

서블릿(Servlet)

서블릿이란?

Server + Applet의 합성어, JAVA 언어를 이용하여 사용자의 요청을 받아 처리하고 그 결과를 다시 사용자에게 전송하는 역할의 Class 파일을 말한다. 즉 웹에서 동적인 페이지를 java로 구현한 서버측 프로그램이라고 보면 된다.

※ 관련 패키지와 클래스는 tomcat에서 제공하는 API문서에서 확인 가능

<https://tomcat.apache.org/tomcat-8.0-doc/servletapi/>

서블릿의 역사

Java 언어의 창시자인 제임스 고슬링(James Gosling)은 1995년 자바를 발표하며 자바로 구현할 수 있는 서버 프로그래밍 기술에 대해서도 염두에 두고 있었지만, 해당 개념이 실제 구현이 가능할 정도의 제품화가 되어 있지 않은 상태였다. 당시 Sun사에서는 이를 서버로 구현할 수 있는 제품이 없어 잠시 미룰 수 밖에 없었다. 즉, Java EE Platform이 제품화되어 있지 않은 시기의 이야기인 것이다.

얼마 뒤, 자바 팀의 일원이었던 파바니 디완지(Pavni Diwanji)는 자바 서버 기술에 대한 필요성을 느껴 서블릿의 개념을 고안하였고, 이 개념을 토대로 프로젝트를 진행하여 서블릿 구현 및 제품화에 성공하였다. 그리고 이 기술은 1997년 6월에 Servlet 1.0을 공식 발표하면서 Java EE의 제품화에 포함되었다.



서블릿 버전 변천사

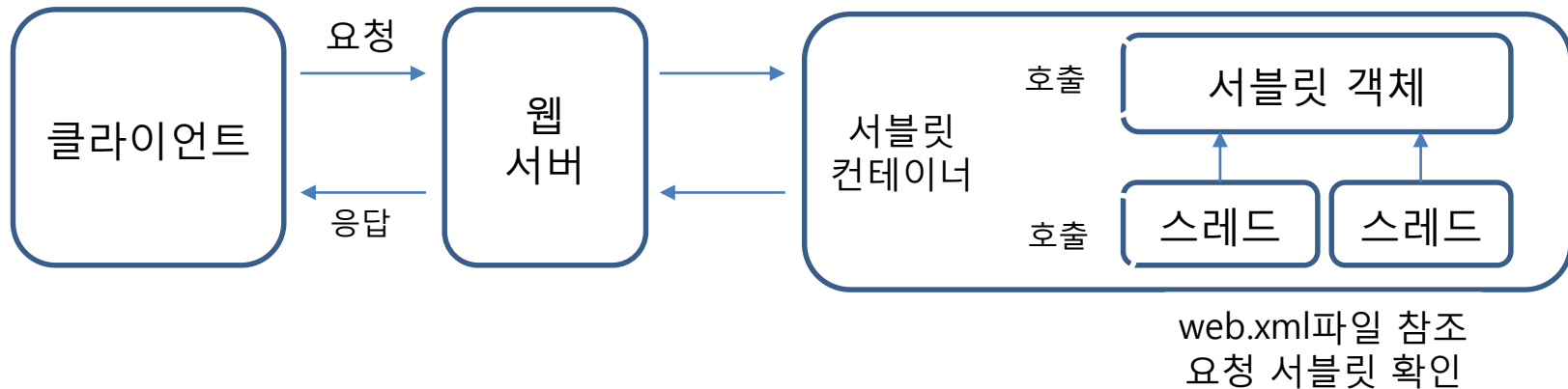
| 서블릿 버전 | 발표일 | 지원 Platform | 주요 변경 내용 |
|-------------|---------|------------------------|--|
| Servlet 1.0 | 1997.06 | . | |
| Servlet 2.1 | 1998.11 | JDK 1.1 | RequestDispatcher, ServletContext 요소를 처음으로 정의 |
| Servlet 2.3 | 2001.08 | J2EE 1.3 J2SE 1.2 | Filter 클래스 추가 |
| Servlet 2.4 | 2003.11 | J2EE 1.4 J2SE 1.3 | web.xml문서를 통한 XML 스키마 사용 |
| Servlet 2.5 | 2005.09 | Java EE 5 Java SE 5 | 어노테이션 지원 기능 추가 |
| Servlet 3.0 | 2009.12 | Java EE 6 Java SE 6 | 어노테이션 지원 범위 확대, 서블릿 보 안 강화, 비동기 서블릿 지원 |
| Servlet 3.1 | 2013.05 | Java EE 7 | Non-blocking I/O 지원, HTTP1.1 지원 |

