Network Programming

1

Network 기본개요

1. Network 기초 개념
2. IP address
3. Port
4. TCP & UDP

1.1. 네트워크의 기초 개념[1/3]

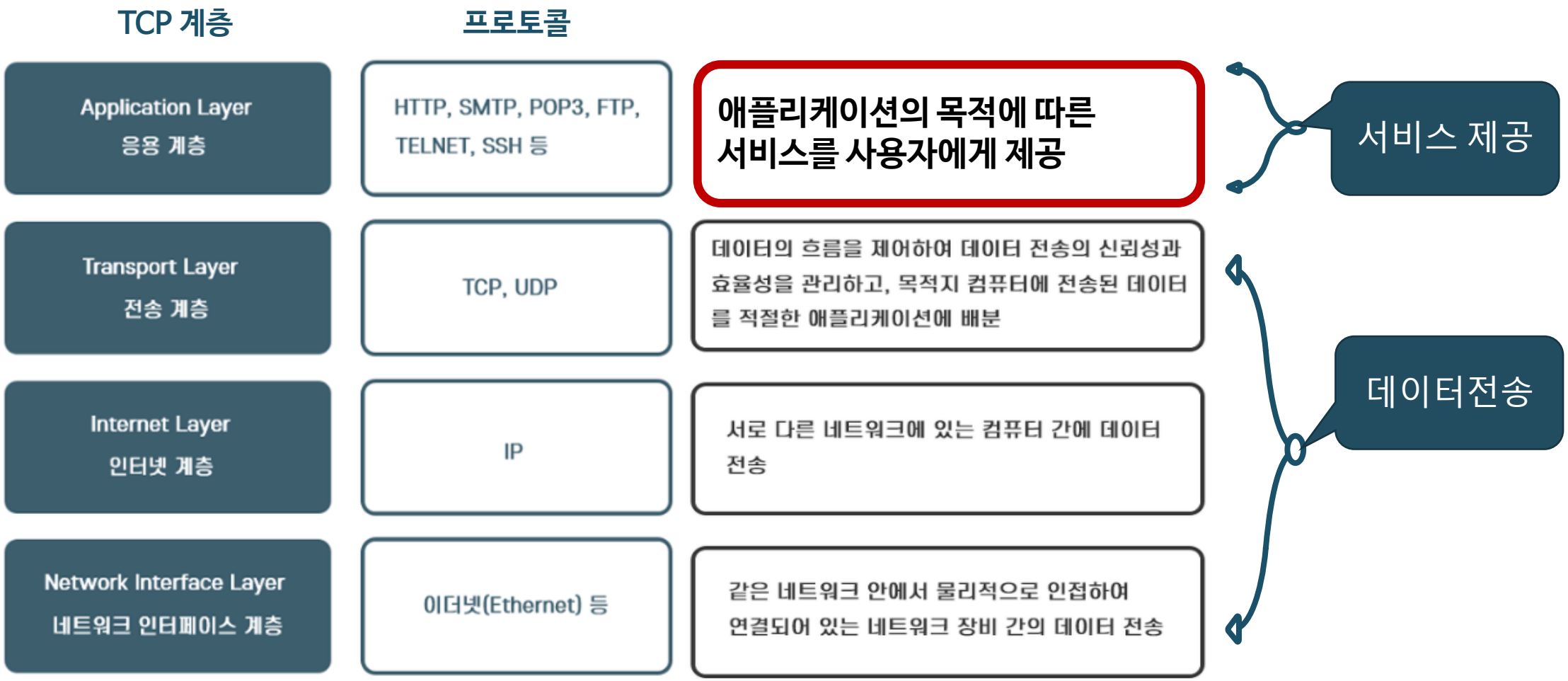
- 네트워크는 컴퓨터와 네트워크 장비를 유무선의 전송 매체로 연결하여 데이터를 전송하는 시스템이다
- 네트워크에서 데이터를 전송하려면
  - ✓ 물리적인 기기들의 하드웨어적인 연결
  - ✓ 데이터를 전송하는 규칙인 프로토콜 두 가지가 필요하다
- 인터넷 : 다수의 네트워크가 연결된 것
- 네트워크 간에 서로 통신이 가능하려면 위에서 얘기했던 통신 프로토콜을 서로 같은 것으로 합의해야 하며 이를 위해서 **인터넷에서는 TCP/IP라는 하나의 통신 프로토콜만을 사용**
- 현재 전 세계 어디에서나 인터넷에 접속할 수 있는 것은 대부분의 컴퓨터와 네트워크 장비가 인터넷의 표준인 TCP/IP 모델의 프로토콜에 따라 만들어졌기 때문이다
  - ✓ 하드웨어는 프로토콜이 정하는 표준 규격에 따라 제작하고
  - ✓ 소프트웨어는 프로토콜이 정한 데이터의 전송 순서와 절차, 데이터 전송 방법 등을 구현할 수 있도록 프로그래밍된다
- 웹 애플리케이션이 동작하는 웹 서버 컴퓨터의 운영체제도, 웹 서버 컴퓨터의 랜카드나 라우터 같은 네트워크 장비도 TCP/IP 모델의 프로토콜에 따라 만들어야 한다

1.1. 네트워크의 기초 개념[2/3] - Protocol

- 네트워크에서 데이터를 주고받기 위해 네트워크를 구성하는 컴퓨터와 네트워크 장비들이 지켜야 할 규칙
- 데이터 전송 상대방, 데이터의 형식, 데이터의 전송 순서와 절차, 데이터 전송 방법 등을 규정
- IP 프로토콜 : 한 컴퓨터에 다른 컴퓨터까지 데이터를 전송하기 위한 절차와 방법을 정한 프로토콜
- TCP 프로토콜 : 네트워크의 혼잡도 등을 고려해 데이터의 흐름을 제어하고 데이터가 정확히 도착할 수 있게 하는 절차와 방법을 정한 프로토콜
- HTTP 프로토콜 : 웹 브라우저로 웹 사이트에 접속할 경우에 사용

1.1. 네트워크의 기초 개념[3/3] - TCP/IP

TCP/IP는 인터넷과 네트워크 통신에서 중요한 역할을 하는 프로토콜 스위트(suite)이다.
데이터 통신을 위한 표준 프로토콜을 제공하여 컴퓨터와 기기 간의 효율적인 통신을 가능하게 하며
웹 브라우징, 이메일, 파일 공유 및 다양한 인터넷 서비스에 사용된다



1.2. IP(Internet Protocol) Adress

- IP address : 네트워크상에서 유일하게 식별될 수 있는 기기(ex:컴퓨터)의 주소
  - 현재 4개의 숫자로 구성 : ex) 255.255.255.31
- Domain : 숫자로 된 주소는 기억하기 어려워 문자열로 구성된 이름으로 치환한 주소 형식
  - DNS(Domain Name Syatem) : IP 주소 ←→ Domain name
- IPv6 : IP주소의 고갈로 128bit의 IPv6 버전의 사용이 늘어나는 추세이다

