2. Servlet

Servlet은 Server Application Let의 줄임 말로 한마디로 서버의 조각을 Servlet이라고 하며 자바 코드 안에 HTML코드를 포함하여 동적인 웹 페이지로 작성하는 것으로 JAVA EE에서 제공하는 하나의 자바 클래스라고 설명할 수 있다. .

1) Servlet 작성과정

- 먼저 jakarta.servlet.http.HttpServlet 클래스로부터 상속받아 Servlet 클래스(java파일)를 생성한다
- 요청 방식과 생성주기에 따른 필요한 메소드를 오버라이딩해 작성한다.
- 필요에 따라 web.xml이나 해당 클래스의 어노테이션(@)을 통해 매핑한다.
  *Servlet 3.0 이상에서는 web.xml을 사용하지 않고 어노테이션을 사용한다.

ex)

```
@WebServlet("/ex01") // 웹서버를 실행했을때 매핑되는 url경로
public class Ex02 extends HttpServlet {
	private static final long serialVersionUID = 1L;

	//컨테이너가 해당 서블릿 객체를 생성할때 최초 한번 실행된다. 주로 초기화를 할때 사용한다.
	public void init() throws ServletException {
		super.init();
	}

	// service()메소드는 요청이 있을때 마다 재호출되는 메소드이다. 요청 방식에 상관없이 호출되며
	// 해당 메소드가 정의되 있으면 doGet()과 doPost()는 호출되지 않는다.
	protected void service(HttpServletRequest arg0, HttpServletResponse arg1)
			throws ServletException, IOException {
		super.service(arg0, arg1);
	}

	// get 방식으로 전달된 데이터를 처리하는 doGet()메소드
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			// 매개변수로 request와 response를 받는다.
			throws ServletException, IOException {
		response.getWriter().append("Served at: ").append(request.getContextPath());
	}

	// post 방식으로 전달된 데이터를 처리하는 doPost()메소드
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
			// 매개변수로 request와 response를 받는다.
			throws ServletException, IOException {
		doGet(request, response);
	}

	//해당 서블릿 객체가 삭제되는 시점(웹 서버의 종료)에 실행되는 메소드
	public void destroy() {
	}
```

*** 매핑이란**

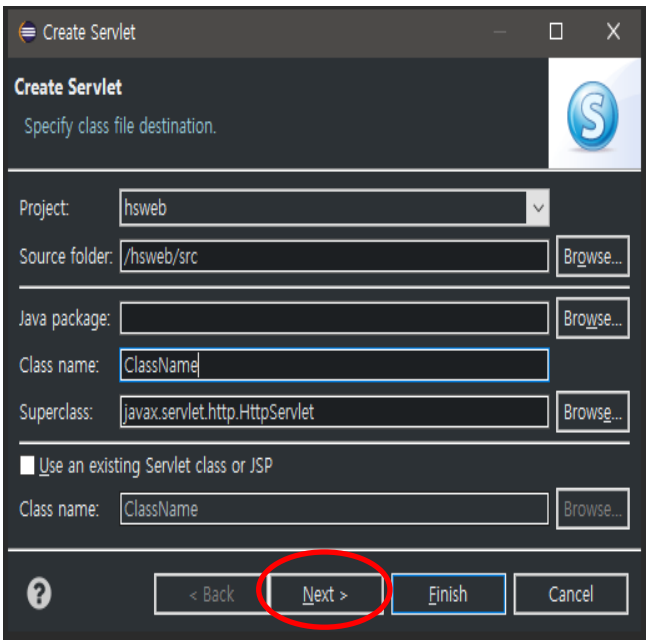
매핑이란 서블릿의 구분을 위한 URL주소를 가상의 경로로 이름을 붙여주는 것이다.

어떤 프로그램이든 특정 파일에 접근하려면 '경로'가 필요하다. 웹 브라우저가 웹 서버에게 요청(Request)을 할 때, 서블릿이 필요하다면 해당 서블릿이 위치한 경로를 이용하면 되는데 이때 경로는 '**가상 경로**'이어야 한다. 이는 서블렛은 서버의 내부 로직 파일이기 때문에 보안을 위해서 내부 파일은 외부로 노출하지 않아야 한다.

