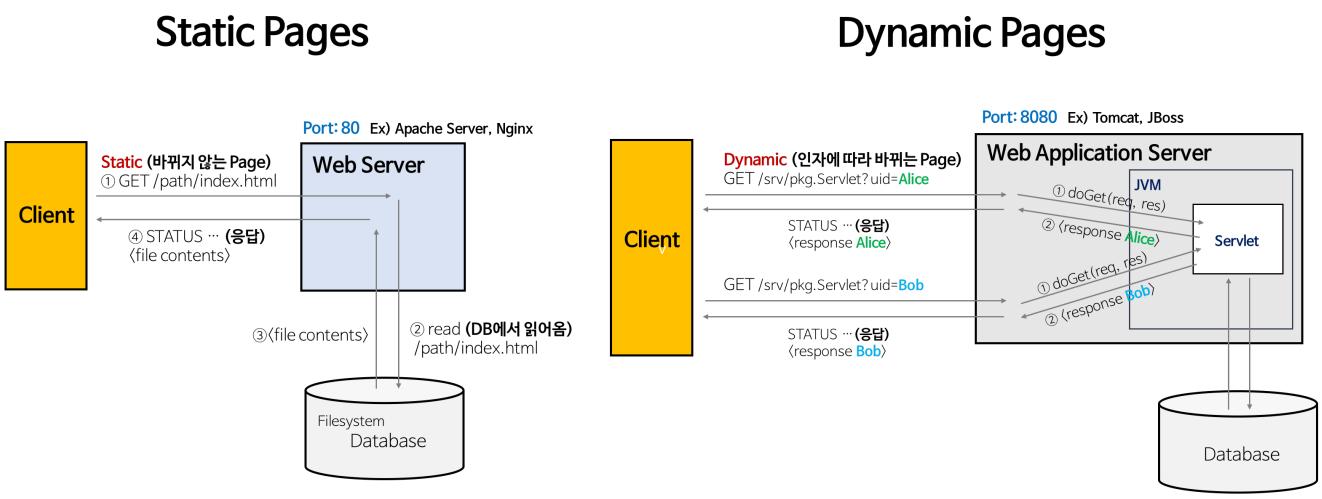
Web Server와 WAS의 차이

웹 서버와 was의 차이점은 무엇일까? 서버 개발에 있어서 기초적인 개념이다.

먼저, 정적 페이지와 동적 페이지를 알아보자



Static Pages

> 바뀌지 않는 페이지

웹 서버는 파일 경로 이름을 받고, 경로와 일치하는 file contents를 반환함

항상 동일한 페이지를 반환함

```
image, html, css, javascript 파일과 같이 컴퓨터에 저장된 파일들
```

Dynamic Pages

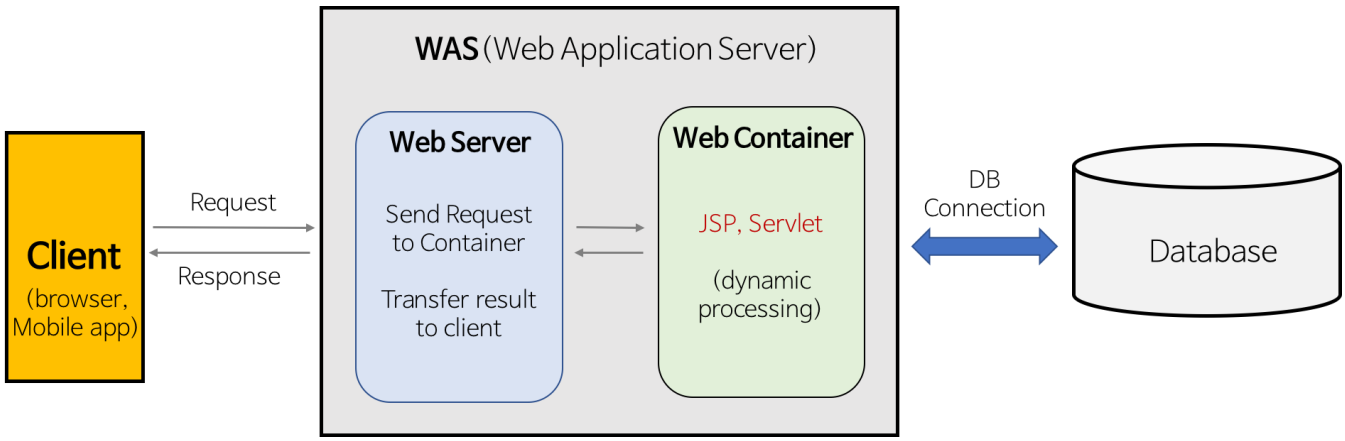
> 인자에 따라 바뀌는 페이지

인자의 내용에 맞게 동적인 contents를 반환함

웹 서버에 의해 실행되는 프로그램을 통해 만들어진 결과물임 (Servlet : was 위에서 돌아가는 자바 프로그램)

개발자는 Servlet에 doGet() 메소드를 구현함

웹 서버와 WAS의 차이



웹 서버

개념에 있어서 하드웨어와 소프트웨어로 구분된다.

