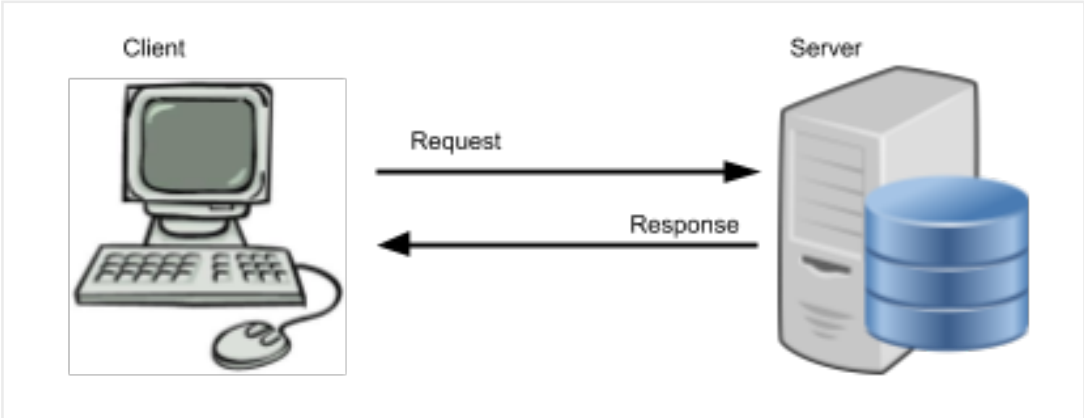
6월 5일
HTTP 프로토콜
HTTP 프로토콜이란?
HTTP는 인터넷 상에서 데이터를 주고 받기 위한 서버/클라이언트 모델을 따르는 프로토콜
TCP/IP위에서 작동함
어떤 종류의 데이터든 전송 할 수 있도록 설계됨 (Ex. HTML문서, 이미지, 동영상, 오디오 등)
하이퍼텍스트 기반으로 데이터를 전송한다 = 링크기반으로 데이터에 접속한다

작동 방식
클라이언트에서 요청(request)를 보내면 서버에서 요청을 처리해서 응답(response)하는 방식



클라이언트 :
서버에 요청하는 클라이언트 소프트웨어(IE, Chrome 등)가 설치된 컴퓨터 이용
클라이언트는 URI를 이용 서버에 접속하고 데이터를 요청할 수 있다

서버:
클라이언트의 요청을 받아, 요청을 해석하고 응답하는 소프트웨어가 설치된 컴퓨터
(Apache,nigix 등) 서버 소프트웨어

Tip. 웹서버는 보통 표준포트인 80번 포트로 서비스함

Connectionless(비연결성) & Stateless(무상태성)
HTTP는 **Connectionless 방식으로 작동하는대 서버에 연결하고, 요청해서 응답을 받으면 연결을 끊어버리는 방식**
기본적으로 자원 하나에 대해서 하나의 연결을 만든다
- 장점
  불특정 다수를 대상으로 하는 서비스 방식에 적합함
  수십만명이 웹 서비스를 사용하더라도 접속유지는 최소한으로 할 수 있기 때문에, 더 많은 유저의 요청 처리가능
- 단점
  연결을 끊어버리기 때문에 클라이언트의 이전 상태를 알 수 없음
  이런 HTTP 특징을 Stateless 라고 하는대, **Connectionless** 로 부터 파생된 특징이다
  클라이언트가 과거에 로그인 성공해도 로그정보 유지할 수 없는 특징이기 때문에 HTTP는 Cookie를 사용하여 이 문제를 해결한다

Tip. Cookie는 클라이언트와 서버의 상태 정보를 담고 있는 정보 조각
Ex. 클라이언트가 로그인에 성공하면, 서버는 로그인 정보를 자신의 DB에 저장하고

동일한 값을 Cookie형태로 클라이언트에 보낸다

URI(Uniform Resource Identifiers)
클라이언트 소프트웨어는 URI를 이용하여 자원의 위치를 찾음
URI는 HTTP와는 독립된 다른 체계이다.
HTTP는 전송 프로토콜이고, URI는 자원의 위치를 알려주기 위한 프로토콜이다
World Wide Web 상에서 접근하고자 하는 자원의 위치를 나타내기 위해서 사용
https://comic.naver.com/index 등이 URI의 예

https : 자원에 접근하기 위해 https 프로토콜을 사용
comic.naver.com : 자원의 인터넷 상에서의 위치로 도메인은 ip주소로 변환되고 ip주소로 서버위치를 찾는다
Index : 요청할 자원의 이름

