## * Servlet이란??

- 웹 서비스를 위한 자바 클래스

(자바를 이용하여 웹을 만들기 위해 필요한 기술)

- 웹 프로그래밍에서 클라이언트의 요청**(Request)**을 처리하고

그 결과를 다시 클라이언트에게 응답**(Response)**하는

Servlet클래스의 구현 규칙을 지킨 자바 프로그래밍 기술

(ex. 사용자가 로그인을 하려고 할 때 아이디와 비밀번호를 입력하고 로그인 버튼을 누르면

서버는 아이디와 비밀번호를 확인하고 다음 페이지를 띄워주는 역할 수행)

즉, **Servlet**은

자바 어플리케이션 코딩을 하듯

웹 브라우저용 출력 화면**(HTML)** 을 만드는 방법

## * 서블릿 특징

- 클라이언트의 요청에 대해 동적으로 작동하는 웹 애플리케이션 컴포넌트.

  -> 클라이언트 요청에 대한 서버 응답 시 미리 만들어둔 화면(정적)이 아닌
  요청을 받을 때 마다 알맞은 화면을 만들어(동적) 응답함.

- HTML을 사용하여 요청에 응답

- MVC Model2패턴에서 Controller로 이용

- http프로토콜 서비스를 지원하는**javax.servlet.http.HttpServlet** 클래스를 상속 받음

## * 서블릿 단점

- servlet에 작성한 html 코드 변경 시 재컴파일 해야 하는 단점이 있음

## * 서블릿 상속 관계

- 서블릿 코드를 작성할 클래스는 반드시
**javax.servlet.http.HttpServlet** 클래스를 상속 받아 메소드를 구현해야 함.

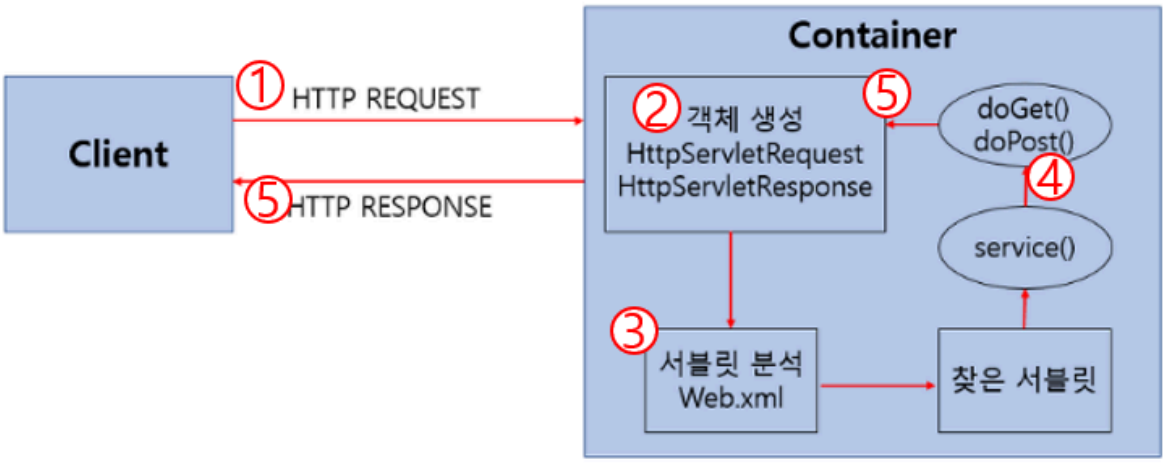
- 서블릿 상속 관계도

javax.servlet.Servlet 인터페이스

  ㄴ javax.servlet.GenericServlet 추상클래스

    ㄴ **javax.servlet.http.HttpServlet** 클래스

## * Servlet 동작 방식




1. 사용자(클라이언트)가 URL(Uniform Resource Locator)을

   클릭하면 **HTTP Request**(요청)를 **Servlet Container**로 전송

   -> 다음 장 설명 참고

2. Http Request를 전송 받은 Servlet Container는 아래 두 객체를 생성

   **HttpServletRequest**(요청 관련 내용이 저장된 객체 : 요청 URL, 파라미터, 헤더 등)),
   **HttpServletResponse**(응답 관련 내용이 저장된 객체 : 응답의 내용과 상태 코드, 헤더 등을 설정 가능)

3. DD (배포서술자, Deployment Descriptor) = **web.xml**은

   사용자가 요청한 URL을 분석하여 어떤 서블릿 클래스에 요청 내용을 전달할지 찾음

   **-> @WebServlet** 로 대체 가능

4. 해당 서블릿에서 init() 메소드를 먼저 호출한 후 service() 메소드를 호출하여

클라이언트로부터 전송 받은 방식인 GET, POST 여부에 따라 해당 메소드(doXXX()) 를 호출함.

