에서의 통신

- App에서 가장 많이 수행하는 처리 중 하나는 Client가 Server에 Data를 Request하고 받아온 Data를 Client의 화면에 표시하는 것
- Client와 Server가 통신하는 2가지 방식
 - JAVA의 전통적인 방식
 - Socket 통신
 - Socket 연결 방식에서는 Client와 Server가 특정 Port를 통해 연결을 계속 유지하고 있기 때문에 실시간으로 **양방향 통신을 할 수 있음**
 - HTTP 통신
 - Client의 Request가 있을 때만 Server가 Response하여 해당 정보를 전송하고 곧바로 연결을 종료하는 **단 방향 통신 방식**
 - Apache Library(HttpClient)를 사용하는 방식



Socket 통신

- Socket 통신은 HTTP 통신과 달리 Server와 Client가 특정 Port를 통해 연결을 성립하고 있어, 실시간으로 **양방향 통신**을 하는 방식
- Client만 필요한 경우에 Request를 보내는 HTTP 통신과 달리 Socket 통신은 Server 역시 Client로 Request를 보낼 수 있으며, 계속 연결을 유지하는 연결 지향형 통신이기 때문에 **실시간 통신이 필요한 경우에 자주 사용**
- 예) 실시간 Streaming 중계나 실시간 Chatting과 같이 즉각적으로 정보를 주고받는 경우에 사용





Socket 통신



■ 특징

- Server와 Client가 계속 연결을 유지하는 양방향 통신
- Server와 Client가 실시간으로 Data를 주고받는 상황이 필요한 경우에 사용
- 실시간 동영상 Streaming이나 Online Game 등과 같은 경우에 자주 사용



HTTP 통신

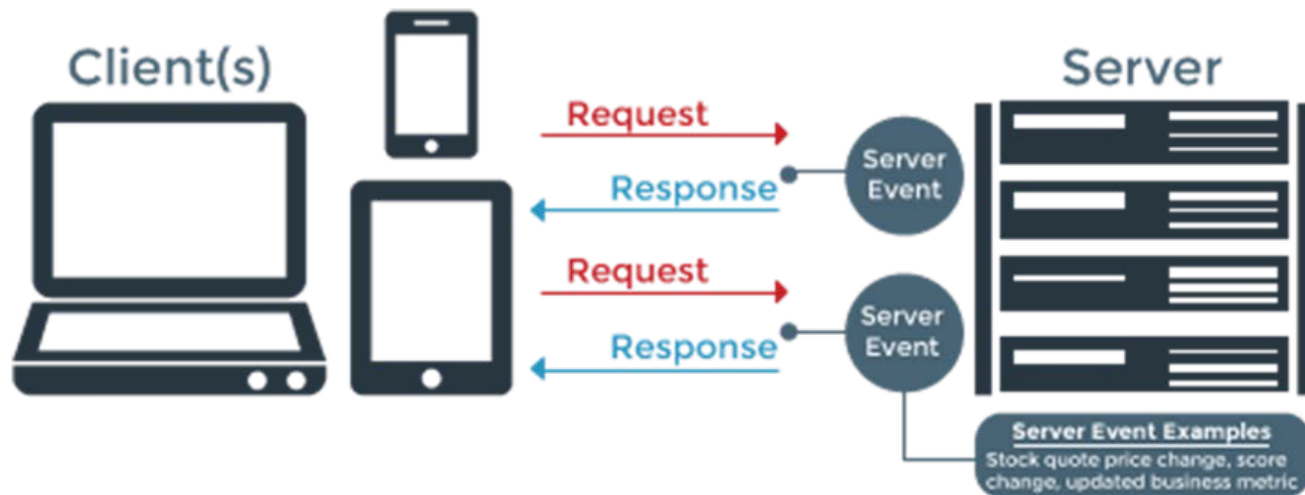


- HTTP(Hypertext Transfer Protocol) 통신
- HTTP는 Server와 Client가 Internet상에서 Data를 주고 받기 위한 Protocol
- HTTP는 Web의 기본 Protocol
- 예) 일반적으로 Mobile Application은 필요한 경우에만 Server로 정보를 Request하는 경우가 많은데, 이러한 Web Server로 HTTP 통신을 주로 사용하며 비용 및 유지보수 등 대부분의 방면에서 좋음
- HTTP는 어떤 종류의 Data도 전송할 수 있도록 설계되어 있음
- 예) Image, Video, Audio, Text 문서 등 종류를 가리지 않고 전송할 수 있음



HTTP 통신

- Client의 Request이 있을 때만 Server가 Response하여 해당 정보를 전송하고 곧바로 연결을 종료하는 방식
- 이러한 연결 방식은 Client가 Request를 보내는 경우에만 Server가 Response하는 단 방향적 통신
- Server가 Client로 Request를 보낼 수는 없음





HTTP 통신



■ HTTP Request

- HTTP Request는 Client(Browser, App 등)가 Server에 특정 작업을 요청하기 위해 보내는 Message
- Client와 Server가 HTTP Protocol을 통해 소통하는 기본 수단으로, Web Page Load, API 호출, File Upload 등 다양한 작업에 사용
- Hand phone에서는 HTTP의 간략화 된 버전인 WAP (Wireless Application Protocol)을 사용했으며 WAP Site 들은 HTML의 군살을 뺀 WML(Wireless Markup Language)로 Contents를 제공
- HTTP Client는 원하는 정보를 URL로 전송
- URL은 Internet상의 자원이나 Service의 유일한 한 지점을 가리키는 주소 값
- RFC 1738에 의해 Format이 정의되어 있으며 Protocol, Host Name, Port, Path, File 등의 기본 정보와 Query, 사용자명, 이즈 정보 등으로 구성