공인 IP

- 전세계적으로 고유한 인터넷 주소
- 인터넷서비스제공업체에 의해 할당
- 해당 장치 또는 네트워크가 인터넷에 직접 접속

사설 IP 범위

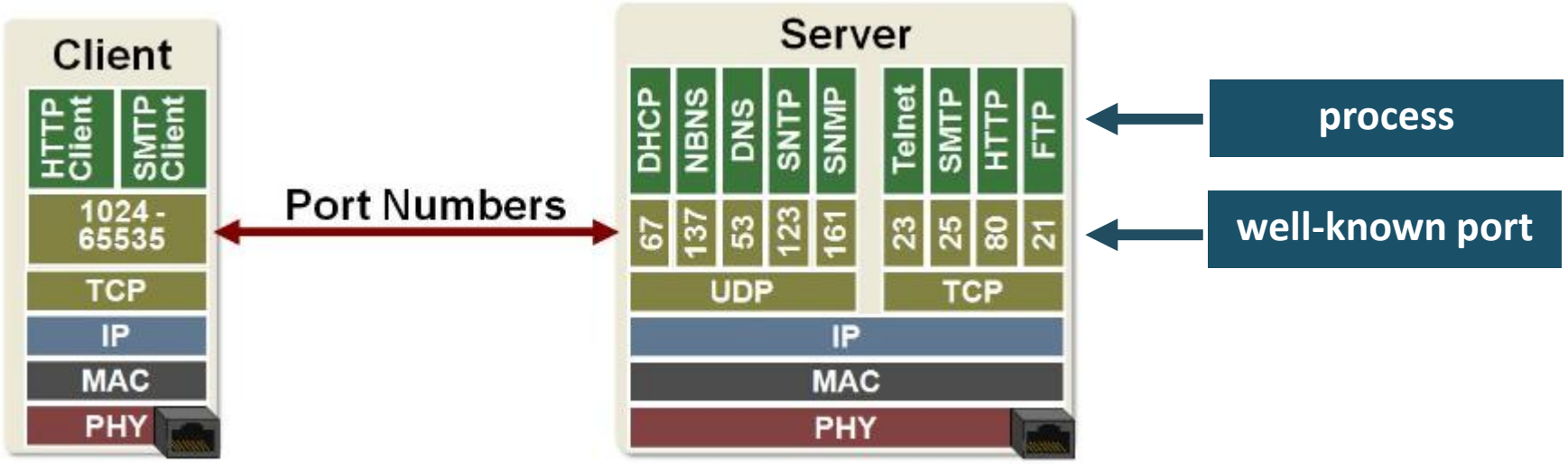
- 10.0.0.0 ~ 10.255.255.255
- 172.16.0.0 ~ 172.31.255.255
- 192.168.0.0 ~ 192.168.255.255

사설 IP

- 로컬 네트워크 내에서 사용되는 IP 주소
- 공인 IP와 달리 고유하지 않음
- 각 로컬 네트워크에서 여러 장치가 동일한 사설 IP 주소를 가질 수 있다.
- 라우터 또는 네트워크 장비에 의해 할당
- 인터넷에 직접 연결할 수 없으며, NAT(Network Address Translation) 또는 포트 포워딩과 같은 기술을 사용하여 공인 IP 주소를 통해 외부와 통신할 수 있다.

1.3. Port

- Host내에서 실행되고 있는 프로세스를 구분하기 위한 16비트의 논리적 할당(0~65535)
  - IP주소 : 네트워크상의 컴퓨터 또는 시스템을 식별하는 주소
  - port 번호 : 해당 포트번호를 이용하여 통신할 응용프로그램을 식별하는 번호
- 모든 응용프로그램은 하나이상의 포트번호를 생성할 수 있다
  - 포트번호를 이용하여 상대방의 응용프로그램과 데이터 교환
- well-known ports : 대표적인 인터넷 프로그램들이 미리 예약하여 사용 (0~1023)
  - 개발자는 포트번호 적용시 프로그램끼리 포트번호 충돌을 피하기위해 웰노운 포트영역은 피하는 것이 좋다



1.4. TCP & UDP[1/2]

Transport Layer (전송계층)에 포함된 프로토콜

TCP

- 두 시스템간에 데이터가 송상없이 안전하게 전송되도록 하는 통신 프로토콜
  - e-mail, FTP, HTTP
- 오류시 재전송하므로 신뢰성이 높다
- 전송데이터의 크기는 제한이 없다
- 파일 전송과 같이 신뢰성이 필요한 서비스에 주로 사용
- 연결형 통신
  - ✓ 한번 연결후 계속 데이터 전송 가능
- 보낸 순서대로 받아 응용프로그램에 전달

UDP

- 주로 실시간 방송과 같은 빠른 성능이 중요시되는 서비스에 사용한다
- 데이터 전송 오류또는 미전달시 전달 데이터를 삭제하므로 데이터의 신뢰성이 낮다
- 1회 전송 데이터의 크기 65,536byte
  - 초과시 나누어서 전송
- 비연결형 통신
  - 통신과정에서 연결 유지 안함

1.4. TCP & UDP[2/2] - TCP & UDP 비교

	TCP	UDP
신뢰도	높음	낮음
연결방식	연결지향	비연결 지향
패킷교환방식	가상회선 방식	데이터그램 방식
전송순서	전송 순서 보장	전송 순서 비보장
수신여부확인	수신 여부 확인	수신 여부 미확인
통신방식	1:1 통신	1:1 또는 1:N 또는 N:N
전송속도	상대적으로 느림	상대적으로 빠름

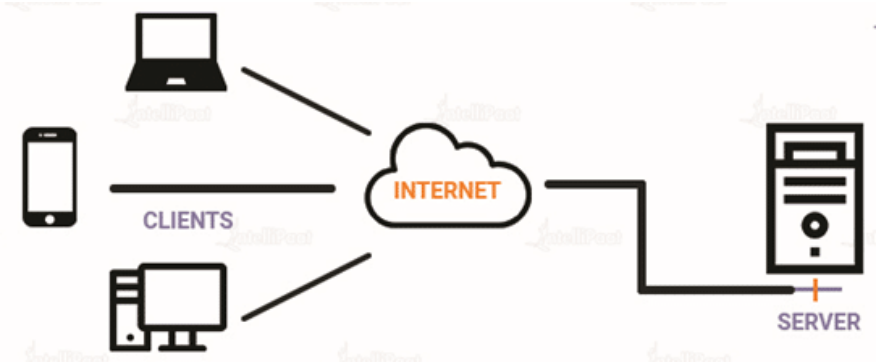
2

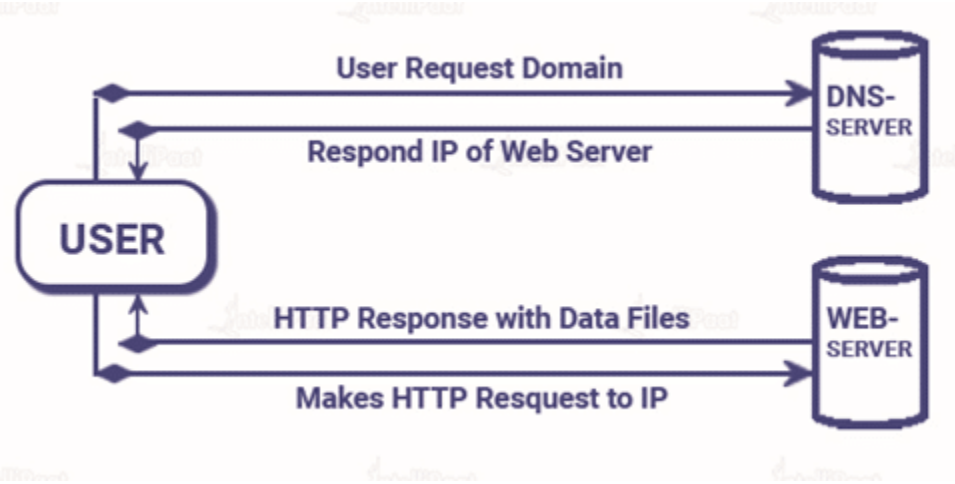
Client - Server Model 구현

1. Client-Server Model
2. 소켓 통신을 이용한 Client-Server Model

2.1. Client-Server Model [1/2] - 개념

- 서버-클라이언트 모델(Server-Client Model)은 컴퓨터 네트워크와 분산 컴퓨팅에서 사용되는 중요한 개념 중 하나이다.
- 역할분담 : 클라이언트는 서버에게 요청을 보내고 응답을 기다리며, 서버는 요청된 작업을 수행하고 결과를 클라이언트에게 제공
- 한 대의 서버에 다수의 클라이언트가 접속하여 서비스를 이용할 수 있는 방식을 서버 클라이언트 모델이라 한다





