Servlet

▶ 서블릿(Servlet)

Server + Applet의 합성어로 JAVA언어를 이용하여 사용자의 요청을 받아 처리하고 그 결과를 다시 사용자에게 전송하는 역할의 Class파일을 말함

즉 웹에서 동적인 페이지를 java로 구현한 서버 측 프로그램

* 관련 패키지 및 클래스는 tomcat에서 제공하는 API문서에서 확인 가능

https://tomcat.apache.org/tomcat-8.5-doc/servletapi/

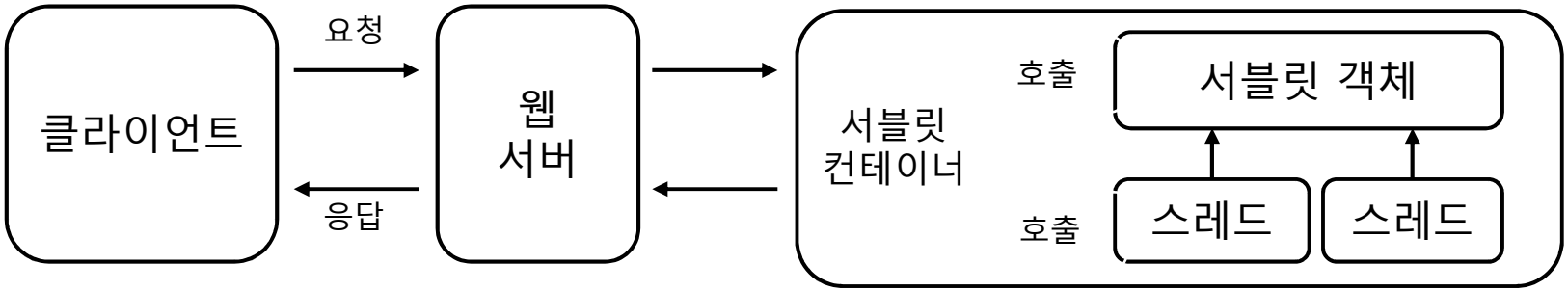
▶ 서블릿의 역사

제임스 고슬링(James Gosling)이 1995년 자바를 발표하면서
자바로 구현할 수 있는 서버 프로그래밍 기술에 대해서도
염두 해두고 있었지만 해당 개념이 실제 구현이 가능할 정도의
제품화가 되어 있지 않은 상태였음
얼마 뒤, 자바 팀의 일원이었던 파바니 디완지(Pavni Diwanji)는
자바 서버 기술에 대한 필요성을 느껴 서블릿의 개념을 고안하였고
이 개념을 토대로 프로젝트를 진행해 서블릿 구현 및 제품화에 성공
그리고 이 기술은 1997년 6월에 Servlet1.0을 공식 발표하면서
Java EE 제품화에 포함되었음

▶ 서블릿 설계 규약

1. 모든 서블릿은 javax.servlet.Servlet 인터페이스를 상속 받아 구현
2. 서블릿 구현 시 Servlet 인터페이스와 ServletConfig 인터페이스를 javax.servlet.GenericServlet에 구현
3. HTTP프로토콜을 사용하는 서블릿은 javax.servlet.http.HttpServlet 클래스는 javax.servlet.GenericServlet을 상속한 클래스로 서블릿은 HttpServlet클래스를 상속 받음
4. 서블릿 Exception처리를 하기 위해서는 javax.servlet.ServletException을 상속 받아야 함

▶ 서블릿 동작 구조



web.xml파일 참조
요청 서블릿 확인

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

▶ 배포 서술자

배포 서술자(DD, Deployment Descriptor)는 애플리케이션에 대한
전체 설정 정보를 가지고 있는 파일을 말하며 이 정보를 가지고
웹 컨테이너가 서블릿 구동, xml파일로 요소(태그)로 이루어져 있음
애플리케이션 폴더의 WEB-INF 폴더에 web.xml파일이 배포 서술자

✓ 설정 정보

- Servlet 정의, Servlet 초기화 파라미터
- Session 설정 파라미터
- Servlet/JSP 매핑, MIME type 매핑
- 보안 설정
- Welcome file list 설정
- 에러 페이지 리스트, 리소스, 환경 변수

▶ 배포 서술자

✓ 파일 세부 내용

<web-app> : 루트 속성, 문법 식별자 및 버전 정보를 속성 값으로 설정

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

<mime-mapping> : 특정 파일 다운로드 시 파일이 깨지는 현상 방지

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

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

ex. default : 공유 자원 제공 및 디렉토리 목록 제공

jsp : jsp컴파일과 실행 담당

<welcome-file-list> : 시작 페이지 설정

<filter> : 필터 정보 등록

<error-page> : 에러 발생 시 안내 페이지 설정

<session-config> : session 기간 설정

<listener> : 이벤트 처리 설정(6가지)

