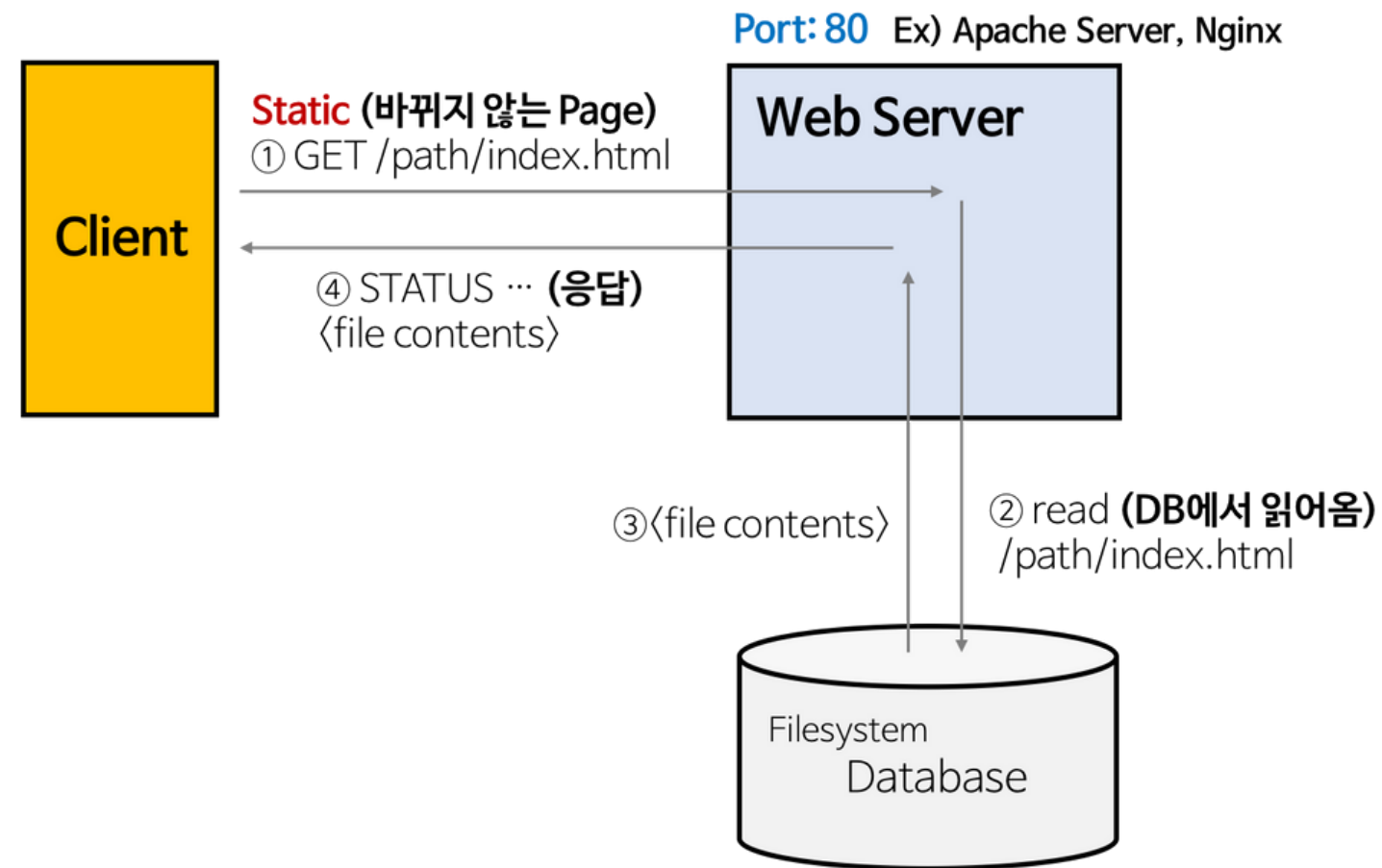


Static , Dynamic

– Static Pages 와 Dynamic Pages

Static Pages

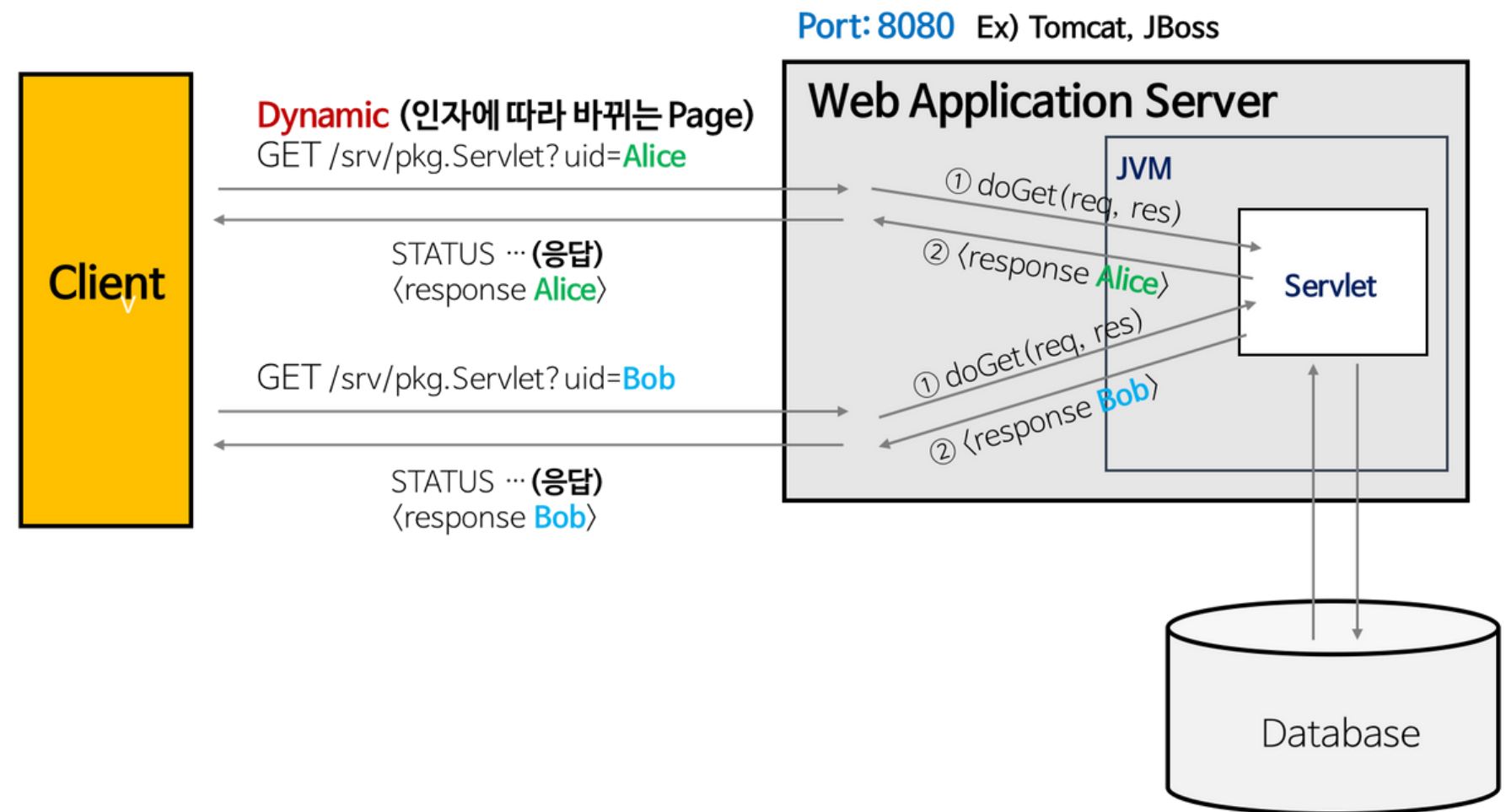


Static Pages

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

Dynamic Pages



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.

Web Server

– Static Pages 와 Dynamic Pages

Web Server

Web Server의 개념

- 1. Web 서버가 설치되어 있는 컴퓨터
- 2. 웹 브라우저 클라이언트로부터 HTTP 요청을 받아 정적인 콘텐츠(.html .jpeg .css 등)를 제공하는 컴퓨터 프로그램

Web Server의 기능

- 1. HTTP 프로토콜을 기반으로 하여 클라이언트(웹 브라우저 또는 웹 크롤러)의 요청을 서비스 하는 기능을 담당한다.
- 2. 정적인 콘텐츠 제공, WAS를 거치지 않고 바로 자원을 제공한다.
- 3. 동적인 콘텐츠 제공을 위한 요청 전달
- 4. 클라이언트의 요청(Request)을 WAS에 보내고, WAS가 처리한 결과를 클라이언트에게 전달(응답, Response)한다.
- 5. 클라이언트는 일반적으로 웹 브라우저를 의미한다.