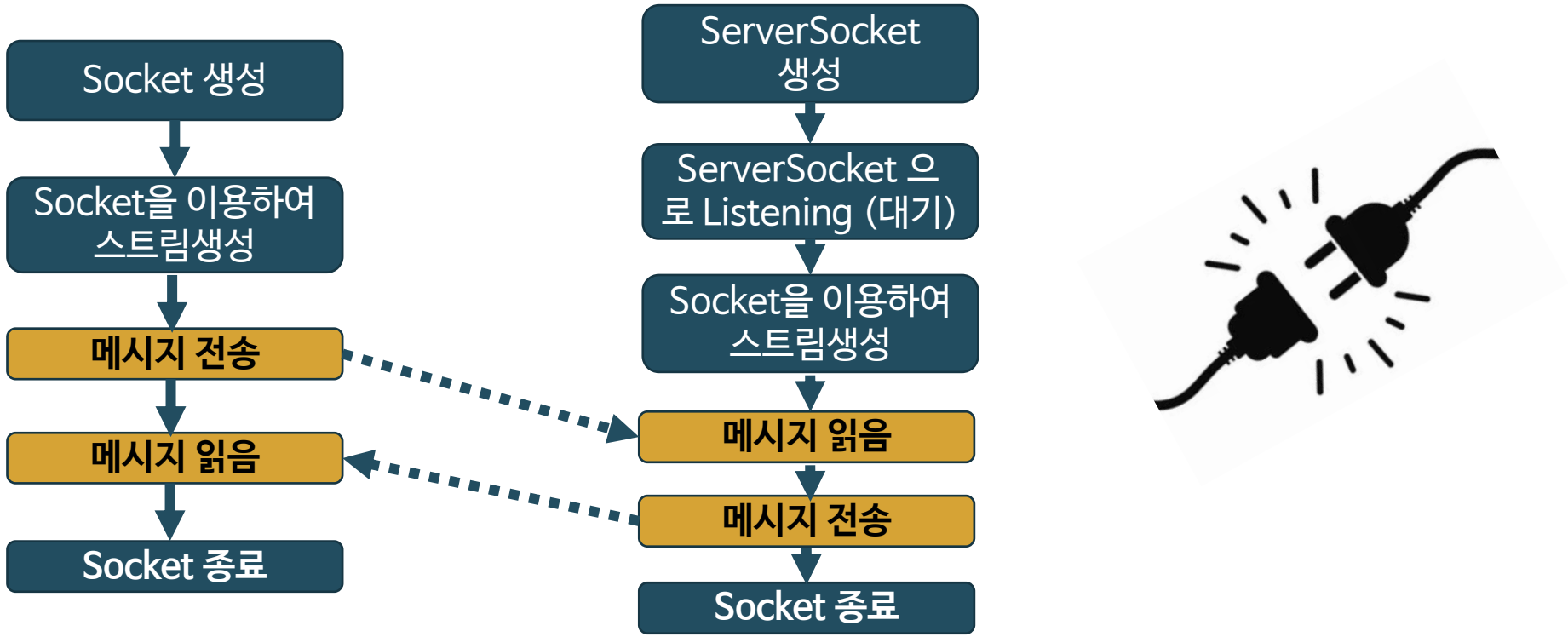
- 클라이언트와 서버 간의 통신은 일반적으로 소켓 프로그래밍을 통해 이루어지며, 자바에서는 java.net 패키지와 같은 라이브러리를 사용하여 이를 구현할 수 있다

2.1. Client-Server Model[2/2] - 종류

- 클라이언트-서버 모델은 여러 다양한 형태와 종류로 나타날 수 있으며, 어플리케이션의 목적과 요구 사항에 따라 다양한 방식으로 구현될 수 있다
- 소켓 기반 클라이언트-서버:
  - ✓ 로우 레벨의 클라이언트-서버 통신 모델이다.
  - ✓ 자바 소켓 프로그래밍을 사용하여 클라이언트와 서버 간의 TCP 또는 UDP 연결을 설정하고 데이터를 주고받는다.
- 웹 애플리케이션 서버 (Web Application Server):
  - ✓ 자바 웹 애플리케이션은 클라이언트(웹 브라우저)와 서버 간의 통신을 기반으로 한다.
  - ✓ 웹 브라우저는 HTTP 프로토콜을 사용하여 서버로 요청을 보내고, 서버는 해당 요청에 대한 응답을 생성한다.
  - ✓ 이전 자바 웹 애플리케이션은 서블릿과 JSP(JavaServer Pages)를 사용하여 개발되었지만 대부분 스프링으로 이전된 상태다
- RESTful 웹 서비스:
  - ✓ REST(Representational State Transfer) 아키텍처를 기반으로 하는 자바 웹 서비스는 HTTP를 사용하여 클라이언트와 서버 간의 통신을 처리한다. 자바에서는 JAX-RS(Java API for RESTful Web Services)를 사용하여 RESTful 웹 서비스를 구현할 수 있다.
- 메시징 시스템:
  - ✓ 자바에서는 JMS(Java Message Service)를 사용하여 메시지 지향 미들웨어(MOM)를 구현할 수 있다.
  - ✓ 이를 통해 클라이언트와 서버 간의 비동기 메시지 교환을 수행할 수 있다
- RMI (Remote Method Invocation): RMI는 자바에서 제공하는 분산 객체 지향 프로그래밍
- SOAP 웹 서비스: SOAP(Simple Object Access Protocol)는 XML 기반의 프로토콜로, 자바에서는 JAX-WS(Java API for XML Web Services)를 사용하여 SOAP 웹 서비스를 개발할 수 있다

2.2. 소켓 통신을 이용한 Client-Server Model[1/7] - Socket 개요

- 일반적으로 클라이언트와 서버시스템 구조는 클라이언트와 서버에서 각각 소켓을 만들어 연결한 뒤 클라이언트와 서버측에서 요청한 메시지를 상호간에 전송하고 받음으로써 요청한 작업을 수행한다
- Socket : TCP/IP 네트워크를 이용하여 통신프로그램을 쉽게 작성하도록 지원하는 기술
  - 두 응용프로그램 간의 양방향 통신 링크의 한쪽 끝 단
  - 소켓은 특정 IP의 포트번호와 결합한다
- OSI 7계층 중 응용 계층에 속하는 프로세스들은 데이터 송수신을 위해 반드시 소켓을 거쳐 전송 계층으로 데이터를 전달해야한다. 즉, 소켓은 전송 계층과 응용 프로그램 사이의 인터페이스 역할을 하며 떨어져 있는 두 호스트를 연결해준다.

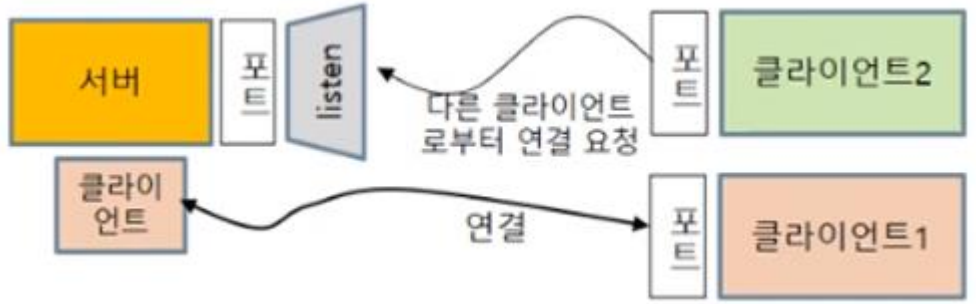


2.2. 소켓 통신을 이용한 Client-Server Model[2/7] - Server와 Client 연결 과정

1. 서버는 항상(Loop) 클라이언트의 연결 요청을 기다린다→ Listen

2. 클라이언트가 서버에게 연결 요청

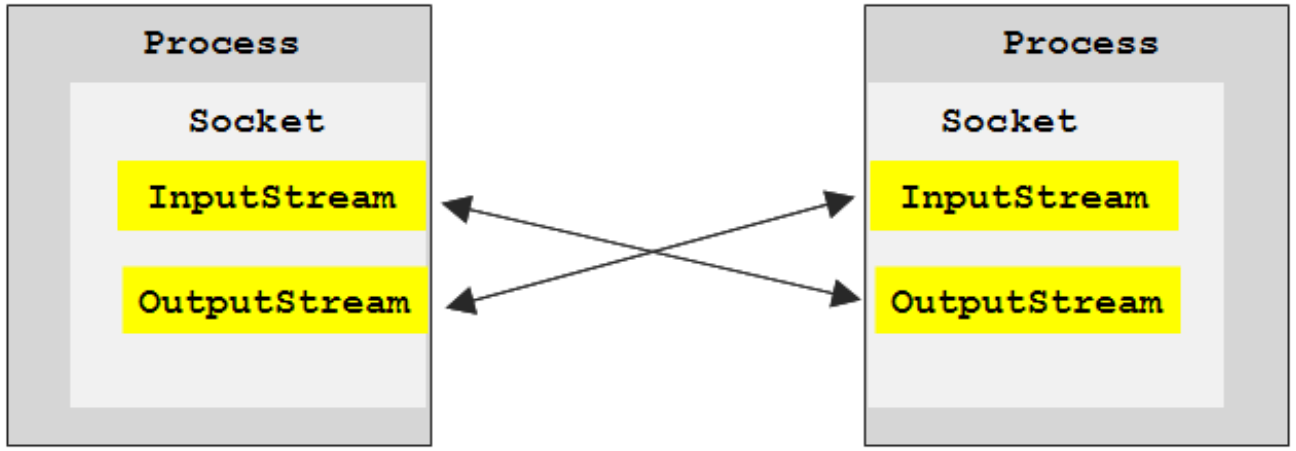
3. 서버가 클라이언트의 연결 요청 수락
   - 새로운 클라이언트 연결 소켓을 만들어 클라이언트와 통신하게 함(1:1)
   - 다시 다른 클라이언트의 연결을 기다림 → 반복