HTTP 통신



- HTTP Request는 Client가 Server와 통신하는 기본 수단
- HTTP 메소드와 Header, Payload를 적절히 구성하여 Data를 전송하거나 Resource를 요청하는 것이 핵심임

- HTTP Request의 구성 요소
 - Request Line (요청 줄)
 - Request의 핵심 정보를 포함하며, 다음과 같은 요소로 구성
 - HTTP Method : Request의 유형을 정의
 - 예) GET, POST, PUT, DELETE
 - URL(Path) : Request 대상의 Resource 경로
 - 예) /users, /products/123
 - HTTP Version : 사용 중인 HTTP Protocol 버전
 - 예) HTTP/1.1, HTTP/2



HTTP 통신



- HTTP Request의 구성 요소

- Headers (헤더)

- Request에 대한 부가 정보를 포함

- <Key-Value> 형태로 표현

- Metadata, 인증 정보, Request Data 형식 등을 전달

- 주요 Header 예시

- Host : 요청 대상 Server의 Domain

- 예) Host : www.example.com

- User-Agent : Request을 보낸 Client의 정보

- 예) User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64)

- Accept : Client가 처리할 수 있는 Data 형식

- 예) Accept : application/json

- Authorization : 인증 정보를 포함

- 예) Authorization: Bearer token123

- Content-Type : Request 본문의 Data 형식 (POST, PUT 등에서 중요)

- 예: Content-Type: application/json



HTTP 통신



- HTTP Request의 구성 요소
 - Body (본문, Payload)
 - Request에 포함된 실제 Data로, Method에 따라 존재 여부가 달라짐
 - 보통 POST, PUT 요청에서 Data를 Server로 전송할 때 사용
 - Data 형식은 Content-Type Header에 따라 결정
 - 예) JSON, XML, Form Data 등



HTTP 통신



- HTTP Response

- Client(Browser나 Application)의 Request에 대해 Server가 보내는 Response Message
- Client는 Server로부터 HTTP Response를 통해 Request한 Resource나 Request 처리 결과를 받음



HTTP 통신



- HTTP Response의 구성요소
 - Status Line (상태 줄)
 - 응답의 상태를 나타냄
 - 구성
 - HTTP Version : 사용된 HTTP Protocol 버전
 - 예) HTTP/1.1
 - Status Code : Request 처리 결과를 나타내는 숫자 Code
 - 예) 200, 404
 - Reason Phrase: Status Code를 설명하는 Text
 - 예) OK, Not Found



HTTP 통신



- HTTP Response의 구성요소

- Headers (헤더)

- Response에 대한 Metadata를 포함

- <Key-Value> 쌍으로 구성

- 주요 Response Header

- Content-Type : 응답 본문의 데이터 형식

- 예) application/json, text/html

- Content-Length : 응답 본문의 길이 (Byte 단위)

- Set-Cookie : Client에 설정할 쿠키 정보

- Cache-Control : 캐싱 정책

- Location : Redirection URL

- Body (본문)

- 응답의 실제 Data를 포함

- 예) HTML, JSON, XML, 텍스트, 이미지 파일 등



HTTP 통신



■ HTTP 작동 방식

- HTTP는 Server/Client Model을 따름
- Client가 Server에게 Request를 보내면, Server는 Client에게 Response를 보냄
- HTTP는 stateless(무 상태) Protocol이며, stateless 방식을 따름
- stateless 방식은, 간단히 말하면 Server가 여러 Client들을 구별할 수 없다는 것
 - 예) Client가 Server에게 Request를 보내고, Server가 Response한 후 연결이 끊겼다고 가정하면, 그 후에 똑같은 Client가 또다시 Server에게 Request를 했을 때, Server는 그 Client가 이전의 Client인지 아닌지 구별할 수 없음. 이것을 stateless 방식이라 함



HTTP 통신



■ stateless 방식의 장·단점

■ 장점

- 불특정 다수를 대상으로 하는 Service에 적합
- Client와 Server가 최대 연결 수 보다 훨씬 많은 Request와 Response를 처리할 수 있음