하드웨어 : Web 서버가 설치되어 있는 컴퓨터

소프트웨어 : 웹 브라우저 클라이언트로부터 HTTP 요청을 받고, 정적인 컨텐츠(html, css 등)를 제공하는 컴퓨터 프로그램

웹 서버 기능

> Http 프로토콜을 기반으로, 클라이언트의 요청을 서비스하는 기능을 담당

요청에 맞게 두가지 기능 중 선택해서 제공해야 한다.

- 정적 컨텐츠 제공

  > WAS를 거치지 않고 바로 자원 제공

- 동적 컨텐츠 제공을 위한 요청 전달

  > 클라이언트 요청을 WAS에 보내고, WAS에서 처리한 결과를 클라이언트에게 전달

웹 서버 종류 : Apache, Nginx, IIS 등

WAS

Web Application Server의 약자

> DB 조회 및 다양한 로직 처리 요구시 **동적인 컨텐츠를 제공**하기 위해 만들어진 애플리케이션 서버

HTTP를 통해 애플리케이션을 수행해주는 미들웨어다.

WAS는 웹 컨테이너 혹은 서블릿 컨테이너라고도 불림

(컨테이너란 JSP, Servlet을 실행시킬 수 있는 소프트웨어. 즉, WAS는 JSP, Servlet 구동 환경을 제공해줌)

역할

WAS = 웹 서버 + 웹 컨테이너

웹 서버의 기능들을 구조적으로 분리하여 처리하는 역할

> 보안, 스레드 처리, 분산 트랜잭션 등 분산 환경에서 사용됨 (주로 DB 서버와 함께 사용)

WAS 주요 기능

1.프로그램 실행 환경 및 DB 접속 기능 제공

2.여러 트랜잭션 관리 기능

3.업무 처리하는 비즈니스 로직 수행

WAS 종류 : Tomcat, JBoss 등

그럼, 둘을 구분하는 이유는?

웹 서버가 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 받아오는 것

따라서 웹 서버를 통해서 정적인 파일을 애플리케이션 서버까지 가지 않고 앞단에 빠르게 보낼 수 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 할때..
>
> 웹 서버만으로는 사용자가 원하는 요청에 대한 결과값을 모두 미리 만들어놓고 서비스하기에는 자원이 절대적으로 부족함
>
> 따라서 WAS를 통해 요청이 들어올 때마다 DB와 비즈니스 로직을 통해 결과물을 만들어 제공!

그러면 WAS로 웹 서버 역할까지 다 처리할 수 있는거 아닌가요?

> WAS는 DB 조회, 다양한 로직을 처리하는 데 집중해야 함. 따라서 단순한 정적 컨텐츠는 웹 서버에게 맡기며 기능을 분리시켜 서버 부하를 방지하는 것
>
> 만약 WAS가 정적 컨텐츠 요청까지 처리하면, 부하가 커지고 동적 컨텐츠 처리가 지연되면서 수행 속도가 느려짐 → 페이지 노출 시간 늘어나는 문제 발생

또한, 여러 대의 WAS를 연결지어 사용이 가능하다.

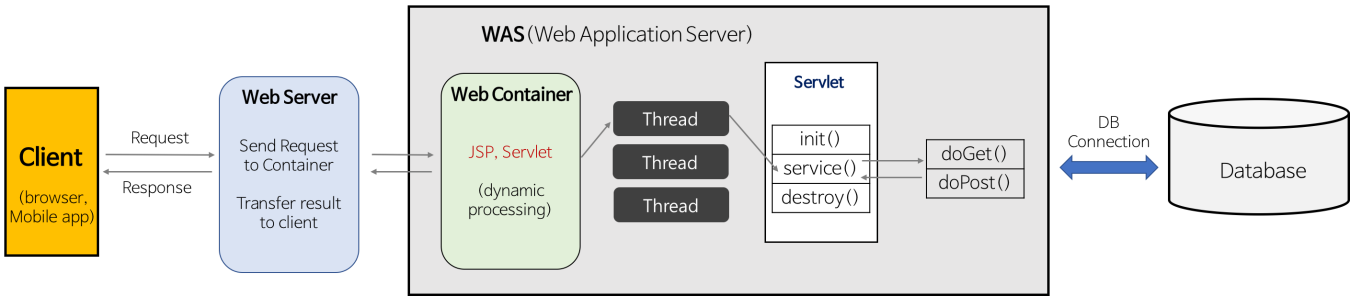
웹 서버를 앞 단에 두고, WAS에 오류가 발생하면 사용자가 이용하지 못하게 막아둔 뒤 재시작하여 해결할 수 있음 (사용자는 오류를 느끼지 못하고 이용 가능)

자원 이용의 효율성 및 장애 극복, 배포 및 유지 보수의 편의성 때문에 웹 서버와 WAS를 분리해서 사용하는 것이다.

가장 효율적인 방법

> 웹 서버를 WAS 앞에 두고, 필요한 WAS들을 웹 서버에 플러그인 형태로 설정하면 효율적인 분산 처리가 가능함

Web Service Architecture



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 담아 WAS에 전달

WAS는 Response 객체를 HttpResponse 형태로 바꿔 웹 서버로 전달

생성된 스레드 종료하고, HttpServletRequest와 HttpServletResponse 객체 제거

[참고자료] : 링크