2.2. 소켓 통신을 이용한 Client-Server Model[3/7] - Java client Socket class

- 클라이언트 소켓에 사용되는 클래스
- java.net 패키지에 포함

생성자	설명
Socket()	연결되지 않은 상태의 소켓 생성
Socket(InetAddress address, int port)	소켓을 생성하고 IP주소와 포트에서 대기하는 서버에 연결
Socket(string host, int port)	소켓을 생성하여 호스트와 포트번호에 대기하는 서버에 연결 호스트이름이 null인 경우는 루프백 주소로 가정



루프백 주소는
로컬 호스트(localhost)를
가리키는데 사용
(127.0.0.1)

2.2. 소켓 통신을 이용한 Client-Server Model[4/7] - Socket 클래스의 메소드

메소드	설명
bind(SocketAdress bindpoint)	소켓에 로컬 IP주소와 로컬포트 지정
void close()	소켓을 닫는다
void connect(SocketAdress endpoint)	서버에 연결
InetAddress getInetAddress()	연결된 서버 IP주소 반환
InputStream getInputStream()	소켓의 입력 스트림 반환, 이 스트림을 이용하여 소켓이 상대방으로부터 받은 데이터를 읽을 수 있음
InetAddress getLocalAddress()	소켓의 로컬주소 반환
int getLocalPort()	소켓의 로컬 포트번호 반환
int getPort()	소켓에 연결된 서버의 포트번호 반환
OutputStream getOutputStream()	소켓의 출력 스트림 반환, 이 스트림을 출력하면 소켓이 상대편으로 데이터 전송
Boolean isBound()	소켓이 로컬주소에 결합되어 있으면 true반환
Boolean isConnected()	소켓이 연결되어 있으면 true반환
Boolean isClosed()	소켓이 닫혀있으면 true반환
void setSoTimeout(int timeout)	데이터 읽기의 타임아웃 시간 지정, 0 → 타임아웃 해제

2.2. 소켓 통신을 이용한 Client-Server Model[5/7] - Client의 Server 통신 과정

1. 클라이언트 소켓 생성 및 서버 접속
   Socket clientSocket = new Socket("255.255.1.190", 9000);
   Socket의 생성자에서 ("255.255.1.190") IP주소의 9000포트에 접속
2. 소켓으로부터 데이터를 전송할 입출력스트림 생성
   BufferedReader in = BufferedReader(
   new InputStreamReader(clientSocket.getInputStream());
   BufferedWriter in = BufferedWriter (
   new InputStreamWriter (clientSocket.getOutputStream());
3. 서버로 데이터 전송
   out.write("Hello");
   out.flush();
4. 서버로부터 데이터 수신
   String line = in.readline(); // 서버로부터 문자열 수신
5. 접속 종료
   clientSocket.close();

2.2. 소켓 통신을 이용한 Client-Server Model[6/7] - ServerSocket 클래스

- 서버소켓에 사용하는 클래스
- java.net 패키지에 포함

생성자	설명
ServerSocket(int port)	포트번호와 결합된 서버 소켓 생성

메소드	설명
Socket accept()	소켓에 로컬 IP주소와 로컬포트 지정
void close()	소켓을 닫는다
InetAddress getInetAddress()	서버소켓의 로컬 IP주소 반환
int getLocalPort()	서버소켓의 로컬 port번호 반환
Boolean isBound()	서버소켓이 로컬주소에 결합되어 있으면 true반환
Boolean isClosed()	서버소켓이 닫혀있으면 true반환
void setSoTimeout(int timeout)	accept()가 대기하는 타임아웃 시간 지정, 0 → 무한대기

2.2. 소켓 통신을 이용한 Client-Server Model[7/7] - Server의 Client 통신 과정

1. 서버 소켓 생성

   ServerSocket ServerSocket = new ServerSocket(9000);

   - Server는 9000포트에서 소켓 생성 후 클라이언트접속을 기다림

2. 접속요청시 접속후 새로운 Socket객체 반환후 새 객체를 통해 클라이언트와 통신

   Socket socket = serverSocket.accecpt();

3. 네트워크 입출력 스트림 생성

```
BufferedReader in = BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream());
BufferedWriter in = BufferedWriter (
                        new InputStreamWriter (clientSocket.getOutputStream());
```