■ 단점

- 연결이 끊어진 후에는 Client가 이전에 무엇을 했는지 알 수 없음
 - 이러한 단점을 보완하기 위해서 cookie라는 기술이 등장



HTTP 통신



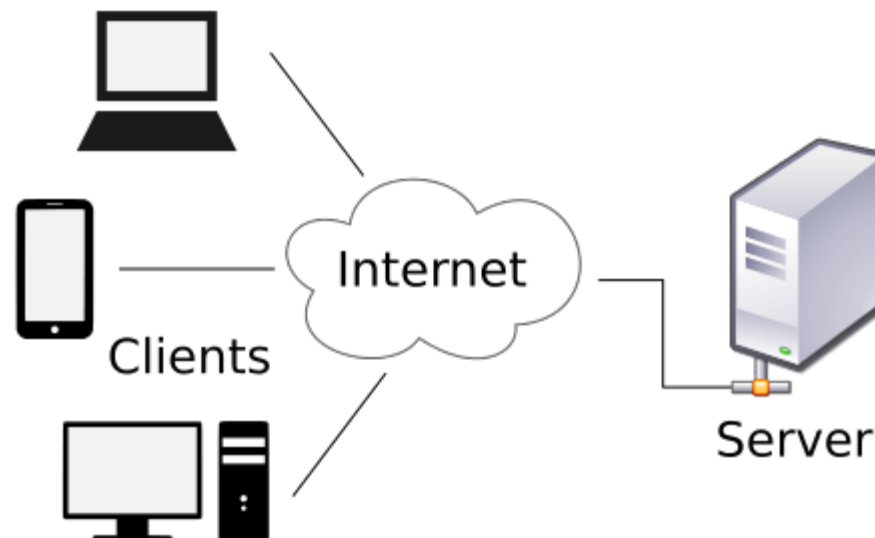
■ HTTP 통신의 특징

- Client가 Request를 보내는 경우에만 Server가 Response하는 단 방향 통신
- Server로부터 Response를 받은 후에는 연결이 바로 종료
- 실시간 연결이 아니고, 필요한 경우에만 Server로 Request를 보내는 상황에 유용
- Request를 보내 Server의 Response를 기다리는 Application의 개발에 주로 사용



HTTP 통신

- HTTP는 80번 Port를 사용하여 Web상에서 정보를 주고 받을 수 있는 Protocol을 말함
- HTTP 통신에서는 Client가 Server에 Header와 Body로 이루어진 Message를 Request
- 그러면 Server는 이 Request를 처리하고 Response Code와 함께 Response을 반환하게 됨





HTTP 통신



■ HTTP 통신 특징

■ Connectionless

- HTTP 연결은 Socket을 사용해서 접속을 만들지만, 용건이 있을 때만 연결했다가 용건이 끝나면 연결을 끊게 되어 있음
- 이것을 Connectionless하다고 표현하는데, 이렇게 하면 Server에 여유가 생겨 더 많은 접속 요구에 대응할 수 있게 됨

■ Stateless

- 통신이 일어날때마다 새로운 접속을 생성하고 삭제하는 Connectionless 특징 때문에 HTTP 통신에서는 기본적으로 **Server가 Client를 식별할 수가 없는데 이것을 Stateless하다고 함**
- Server가 Client를 기억해야 할 경우 Cookie나 Session, 혹은 Token이라는 기술을 사용하게 됨



HTTP 통신



■ HTTP Method

- Client가 Server에 Message를 보낼 때 어떠한 목적을 가졌는지 밝히는 것
- 일반적으로 많이 사용되는 GET, POST, PUT, DELETE 이외에도 여러가지 명령어들이 있음



HTTP 통신



■ HTTP Method

Method	설명
GET	Server에 Resource를 요청 Server를 수정하지 않기 때문에 safe method로 분류
HEAD	GET과 같지만 Server가 Body를 포함하지 않음
POST	Client에서 Request한 URL에 Body의 내용으로 새로운 Resource를 생성
PUT	Client가 Request한 내용으로 Server의 Resource를 수정
DELETE	Server의 리소스를 삭제
CONNECT	요청 리소스에 대해 양방향 연결을 수립, 예로 프록시를 통한 SSL 연결 수립에 사용될 수 있음
OPTIONS	서버가 어떤 HTTP 메소드를 지원하는지 물어봄
TRACE	메시지의 변조 여부 확인을 위해, 서버가 수신한 클라이언트의 메시지를 반환하도록 함
PATCH	PUT은 리소스 전체를 수정하나, PATCH는 해당 리소스의 일부만을 수정



HTTP 통신



■ HTTP Method

- GET 방식과 POST 방식(Data를 전송하는 방식으로 구분)





HTTP 통신



■ GET 방식

- 주로 Data를 Server로 Request하는 데 사용
- URL에 Data를 Parameter를 붙여서 전송한다는 것
 - Data는 URL끝에 '?'를 붙이고 'key=value' 형태로 표시
 - 이러한 Parameter는 '&'로 구분됨
- 예) `http://www.tistory.com/test?par1=val1&par2=val2`
- 예) 검색어를 보내는 경우 다음과 같이 URL에 Parameter를 포함시킴

<http://example.com/search?query=keyword>

- Data를 URL에 Parameter로 전송하기 때문에 Body 영역을 사용하지 않음
- 또한 URL에 Data를 실어 보내기 때문에 대용량 Data 전송을 하기에 제한 사항이 있음
- 한번 Request 시 URL 포함 255자 까지 전송이 가능하며 HTTP/1.1에서는 2,048자 까지 가능





HTTP 통신



■ POST 방식

- Data를 Server로 전송하기 위해 Body에 Data를 담아 전송
- POST로 Data를 전송할 때에는 Body 영역 Data Type을 Header Content-Type에 명시해줘야 함
- Data는 노출되지 않고 Request Body에 포함되므로, GET 방식보다 보안성이 높음
- GET 방식과는 달리 보내는 Data를 URL를 통해 볼 수 없어 보안적으로 안전하다고 하지만 다른 Tool을 사용하여 POST 영역의 Data를 확인이 가능하기 때문에 안심해서는 안됨
- Body에 Data를 실어 보내기 때문에 Data 전송 양의 크기에 제한이 없으므로 대용량 Data를 보내는데 적합
- 사용자가 입력한 Form Data를 Server로 제출하거나 중요한 정보를 전송할 때 주로 사용



