1. Static Pages와 Dynamic Pages
2. Static Pages

* Web Server는 파일 경로 이름을 받아 경로와 일치하는 file contents를 반환한다.
* 항상 동일한 페이지를 반환한다.
* Ex) image, html, css, javascript 파일과 같이 컴퓨터에 저장되어 있는 파일들

1. Dynamic Pages

* 인자의 내용에 맞게 동적인 contents를 반환한다.
* 즉, 웹 서버에 의해서 실행되는 프로그램을 통해서 만들어진 결과물 \* Servlet: WAS 위에서 돌아가는 Java Program
* 개발자는 Servlet에 doGet()을 구현한다

1. Web Sever와 WAS의 차이
2. Web Sever의 개념

* 소프트웨어와 하드웨어로 구분한다.
* 하드웨어
* Web서버가 설치되어 있는 컴퓨터
* 소프트웨어
* 웹 브라우저 클라이언트로부터 HTTP 요청을 받아 정적인 컨텐츠(.html .jpeg .css .png)를 제공하는 컴퓨터 프로그램

1. Web Sever의 기능

* HTTP 프로토콜을 기반으로 하여 클라이언트(웹 브라우저 또는 웹 크롤러)의 요청을 서비스하는 기능을 담당한다,
* 요청에 따라 아래의 두가지 기능 중 적절하게 선택하여 수행한다.
* 기능1
  + 정적인 컨텐츠 제공
  + WAS를 거치지 않고 바로 자원을 제공한다.
* 기능2
  + 동적인 컨텐츠 제공을 위한 요청 전달
  + 클라이언트의 요청(Request)을 WAS에 보내고, WAS가 처리한 결과를 클라이언트에게 전달(응답, Response)한다.
  + 클라이언트는 일반적으로 웹 브라우저를 의미한다.
* Web Server의 예
* Ex) Apache Server, Nginx, IIS(Windows 전용 Web 서버) 등

1. WAS(Web Application Server)
2. WAS의 개념

* DB 조회나 다양한 로직 처리를 요구하는 동적인 컨텐츠를 제공하기 위해 만들어진 Application Server
* HTTP를 통해 컴퓨터나 장치에 애플리케이션을 수행해주는 미들웨어(소프트웨어 엔진)이다.
* “웹 컨테이너(Web Container)” 혹은 “서블릿 컨테이너(Servlet Container)”라고도 불린다.
  + Container란 JSP, Servlet을 실행시킬 수 있는 소프트웨어를 말한다.
  + 즉, WAS는 JSP, Servlet 구동 환경을 제공한다.

1. WAS의 역할

* WAS = Web Server + Web Container
* Web Server 기능들을 구조적으로 분리하여 처리하고자하는 목적으로 제시되었다.
  + 분산 트랜잭션, 보안, 메시징, 쓰레드 처리 등의 기능을 처리하는 분산 환경에서 사용된다.
  + 주로 DB 서버와 같이 수행된다.
* 현재는 WAS가 가지고 있는 Web Server도 정적인 컨텐츠를 처리하는 데 있어서 성능상 큰 차이가 없다.

1. WAS의 주요 기능

* 프로그램 실행 환경과 DB 접속 기능 제공
* 여러 개의 트랜잭션(논리적인 작업 단위) 관리 기능
* 업무를 처리하는 비즈니스 로직 수행

1. WAS의 예

* Ex) Tomcat, JBoss, Jeus, Web Sphere등

1. Web Server와 WAS를 구분하는 이유
2. Web Server가 필요한 이유?

* 클라이언트(웹 브라우저)에 이미지 파일(정적 컨텐츠)을 보내는 과정을 생각해보자.
  + 이미지 파일과 같은 정적인 파일들은 웹 문서(HTML 문서)가 클라이언트로 보내질 때 함께 가는 것이 아니다.
  + 클라이언트는 HTML 문서를 먼저 받고 그에 맞게 필요한 이미지 파일들을 다시 서버로 요청하면 그때서야 이미지 파일을 받아온다.
  + Web Server를 통해 정적인 파일들을 Application Server까지 가지 않고 앞단에서 빠르게 보내줄 수 있다.
* 따라서 Web Server에서는 정적 컨텐츠만 처리하도록 기능을 분배하여 서버의 부담을 줄일 수 있다.

1. WAS가 필요한 이유?

* 웹 페이지는 정적 컨텐츠와 동적 컨텐츠가 모두 존재한다.
  + 사용자의 요청에 맞게 적절한 동적 컨텐츠를 만들어서 제공해야 한다.
  + 이때, Web Server만을 이용한다면 사용자가 원하는 요청에 대한 결과값을 모두 미리 만들어 놓고 서비스를 해야 한다.
  + 하지만 이렇게 수행하기에는 자원이 절대적으로 부족하다.
* 따라서 WAS를 통해 요청에 맞는 데이터를 DB에서 가져와서 비즈니스 로직에 맞게 그때 그때 결과를 만들어서 제공함으로써 자원을 효율적으로 사용할 수 있다.

