01_Servlet이란

Servlet이란

- **웹 서비스를 위한 자바 클래스** (자바를 이용하여 웹을 만들기 위해 필요한 기술)

- 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 **(요청)**
  서버는 아이디와 비밀번호를 확인하고 다음 페이지(로그인 성공 또는 실패)를
  보여주는 역할 수행 **(응답)**

- Servlet은 자바로 구현된 **CGI라고 함

> CGI는 특별한 라이브러리나 도구를 의미하는 것이 아닌,
> 별도로 제작된 웹 서버와 프로그램간의 교환 방식

Servlet의 특징

클라이언트의 요청에 따라
Java Application(자바 프로그램) 으로
웹 브라우저용 출력 화면(HTML) 을 만들어 응답하는 기술

- 클라이언트의 요청에 대해 동적으로 작동하는 웹 애플리케이션 컴포넌트.

- http프로토콜 서비스를 지원하는
  javax.servlet.http.HttpServlet 또는
  jakarta.servlet.http.HttpServlet 클래스를 상속 받음 (jdk, tomcat 버전 따라 다름)
- 클라이언트 요청에 대한 서버 응답 시
  미리 만들어 놓은 화면(정적, html파일)이 아닌
  요청을 받을 때 마다 알맞은 **화면을 만들어(동적, Java에서 HTML 코드를 작성)** 응답함.

- java thread를 이용하여 동작. (요청마다 별도 thread가 생성됨)
- MVC Model2패턴에서 Controller로 이용

Servlet의 단점

- Servlet은 Java 코드 내부에 HTML 코드를 작성하기 때문에
  HTML 코드를 수정하는 경우 컴파일을 다시 해야 되는 문제가 있음.

Servlet Container

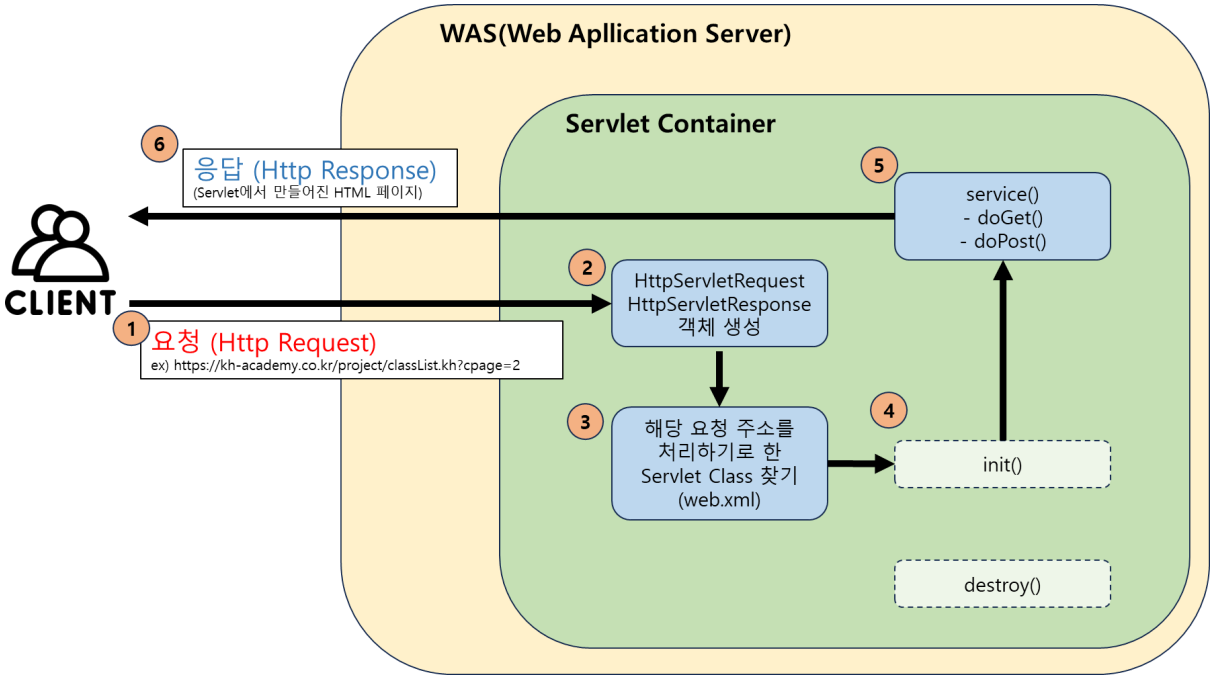
- 배포를 위한 포트 연결, 웹 서버 통신을 위한 소켓, 입/출력 스트림을 생성하는 역할을 함.
  WAS(Web Application Server)가 Servlet Container에 해당하며 대표적으로 Tomcat이 있음.