HTTP 통신



■ Query String

- Web Browser에서 주소 창으로 전달하는 GET Parameter는 '?'로 시작하고 각 항목은 <키=값> 형태로 구성하며, 항목간의 연결에는 '&'를 사용
- Web Browser의 주소 창으로 전달되는 Query String Parameter는 PHP에서 연관 배열 형식의 \$_GET 슈퍼 글로벌 변수를 이용해 \$_GET[파라미터 키] 형식으로 읽어올 수 있음

Secure | https://en.wikipedia.org/w/index.php?title=Query_string&action=edit



HTTP 통신



처리 방식	GET 방식	POST 방식
URL에 데이터 노출 여부	O	X
URL 예시	http://localhost:8080/boardList?name=제목&contents=내용	http://localhost:8080/addBoard
데이터의 위치	Header(헤더)	Body(바디)
캐싱 가능 여부	O	X



HTTP 통신



- GET 방식으로 Data 보내기
 - Client의 Data를 URL뒤에 붙여서 보냄
 - URL 뒤에 "?" Mark를 통해 URL의 끝을 알리면서, Data 표현의 시작점을 알림
 - Data는 key와 value 쌍으로 넣어야 함
 - 중간에 "&" Mark는 구분자 임
 - 2개 이상의 key - value 쌍 Data를 보낼 때는 "&" Mark로 구분해 줌
 - URL에 붙이므로, HTTP Packet의 Header에 포함되어 Server에 요청
 - GET 방식에서 BODY에 특별한 내용을 넣을 것이 없으므로 BODY가 빈 상태로 보내 짐
 - 간단한 Data를 넣도록 설계되어, Data를 보내는 양의 한계가 있음



HTTP 통신



■ GET 방식의 특징

■ URL에 변수(Data)를 포함시켜 요청

■ 간단한 Data를 URL에 넣도록 설계된 방식으로 Data를 보내는 Data의 양에 한계가 있음

■ Data를 Header에 포함하여 전송

■ URL에 Data가 노출되어 보안에 취약함

■ 캐싱할 수 있음



HTTP 통신



- POST 방식으로 Data 보내기
 - GET 방식과 달리, Data 전송을 기반으로 한 Request 메소드
 - GET 방식은 URL에 Data를 붙여서 보내는 반면, POST 방식은 URL에 붙여서 보내지 않고, BODY에 Data를 넣어서 보냄
 - Header Field 중 BODY의 Data를 설명하는 Content-Type이라는 Header Field가 들어가고 어떤 Data Type인지 명시
 - Contents Type으로는 다음과 같이 여러가지가 있음
 - application/x-www-form-urlencoded
 - text/plain
 - multipart/form-data
 - POST 방식으로 Data를 보낼 때는 위와 같이 Contents Type을 꼭 명시해줘야 함



HTTP 통신



- POST 방식으로 Data를 보내기
 - 보통 작성하지 않는 경우는 1번의 Contents Type으로 Setting 됨
 - 1번의 Contents Type은 GET 방식과 마찬가지로 BODY에 key와 value 쌍으로 Data를 넣음. 똑같이 구분자 "&"를 씀
 - 2번의 Contents Type은 BODY에 단순 txt를 넣음
 - 3번의 Contents Type은 File 전송을 할 때 많이 쓰는데 BODY의 Data를 Binary Data로 넣는다는 걸 알려줌
 - JAVA와 같이 OOP Programming에서는 BODY에 Data를 InputStream/OutputStream 클래스를 통해서 읽고/쓰



HTTP 통신



■ POST 방식의 특징

- URL에 Data를 노출하지 않고 Request
- Data를 Body에 포함
 - Data를 Body에 포함시키는 이점 때문에 Message 길이의 제한은 없지만 최대 Request를 받는 시간인 Time Out이 존재하므로 Client에서 Page를 Request하고 기다리는 시간이 존재
- URL에 Data가 노출되지 않아서 기본 보안은 됨
- Caching할 수 없음



HTTP 통신



- 언제 GET과 POST를 사용하는가?
 - GET은 URL에 Data가 나오고 POST는 URL에 Data가 나타나지 않음
 - 간단한 Text Data들은 GET으로 넘겨도 되나, 너무 긴 내용은 POST로 넘겨야 함
 - Image File 같이 Binary 형태의 Data는 GET보다는 POST로 넘겨야 함



HTTP 통신




- GET 방식과 POST 방식
 - POST 방식이 GET 방식보다 보안측면에서 더 좋다 ?
 - POST든 GET이든 보내는 Data는 전부 Client 측에서 볼 수 있음
 - 단지 GET 방식은 URL에 Data가 표시되어 별다른 노력 없이 볼 수 있어서 지, 두 방식 전부 보안을 생각한다면 암호화 해야함
- GET 방식이 POST 방식보다 속도가 빠르다?
 - 빠른 건 맞음
 - 이유는 GET 방식의 Request은 캐싱(한번 접근 후, 또 Request할 시 빠르게 접근하기 위해 Data를 저장시켜 놓음)때문에 빠른 것



HTTP 통신

■ HTTP Status Code

■ https://en.wikipedia.org/wiki/List_of_HTTP_status_codes



WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Current events
- Random article
- About Wikipedia
- Contact us
- Donate