▶ 서블릿 매핑

클라이언트가 servlet에 접근할 때 원본 클래스 명이 아닌

다른 명칭으로 접근 시 사용

설정 방법은 web.xml과 @annotation을 이용하는 방법이 있음

▶ 서블릿 매핑

✓ web.xml 예시

```
<servlet>
    <servlet-name>mapping 명칭</servlet-name>
    <servlet-class>실제 클래스 명칭</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>mapping 명칭</servlet-name>
    <url-pattern>사용자 접근 명칭</url-pattern>
</servlet-mapping>
```

✓ @annotation 예시

```
@web-Servlet("/매핑명칭")
public class 서블릿명칭 extends HttpServlet{
            // servlet code
}
```

▶ 서블릿 초기 값 설정

✓ ServletConfig

web.xml에 대한 정보를 가져오는 객체(interface)로

저장할 객체를 만들어 저장하고 getServlet()으로 정보가 저장된

객체 호출 가능

getInitParam("저장 이름")으로 초기화 된 값을 가져올 수 있음

* 지정된 servlet에서만 활용 가능한 초기 값, 동적 수정이 불가해 상수 값으로 보면 됨

* servlet이 초기화된 이후에 사용 가능

```
<web-app>
        <servlet>
                <servlet-name>mapping 명칭</servlet-name>
                <servlet-class>설정 클래스 명칭</servlet-class>
                <init-param>
                        <param-name>저장 이름</param-name>
                        <param-value>저장 값</param-value>
                </init-param>
        </servlet>
</web-app>
```

▶ 서블릿 초기 값 설정

✓ ServletContext

ServletConfig의 초기 값은 지정된 Servlet에서만 사용 가능하나 ServletContext는 모든 애플리케이션이 고용으로 사용하는 초기 값 설정, 값은 getContext().getInitParam("저장 이름")으로 호출

* <servlet>태그 안이 아니라 <web-app>내부에 설정
* getContext앞에는 getConfig() / this가 생략되어 있음

```
<web-app>
        <context-param>
                <param-name>저장 이름</param-name>
                <param-value>설정 값</param-value>
        </context-param>
</web-app>
```

▶ WAS서버 설정 변경

✓ server.xml 예시

WAS서버에 대한 설정을 변경할 수 있는 파일

✓ 설정 정보

- Context path설정(서버 내 애플리케이션 설정)
- 애플리케이션 포트 설정
- default 접속 경로 설정(localhost 설정)
- 특정 이벤트 설정 등

▶ 웹 애플리케이션 접근

✓ Context Path

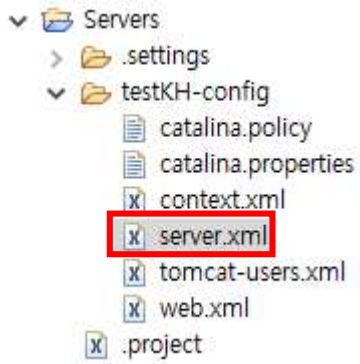
애플리케이션에 접근하는 경로 * 컨테이너에서 애플리케이션 구분

즉, 애플리케이션의 root경로

http://localhost:[port번호]/[프로젝트 별칭]/[servlet 명]

ex. http://localhost:9080/first/test1.do

* 프로젝트의 별칭은 톰캣 서버 설정의 server.xml 내 <context> 설정을 따름

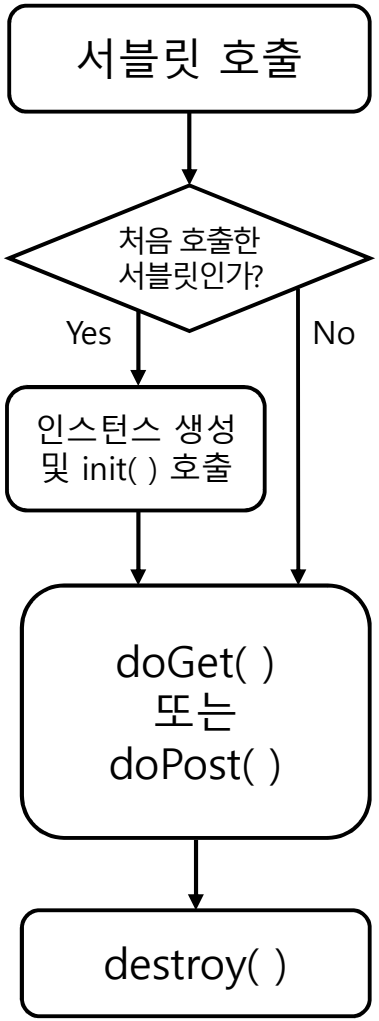



```
<Context docBase="firstProject" path="/first" reloadable="true"
 source="org.eclipse.jst.jee.server:firstProject"/>
```

