어플리케이션 : 대개 특정 목적을 위해 작업을 수행하도록 설계된 하나 이상의 프로그램

인터넷 : 여러 통신망을 하나로 연결한다는 의미의 인터 네트워크라는 말에서 시작되었으며 이제는 전 세계 컴퓨터들을 하나로 연결하는 거대한 컴퓨터 통신망을 의미

http : 인터넷상에서 데이터를 주고 받기 위한 서버/클라이언트 모델을 따르는 프로토콜로

어떤 종류의 데이터든지 전송할 수 있도록 설계되어 있다.

Ip : 송신 호스트와 수신 호스트가 패킷 교환 네트워크에서 정보를 주고받는데 사용하는 정보의 규약

리소스 : 자원. 사용할 수 있는 어떤 항목. 웹사이트에 접속하거나 API 호출을 할 때, 서버로부터 제공받는 식별가능한 자원들을 의미

정적리소스 : 프로그램 실행시 변경되는 정보가 거의 없는 자원. 웹사이트에 고정적으로 박혀 있는 이미지, HTML 소스 등이 해당된다.

동적리소스 : 프로그램을 실행하기 위해 자원을 배당할 시 적용되는 기준에 따라 결정되는 배당 기법

웹서버 : HTTP 요청을 받아 정적인 콘텐츠를 제공하는 서버나 프로그램

웹앱서버 : 다양한 서버 내 알고리즘, 비즈니스 로직, DB 조회 등 클라이언트 요청에 따라 동적인 컨텐츠를 제공하는 서버, 프로그램

모바일 웹 : 모바일 화면에 맞게 구성한 웹

웹앱 : 모바일 웹과 비슷하지만 구동방식이 앱처럼 보이게 한 앱

아파치 : 세계에서 가장 많이 쓰이는 웹 서버 중 하나로 서버가 굉장히 다양하고 기능적인 면에서 우수. 또한 구축이 쉬움. 하지만 apache 자체만으로 무겁고 slowloris와 같은 취약점이 존재

톰캣 : 아파치 소프트웨어 재단의 어플리케이션 서버로서 자바 서블릿을 실행 시키고 jsp 코드가 포함되어 있는 웹페이지를 만들어준다.

쉽게 말해 웹서버에서 넘어온 동적인 페이지를 읽어들여 프로그래밍을 실행하고 그 결과를 다시 html로 재구성하여 아파치에게 되돌려준다.

객체 : 넓은 의미로는 실세계에 존재하거나 생각할 수 있는 것

프로그래밍에서의 객체는 속성과 기능을 가지는 프로그램 단위

oop : object oriented programming 의 줄임말

인간 중심적 프로그래밍 패러다임(견해, 사고법)으로 현실 세계를 프로그래밍으로 옮겨와 프로그래밍 하는 것을 말한다.

프로그래밍에서 필요한 데이터를 추상화시켜서 상태와 행위를 가진 객체로 만들고 그 객체간의 상호작용을 통해 로직을 구성하는 방법을 의미

자료형 : 값의 종류에 값이 저장될 공간의 킉와 저장 형식을 정의한 것

기본형 : 실제 값을 저장 (boolean, char, byte, short, int, long, float, double 8가지 존재)

참조형 : 어떤 값이 저장되어 있는 메모리 주소를 값으로 저장

(기본형이 아닌 자료형은 모두 참조형)

데이터 타입 기본형과 자료형

Wrapper class : 기본 타입의 데이터를 객체로 포장해주는 클래스

(Boolean, Character, Byte, Short, Integer, Long, Float, Double)

패키지 : 자바에서의 패키지란 클래스들의 모음집

패키지를 통해 라이브러리끼리 구분이 가능. 사용하는 이유는 클래스의 고유성을 보장하기 위함

URL : Uniform Resource Locator의 약자로 인터넷의 리소스를 가리키는 표준이름

웹 상의 소스가 어디 있고 어떻게 접근할 수 있는지 알려주는 역할

http : HyperText Transfer Protocol의 약자. 하이퍼텍스트(HTML) 문서를 교환하기 위해 만들어진 Protool. 특징으로는 Stateless. 즉 State를 저장하지 않는다. 다음 요청을 보낼때 그전에 보낸 요청/응답에 대해 알지 못한다.

http 요청, 응답 : 요청(request)와 응답(response)의 구조로 되어있다.

클라이언트가 서버에 HTTP request를 보내고 서버가 HTTP response를 돌려보낸다.

클라이언트와 서버의 모든 통신이 요청과 응답으로 이루어진다

Get 방식 : 클라이언트가 서버로 데이터를 요청하기 위해 사용되는 Method이며 Get요청을 할 때는 Body 부분은 비어있고 헤더에 Body의 콘텐츠 타입을 명시하는 Content-Type 헤더 필드도 적지 않는다. URL 뒤에 쿼리스트링을 붙이고 HTTTP 패킷의 헤더에 포함해서 서버에 데이터 요청 URL의 ? 뒤에서부터 쿼리스트링으로 데이터를 표현하기 때문에 중요한 값들이 노출되는 문제가 있고 브라우저마다 글자 수 제한이 있기에 게시판의 게시물, 목록 조회와 같은 간단한 데이터 요청할 때 적합하다. 장점으로는 캐싱이 가능하므로 속도가 빠르다

• **URL에 쿼리 스트링(데이터)을 붙여서 서버에 전송**

• **데이터를 Header(헤더)에 포함하여 전송**

• **URL에 정보들이 그대로 노출되기 때문에 POST 방식보다 상대적으로 보안에 취약**

• **캐싱이 가능**

• **POST 방식보다 상대적으로 전송 속도가 빠름**

• **전송하는 데이터양에 한계가 있음 (브라우저마다 GET 요청 길이 제한 존재)**

• **브라우저 히스토리에 기록이 남음**

Post 방식 : GET방식 과는 다르게 URL에 표시되지 않고 HTTP 패킷 Body에 담겨 서버로 데이터가 전송. 따라서 보내는 데이터 양에 제한이 없어 대용량 데이터 전송에 적합. URL에 데이터가 표시되지 않아 상대적으론 보안적이지만 body의 데이터도 크롬 개발자 도구 등으로 확인이 가능하기에 암호화가 필요하다. 보통 서버의 값이나 상태를 변경하기 위해 사용되므로 글쓰기, 글 수정과 같은 작업에 적합

• **데이터들을 URL뒤에 붙여서 서버로 보내는 것이 아닌 Body에 담아서 전송**

• **요청 헤더의 Content-Type에 콘텐츠 타입을 명시**

• **데이터들이 URL에 노출되지 않기 때문에 GET 방식보다 상대적으로 보안적**

• **데이터들을 Body에 담기 때문에 서버로 보내는 데이터의 양은 제한 없음**

• **URL에 데이터가 노출되지 않으므로 캐싱 불가**

• **클라이언트에서 인코딩, 서버에서 디코딩**

• **요청받는 시간제한 존재**

• **브라우저 히스토리에 기록이 남지 않음**