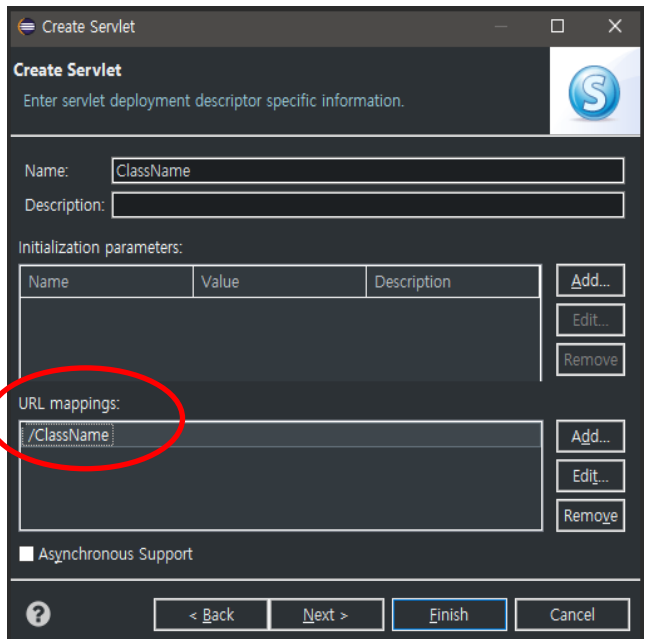
또한 경로를 매핑을 통해 함축적으로 간결하게 표시해 관리할 수 있다는 장점도 있다.

(1) servlet의 **@WebServlet("/URL")** 어노테이션을 사용한 매핑

- 첫번째 방법은 이클립스에서 servlet파일을 생성할 때 같이 매핑하거나



->



- 두번째는 servlet 클래스 생성후 어노테이션을 통해 매핑하는 방법이다.

ex)

```
  @WebServlet("/ex01")
//@WebServlet("/URL")의 URL 주소로 접속하면 톰켓 서버의 컨테이너가 매핑된 서블릿을 찾아 실행해 준다
// 어노테이션을 이용해서 만드는 서블릿 클래스는 반드시 HttpServlet을 상속받아야 한다.
public class Ex02 extends HttpServlet {
}
```

* 배포 서술자(web.xml)를 이용한 매핑 방법도 있었지만 서블릿 3.0 부터는 web.xml 대신 서블릿 클래스를 통해서만 설정을 할 수 있기 때문에 따로 정리하지 않는다.

* 배포 서술자란(web.xml) Deployment Descriptor라고도 하며, 웹 어플리케이션에 대한 전반적인 설정을 하는 XML형식의 파일로 이곳에 웹 어플리케이션 운용 시 사용하는 설정(Error Page, Fillter등) 값들을 정의할 수 있다.

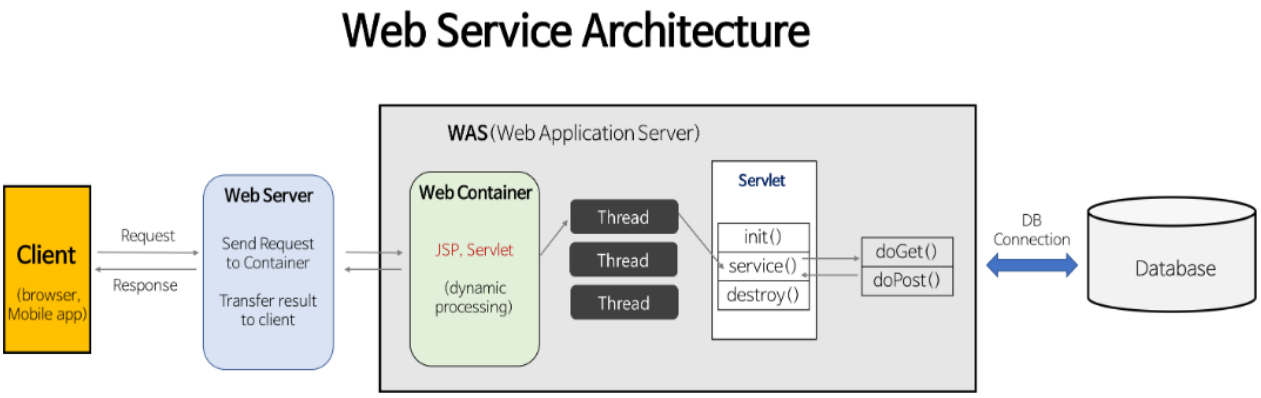
배포 서술자는 WAS구동시 web.xml을 읽고 웹 어플리케이션 설정을 하므로 web.xml파일을 수정할 경우엔 WAS를 재 시작 해야한다. 또한 다음과 같은 특징이 있다.

- 하나의 웹 프로젝트에 하나만 존재한다.