위 내용을 합쳐보면 https 프로토콜을 사용해 도메인(ip주소)으로 서버위치를 찾아 index 를 요청

Method(메서드)
메서드는 서버에게 요청의 종류를 알려주기 위해 사용

메서드 종류
GET : 정보를 요청하기 위해서 사용(SELECT)
POST : 정보를 넣어주기 위해 사용(INSERT)
PUT : 정보를 업데이트 하기위해 사용(UPDATE)
DELETE : 정보를 삭제하기 위해 사용(DELETE)

HEAD : 해더 정보만 요청. 해당 자원이 존재하는지 혹은 서버에 문제가 없는지 확인하기 위해 사용
OPTIONS : 웹서버가 지원하는 메서드의 종류를 요청
TRACE : 클라이언트의 요청을 그대로 반환 ex) echo 서비스로 서버 상태를 확인하기 위한 목적으로 주로 사용

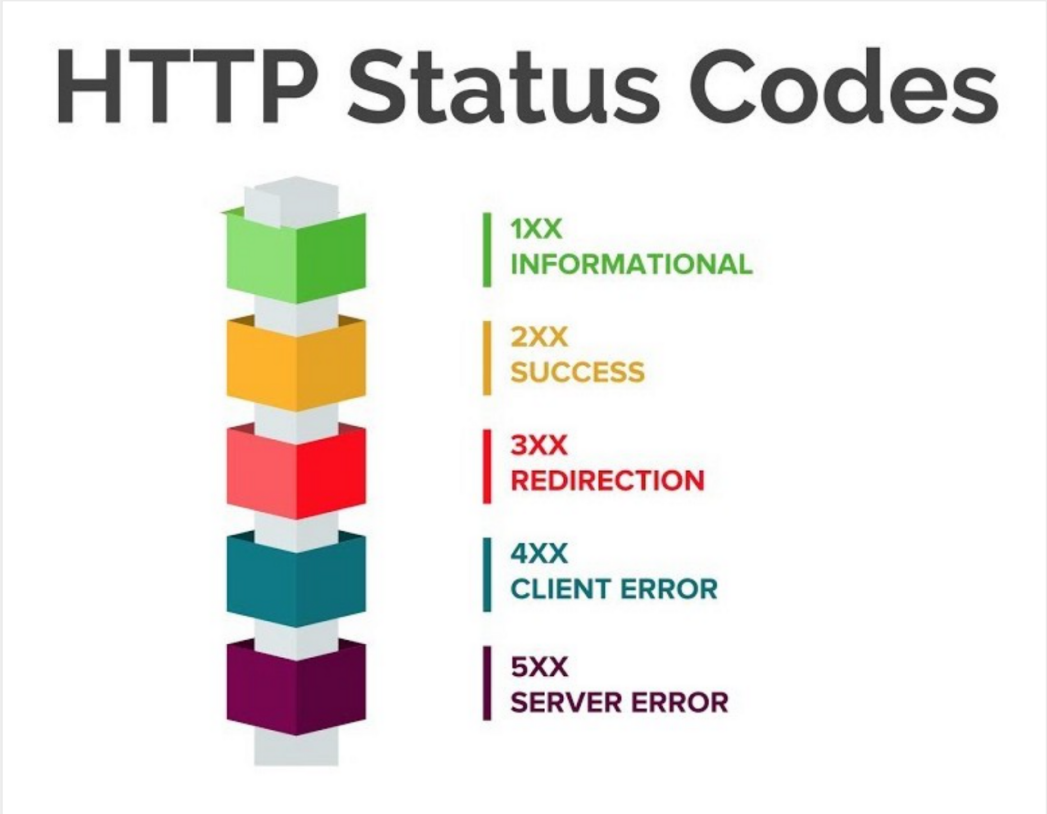
각 용도에 맞는 메서드들이 있지만 GET과 POST 만으로도 모든 종류 요청 표현가능

메서드를 용도에 맞게 사용하지 않아도 웹 서비스 개발에는 큰 문제는 없지만
Restful API 서버의 경우 GET, POST, DELETE, PUT을 명시적으로 구별하여 사용한다

응답헤더 포맷
프로토콜과 응답코드 : (HTTP/1.1 200 OK)
날짜 : (Date: Sun, 12 Aug 2018 11:30:00 GMT)
서버 프로그램및 스크립트 정보 : (Apache/2.2.4 (Unix) PHP/5.2.0)
응답헤더에는 다양한 정보를 추가할 수가 있다.
컨텐츠의 마지막 수정일
캐쉬 제어 방식.
컨텐츠 길이.