Contribute

Help

- Community portal
- Recent changes
- Upload file

Tools

- What links here
- Related changes
- Special pages
- Permanent link
- Page information
- Cite this page
- Wikidata item

Print/export

- Download as PDF
- Printable version

Not logged in · Talk · Contributions · Create account · Log in

Article · **Talk** · Read · View source · View history

Search Wikipedia

List of HTTP status codes

From Wikipedia, the free encyclopedia

This is a list of [Hypertext Transfer Protocol](#) (HTTP) response status codes. Status codes are issued by a server in response to a [client's request](#) made to the server. It includes codes from IETF [Request for Comments](#) (RFCs), other specifications, and some additional codes used in some common applications of the HTTP. The first digit of the status code specifies one of five standard classes of responses. The message phrases shown are typical, but any human-readable alternative may be provided. Unless otherwise stated, the status code is part of the HTTP/1.1 standard (RFC 7231).^[1]

The [Internet Assigned Numbers Authority](#) (IANA) maintains the official registry of HTTP status codes.^[2]

All HTTP response status codes are separated into five classes or categories. The first digit of the status code defines the class of response, while the last two digits do not have any classifying or categorization role. There are five classes defined by the standard:

- *1xx informational response* – the request was received, continuing process
- *2xx successful* – the request was successfully received, understood, and accepted
- *3xx redirection* – further action needs to be taken in order to complete the request
- *4xx client error* – the request contains bad syntax or cannot be fulfilled
- *5xx server error* – the server failed to fulfil an apparently valid request

Contents [\[hide\]](#)

- 1 [1xx Informational response](#)
- 2 [2xx Success](#)
- 3 [3xx Redirection](#)
- 4 [4xx Client errors](#)
- 5 [5xx Server errors](#)
- 6 [Unofficial codes](#)

6.1 [Internet Information Services](#)

HTTP

[Persistence](#) · [Compression](#) · [HTTPS](#) · [QUIC](#)

Request methods

[OPTIONS](#) · [GET](#) · [HEAD](#) · [POST](#) · [PUT](#) · [DELETE](#) · [TRACE](#) · [CONNECT](#) · [PATCH](#)

Header fields

[Cookie](#) · [ETag](#) · [Location](#) · [HTTP referer](#) · [DNT](#) · [X-Forwarded-For](#)

Status codes

[301 Moved Permanently](#) · [302 Found](#) · [303 See Other](#) · [403 Forbidden](#) · [404 Not Found](#) · [451 Unavailable For Legal Reasons](#)

Security access control methods

[Basic access authentication](#) · [Digest access authentication](#)

Security vulnerabilities

[HTTP header injection](#) · [HTTP request smuggling](#) · [HTTP response splitting](#) · [HTTP parameter pollution](#)

V · T · E



Futuristic Innovator

京福大學校
KYUNGBOK UNIVERSITY



HTTP 통신



■ HTTP Status Code

- HTTP Status Code는 Server가 Client의 Request를 처리한 결과를 나타내는 숫자 Code
- 1xx : Informational(정보)
 - Request를 처리 중임을 알리는 상태
 - 예) 100 Continue : Client가 Request를 계속해도 됨
 - 예) 101 Switching Protocols : Protocol 변경 Request 승인
- 2xx : Success (성공)
 - Request가 성공적으로 처리되었음을 나타냄
 - 200 OK : Request가 성공적으로 처리됨
 - 201 Created : Resource가 성공적으로 생성됨
 - 204 No Content : 성공했으나 본문이 없음



HTTP 통신



■ HTTP Status Code

■ 3xx: Redirection (리다이렉션)

- 요청한 Resource가 다른 위치에 있거나 Client가 추가 작업을 해야 함
- 301 Moved Permanently : Resource가 영구적으로 이동됨
- 302 Found : Resource가 임시적으로 이동됨
- 304 Not Modified : Resource가 수정되지 않았으므로 Cache된 Data를 사용
- 307 : 임시 Page로 Redirect



HTTP 통신



■ HTTP Status Code

■ 4xx: Client Error (클라이언트 오류)

- Client의 Request에 문제가 있을 때 사용

- 400 Bad Request : Request 문법이 잘못되었거나 유효하지 않음

- 401 Unauthorized : 인증이 필요함

- 403 Forbidden : 접근 권한이 없음

- 404 Not Found : 요청한 Resource를 찾을 수 없음
403 대신에 사용할 수도 있음(해커들의 공격을 방지하고자 페이지가 없는 것처럼 위장함)

- 408 : Request 시간이 초과됨

- 409 : Server가 Request를 처리하는 과정에서 충돌이 발생한 경우 (회원가입 중 중복된 아이디인 경우)

- 410 : 영구적으로 사용할 수 없는 Page



HTTP 통신



■ HTTP Status Code

■ 5xx: Server Error (서버 오류)

- Server에서 Request를 처리하지 못했을 때 사용
- 500 Internal Server Error : Server에 일반적인 문제가 발생
- 502 Bad Gateway : Gateway이나 Proxy Server가 잘못된 Response를 받음
- 503 Service Unavailable : Server가 Request를 처리할 수 없음 (과부하, 유지보수 등)
- 504 : Server Gateway에 문제가 생겨 시간 초과가 된 경우
- 505 : HTTP 버전이 달라 Request를 처리할 수 없음