- 클라이언트의 요청을 받을 때 마다 새로운 자바 스레드(Thread)를 생성하여 요청을 처리하고 응답을 해줌.

Servlet Container의 역할

1. **웹 서버와의 통신 지원**
   서블릿과 웹 서버가 손 쉽게 통신할 수 있게 함.
   일반적으로 소켓을 만들고 listen, accept 등을 해야 하지만
   서블릿 컨테이너는 이러한 기능을 API로 제공하여 복잡한 과정 생략하게 함
2. **서블릿 생명주기(Life Cycle) 관리**
   서블릿 클래스를 로딩하여 인스턴스화 하고, 초기화 메서드를 호출하고,
   요청이 들어오면 적절한 서블릿 메서드 호출.
   서블릿이 생명을 다 한 순간에는 적절하게 가비지 컬렉션을 이용해 메모리 관리 편의 제공
3. **멀티쓰레드 지원 및 관리**
   서블릿 컨테이너는 요청이 올 때마다 새로운 자바 스레드를 생성하는데
   응답이 완료되면 스레드는 자동으로 사라짐.
4. **선언적인 보안 관리**
   서블릿 컨테이너를 사용하면 개발자는 보안에 관련된 내용을
   서블릿 또는 자바 클래스에 구현하지 않아도 됨.
   일반적으로 보안 관리는 xml 배포 서술자 (Deployment Descriptor (web.xml))에 기록하므로 보안에 대해 수정할 일이 생겨도
   자바 코드를 수정하여 다시 컴파일 하지 않아도 보안 관리 가능

Servlet 동작 방식(순서)



1. 사용자(클라이언트)가 URL(Uniform Resource Locator)을
   입력하면 **HTTP Request**(HTTP 방식의 요청)를 **Servlet Container**로 전송
2. Http Request를 전송 받은 Servlet Container는
   HttpServletRequest(요청 관련 내용이 저장된 객체),
   HttpServletResponse(응답 관련 내용이 저장된 객체)
   객체를 생성
3. DD (배포서술자, Deployment Descriptor) = **web.xml**을 이용해
   사용자가 요청한 URL을 분석하여 **어떤 Serlvet Class 에 요청 내용을 전달할지 찾음**
4. 메모리에 해당 Serlvet Class가 load(적재)되어 있지 않으면(객체로 만들어져 있지 않다면)
   해당 Servlet Class의 **init()** 메서드를 호출해서 객체 생성 및 초기화 진행
5. service() 메서드를 호출하여
   클라이언트로부터 전송 받은 방식인 GET, POST 여부에 따라

doGet() 또는 doPost() 메서드를 호출하여 수행
→ 해당 메서드에서 응답을 위한 코드(java, html)를 작성함

6. doGet() / doPost()로 동적 페이지를 생성 후 **HttpServletResponse**객체에 응답을 보냄

> destory() 메서드는 서버 실행 중 Servlet Class의 내용이 변하게 되는 경우
> 기존 Servlet 객체를 파괴해야 할 때 자동으로 수행.