Keep Alive기능 설정

응답코드



1XX (정보) : 요청을 받았으며 프로세스를 처리중이다.
2XX (성공) : 요청을 성공적으로 받아 정상 처리했다.
3XX (리다이렉션) : 요청 완료를 위해 추가 작업 조치가 필요하다.
4XX (클라이언트 오류) : 요청의 문법이 잘못되었거나 요청을 처리할 수 없다.
5XX (서버 오류) : 서버가 클라이언트의 요청에 대한 응답에 실패했다.

Keep Alive
HTTP 1.1 부터 지원하는 기능

HTTP는 하나의 연결에 하나의 요청을 기준으로 설계됨
하지만 요즘 웹 서비스는 간단한 페이지라도 수십개의 데이터가 있는대 원래 HTTP규격대로라면
웹페이지 하나당 연결을 맺고 끊는 과정을 수십번 반복해야 한다.
이렇게 연결을 계속 맺고 끊는 상황은 매우 비효율적일 수 밖에 없는데 연결을 맺고 끊는 것은
TCP 통신과정에서 가장 많은 비용이 소모 되는 작업이기 때문이다
예를 들어 Keep alive를 지원 하지 않는 상황에 과정을 살펴보자

1.
웹 서버에 연결한다.
HTML 문서를 다운로드 한다.
웹 서버 연결을 끊는다.

2.
HTML 문서의 image, css, javascript 링크들을 읽어서 다운로드해야할 경로를 저장한다.
웹 서버에 연결한다.
첫번째 이미지를 다운로드
연결을 끊는다.

3.
웹 서버에 연결한다.
두번째 이미지를 다운로드
연결을 끊는다.

모든 링크를 다운로드 할 때까지 반복한다

이제 Keep alive를 지원할 경우를 살펴보면 아래처럼 지정된 시간동안 끊지않고 요청을 계속보낼 수 있다

1.
웹 서버에 연결한다.
HTML 문서를 다운로드 한다.
Image, css, javascript 들을 다운로드 한다.
모든 문서를 다운로드 받았다면 연결을 끊는다.

Connection: Keep-Alive
Keep-Alive:timeout=5, max=200

아래 내용은 하나의 연결당 5초동안 유지하고 Keep alive 연결을 최대 200개 까지 하용한다는 내용인대
Keep alive는 하나의 연결로 여러 요청을 처리하기 때문에 효율적이지만
그만큼 연결이 길어져 동시간대 연결이 늘어난다 하지만 운영체제에 있어 연결은 “유한한 자원”
이기 때문에 연결을 다 써버리면, 서버는 더 이상 연결을 받을 수 없게 된다.
따라서 Max값을 이용해 연결 제한을 설정하게된다

Restful API

Rest(Representational State Transfer) **API란?**
REST API(RESTful API)란 REST 아키텍처의 제약 조건을 준수하는 API
RESTful API를 통해 요청이 수행될 때 RESTful API는 리소스 상태에 대한 표현을 요청자에게 전송함
이 정보 또는 표현은 HTTP: JSON, HTML, XLT 또는 일반 텍스트를 통해 몇 가지 형식으로 전송됨
JSON은 사용 언어와 상관이 없을 뿐 아니라 인간과 머신이 모두 읽을 수 있기 때문에 가장 널리 사용됨

API가 RESTful로 간주되려면

- 클라이언트, 서버 및 리소스로 구성되며 요청이 HTTP를 통해 관리되는 클라이언트-서버 아키텍처
- **Stateless** 클라이언트-서버 커뮤니케이션: 요청 간에 클라이언트 정보가 저장되지 않으며, 각 요청이 분리되어 있고 서로 연결되어 있지 않음
- 클라이언트-서버 상호 작용을 간소화하는 캐시 가능 데이터
- 정보가 표준 형식으로 전송되도록 하기 위한 구성 요소 간 통합 인터페이스.