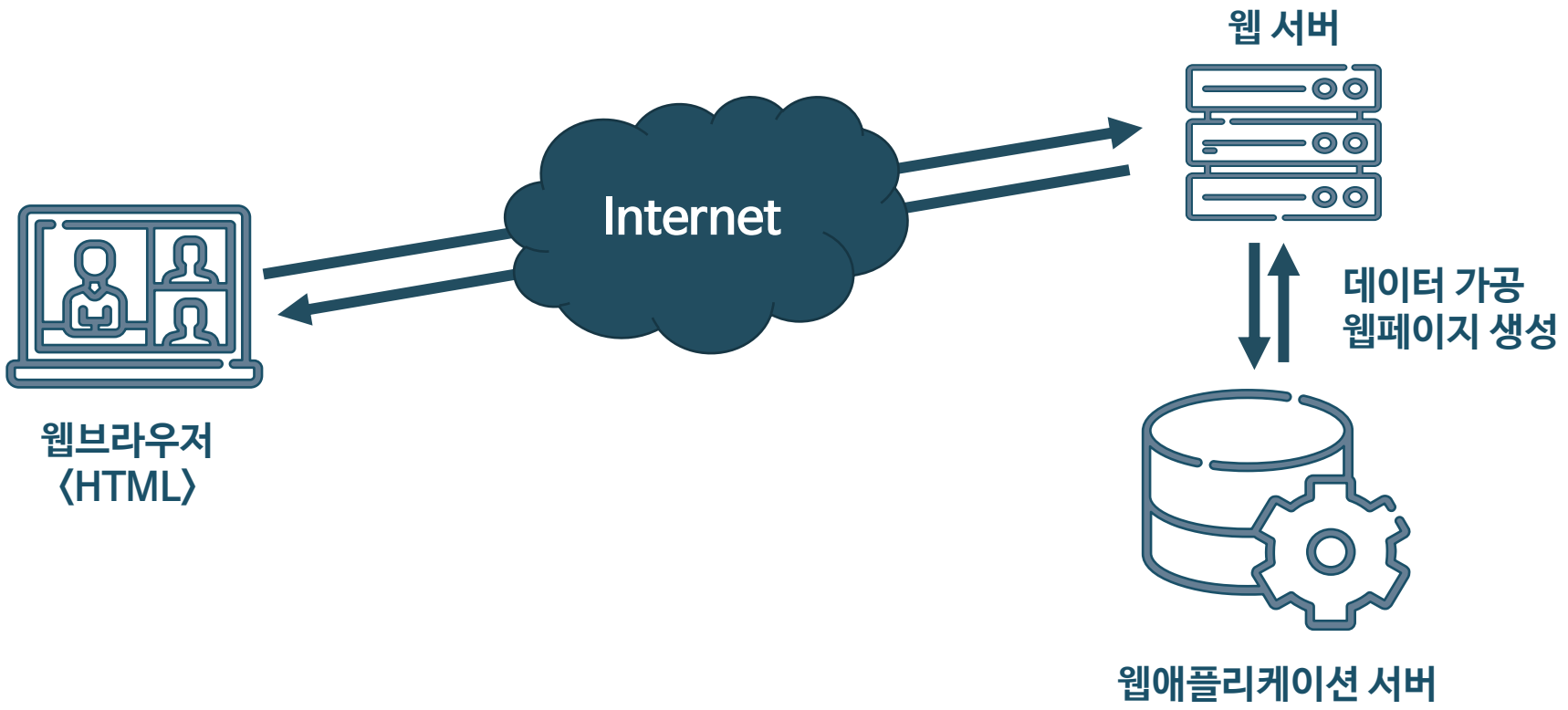
3

RESTful API

1. Web Application Server
2. HTTP
3. REST API

3.1. 웹 애플리케이션 서버(Web Application Server)-WAS

- 웹 서비스의 클라이언트 애플리케이션으로 사용되는 웹 브라우저는 HTTP 프로토콜에 따라 구현된다
- 웹의 형태로 새로운 서비스를 만들 때, 즉 웹 애플리케이션을 만들 때는 이미 개발되어 있는 다양한 웹 브라우저와 HTTP 프로토콜을 이용하면 된다
- 웹 애플리케이션 서버는 제공하고자 하는 서비스에 맞게 데이터를 가공하거나 다른 서버와 상호작용하면서 즉석에서 웹페이지를 만들어 제공하는 응용 프로그램을 웹서버와 구분하기 위해 웹 애플리케이션 서버라고 한다



3.2. HTTP (HyperText Transfer Protocol) 개요[1/6]

- HTTP는 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜이다
- 주로 HTML 문서를 주고받는 데에 쓰인다. 주로 TCP를 사용하고 HTTP/3부터는 UDP를 사용하며, 80번 포트를 사용한다
- 예를 들면, 클라이언트인 웹 브라우저가 HTTP를 통하여 서버로부터 웹페이지(HTML)나 그림 정보를 요청하면, 서버는 이 요청에 응답하여 필요한 정보를 해당 사용자에게 전달하게 된다.
- HTTP는 Connectionless 방식으로 작동한다
  - ✓ 서버에 연결하고, 요청해서 응답을 받으면 연결이 끊어지는 구조이며 기본적으로는 자원 하나에 대해서 하나의 연결을 만든다
  - ✓ 연결을 끊어버리기 때문에, 클라이언트의 이전 상태를 알 수가 없다는 단점이 있는데 해결책으로 session과 cookie를 사용한다
- HTTP를 통해 전달되는 자료는 http:로 시작하는 URL(인터넷 주소)로 조회할 수 있다.

HTML(Hyper Text Markup Language)

- HTML문서는 HTTP프로토콜 방식을 이용해 전송한다
- HTML 은 단순한 text 의미를 넘어서 링크, 이미지 등 다양한 것들을 표현할 수 있다는 의미다.
- HTML은 웹 문서의 뼈대를 구성하는 언어이며, 모든 웹 문서는 HTML로 이루어져 있고, HTML로 이루어진 문서만이 브라우저를 통해 웹 문서로서 읽어질 수 있다.

3.2. HTTP[2/6] - 주요 특징

- 대부분의 파일 형식 전송 가능
  - ✓ JSON, TEXT, IMAGE 파일은 물론 음성 파일 등의 거의 모든 파일 형식을 HTTP 통신을 이용해 전송 가능
- 클라이언트 – 서버 구조
  - ✓ HTTP는 클라이언트에서 서버에 요청을 하는 단방향 통신이다.
  - ✓ 서버는 클라이언트에 요청을 하지 않으며 클라이언트의 요청에 대한 응답만을 할 뿐이다.
  - ✓ 클라이언트의 요청에 대한 서버의 응답에는 요청 처리 결과에 따라 응답 코드가 다르게 온다.
  - ✓ 응답 코드 별로 처리 로직을 만들어 서버의 상황에 대한 대응이 가능해진다
- Stateless
  - ✓ HTTP 통신에서 서버는 클라이언트의 상태를 저장하지 않는다.
  - ✓ 클라이언트가 이전에 했던 요청이 무엇인지에 따라 반응이 달라지지 않는다는 것이다.
  - ✓ 정보 공유가 최소화되어 정보를 공유하기 위한 비용을 최소화 할 수 있다
- Connectionless
  - ✓ HTTP 통신은 연결(Connection)을 유지하지 않는 것을 기본 동작으로 가진다.
  - ✓ 지속적인 리소스 사용을 방지하기 위해서이다

3.2. HTTP[3/6] - HTTP vs HTTPS

- HTTP 프로토콜은 네트워크 통신을 작동하게 하는 기본 기술
- HTTPS에서는 브라우저와 서버가 데이터를 전송하기 전에 안전하고 암호화된 연결을 설정

	HTTP	HTTPS
	Hypertext Transfer Protocol	Hypertext Transfer Protocol Secure
기본 프로토콜	HTTP/1과 HTTP/2는 TCP/IP HTTP/3은 QUIC 프로토콜	HTTP 요청 및 응답을 추가로 암호화하기 위해 SSL/TLS와 함께 HTTP/2 사용
포트	기본 포트 80	기본 포트 443
용도	이전 텍스트 기반 웹 사이트	모든 최신 웹 사이트
보안	추가 보안 기능 없음	퍼블릭 키 암호화에 SSL 인증서 사용
장점	인터넷을 통한 통신 지원	웹 사이트에 대한 권위, 신뢰성 및 검색 엔진 순위 개선

3.2. HTTP[4/6] - HTTP 통신 패킷

- 패킷이란 웹 통신이 요청을 보내고 응답을 받을 때 전체 처리할 데이터를 일정 크기로 나누어 주고 받는데 이때 주고 받는 데이터의 블록을 의미한다
- 용량이 큰 파일들은 한번에 처리하면 오래 걸리고 중간에 전송 실패에 대한 대책과 동시 다수의 클라이언트 요청을 병렬처리하기 위한 것이다
- 패킷의 구조
  - HTTP 통신으로 보내는 패킷은 크게 헤더(Header)와 바디(Body)부분으로 나뉜다.
    - ✓ 헤더

      데이터 이외에 HTTP 선두에 삽입되는 부분을 말하며, 목적에따라 응답/요청 헤더로 나뉜다.
    - ✓ 바디

      실제 데이터 부분이다.

      HTML이외에도 JSON, TEXT, IMAGE 파일은 물론 음성 파일 등의 거의 모든 파일 형식이 가능하다

3.2. HTTP[5/6] - HTTP 통신 메시지 구조

- HTTP 메시지 구조는 웹에서 데이터를 전송하는 데 사용되는 규칙과 형식을 정의하는 방법이다
- HTTP 메시지는 클라이언트와 서버 간에 요청과 응답을 교환할 때 사용된다

- 요청(Request) 메시지
  - ✓ 요청 라인: 요청 메소드 (GET, POST 등), 요청 URI (Uniform Resource Identifier) 및 HTTP 버전 정보 포함
  - ✓ 헤더 필드: 추가 정보를 담고 있는 헤더 필드가 포함. 요청의 속성 및 요청을 처리하는 방법 정의
  - ✓ 빈줄 (empty line)
  - ✓ 본문(BODY): 선택적으로 요청 데이터 (예: HTML 양식 데이터 또는 업로드 파일)가 포함될 수 있다.
  - ❖ 요청 내용과 헤더 필드는 〈CR〉〈LF〉로 끝나야 한다. 즉, 캐리지 리턴(Carriage Return) 다음에 라인 피드(Line Feed)가 와야 한다. 빈 줄(empty line)은 〈CR〉〈LF〉로 구성되며 그 외 다른 화이트스페이스(whitespace)가 있어서는 안된다.