XML 문서 정보 확인 방법

- [Windows + R]를 누른 뒤에 cmd를 입력하여 명령 프롬프트를 연다

```
C:/>curl -i [웹 url]
```

```
명령 프롬프트
C:\Users\bae>curl https://naver.com
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center> NWS </center>
</body>
</html>

C:\Users\bae>curl -i https://naver.com
HTTP/1.1 301 Moved Permanently
Server: NWS
Date: Wed, 30 Nov 2022 05:52:12 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Location: http://www.naver.com/
Vary: Accept-Encoding,User-Agent
Referrer-Policy: unsafe-url

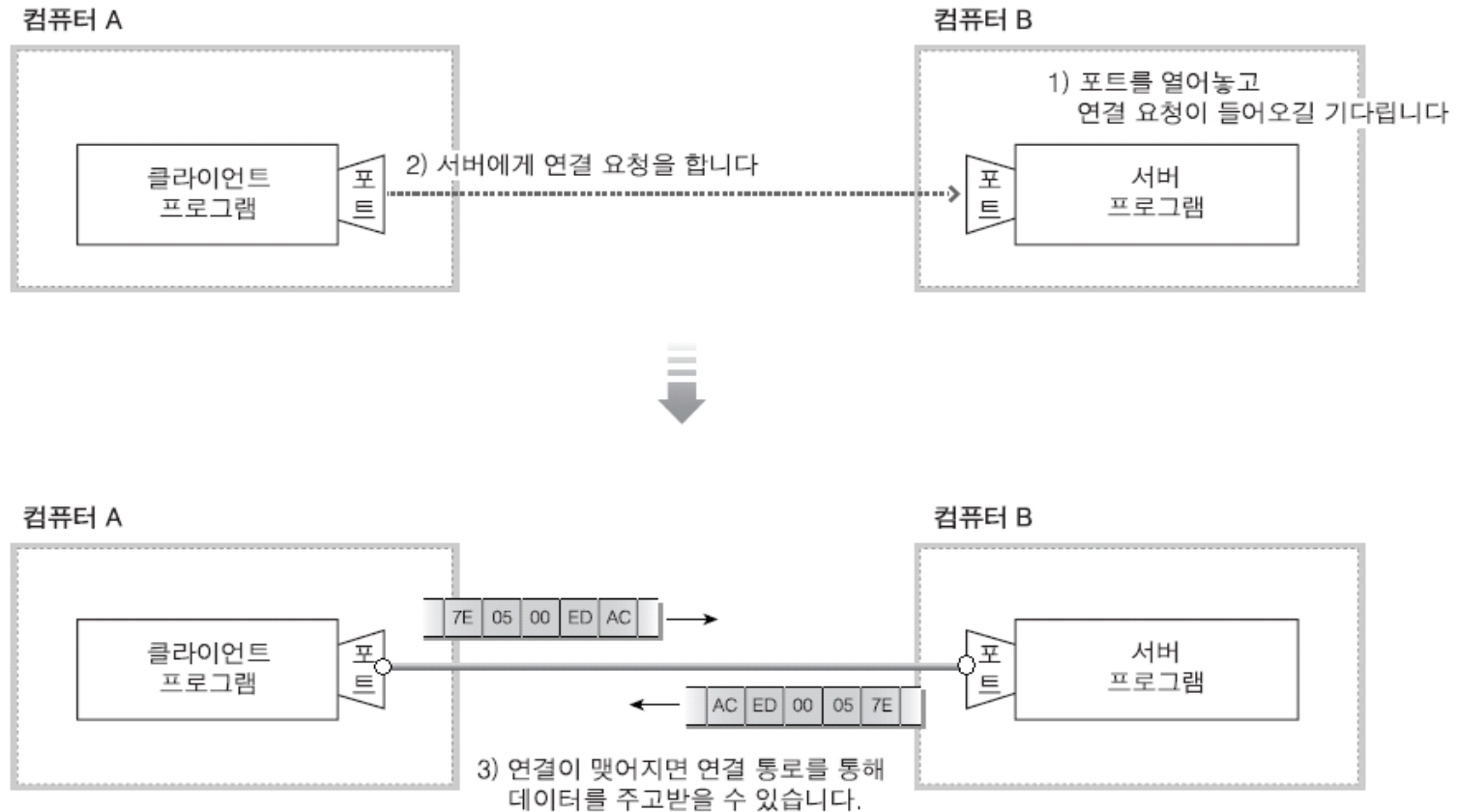
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center> NWS </center>
</body>
</html>

C:\Users\bae>
```



Network 통신 Program

■ Client Program과 Server Program의 통신 과정





Network 통신 Program



■ java.net 패키지

- JAVA는 사용자가 Network의 세부 구조를 모르고도 Network 기능을 편리하게 사용할 수 있는 기능들을 **java.net 패키지**로 제공
- 사용자는 java.net 패키지에서 제공되는 클래스들을 이용하여 Networking 관련 Program을 작성
- Berkeley UNIX인 4.2BSD에서 처음으로 제안
→ 소켓(Socket) 개념을 캡슐화