WAS(Web Application Server)

WAS의 개념

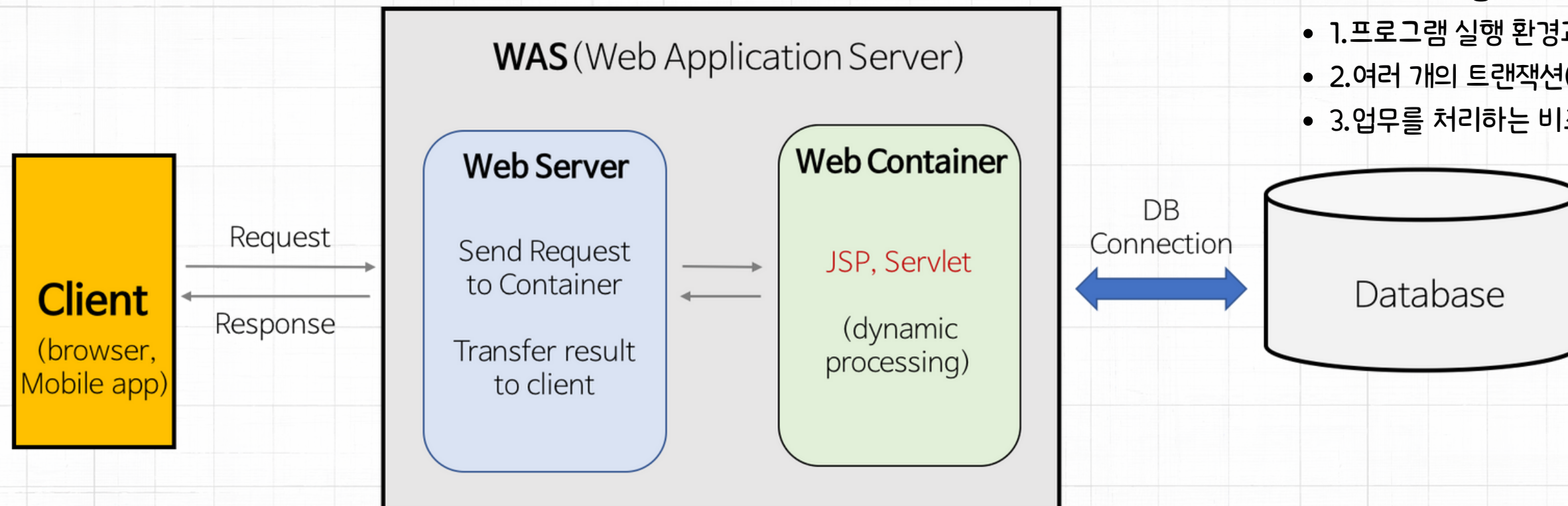
- 1. DB 조회나 다양한 로직 처리를 요구하는 동적인 콘텐츠를 제공하기 위해 만들어진 Application Server
- 2. HTTP를 통해 컴퓨터나 장치에 애플리케이션을 수행해주는 미들웨어(소프트웨어 엔진)이다.
- 3. “웹 컨테이너(Web Container)” 혹은 “서블릿 컨테이너(Servlet Container)”라고도 불린다.
- Container란 JSP, Servlet을 실행시킬 수 있는 소프트웨어를 말한다.
- 즉, WAS는 JSP, Servlet 구동 환경을 제공한다.

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 콘텐츠를 처리하는 데 있어서 성능상 큰 차이가 없다.