※ J2EE, J2SE 1.5 버전 부터 표기 방식을 Java EE / SE 로 변경

서블릿 설계 규약

- I. 모든 서블릿은 **javax.servlet.Servlet** 인터페이스를 상속 받아 구현한다.
- II. 서블릿을 구현 시 Servlet 인터페이스와 **ServletConfig** 인터페이스를 **javax.servlet.GenericServlet**에 구현한다.
- III. HTTP 프로토콜을 사용하는 서블릿은 **javax.servlet.http.HttpServlet** 클래스는 **javax.servlet.GenericServlet**를 상속한 클래스로 서블릿은 **HttpServlet**클래스를 상속받는다.
- IV. 서블릿의 Exception을 처리하기 위해서는 **javax.servlet.ServletException**을 상속 받아야 한다.

※ 소스 코드로 확인 구현 확인



☞ 서블릿 컨테이너란

웹 서버 또는 응용 프로그램 서버의 일부로, 웹 서버에서 온 요청을 받아 서블릿 class를 관리하는 역할(생명주기)을 한다. 컨테이너의 서블릿에 대한 설정은 Deployment Descriptor(web.xml)파일을 이용

배포서술자(DD)

어플리케이션에 대한 전체 설정정보를 가지고 있는 파일로 이정보를 가지고 웹 컨테이너가 서블릿을 구동, xml파일로 요소(태그)로 이루어져 있음. 어플리케이션 폴더의 WEB-INF폴더에 **web.xml**이라는 파일이다.

설정정보

- Servlet정의 / Servlet 초기화 파라미터
- Session설정 파라미터
- Servlet/jsp 매핑 / MIME type 매핑
- 보안설정
- Welcome file list설정
- 에러 페이지 리스트, 리소스 그리고 환경변수

파일세부내용

<web-app> : 루트속성, 문법식별자 및 버전의 정보를 속성값으로 설정

<context-param> : 웹 어플리케이션에서 공유하기 위한 파라미터 설정

<mime-mapping> : 특정파일 다운로드시 파일이 깨지는 현상방지

<servlet> ~ <servlet-class> / <servlet-mapping> : 서블릿 맵핑

<servlet> ~ <servlet-class> : 컨테이너에 서블릿 설정

예 default : 공유자원제공 및 디렉토리목록 제공, jsp : jsp컴파일과 실행 담당

<welcome-file-list> : 시작페이지 설정

<filter> : 필터정보 등록

<error-page> : 에러발생시 안내페이지설정

<session-coiifig> : session기간 설정

<listener> : 이벤트 처리 설정(6가지)

서블릿 mapping

client가 servlet에 접근할 때 원본 클래스명이 아닌 다른 명칭으로 접근시 사용 설정방법은 web.xml을 이용하여 적용하는 방법과 @annontation를 이용하는 방법이 있음.

web.xml 이용

```
<servlet>
    <servlet-name>mapping명칭</servlet-name>
    <servlet-class>실제 클래스명칭</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>mapping명칭</servlet-name>
    <url-pattern>사용자 접근명칭</url-pattern>
</servlet-mapping>
```

@annotation 이용

@web-Servlet("/매핑명칭")

public class 서블릿명칭 extends HttpServlet {

 servlet코드.....