■ Internet 기술

- TCP/IP 스택(Stack)에 바탕을 둠
- TCP/IP Stack은 계층 아키텍처에 놓여 있는 Protocol의 집합



Network 통신 Program



- Network Programming을 위한 패키지는 java.net
 - TCP를 위한 클래스
 - URL
 - URLConnection
 - Socket
 - ServerSocket
 - UDP를 위한 클래스
 - DatagramPacket
 - DatagramSocket
 - MulticastSocket



Network 통신 Program

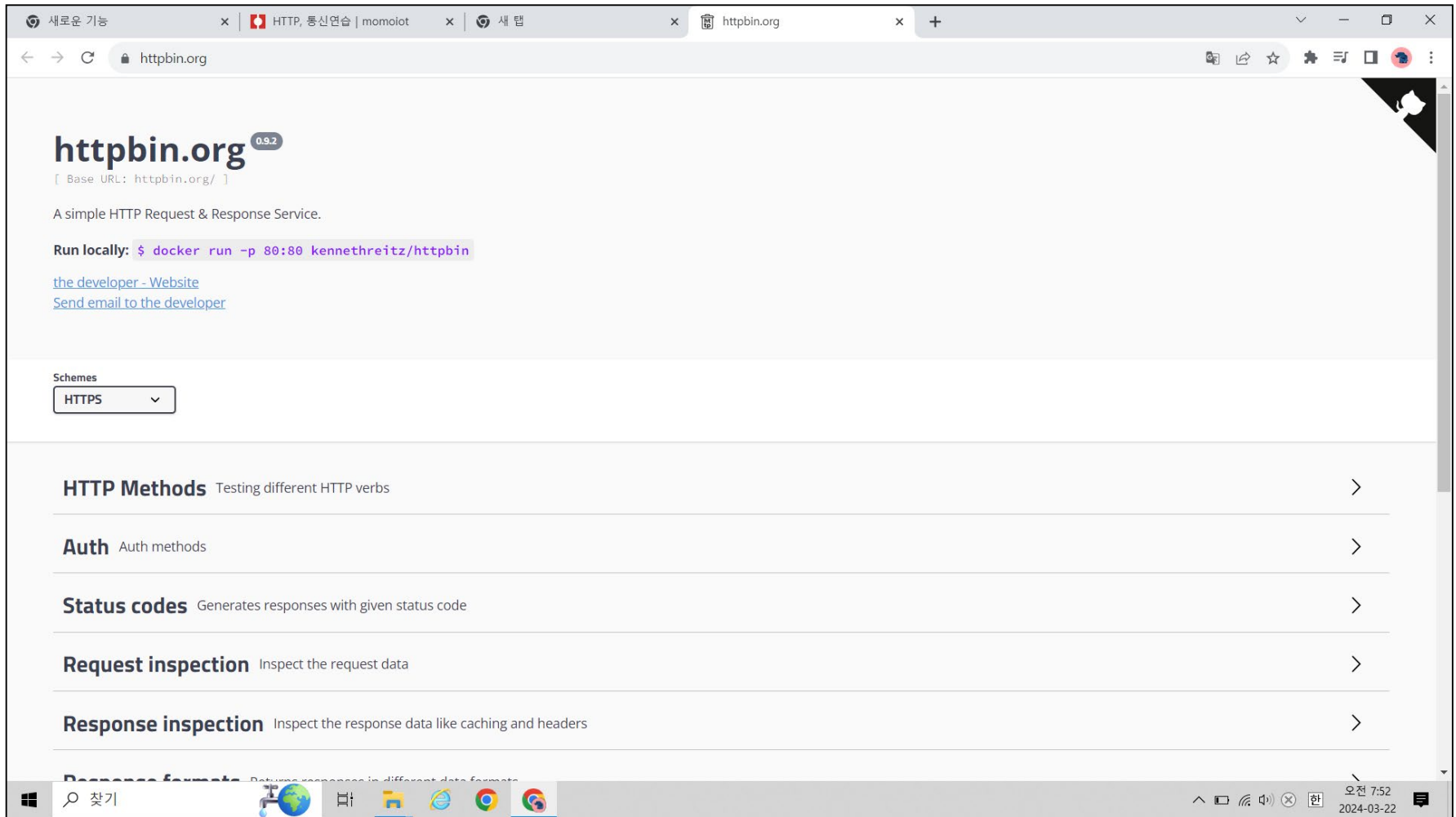


- Networking Programming 시 주의할 점
 - Networking을 사용할 때는 반드시 Thread 사용
 - Thread를 사용하므로 UI Update를 위해서는 반드시 Handler를 사용



HTTP 통신 연습

■ <http://httpbin.org/>





HTTP 통신 연습



- Local Web(web/http) Server
- 가상 Local Web Server
- Online Http Server 환경을 제공하는 Site
 - XML
 - <https://www.w3schools.com/xml/note.php>
 - <https://www.w3schools.com/xml/books.xml>
 - JSON
 - <http://httpbin.org/>
 - <http://jsonplaceholder.typicode.com/>
 - <https://reqres.in/>



HTTP 통신 연습

<https://www.w3schools.com>

XML 예제

XML 트리

XML 구문

XML 요소

XML 속성

XML 네임스페이스

XML 디스플레이

XML HTTP 요청

XML 파서

XML DOM

XML XPath

XML XSLT

XML XQuery

XML 링크

XML 검사기

XML DTD

XML 스키마

XML 서버

XML 예제

XML 퀴즈

XML 인증서

XML AJAX

XML 파일 보기

단순 XML 파일(note.xml)

보기 오류가 있는 동일한 XML 파일

보기 XML CD 카탈로그

보기 XML 식물 카탈로그

보기 XML 음식 메뉴 보기

예시 설명

XML 및 CSS

JetBrains Academy Learn Python. Develop Apps. Repeat.