- 응답 (Responses) 메시지
  - ✓ 상태 라인: 상태 코드 (예: 200 OK, 404 Not Found)와 상태 메시지가 포함된다.
  - ✓ 헤더 필드: 추가 정보를 담고 있는 헤더 필드가 포함. 응답의 속성 및 클라이언트에게 전송되는 데이터에 대한 정보를 제공한다
  - ✓ 빈줄 (empty line)
  - ✓ 본문(BODY): 선택적으로 응답 데이터 (예: HTML 페이지, 이미지, JSON 데이터)가 포함될 수 있다

3.2. HTTP[6/6] - HTTP 메소드

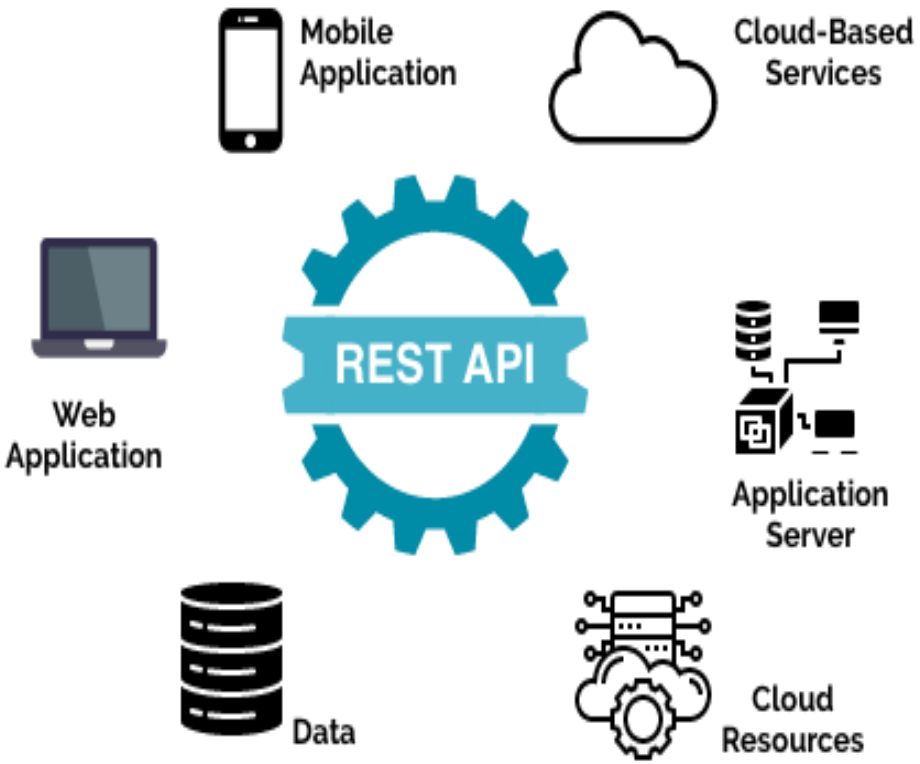
- 클라이언트가 브라우저를 통해서 어떠한 서비스 url을 통하거나 다른 방법으로 요청(request)를 하면, 서버에서는 해당 요청사항에 맞는 결과를 찾아서 사용자에게 응답(response)하는 형태로 동작한다.
- HTTP Method로 주고 받는 자료의 형태는 HTML 뿐 아니라 JSON 데이터 및 XML과 같은 형태의 정보도 주고 받을 수 있으며, 보통은 클라이언트가 어떤 정보를 HTML 형태로 받고 싶은지, JSON 형태로 받고 싶은지 명시해주는 경우가 많다.

- **HTTP METHOD**

✓ Create : 생성(POST)
✓ Read : 조회(GET)
✓ Update : 수정(PUT)
✓ Delete : 삭제(DELETE)
✓ HEAD: header 정보 조회(HEAD)

코드	HTTP 응답 상태코드 설명
2XX	클라이언트의 요청이 성공적으로 수행됨
3XX	클라이언트는 요청을 완료하기 위해 추가적인 행동을 취해야 함
4XX	클라이언트의 잘못된 요청
5XX	서버쪽 오류로 인한 상태코드

3.3. REST(Representational State Transfer) 개념[1/9]



✓ REST는 HTTP 프로토콜을 기반으로 하며, 리소스를 고유한 URI(Uniform Resource Identifier)로 식별하고, HTTP Method를 통해 Resource를 처리하도록 설계된 아키텍쳐를 의미한다

- REST가 필요한 이유
  - ✓ 다양한 클라이언트의 등장 → 안드로이드, 아이폰, 다양한 브라우저
  - ✓ 최근의 서버 프로그램은 다양한 클라이언트와 통신을 할 수 있어야 한다.
  - ✓ 이러한 멀티 플랫폼에 대한 지원을 위해 서비스 자원에 대한 아키텍처를 세우고 이용하는 방법을 모색한 결과, REST에 관심을 가지게 되었다

3.3. REST 구성요소[2/9]

- 자원(Resource):
  - ✓ URI모든 자원에 고유한 ID가 존재하고, 이 자원은 Server에 존재한다.
  - ✓ 자원을 구별하는 ID는 '/articles/article001'과 같은 HTTP URI 다.
  - ✓ Client는 URI를 이용해서 자원을 지정하고 해당 자원의 상태(정보)에 대한 조작을 Server에 요청한다.
- 행위(Verb):
  - ✓ Restful API의 행위는 HTTP Method를 이용한 자원에 대한 행위(조회, 생성, 수정, 삭제)를 의미한다
  - ✓ HTTP 프로토콜은 GET, POST, PUT, DELETE 와 같은 메소드를 제공한다.
- 표현(Representation of Resource)
  - ✓ Client가 자원의 상태(정보)에 대한 조작을 요청하면 Server는 이에 적절한 응답(Representation)을 보낸다.
  - ✓ REST에서 하나의 자원은 JSON, XML, TEXT, RSS 등 여러 형태의 Representation으로 나타낼 수 있다.
  - ✓ JSON 이나 XML를 통해 데이터를 주고 받는 것이 일반적이다

URI & URL

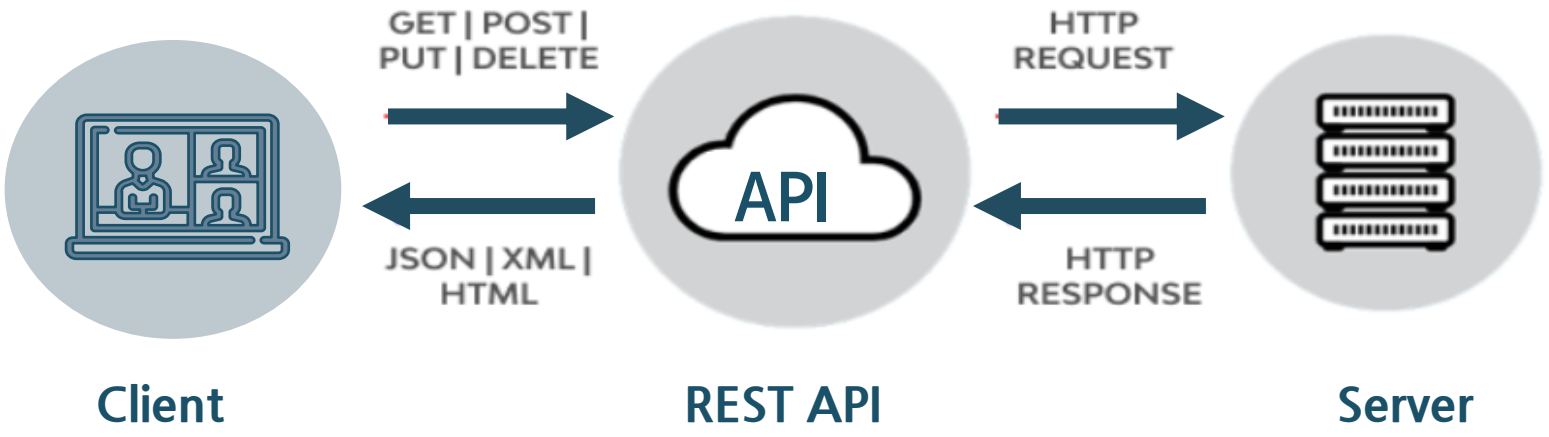
- URI(Uniform Resource Identifier) 는 인터넷상의 모든 자원을 표기하기 위한 규격이다
  - URI는 프로그램의 실행경로와 파라미터등을 포함한 형태 까지를 말한다
  - http://www.naver.com/product/productlist?pid=1001021 -> URI
- URL(Uniform Resource Locator)는 URI의 서브셋으로 대표적인 URI 사용의 한 형태 이다.
  - URL은 구체적인 파일자원등에 대한 주소를 말한다
  - http://www.naver.com/product/productlist.html -> URL