WAS의 주요 기능

- 1. 프로그램 실행 환경과 DB 접속 기능 제공
- 2. 여러 개의 트랜잭션(논리적인 작업 단위) 관리 기능
- 3. 업무를 처리하는 비즈니스 로직 수행



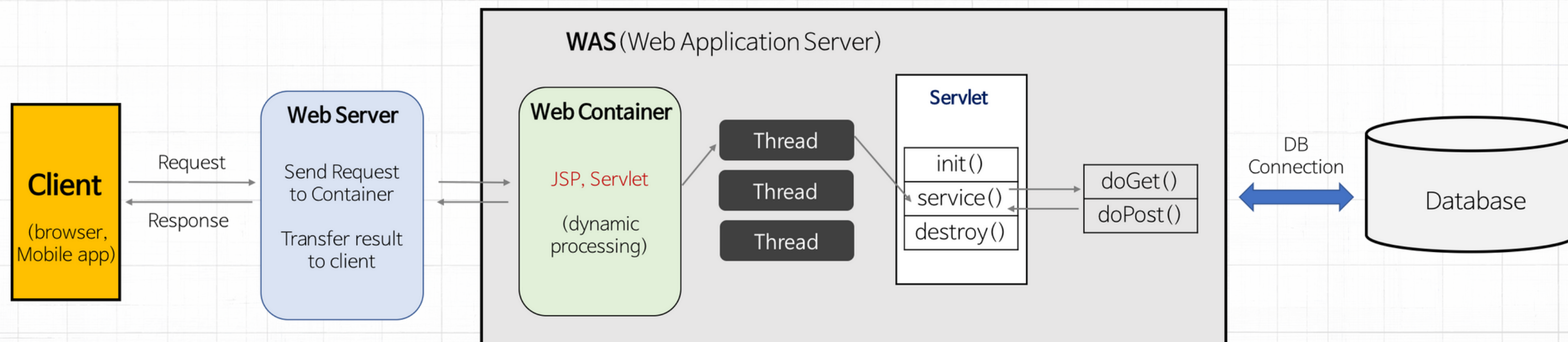
Web Server Architecture

– Client → Web Server → WAS → DB

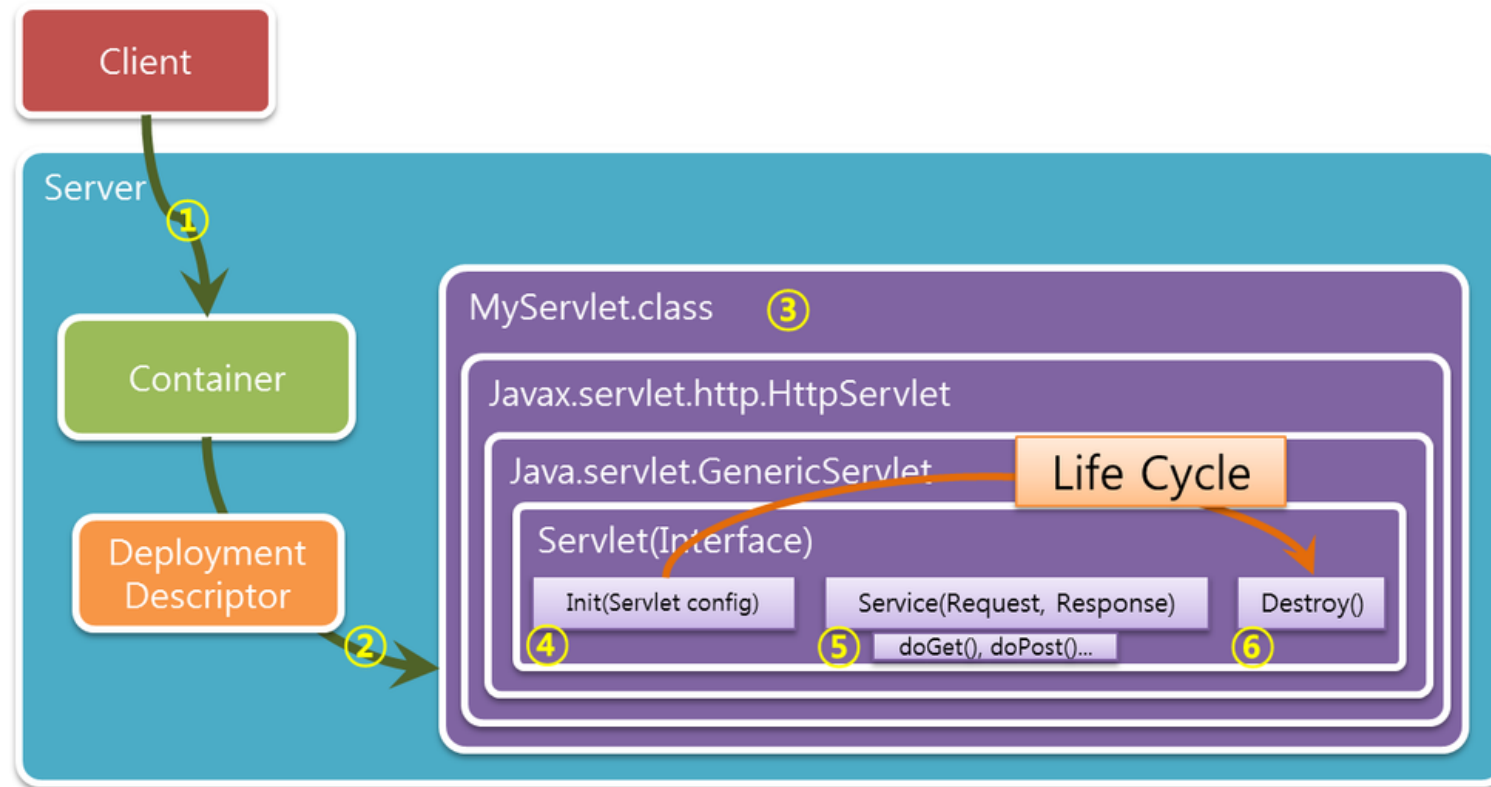
동작구조

- 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.
- 6. protected doGet(HttpServletRequest request, HttpServletResponse response)
- 7. doGet() 또는 doPost() 메서드는 인자에 맞게 생성된 적절한 동적 페이지를 Response 객체에 담아 WAS에 전달한다.
- 8. WAS는 Response 객체를 HttpServletResponse 형태로 바꾸어 Web Server에 전달한다.
- 9. 생성된 Thread를 종료하고, HttpServletRequest와 HttpServletResponse 객체를 제거한다