1. 그렇다면 WAS가 Web Server의 기능도 모두 수행하면 되지 않을까?
2. 기능을 분리하여 서버 부하 방지
   * WAS는 DB 조회나 다양한 로직을 처리하느라 바쁘기 때문에 단순한 정적 컨텐츠는 Web Server에서 빠르게 클라이언트에 제공하는 것이 좋다.
   * WAS는 기본적으로 동적 컨텐츠를 제공하기 위해 존재하는 서버이다.
   * 만약 정적 컨텐츠 요청까지 WAS가 처리한다면 정적 데이터 처리로 인해 부하가 커지게 되고, 동적 컨텐츠의 처리가 지연됨에 따라 수행 속도가 느려진다.
   * 즉, 이로 인해 페이지 노출 시간이 늘어나게 될 것이다.
3. 물리적으로 분리하여 보안 강화

* SSL에 대한 암복호화 처리에 Web Server를 사용
* 웹 서버 없이 WAS에 직접 접근할 경우 WAS의 port number가 노출되어 보안 상 좋지 않다

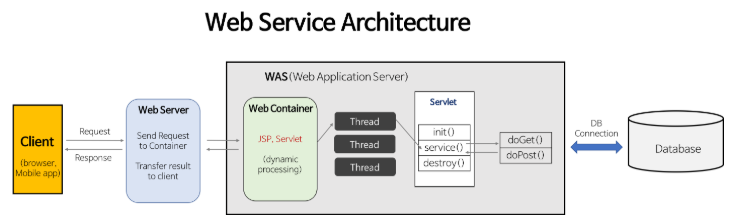
1. 여러 대의 WAS를 연결 가능
   * Load Balancing을 위해서 Web Server를 사용
   * fail over(장애 극복), fail back 처리에 유리
   * 특히 대용량 웹 어플리케이션의 경우(여러 개의 서버 사용) Web Server와 WAS를 분리하여 무중단 운영을 위한 장애 극복에 쉽게 대응할 수 있다.
   * 예를 들어, 앞 단의 Web Server에서 오류가 발생한 WAS를 이용하지 못하도록 한 후 WAS를 재시작함으로써 사용자는 오류를 느끼지 못하고 이용할 수 있다.
2. 여러 웹 어플리케이션 서비스 가능

* 예를 들어, 하나의 서버에서 PHP Application과 Java Application을 함께 사용하는 경우

1. 기타

* 접근 허용 IP 관리, 2대 이상의 서버에서의 세션 관리 등도 Web Server에서 처리하면 효율적이다.

1. 즉, 자원 이용의 효율성 및 장애 극복, 배포 및 유지보수의 편의성을 위해 Web Server와 WAS를 분리한다.
2. Web Server를 WAS 앞에 두고 필요한 WAS들을 Web Server에 플러그인 형태로 설정하면 더욱 효율적인 분산 처리가 가능하다.
3. Web Service Architecture
4. 다양한 구조를 가질 수 있다.
   * Client -> Web Server -> DB
   * Client -> WAS -> DB
   * Client -> Web Server -> WAS -> DB



3번 구조의 동작과정

* + - 1. Web Server는 웹 브라우저 클라이언트로부터 HTTP 요청을 받는다.
      2. Web Server는 클라이언트의 요청(Request)을 WAS에 보낸다.
      3. WAS는 관련된 Servlet을 메모리에 올린다.
      4. WAS는 web.xml을 참조하여 해당 Servlet에 대한 Thread를 생성한다. (Thread Pool 이용)
      5. HttpServletRequest와 HttpServletResponse 객체를 생성하여 Servlet에 전달한다.

5-1. Thread는 Servlet의 service() 메서드를 호출한다.

5-2. service() 메서드는 요청에 맞게 doGet() 또는 doPost() 메서드를 호출한다.

protected doGet(HttpServletRequest request, HttpServletResponse response)

* + - 1. doGet() 또는 doPost() 메서드는 인자에 맞게 생성된 적절한 동적 페이지를 Response 객체에 담아 WAS에 전달한다.
      2. WAS는 Response 객체를 HttpResponse 형태로 바꾸어 Web Server에 전달한다.
      3. 생성된 Thread를 종료하고, HttpServletRequest와 HttpServletResponse 객체를 제거한다

1. 서버 만들 때 필요한 툴

* 웹서버는 '컴퓨터 + 랜선 + 고정 IP'가 기본
* 컴퓨터: 서버운용을 위한 서버용 컴퓨터, 일반 고객이 사용하는 컴퓨터보다 전력 사용량이 낮으며 더 오래 사용 가능하도록 설계되어 있음
* 고정 IP: 인터넷과 연결하는 네트워크
* 서버 운용을 위해 보통 서버용 컴퓨터가 따로 나온다. 일반 고객이 사용하는 컴퓨터보다 전력 사용량이 낮으며 더 오래 사용 가능하도록 설계되어 있습니다.