- `init()`: 서블릿이 처음 생성될 때 한 번만 호출되며, 초기화 작업을 수행.
- `service()`: 클라이언트 요청이 들어올 때마다 호출됨. 요청 방식(GET, POST 등)에 따라

적절한 `doGet()`, `doPost()` 등의 메서드를 호출하는 역할.

5. doGet() / doPost() 메소드는 동적 페이지를 생성 후 **HttpServletResponse**객체에 응답을 보냄

6. 응답 종료 시 **HttpServletRequest, HttpServletResponse** 객체 소멸

## * Servlet Container

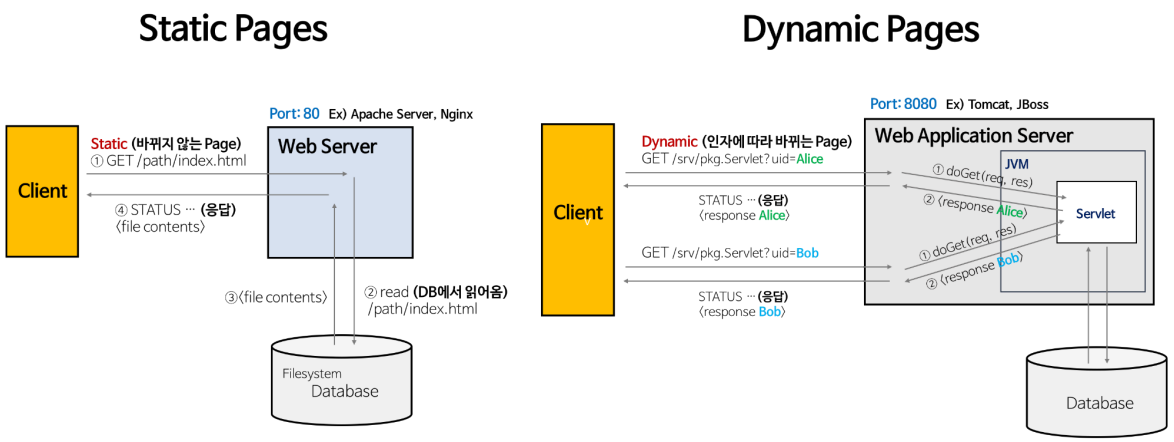
- 배포를 위한 포트 연결, 웹 서버 통신을 위한 소켓, 입/출력 스트림을 생성하는 역할을 함.

 Servlet Container는 **WAS(Web Application Server)**의 일부에 해당하며 대표적인 서블릿 컨테이너로 **Tomcat**이 있음.

- 클라이언트의 요청을 받을 때 마다 새로운 자바 스레드(Thread)를 만들어

요청을 처리하고 응답을 해줌.

( 단, 모든 요청마다 새로운 스레드를 생성하는 것이 아니라 이미 생성된 스레드를 재사용하는 경우도 있음(스레드 풀(Thread Pool) 이용) )



*** Get**과 **Post**방식의 비교**/**차이

- 클라이언트가 서버로 요청을 보내는 방법

**1. GET**방식 **: (**데이터를**)** 가져오다**,** 얻어오다

- URL에 변수(데이터)를 포함시켜 요청

보안 유지를 안 하기 때문에 로그인 같은 경우는 get방식으로 하면 부적합

- 데이터를 HTTP Header에 포함하여 전송

GET방식에서 바디는 보통 빈 상태로 전송 되며

헤더의 내용 중 Body의 데이터를 설명하는 Content-type헤더필드도 들어가지 않음

- 전송하는 길이 제한(보내는 길이가 너무 길면 초과데이터는 절단됨)

- 캐싱 가능 (ex. 즐겨찾기, 북마크)

(한번 접근 후, 또 요청할 시 빠르게 접근하기 위해 데이터를 저장시켜 놓는 것)

**2. POST**방식 **: (**데이터를**)** 부치다**.**

- 데이터를 서버로 제출하여 추가 또는 수정하기 위해 데이터를 전송하는 방식

- URL에 변수(데이터)를 노출하지 않고 요청 데이터를 HTTP Body에 포함하여 전송

- 헤더필드 중 Body의 데이터를 설명하는 Content-Type이라는 헤더필드가 들어가고
어떤 데이터 타입인지 명시해주어야 함

- 전송하는 길이 제한이 없음.
Body에 데이터가 들어가기 때문에 길이에 제한이 없지만
최대 요청을 받는 시간(Time Out)이 존재해서
페이지 요청, 기다리는 시간 존재

- 캐싱할 수 없음.
URL에 데이터가 노출 되지 않으므로 즐겨찾기나 캐싱 불가능
하지만 쿼리스트링(문자열)데이터, 라디오 버튼, 텍스트 박스와 같은
객체들의 값도 전송 가능