3.3. REST의 특징 [3/9]

- Server-Client(서버-클라이언트 구조)
  - 자원이 있는 쪽이 Server, 자원을 요청하는 쪽이 Client가 된다.
  - 서로 간 의존성이 줄어든다.
- Stateless(무상태)
  - HTTP 프로토콜은 Stateless Protocol이므로 REST 역시 무상태성을 갖는다.
  - Client의 context를 Server에 저장하지 않는다.
  - Server는 각각의 요청을 완전히 별개의 것으로 인식하고 처리한다.
- Cacheable(캐시 처리 가능)
  - 웹 표준 HTTP 프로토콜을 그대로 사용하므로 웹에서 사용하는 기존의 인프라를 그대로 활용할 수 있다.
  - 캐시 사용을 통해 응답시간이 빨라지고 REST Server 트랜잭션이 발생하지 않기 때문에 전체 응답시간, 성능, 서버의 자원 이용률을 향상시킬 수 있다.
- Layered System(계층화)
  - Client는 REST API Server만 호출한다.
  - REST Server는 다중 계층으로 구성될 수 있다.
  - PROXY, 게이트웨이 같은 네트워크 기반의 중간 매체를 사용할 수 있다.
- Uniform Interface(인터페이스 일관성)
  - URI로 지정한 Resource에 대한 조작을 통일되고 한정적인 인터페이스로 수행한다.
  - HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용이 가능하다.
  - 특정 언어나 기술에 종속되지 않는다.

3.3. REST API 개요[4/9]

- API(Application Programming Interface)
  - ✓ 데이터와 기능의 집합을 제공하여 컴퓨터 프로그램간 상호작용을 촉진하며, 서로 정보를 교환가능 하도록 하는 것
- REST API
  - ✓ 웹상에서 사용되는 여러 리소스를 HTTP URI로 표현하고, 해당 리소스에 대한 행위를 HTTP Method로 정의하는 방식이다
  - ✓ REST API 설계는 아키텍처 스타일을 의미하며 표준은 아니다

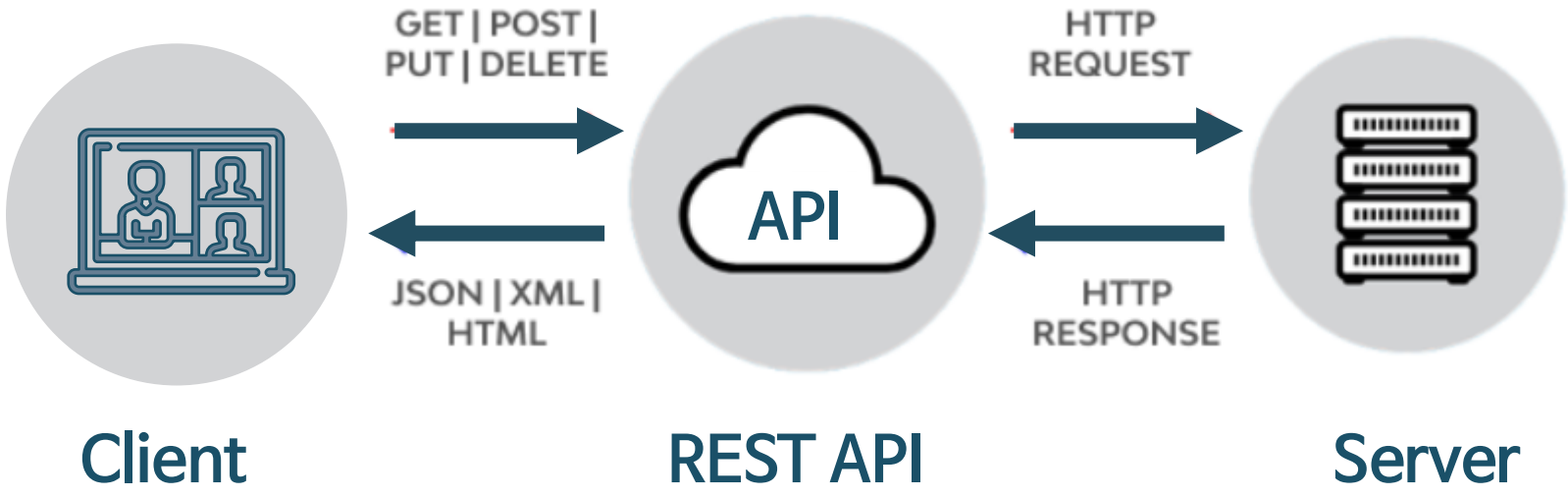


3.3. REST API 사용시 이점[5/9]

- **확장성**
  - ✓ REST가 클라이언트-서버 상호 작용을 최적화하기 때문에 효율적으로 크기를 조정할 수 있다.
  - ✓ 무상태는 서버가 과거 클라이언트 요청 정보를 유지할 필요가 없기 때문에 서버 로드를 제거
  - ✓ 잘 관리된 캐싱은 일부 클라이언트-서버 상호 작용을 부분적으로 또는 완전히 제거
  - ❖ 이러한 기능들은 성능을 저하시키는 통신 병목 현상을 일으키지 않으면서 확장성을 지원한다.
- **유연성**
  - ✓ RESTful 웹 서비스는 완전한 클라이언트-서버 분리를 지원.
  - ✓ 각 부분이 독립적으로 발전할 수 있도록 다양한 서버 구성 요소를 단순화하고 분리.
  - ✓ 서버 애플리케이션의 플랫폼 또는 기술 변경은 클라이언트 애플리케이션에 영향을 주지 않는다.
- **독립성**
  - ✓ REST API는 사용되는 기술에 영향을 받지 않는다
  - ✓ API 설계에 영향을 주지 않고 다양한 프로그래밍 언어로 클라이언트 및 서버 애플리케이션을 모두 작성할 수 있다. 또한 통신에 영향을 주지 않고 양쪽의 기본 기술을 변경할 수 있다.

3.3. REST API 작동[6/9]

1. 클라이언트가 서버에 요청을 전송. 클라이언트가 API 문서에 따라 서버가 이해하는 방식으로 요청 형식을 지정
2. 서버는 클라이언트를 인증하고 해당 요청을 수행할 수 있는 권한이 클라이언트에 있는지 확인
3. 서버가 요청을 수신하고 내부적으로 처리.
4. 서버가 클라이언트에 응답 반환. 응답에는 요청이 성공했는지 여부를 클라이언트에 알려주는 정보가 포함. 응답에는 클라이언트가 요청한 모든 정보도 포함



3.3. REST API 디자인 가이드[7/9]

- URI는 정보의 자원을 표현해야 한다. (리소스명은 명사를 사용)
  - resource별로 두개의 기준 url을 사용한다(복수와 단수)
    - ✓ 목록(Collection)을 위한 url : /articles
    - ✓ 목록중 특정 자원 개체(Document)를 위한 url : /articles/article001
  - 기준 url에는 동사를 두지 않는다
- 간결하고 직관적인 기준 URL유지 → 행동 유도성(별도의 가이드가 없어도 사용자가 직관적으로 알수 있게)
- 컬렉션이나 요소들을 다루는 자원에 대한 행위는 HTTP메소드를 사용한다

Resource	POST(create)	GET(read)	PUT(update)	DELETE(delete)
/articles	새로운 article 생성	article 목록	article 에 대한 대량 업데이트	모든 article 삭제
/articles/article001	에러	해당 article 보기	해당 개체가 있으면 업데이트 없으면 에러	해당 article 삭제

3.3. REST API 디자인 가이드 - URI Naming시 주의할 점[8/9]

URI Naming시 주의점

- 슬래시 구분자(/)는 계층 관계를 나타내는 데 사용
- URI 마지막 문자로 슬래시(/)를 포함하지 않는다
- 하이픈(-)은 URI 가독성을 높이는데 사용
- 언더바(_)는 URI에 사용하지 않는다.
- URI 경로에는 소문자가 적합하다.
- 파일 확장자는 URI에 포함시키지 않는다