Get Started

JET BRAINS

광고

내꺼이
평균
은
범위를

(후방주의)
업무 범위
를 넓혀,
사람이라면 지
정보를

사람인

확인

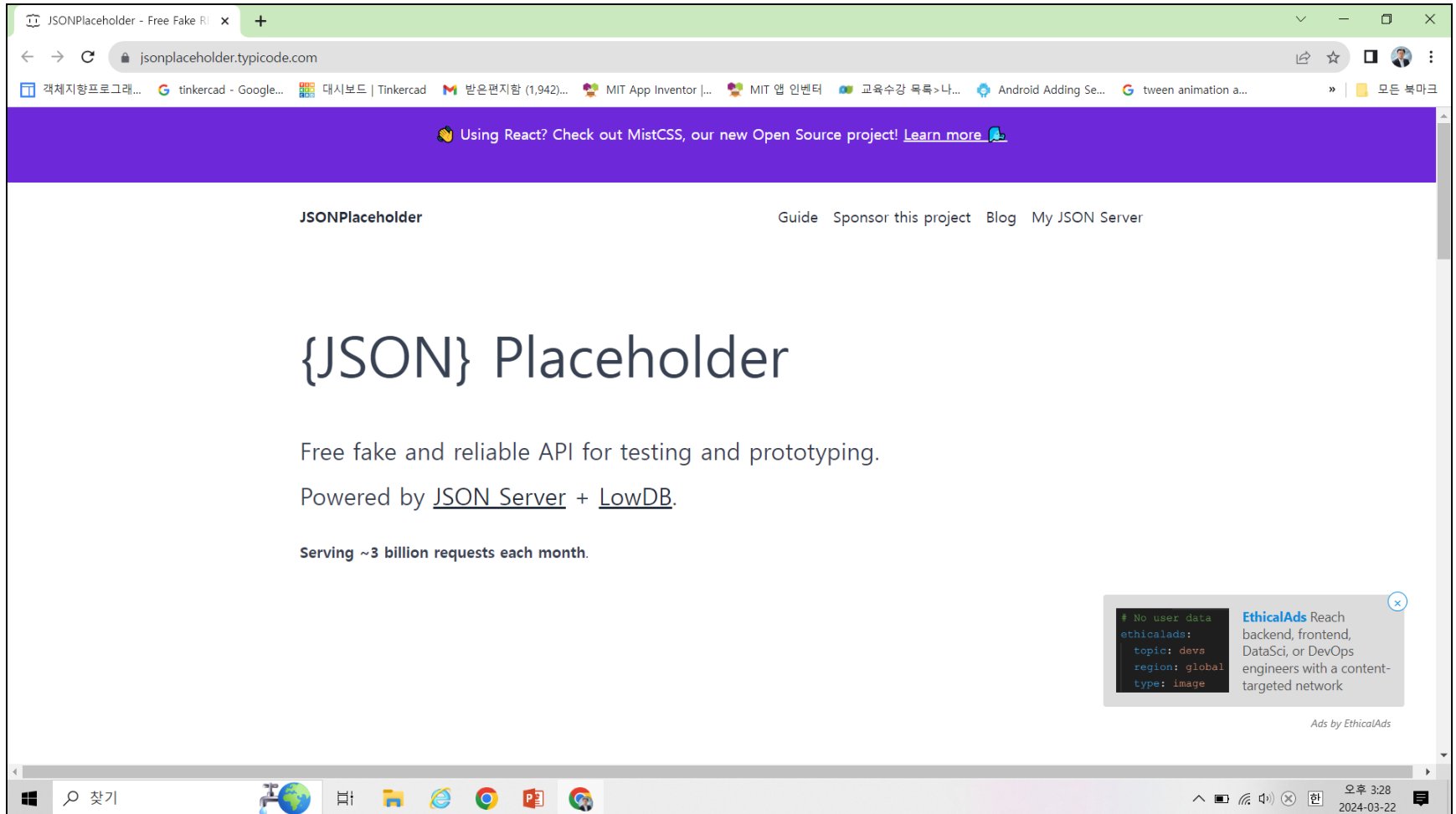
오전 9:14
2022-11-19





HTTP 통신 연습

■ <http://jsonplaceholder.typicode.com/>





HTTP 통신 연습

■ <https://reqres.in/>

JSONPlaceholder - Free Fake REST API x Reqres - A hosted REST-API resource x +

reqres.in

객체지향프로그래... tinkercad - Google... 대시보드 | Tinkercad 받은편지함 (1,942)... MIT App Inventor | MIT 앱 인벤터 교육수강 목록>나... Android Adding Se... tween animation a... 모든 북마크

REQRES

Test your front-end against a real API

- Fake data**
No more tedious sample data creation, we've got it covered.
- Real responses**
Develop with real response codes. GET, POST, PUT & DELETE supported.
- Always-on**
24/7 *free* access in your development phases. Go nuts.

St Adobe Stock Get 10 Free Images From Adobe Stock. Start Now.
ADS VIA CARBON

Windows taskbar: 3:30 PM 2024-03-22