Web Server와 WAS

들어가기 전 - Static Pages와 Dynamic Pages

Static Pages

- Web Server는 파일 경로 이름을 받아 경로와 일치하는 file contnets를 반환.
- 항상 동일한 페이지를 반환 하는 게 특징.
- ~~~.html과 같이 파일을 직접 여는 것으로 이해하면 쉽다.

Dynamic Pages

- 요청 인자에 맞게 동적인 Contents를 반환.
- 웹 서버에 의해서 실행되는 프로그램을 통해 만들어진 결과물
- 이때 프로그램의 예시로 서블릿이 있다.
- Servlet은 WAS 위에서 돌아가는 자바 프로그램.
    - 개발자는 Servlet에 doGet()메소드를 구현한다.

Web Server와 WAS의 차이

Web Server

- Web Server의 개념
    - **소프트웨어와 파드웨어로 구분된다.**
    - 1) 하드웨어
        - Web 서버가 설치되어 있는 컴퓨터
    - 2) 소프트웨어
        - 웹 브라우저 클라이언트로부터 **HTTP요청(ex. GET/path/index.html)**을 받아 정적인 컨텐츠(~~.html 등)를 제공하는 컴퓨터 프로그램
- Web Server의 기능

    - HTTP 프로토콜을 기반으로 하여 클라이언트(웹 브라우저 또는 웹 크롤러)의 요청을 서비스 하는 기능을 담당
    - 요청에 따라 아래의 두 가지 기능 중 적절하게 선택하여 수행.
    - 기능 1)
        - 정적인 컨텐츠 제공
        - WAS를 거치지 않고 바로 자원을 제공.
    - 기능 2)
        - 동적인 컨텐츠 제공을 위한 요청 전달.
        - 클라이언트의 요청을 WAS에 보내고, WAS가 처리한 결과를 클라이언트에게 전달(응답)
- Web Server의 예
    - Apache Server, Nginx, IIS 등

WAS(Web Application Server)

- WAS의 개념
    - DB 조회나 다양한 로직 처리를 요구하는 **동적인 컨텐츠를 제공하기 위해 만들어진 Application Server**
    - HTTP를 통해 컴퓨터나 장치에 애플리케이션을 수행해주는 미들웨어(소프트웨어 엔진)
    - **"웹 컨터에너(Web Container)" 혹은 "서블릿 컨테이너(Servlet Container)"**라고도 불림.
        - Container란 JSP, Servlet을 실행시킬 수 있는 소프트웨어.
        - 즉, WAS는 JSP, Servlet 구동 환경을 제공함을 의미.
- WAS의 역할
    - **WAS = Web Server + Web Container**
    - Web Server 기능들을 구조적으로 분리하여 처리하고자 하는 목적으로 제시 됨.
        - 분산 트랜젝션, 보안, 메시징, 쓰레드 처리 등의 기능을 처리하는 분산 환경에 사용됨.
        - 주로 DB서버와 같이 수행 됨.

- 현재는 WAS가 가지고 있는 Web Server도 정적인 컨텐츠를 처리하는 데 있어서 성능상 큰 차이가 없음.
- WAS의 주요 기능
  - 프로그램 실행 환경과 DB접속 기능 제공
  - 여러 개의 트랜잭션(논리적인 작업 단위) 관리 기능
  - 업무를 처리하는 비즈니스 로직 수행
- WAS의 예
  - Tomcat, JBoss, Jeus, Web Sphere 등

Web Server와 WAS를 구분하는 이유

Web Server가 필요한 이유

▼ 웹 서버에서는 정적 컨텐츠만 처리하도록 기능 분배를 해서 서버 부담을 줄이는 것

- 클라이언트가 이미지 파일(정적 컨텐츠)를 보낼 때