Collection 과 Document

- 컬렉션과 도큐먼트는 모두 리소스라고 표현할 수 있으며 URI에 표현된다
- DOCUMENT는 단순히 문서 또는 한 객체라고 이해하고
- 컬렉션은 문서들의 집합, 객체들의 집합인 디렉터리 개념으로 이해할 수 있겠다

3.3. REST API - URI & HTTP Method 예시[9/9]

CRUD	HTTP	URI
전체 리소스 조회	GET	/resources
특정 리소스 조회	GET	/resources/:id
리소스 생성	POST	/resources
리소스 전체 수정	PUT	/resources/:id
특정 리소스 삭제	DELETE	/resources/:id

✓ :id와 같이 변하는 값은 하나의 특정 resource를 나타내는 고유값이어야 한다

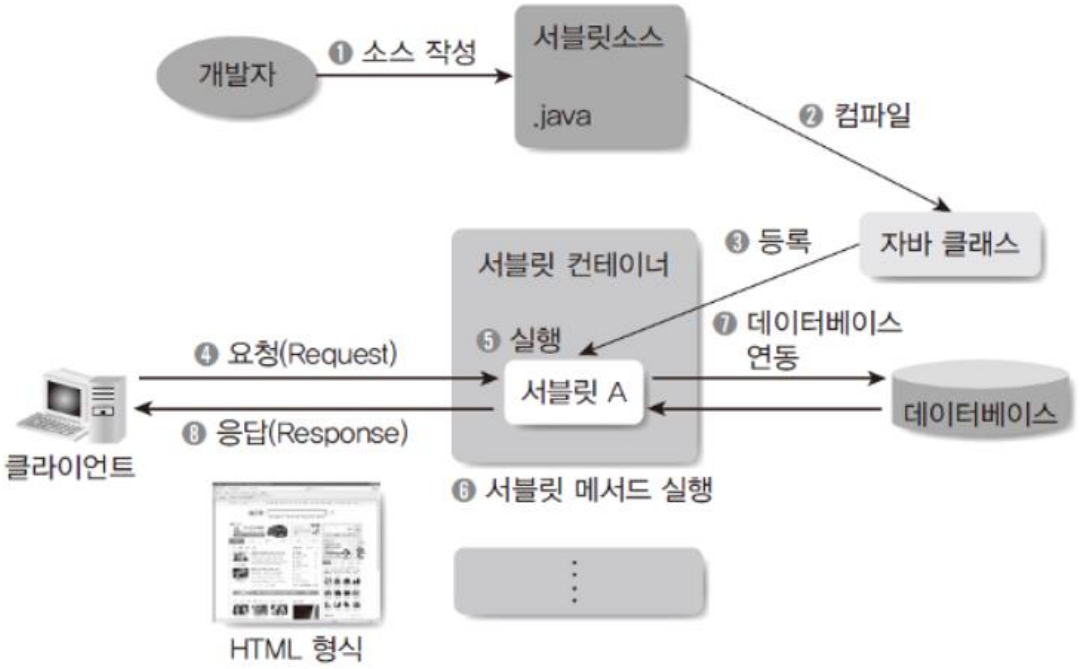
4

Servlet

1. 서블릿(Servlet) 개요
2. 서블릿(Servlet) 장단점
3. 서블릿& JSP와 모던 웹 개발기술
4. 서블릿& JSP는 왜 사라지고 있는가?

4.1. 서블릿(Servlet) 개요

- Dynamic Web Page를 만들 때 사용되는 자바 기반의 웹 애플리케이션 프로그래밍 기술
- 웹 서버에서 동작하며, 클라이언트의 요청을 처리하고 동적인 웹 페이지를 생성하는 데 사용된다.
- 주로 JSP와 함께 사용된다
- 서블릿 실행을 위해서는 웹 애플리케이션 형식으로 패키징 하는 과정이 필요하며 실행은 서블릿 컨테이너를 통해 이루어진다



1. HttpServlet 클래스를 상속받는 서블릿 클래스 작성
2. 컴파일후 웹 애플리케이션으로 패키징
3. 서블릿 컨테이너에 배포
4. 클라이언트의 URL 요청
5. 애너테이션에 등록된 URL 매핑정보를 참고해 해당 서블릿 실행
6. 요청 메소드에 따라 서블릿의 doGet(), doPost() 등의 메소드 호출
7. 서블릿은 데이터베이스 연동 등 필요한 작업 수행
8. 데이터를 포함한 HTML 형식의 데이터를 클라이언트에게 전달

4.2. 서블릿(Servlet) 장단점

장점

- 자바를 기반으로 하므로 자바 API를 모두 사용할 수 있다.
- 운영체제나 하드웨어에 영향을 받지 않으므로, 한 번 개발된 애플리케이션은 다양한 서버 환경에서도 실행할 수 있다.
- 웹 애플리케이션에서 효율적인 자료 공유 방법을 제공한다.
- 다양한 오픈소스 라이브러리와 개발도구를 활용할 수 있다

단점

- HTML 응답을 위해서는 출력문으로 문자열 결합을 사용해야 한다.
- HTML 을 서블릿에서 포함할 경우 화면 수정이 어렵다.
- HTML form 데이터 처리가 불편하다.
- 기본적으로 단일 요청과 응답을 처리하는 구조로 다양한 경로의 URL 접근을 하나의 클래스에서 처리하기 어렵다. (예를들면 Rest API 구현)

4.3. 서블릿& JSP와 모던 웹 개발기술

- **서블릿 (Servlet)**:
  - **정의**: 자바로 작성된 웹 애플리케이션 컴포넌트로, 클라이언트 요청을 처리하고 동적 웹 페이지 생성에 사용.
  - **기능**: 요청 처리, 데이터베이스 연동, 비즈니스 로직 실행, HTML 코드 생성 등의 기능을 수행
  - **장점**: 자바 기반, 유연성, 컨트롤 가능성, 재사용성이 높음.
- **JSP (JavaServer Pages)**:
  - **정의**: HTML 내에 자바 코드를 삽입하여 동적 웹 페이지를 생성하는 기술
  - **기능**: 서블릿과 유사한 기능을 제공하며, HTML과 자바 코드를 혼합하여 웹 페이지를 생성
  - **장점**: HTML과 자바를 쉽게 통합하여 웹 페이지를 구성할 수 있음

✓ 현재는 서블릿과 JSP보다 더 현대적이고 효율적인 웹 개발 기술과 프레임워크가 널리 사용되고 있으며
✓ 이러한 기술은 더 빠른 개발과 더 나은 성능을 제공하고 모던 웹 개발에 필요한 다양한 기능과 보안을 더 쉽게 제공하여 특별히 유지보수 기능외에는 사용하지 않는 추세이다.

4.4. 서블릿& JSP는 왜 사라지고 있는가?

- **복잡성과 유지보수 어려움**:
  - ✓ 서블릿과 JSP는 자바 코드와 HTML을 혼합해 사용하므로 코드 복잡성이 증가하고 유지보수가 어려워짐
- **모던 웹 프레임워크의 등장**:
  - ✓ 현재는 더 현대적이고 강력한 웹 프레임워크 및 라이브러리(예: Spring, Django, Ruby on Rails)가 등장하여 개발 생산성과 효율성이 향상되었음.
- **RESTful API와 단일 페이지 애플리케이션(SPA)의 인기**:
  - RESTful API등을 사용하여 더 빠르고 반응성 있는 웹 애플리케이션을 구축할 수 있어서 서블릿과 JSP보다 선호됨
- **마이크로서비스 아키텍처의 확산**:
  - 서블릿과 JSP는 현재의 마이크로서비스 아키텍처와의 통합이 어렵다는 문제가 있다
- **성능 문제**:
  - 서블릿과 JSP는 요청마다 자바 스레드를 생성하고 메모리를 소비하기 때문에 대규모 트래픽을 다루기에 적합하지 않으며 더 효율적인 웹 서버 기술이 사용된다.
- **컨테이너 종속성**:
  - 서블릿과 JSP는 Java EE 컨테이너에 종속적이기 때문에 다른 언어나 환경으로 마이그레이션하기 어려울 수 있다.
- **보안 문제**:
  - 서블릿과 JSP에서 개발자의 주의가 부족하면 보안 취약점이 발생할 수 있다
  - 스프링부트 같은 웹 프레임워크는 이를 보완하여 향상된 보안 기능을 제공한다.