- 위치는 항상 동일하며 Web Document Root 디렉토리에 WEB-INF폴더 하위에 위치하게 된다.
  ex)/WebContent/WEB-INF/web.xml

- 서블릿 매핑 등 다양한 어플리케이션 정보를 텍스트 기반으로 설정할 수 있어 서비스 운영 중 프로그램을 수정하지 않고도 어플리케이션 동작 등에 대한 조정이 가능하다.
  즉 소스코드가 없더라도, 애플리케이션을 목적에 맞게 수정할 수 있다.

2) servlet 작동순서



1. 먼저 사용자(클라이언트)가 웹 서버에 요청(URL을 입력)을 했을 때 그 요청이 서블릿에 대한 요청일 경우 해당 요청이 Web Container로 전송된다.

2. 요청을 전송 받은 Web Container는 HttpSetvletRequest, HttpServletResponse 객체를 생성한다.
   - 이때 Web Container는 객체 생성 유무를 파악해 해당 서블릿이 최초 요청 일시 서블릿 객체를 생성하고 이후 요청 시 앞서 생성한 서블릿 객체를 그대로 사용한다. 스레드를 생성한다.

3. Web Container에서 web.xml(배포 서술자)을 기반으로 사용자가 요청한 URL이 어느 서블릿에 대한 요청인지 찾는다.

4. 해당 서블릿에서 service메소드를 호출한 후 클라이언트의 GET, POST 여부에 따라 doGet() 또는 doPost() 메소드를 호출한다.

5. doGet() 또는 doPost() 메소드에서 실행된 동적 페이지의 결과물을 생성한 후 HttpServletResponse 객체에 응답을 보낸다.

6. 응답이 끝나면 HttpServletRequest, HttpServletResponse 두 객체를 소멸 및 스레드를 종료한다.

3) servlet 요청 처리 방식

클라이언트에서 WAS로 정보를 요청하는 방식은 두 가지가 있는데 (이때 전송 방식은 HTML코드안에 명시된 방법으로 정해진다.) Get과 Post방식이다. servlet은 이런 방식(get, post)에 따라 doGet()과 doPost() 메소드로 요청을 처리한다. 특징은 다음과 같다.

(1) Get방식

- URL에 보낸 정보가 표시되어 보안에 약하다.
- URL에 데이터를 포함시켜 요청한다. (?로 구분) ex) http://localhost:/ch01/ex6.jsp**?su1=1**
- 전송하는 데이터의 한계(255자 이내)가 있으며 보안성이 없다

(2) Post방식

- URL에 보낸 정보가 표시되지 않고 요청한다..
- 전송하는 데이터의 크기에 제한이 없으며 URL에 표시되지 않아 보안성이 좋다.
- GET은 URL에 데이터를 붙여서 전송하는 반면, POST는 BODY에 데이터를 넣어서 전송한다
  따라서 어떤 데이터 타입인지 명시해 주어야 한다. ex) 한글 전송시 request.setCharacterEncoding("utf-8")

4) Servlet container

웹 컨테이너, JSP 컨테이너라고도 하며 JSP, Servlet을 실행시킬 수 있는 WAS에 포함된 소프트웨어이다.
서버에 서블릿을 만들었다고 해서 스스로 작동하는 것이 아니고 서블릿을 관리해주는 것이 필요한데,
그러한 역할을 하는 것이 바로 서블릿 컨테이너다. 서블릿 컨테이너는 클라이언트의 요청(Request)을 받아주고 응답(Response)할 수 있게, 웹서버와 소켓으로 통신하며 JSP, Servlet의 구동 환경을 제공한다고 볼 수 있다.

(1) 서블릿 컨테이너의 역할

1. 웹서버와의 통신 지원

- 서블릿과 웹 서버가 통신할 수 있는 쉬운 방법을 제공해준다. 일반적으로 개발자는 웹 서버에서 소켓(네트워크로 연결된 컴퓨터 간 데이터 통로)을 만들고 listen, accept 등을 해야 하지만, 서블릿 컨테이너가 이러한 기능을 API로 제공해 개발자에게 복잡한 과정을 생략하게 하고 편의를 제공한다.
  다시 말해 개발자에게 서블릿에 구현해야 할 비즈니스 로직에 초점을 둘 수 있게 도와준다

2. 서블릿 생명 주기(Life Cycle) 관리

- 먼저 요청시 서블릿 클래스를 로딩하여 인스턴스화 후 초기화 및 요청이 들어오면 적절한 서블릿 메소드 호출하고, 서블릿 소멸 시 Garbage Collection(가비지 컬렉션)을 진행하여 서블릿의 생성과 제거 관리한다.