- 웹 문서(html 문서)가 클라이언트로 보내질 때, 이미지 파일과 같은 정적 파일은 함께 보내지지 않음.
- 먼저 html 문서를 받고, 이에 필요한 이미지 파일들을 다시 서버로 요청해서 받아옴.
- 따라서, 웹 서버를 통해서 정적인 파일을 어플리케이션 서버까지 가지 않고 앞단에 빠르게 보낼 수 있다는 장점이 있음.

WAS가 필요한 이유

▼ WAS를 통해 요청에 맞는 데이터를 DB에서 가져와 비즈니스 로직에 맞게 그때마다 결과를 만들고 제공하면서 자원을 효율적으로 사용할 수 있음.

- 동적인 컨텐츠를 제공해야 할 때
- 웹 서버만으로는 사용자가 원하는 요청에 대한 결과값을 모두 미리 만들어 놓고 서비스 하기에는 자원이 절대적으로 부족.
- 즉, WAS를 통해 요청이 들어올 때마다 DB와 비즈니스 로직을 통해 결과물을 만들어 제공.

WAS로 Web Server역할까지 처리하지 않는 이유

- 기능을 분리하여 서버 부하 방지.
    - WAS는 기본적으로 동적 컨텐츠를 제공하기 위해 존재하는 서버이다.
    - WAS는 DB조회나 다양한 로직을 처리하느라 바쁘기 때문에 단순한 정적 컨텐츠는 Web Server에서 빠르게 클라이언트에 제공하는 것이 좋음.
    - 만약, 정적 컨텐츠 요청까지 WAS가 처리한다면, 정적 데이터 처리로 인해 부하가 커지고
    - 동적 컨텐츠 처리가 지연됨에 따라 수행 속도가 느려질 것.
    - 즉, 이로 인해 페이지 노출 시간이 늘어나게 될 것.
- 물리적으로 분리하여 보안 강화
    - SSL에 대한 암복호화 처리에 Web Server를 사용
- 여러 대의 WAS를 연결 가능
    - Load Balancing(부하 분산)을 위해서 Web Server를 사용
    - fail over(장애 극복), fail back처리에 유리
    - 특히, 대용량 Web Application의 경우(여러 개의 서버 사용)Web Server와 WAS를 분리하여 무중단 운영을 위한 장애 극복에 쉽게 대응할 수 있음.
    - 예로, 앞 단의 Web Server에서 오류가 발생한 WAS를 이용하지 못 하도록 한 후 WAS를 재시작함으로써 사용자는 오류를 느끼지 못 하고 이용할 수 있다.
- 여러 Web Application 서비스 가능
    - 하나의 서버에서 PHP Application과 Java Application을 함께 사용하는 경우
- 기타
    - 접근 허용 IP 관리, 2대 이상의 서버에서의 세션 관리 등도 Web Serverd에서 처리하면 효율적

즉, 자원 이용의 효율성 및 장애 극복, 배포 및 유지보수의 편의성을 위해 Web Server와 WAS를 분리.

가장 효율적인 구성(방법)

- **Web Server를 WAS 앞에 두고 필요한 WAS들을 Web Server에 플러그인 형태로 설정하면 더욱 효율적인 분산 처리가 가능.**

- 클라이언트의 요청을 먼저 웹 서버가 받은 다음, WAS에게 보내 관련된 Servlet을 메모리에 올림.
- WAS는 web.xml을 참조해 해당 Servlet에 대한 스레드를 생성(스레드 풀 이용)
- 이때, HttpServletRequest와 HttpServletResponse 객체를 생성해 Servlet에게 전달
    - 스레드는 Servlet의 service() 메소드를 호출
    - service() 메소드는 요청에 맞게 doGet()이나 doPost() 메소드를 호출
- doGet()이나 doPost() 메소드는 인자에 맞게 생성된 적절한 동적 페이지를 Response 객체에 댐아 WAS에 전달
- WAS는 Response 객체를 HttpResponse형태로 바꿔 웹 서버로 전달
- 생성된 스레드를 종료하고, HttpServletRequest와 HttpServletResponse 객체 제거