DNS, 서버, 클라우드

지난 주차에서는 백엔드와 프론트가 무엇인지, API에 대한 기본 개념

그리고 네트워크에 대해 간략히 알아보았습니다.

백엔드와 프론트가 어떤 역할은 한다고 했었죠?

개발은 우리 눈에 보이는 영역을 개발하는 일과 눈에 보이지 않는 뒷단을 개발하는 일 크게 두 분야로로 나뉜다고 했죠.

여기서 저희가 맡은 백엔드는 **프론트엔드에 있는 사용자들이 원하는 행동(이벤트)을 처리하는 역할 입니다.**

그럼 API란 무엇이였나요?
API란? Application Programming Interface.

프로그램들이 서로 상호작용하는 것을 도와주는 매개체 역할

즉, 프론트와 백엔드 간의 정보전달 매개체 역할을 해주었죠.

프론트와 백엔드의 연결에 대해 알게되었으니

실제 통신이 어떻게 이뤄지는지도 저희가 알아봤었죠.

전 세계의 컴퓨터가 인터넷이라는 네트워크에 묶여

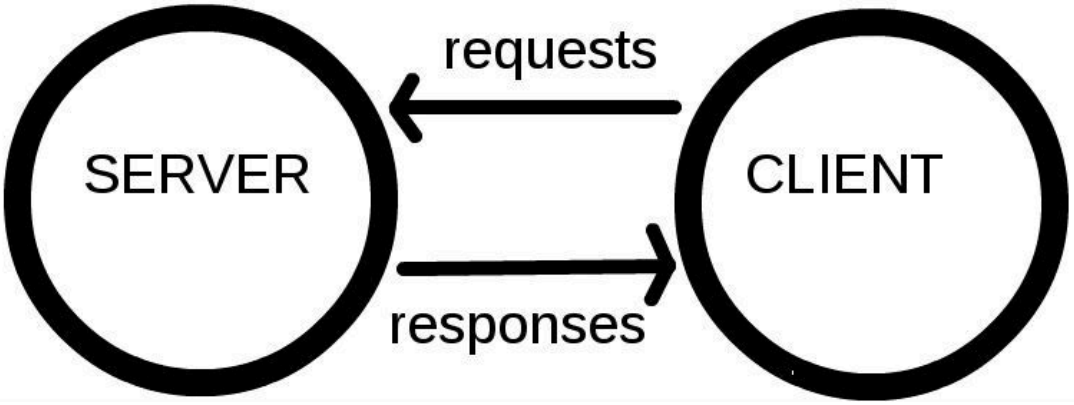
IP 주소를 통해 서로를 식별하고 찾아갈 수 있다는 건 알게되었지만,

우리가 구글을 접속할때 “216.58.200.14” 가 아닌 www.google.com 을 입력하였을 때,

접속이 되는 이유는 무엇일까요?