3. 멀티스레드 지원 및 관리

- 서블릿 컨테이너는 요청이 들어올 때마다 자바 스레드 생성, HTTP 서비스 메소드를 실행하고 나면 스레드는 자동으로 죽게 되는데 원래는 스레드를 관리해야 하지만 서버가 다중 스레드를 생성 및 운영을 해준다

4. 선언적인 보안 관리

- 서블릿 컨테이너를 사용하면 개발자는 보안 관련 내용을 서블릿 or 자바 클래스에 구현해 놓지 않아도 된다. 일반적으로 보안관리는 XML 배포 서술자에 기록하므로, 보안에 대해 수정할 일이 생겨도 자바 소스 코드를 수정하여 다시 컴파일 하지 않아도 보안관리가 가능하다.

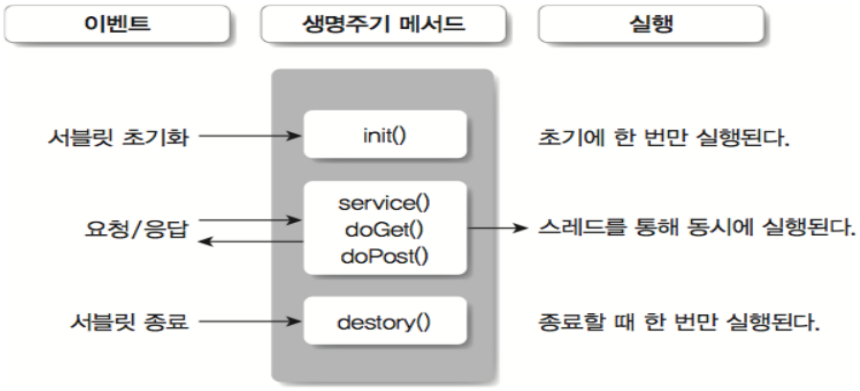
5) Servlet 생명주기(Life Cycle)

클라이언트가 서블릿에 요청을 하면, 서블릿은 바로 호출이 되지 않고 객체를 생성하고 초기화 작업을 거친 후, 요청을 처리하는 생명 주기를 갖고 있다.

1. 먼저 WAS는 클라이언트로부터 요청을 받으면 해당 서블릿이 메모리에 있는지 확인한다.
2. (해당 서블릿이 최초 요청되어 메모리에 없을시) 서블릿 클래스를 메모리에 올리고 init()메소드와 service()메소드를 실행한다.
   * 서블릿 객체를 생성하고 초기화하는 작업은 비용이 많은 작업이므로, 다음에 또 요청이 올 때를 대비하여 이미 생성된 서블릿객체는 메모리에 남겨둡니다.
3. (해당 서블릿이 메모리에 있으면) service()메소드만 실행한다.
4. WAS가 종료되거나 웹 어플리케이션이 갱신되어 서블릿 종료 요청이 있을때 destory()메소드를 실행한다.

※ 서블릿은 최초 요청시 객체가 생성되고 이때 생성된 객체가 재사용되며, 서버가 중지될 때 서블릿 객체는 삭제된다.

(1) Servlet 생명주기 메소드



- init( )

서블릿 요청시 객체가 생성된 다음에 최초 한번 호출되는 메소드로서, 재정의가 가능하다.

Init()메소드는 생명주기중 오직 서블릿이 생성 될 때만 호출되며 이후 사용자의 요청에 호출되지 않는다.

주로 기본적으로 반복하는 기본 셋팅을 담으며 일반적으로 서블릿 생성 시 초기화 작업을 주로 수행한다

- service( )

요청/응답(request/response)를 처리하는 메서드로 웹 서버는 서블릿에 대한 요청을 수신하면 service()

메서드를 호출하는 새로운 스레드를 생성한다. 서블릿 요청이 있을 때 마다 재사용되어 호출된다.

클라이언트의 요청이 가장 먼저 처리되는 과정으로 매개변수로 request와 response객체가 제공된다.

재정의 하지 않을 시 요청정보 헤더의 요청방식에 따라 서로 다른 메소드 doGet()과 doPost()를 호출한다

- destroy()

해당 서블릿 객체의 생명주기가 끝날 때, 서블릿이 기능을 수행하고 메모리에서 소멸될 때 한번 호출된다.

주로 DB의 연결을 해제하거나, 서버의 리소스들을 다시 반납해주는등 마무리 작업을 담는다.