    - 요청된 리소스가 식별 가능하며 클라이언트에 전송된 표현과 분리되어야함
    - 수신한 표현을 통해 클라이언트가 리소스를 조작할 수 있어야 함
    - 클라이언트에 반환되는 자기 기술적(self-descriptive) 메시지에 클라이언트가 정보를 어떻게 처리해야 할지 설명하는 정보가 충분히 포함되어 있어야함
    - 하이퍼미디어: 클라이언트가 리소스에 액세스한 후 하이퍼링크를 사용해 현재 수행 가능한 기타 모든 작업을 찾을 수 있어야 함
- 요청된 정보를 검색하는 데 관련된 서버(보안, 로드 밸런싱 등을 담당)의 각 유형을 클라이언트가 볼 수 없는 계층 구조로 체계화하는 계층화된 시스템.
- 코드 온디맨드(선택 사항): 요청을 받으면 서버에서 클라이언트로 실행 가능한 코드를 전송하여 클라이언트 기능을 확장할 수 있는 기능.

Rest API 사용성
이처럼 REST API는 따라야 할 기준이 있지만, 속도를 저하시키고 더 무겁게 만드는 XML 메시징, 빌트인 보안 및 트랜잭션 컴플라이언스처럼 특정 요구 사항이 있는 SOAP(Simple Object Access Protocol) 등의 규정된 프로토콜보다 사용하기 쉬움
REST는 필요에 따라 구현할 수 있는 일련의 지침으로, 이를 통해 REST API는 더 빨라지고 경량화되며 사물인터넷(IoT) 및 모바일 앱 개발에 가장 적합한 API임

CS 스터디

38번 여러 작업을 수행하는 애플리케이션

- 애플리케이션

운영체제를 플랫폼으로 삼아 작업을 수행하는 온갖 종류의 프로그램이나 소프트웨어를 총칭하는 용어
애플리케이션은 하나의 특정 과제에 집중하거나 폭넓은 기능을 처리할 수도 있고 판매되거나 무료로 배포 할 수도 있음
애플리케이션의 코드는 소유권이 강하게 보호되거나 자유롭게 사용할 수 있는 오픈소스이거나 사용에 아무런 제한이 없음
애플리케이션의 크기는 천차만별이라 한가지 기능만 수행하는 독립적 프로그램부터 워드나 포토샵처럼 여러 가지 복잡합 작업을 수행하는 대형프로그램까지 다양하다

- 간단한 애플리캐이션 예 : 날짜,시간등을 나타내는 date라는 유닉스 프로그램, 파일과 폴더를 나열하는 ls프로그램
- 복잡한 애플리케이션 예 : 사용자가 파일을 열고, 내용을 읽고 문서를 저장하는 시스템부터 텍스트가 바뀜에 따라 디스플레이를 계속 갱신하는 알고리즘 그리고 글자 크기, 글꼴, 색상 등을 조정하는 인터페이스를 지원하는 문서에 대한 다양한 기능들을 가지고있는 워드

- 브라우저 : 규모가 크고 무료이고 오픈소스로 개발되는 애플리케이션
- 브라우저의 비동기적 이벤트 : 예측할 수 없는 시점에 일정한 순서를 따르지 않고 발생하는 이벤트
  예시) 사용자가 링크를 클릭하면 브라우저는 페이지에 대한 요청을 보내는데, 브라우저는 해당 응답을 기다리고 있지 않고 사용자가 현재 페이지를 스크롤하면 즉각 반응하고 뒤로가기 버튼을 누르거나 다른 링크를 클릭하면 요청된페이지가 오는 중이라도 취소 하며 사용자가 창의 모양을 바꾸면 디스플레이를 계속 갱신해줘야함

- 브라우저는 정적인 텍스트부터 대화형 프로그램까지 많은 종류에 콘텐츠를 지원해야하는대 이중 일부를 확장프로그램에 콘텐츠 처리를 위임 할 수 있다

PDF나 동영상같은 표준 포맷을 처리하는 데에 해당 방식이 일반적

추후 추가 예정