DNS

유튜브(www.youtube.com)에 접속하여 영상을 본다고 가정 하였을 경우

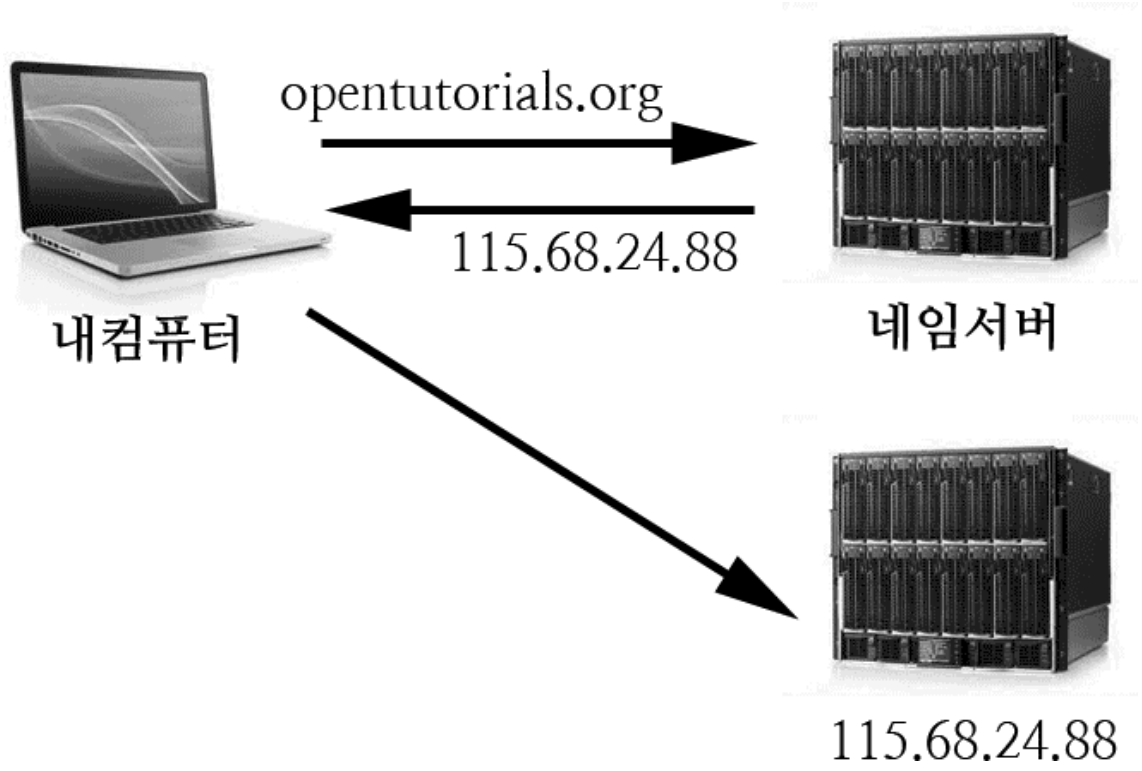


(클라이언트는 유튜브를 볼 디바이스를 의미하고, 서버는 유튜브 내부의 영상이 저장되어있는 데이터센터)

네트워크는 위에서 설명한대로 라우터를 활용한 광케이블 통신으로 구성되어있을것이고 서버의 IP주소를 통해 해당하는 서버가 있는 네트워크로 찾아가게 됨.

여기서 문제가 있습니다. 과연 모든 웹페이지의 IP주소를 어떻게 다 외울 수 있을까...?
IP주소는 단순한 숫자의 나열이므로 외우기가 쉽지 않음. 이 문제를 해결하기 위해 "**도메인**"이라는 개념이 등장

> 💡 **도메인**이란?
>
> IP 주소는사람이 기억하기 어렵기 때문에 각 IP에 기억이 쉽게끔 이름을 부여할 수 있게 했는데, 이것을 도메인이라 부름.
>
> ex) www.naver.com
> www.google.com



내 컴퓨터 → 도메인 입력 → DNS 서버 → IP 획득 → 라우터 → ARP 프로토콜 → MAC 주소 획득 → 서버 → HTTP 요청과 응답

```
POST /payment-sync HTTP/1.1

Accept: application/json
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 83
Content-Type: application/json
Host: intropython.com
User-Agent: HTTPie/0.9.3

{
    "imp_uid": "imp_1234567890",
    "merchant_uid": "order_id_8237352",
    "status": "paid"
}
```

정리하자면

1. 브라우저는 DNS서버로 가서 도메인을 확인하고 웹사이트가 있는 서버의 IP주소를 찾음
2. 서버에게 웹사이트의 사본을 클라이언트에게 보내달라는 HTTP 요청 메시지를 서버로 전송
3. 서버는 클라이언트의 요청을 승인하고 웹 사이트의 정보를 패킷으로 묶어서 브라우저로 보냄
4. 브라우저는 이 패킷들을 활용하여 웹사이트로 만들어서 사용자에게 보여줌

좋아요
그럼 지금까지 네트워크에서의 연결에 대해서 알아보았고,
실제 네트워크는 IP를 통해 목적지까지 길을 찾아가지만
사람들은 그 많은 사이트의 IP 주소를 기억할 수 없으니 좀 더 기억하기 쉬운 도메인을 통해
사이트에 접근하게 된다는 걸 알게 되었습니다.

이번에는 도메인을 통해 해당 사이트에 접근하게 되었을 때
저희에게 사이트에 대한 정보를 보여주는 “서버”에 대해서 알아 보도록 하겠습니다.

서버

서버에 대해서 다들 한번쯤을 들어보셨을꺼라 생각합니다.

서버란 무엇일까요?

서버(server)의 사전적 의미 자체는 ‘서비스 (service)를 제공하는 사람’

현대의 ‘서버’라는 의미는 좀 더 광범위하게 변화를 하게 되는데요,

인터넷 네트워크 환경에서 다른 컴퓨터에게
“다양한 기능, 데이터, 서비스를 제공하는 컴퓨터나 소프트웨어 전반”
을 가리키는 의미로 발전하게 됩니다.