}

ServletConfig

web.xml에 대한 정보를 가져오는 객체(interface)로 저장할 객체를 만들어 저장하고 `getServlet` 메소드로 정보가 저장된 객체 호출가능, `getInitParam("저장이름")`으로 초기화된 값을 가져올 수 있음

** 지정된 servlet에서만 활용 가능한 초기값임 / 동적 수정이 불가하여 상수값으로 보면 됨.

** servlet이 초기화된 이후에 활용가능

```
<web-app>
```

```
  <servlet>
```

```
    <servlet-name>mappin명칭</servlet-name>
```

```
    <servlet-class>설정 클래스명칭</servlet-class>
```

```
    <init-param>
```

```
      <param-name>저장 이름</param-name>
```

```
      <param-value>저장값</param-value>
```

```
    </init-param>
```

```
  </servlet>
```

```
</web-app>
```

ServletContext

ServletConfig의 초기값은 지정된 Servlet에서만 사용이 가능하나 ServletContext는 모든 어플리케이션이 공용으로 사용하는 초기값을 설정, 값은 `getContext().getInitParam("저장이름")`으로 호출

** <servlet> 태그안이 아니라 <web-app>내부에 설정

** getContext앞에는 getConfig() / this가 생략되어 있음.

```
<web-app>
```

```
  <context-param>
```

```
    <param-name>저장 이름</param-name>
```

```
    <param-value>설정값</param-value>
```

```
  </context-param>
```

```
</web-app>
```

sever.xml

WAS서버에 대한 설정을 변경할 수 있는 파일

설정정보

- Context path설정(서버내 애플리케이션 설정)
- 어플리케이션 포트설정
- default접속 경로 설정(localhost설정)
- 특정 이벤트 설정 등이 있음

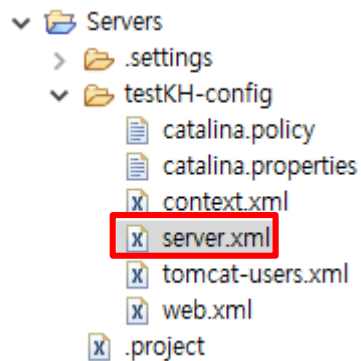
컨텍스트 패스(Context Path)

어플리케이션에 접근하는 경로를 말한다. * 컨테이너에서 어플리케이션 구분
즉, 어플리케이션의 root경로라고 볼 수 있다.

http://localhost:[PORT번호]/[프로젝트 별칭]/[Servlet 명]

EX) http://localhost:8800/**first/test1.do**

프로젝트의 별칭은 톰캣 서버 설정의 server.xml 내 <Context> 설정을 따른다.

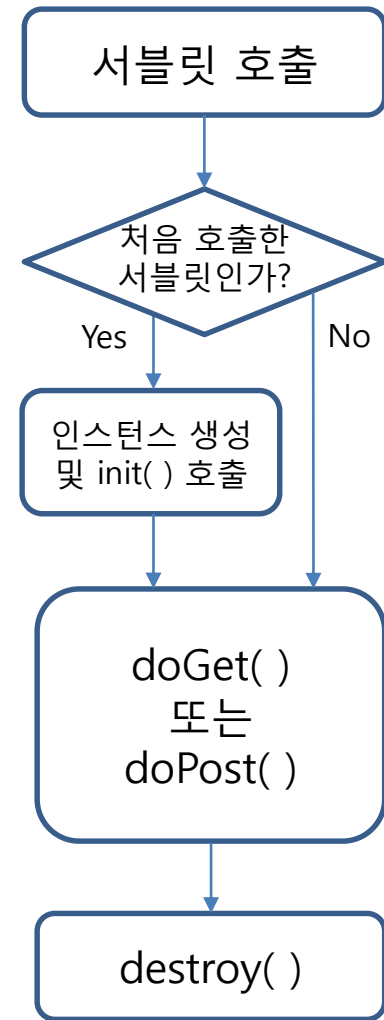


```
<Context docBase="firstProject" path="/first" reloadable="true"  
source="org.eclipse.jst.jee.server:firstProject"/>
```

서블릿 라이프사이클

1. 첫 번째 요청일 경우, 객체를 생성하며 `init()` 메소드를 호출한다.
2. 이 후 작업이 실행 될 때마다 `service()` 메소드가 요청한 HTTP Type에 따른 `doGet()`, `doPost()` 메소드 호출
3. 최종적으로 서블릿이 서비스 되지 않을 때 `destroy()` 메소드가 호출.

** `destroy()` 는 보통 서버가 종료되었을 때,
서블릿의 내용이 변경되어 재 컴파일 될 때 호출한다



Servlet 구동