Web Service Architecture



Servlet Life Cycle

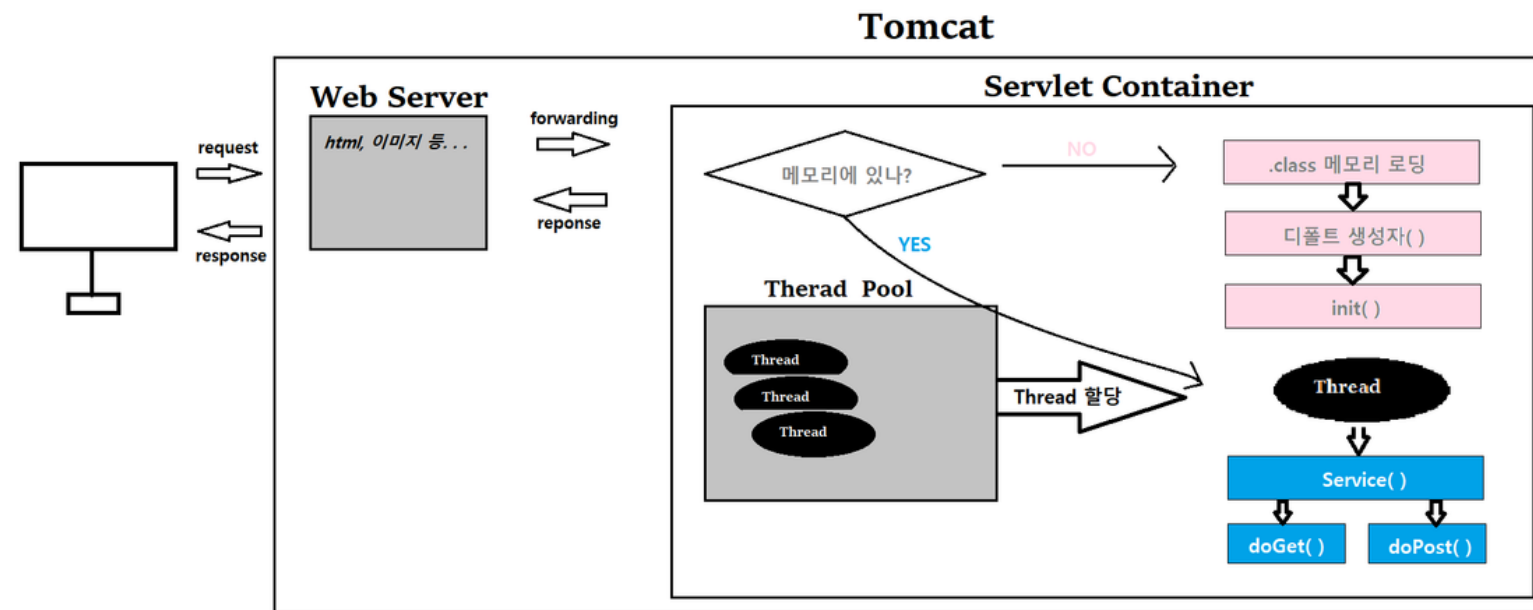
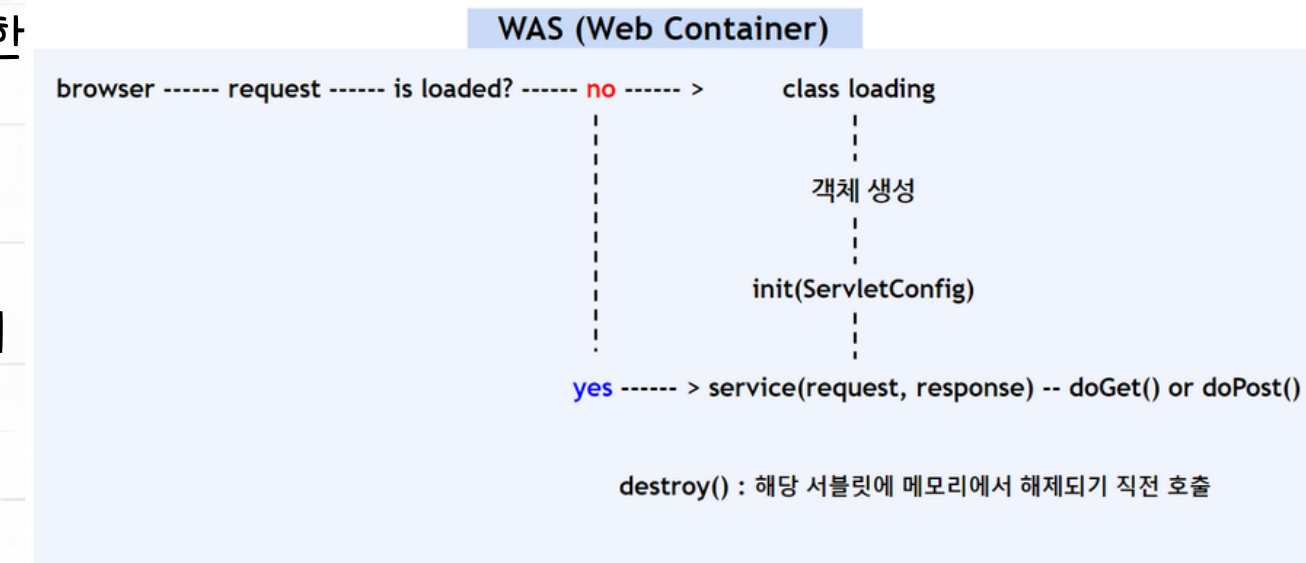


서블릿 생명주기 관련 메소드

→ `init(ServletConfig config)` : 해당 서블릿의 초기화 작업을 위해 정의, 서블릿 당 한번 실행

→ `service(request, response)`
: 해당 서블릿이 사용자에게 서비스하기 위해 정의, 클라이언트 요청시마다 호출
: `doGet()` / `doPost()`와 연결

→ `destroy()` : 해당 서블릿이 메모리에서 해제되기 전에 실행



서블릿 생명주기

① 사용자로부터 특정 페이지에 대한 요청이 들어오면 컨테이너로 요청 정보가 전해진다

② 컨테이너는 배포서술자(DD:Deployment Descriptor)의 서블릿 �핑 정보를 참조하여 해당 서블릿을 호출한다.

is loaded ? 해당 Servlet / JSP 객체가 메모리에 적재되었는지 여부를 확인한다

no → ③번으로 (최초 요청 시에만 실행 : 1번 실행)

yes → ④번으로 (요청 시마다 실행 : n번 실행)

③ 호출된 MyServlet이 로딩(class loading) → Servlet or JSP 객체가 생성 된다.

④ 서블릿이 초기화(init) 된다.

⑤ 요청에 대한 내용을 처리하고 응답해준다.

⑥ 서블릿이 소멸된다.(서비스 종료시, 서블릿 인스턴스가 메모리에서 해제되기 전 실행)