‘사람’에 국한되던 원래 의미에서
컴퓨터나 장치, 소프트웨어 전반으로 확장되어
사용하게 된 것이죠.

‘서버’는 누군가에게 서비스를 ‘제공’하는 ‘역할’ 자체를 지칭하게 되버린거죠.

누군가에게 서비스를 제공할 때, 그 서비스가 한 가지만 있는 것은 아닙니다.
이처럼 서버에도 여려가지가 있는데요,

웹 서버 : 웹사이트 서비스를 제공하기 위한 서버
도메인 서버 : 도메인을 관리하기 위한 서버
이미지 서버 : 이미지를 관리하기 위한 서버
이메일 서버 : 이메일을 관리하기 위한 서버
DB 서버 : 데이터 정보를 관리하기 위한 서버
게임 서버 : 게임을 제공하기 위한 서버
미디어 서버 : 미디어를 제공 관리하기 위한 서버

등등...

‘서버’라는 같은 단어로 통칭되지만,
수없이 많은 기능들을 제공하는 컴퓨터와 소프트웨어들이 존재합니다.

서버는 컴퓨터에만 해당하지 않습니다.

서버라는 이름을 붙일 수 있는 것에 컴퓨터나 장치 같은 ‘하드웨어’만 있는 것이 아닙니다.

기능을 제공하게끔 도와주는 ‘소프트웨어’까지도 서버라는 개념에 포함됩니다.

예를 들면
홈페이지가 인터넷에서 구현되기 위해서는 여러 종류의 서버가 필요합니다.

그중에 가장 중요한 1가지가 웹 서버 (web server) 라고 할 수 있습니다.

홈페이지 제작이 되기 위한 서버를
하드웨어적 측면과 소프트웨어적 측면으로 나눠서 살펴보자면,

우선, 하드웨어적인 측면에서 '웹 서버 컴퓨터'가 필요합니다.

(각각의 목적에 맞춰 여러 대가 될 수도 있는)

다음으로, 소프트웨어적인 측면에서 웹 서버 컴퓨터 (하드웨어)에 들어있는 홈페이지 재료들을

웹에 뿌려지고 동작하게 하는 '웹 서버 프로그램'이 필요합니다.

웹 서버 프로그램에는 아파치 (Apache), 엔진엑스 (Nginx) 등 여러 가지가 있습니다.

그렇다면 저희는 어떤 서버를 만들게 되는 걸까요?

WAS
웹 어플리케이션 서버(WAS, Web Application Server)

DB 조회나 로직 처리를 요구하는 동적 컨텐츠를 제공하기 위해 만들어진 Application Server

✅ 웹 서버의 역할과 예시

웹 서버는 클라이언트가 웹 브라우저를 통해 요청한 정적 콘텐츠를 제공하는 역할을 합니다. 웹 서버는 주로 HTTP 프로토콜을 사용하여 작동하며, 클라이언트가 URL을 통해 요청한 웹 페이지를 찾아 전송해줍니다.

> 예시: 회사 홈페이지, 블로그, 뉴스 사이트 등

✅ WAS 서버의 역할과 예시

WAS 서버는 웹 애플리케이션을 실행하여 동적 콘텐츠를 생성하고, 웹 서버와 클라이언트 간의 데이터 처리를 담당하는 역할을 합니다. WAS 서버는 클라이언트의 요청에 따라 데이터베이스에서 정보를 가져오거나, 웹 애플리케이션을 실행하여 동적인 웹 페이지를 생성한 후 결과를 웹 서버에 전달합니다. 웹 서버는 이를 받아 클라이언트에게 전달합니다.

> 예시: 온라인 쇼핑몰, 은행 인터넷 뱅킹, SNS 등

실제 웹 서비스 환경에서는 웹 서버와 WAS 서버가 협업하여 작동합니다. 일반적으로 웹 서버는 정적 콘텐츠를 처리하고 WAS 서버는 동적 콘텐츠를 처리하는 역할을 맡아, 사용자에게 원활하고 다양한 웹 서비스를 제공하게 됩니다. 이를 통해 웹 사이트의 로딩 속도와 서비스 품질이 향상되며, 웹 애플리케이션의 성능이 개선됩니다.