1. http server: 고객의 자료 요청과 웹 자료 전달의 처리 등 전반적인 서비스 담당

* Ex) 아파치(apache)

1. Database: 고객이 전달한 자료를 저장해줄 프로그램,

* Ex) mysql, oracle

1. 서버 사이드 스크립트 언어(server-side script): 고객이 요청한 자료를 서버에서 가공하여 넘겨줄 때 가용하는 프로그램

* Ex) php, jsp

⇒ 즉, 웹서버는 '컴퓨터 + 랜선 + 고정 IP'를 기본으로  
컴퓨터 내부에서는 'http server + database + server-side script' (예시로 아파치 + mysql + php) 프로그램으로 동작

1. 문서 전송 프로토콜
2. HTTP(Hyper Text Transfer Protocol)

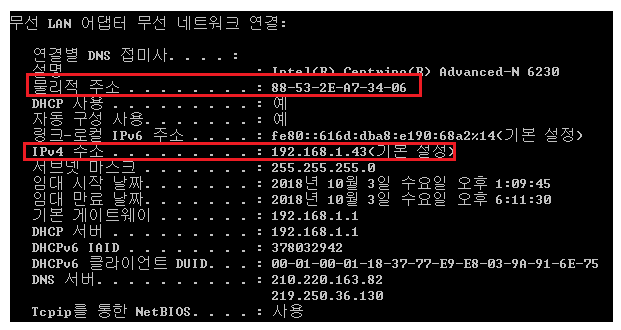
* Hyper Text를 전송하기 위한 프로토콜
* Hyper Text: 웹 문서를 구성하고 있는 언어인 HTML
* 기본적으로 '요청(Request)'과 '응답(Response)'으로 이루어짐
* HTTP 통신은 기본적으로 연결이 되어 있지 않음.

⇒클라이언트가 서버에 요청을 보내고 응답을 받으면, 그것으로 통신이 종료된다는 것을 의미함. 서버는 클라이언트가 웹 사이트에 접속해 있는지, 알 수 없음.

* 요청을 보내고 응답을 받을 때 패킷(Packet)을 이용

1. 추가적인 프로토콜
2. IP / MAC / ARP

* IP( Internet Protocol )는 각각의 패킷을 IP 주소와 MAC 주소를 통해 상대방에게 전달하는 역할을 합니다.
* IP 주소
* 각 노드에 부여된 주소를 가르킨다.
* MAC 주소
* 각 네트워크 카드( NIC )에 할당된 고유의 주소를 말한다.
* ARP 프로토콜을 이용해서 IP 주소를 MAC 주소로 변환하여 목적지를 찾아간다.
* IP주소는 유동적이기 때문에 고유의 대상이 될 수 없지만, MAC 주소는 고유하기 때문에 ARP 프로토콜을 사용한다.



1. TCP / UDP

* TCP( Transmission Control Protocol ) 전송 제어 프로토콜
* 데이터를 안정적으로 전달
* 데이터를 순서대로 전달
* 데이터를 에러없이 교환할 수 있도록 전달
* HTTP는 자신의 메시지를 전달하기 위해 TCP 프로토콜을 이용
* 클라이언트와 서버는 TCP 커넥션이 맺어지면 메시지가 절대 사라지거나 손상되지 않고, 순서가 뒤바뀌는 일도 없는 안정성 있는 프로토콜
* UDP ( User Datagram Protocol )
* 상대방이 데이터를 잘 받았든 말든 상관하지 않는다.

⇒ 데이터 전달에 대한 안정성을 보장하지 않습니다.

* UDP가 TCP에 비해 속도가 빠르다.

⇒ 서비스에 따라 사용하는 프로토콜이 다릅니다.

* IP 프로토콜과 함께 TCP / IP 라는 명칭으로 사용되며, TCP/IP는 OSI 7 계층을 4계층으로 단순화 한 모델입니다.

1. DNS (Domain Name Server

* DNS는 도메인 주소를 IP 주소로 변환해주는 역할을 하기 때문에, 사용자는 도메인 이름을 입력하면 원하는 사이트에 접근할 수 있다.

<https://velog.io/@ikswary/%EC%9B%B9%EC%84%9C%EB%B2%84-%EA%B5%AC%EC%A1%B0>

<https://developer.mozilla.org/ko/docs/Learn/Common_questions/What_is_a_web_server>

<https://gmlwjd9405.github.io/2018/10/27/webserver-vs-was.html>

<https://victorydntmd.tistory.com/285>

<https://cordelia273.space/11>