저희가 JAVA/SPRING을 통해 개발하게 될 API들도

WAS인 톰캣서버에서 돌아가게 됩니다.

그렇다면 위와같은 서버는 어떻게 실행시키고 배포하는 걸까요?

우선 서버를 배포하기 위해서는 하드웨어 장비가 필요합니다.

하지만 하드웨어 장비를 구매하거나 구성하기 위해서는

많은 돈과 시간, 그리고 인프라 구축이 들기에

배포하는데 큰 어려움을 겪게 됩니다.

이를 해결하기 위해 나온 것이 클라우드 입니다.

클라우드(AWS)

클라우드(Cloud)

클라우드(Cloud)란 **'인터넷'**입니다.

즉, 모든 가상화 서비스가 이뤄지는 공간을 말합니다.

클라우드 서비스(Cloud Service)

클라우드 서비스(Cloud Service)란 **'인터넷 서비스'**

인터넷을 이용해 서비스를 제공하는 것을 '클라우드 서비스' 또는 'SaaS' 라고 부른다.

예를 들어, 과거에는 문서를 작성하면 내 컴퓨터 안의 폴더에 저장했다. 인터넷 연결이 필요 없던 시절이다.

현재는 내 컴퓨터뿐만 아니라 구글 클라우드 내에 문서를 함께 저장한다. 지금은 인터넷 연결이 필요하다.

현재의 방법은 인터넷이 가능한 어느 곳에서나 문서를 확인할 수 있다. 즉, **클라우드 서비스는 '인터넷'이 가능한 환경에서만 사용**할 수 있다. 네이버 클라우드, 구글 클라우드가 다음의 예이다.

클라우드 컴퓨팅(Cloud Computing)

클라우드 컴퓨팅(Cloud Computing)이란 내 컴퓨터의 서버, 네트워크 등을 사용하는 것이 아닌 **'컴퓨팅 리소스'**를 제공하는 회사를 통해 서버, 네트워크를 제공받아 사용하는 것이다.

컴퓨팅 리소스를 제공하는 대표적인 클라우드 서비스 제공자에는 'Goggle Cloud', 'MS Azure', 'AWS'가 있다.
이런 종류의 클라우드 특성에 따라 IaaS, PaaS로 나눌 수 있다.

① **IaaS**(=Infra as a Service, 인프라 서비스)

- 클라우드의 가장 기본적인 제공 형태

- 서버, 스토리지, 네트워크 장비, 서버용 운영체제 등을 빌려주는 서비스

② **PaaS**(=Platform as a Service, 플랫폼 서비스)

- 인프라 서비스에서 한단계 더 발전한 클라우드 서비스
- 인프라와 IT기술을 빌려주고, 다양한 지원 서비스도 함께 제공
- 임대 서버가 이에 해당함, OS가 설치된 서버에 사용자가 애플리케이션 등을 설치해서 사용해야함

③ **SaaS**(=Software as a Service, 소프트웨어 서비스)
- 인프라나 플랫폼 뿐만 아니라 애플리케이션까지 제공
- 과거에 pc나 서버 등에 설치해서 이용해야 했던 소프트웨어를 클라우드를 통해 제공하는 서비스

④ **EaaS**(=Everything as a Service)
- Iaas, Paas, SaaS 세가지를 통칭하는 말
- XaaS 라고도 함
- AWS는 EaaS라고 할 수 있음

[사용 이유]

비용 절감, 즉 **경제성** 측면에서 좋기 때문이다.

우리는 서버 한대를 구축하려면 100만원 이상이 필요하다. 이후, 서버가 더 필요하다면 추가 구매 비용이 들 것이다. 하지만 클라우드를 사용한다면 초기 비용과 운영 비용에 드는 시간과 비용을 절감할 수 있을 것이다.

많이 들어보신 AWS의 EC2가 클라우드 컴퓨팅의 일종입니다.

클라우드를 통해 어떻게 서버를 구축하고 배포하는지는 4주차에 설명드리도록 하겠습니다.

요약하자면

클라우드 사용 이유

- 사용한 만큼 비용을 지불하기 때문에 비용 절감, 경제성이 좋다.
- 서버를 구축해주기 때문에 인프라 운영이 쉽다.
- 클라우드는 'AWS'과 같은 UI/UX를 통해서 편하게 서버 스펙을 바꿀 수 있다.