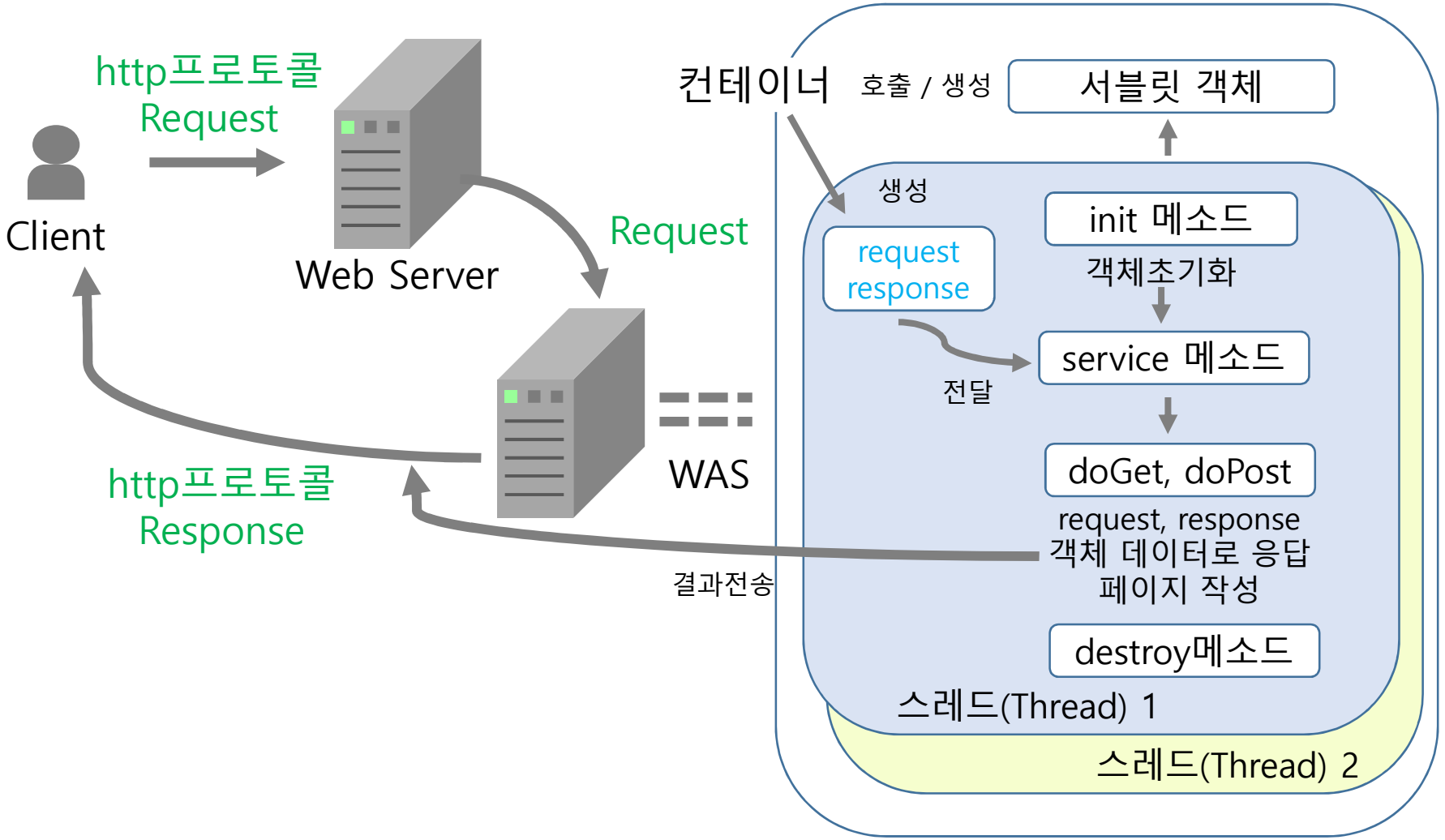
▶ 서블릿 라이프 사이클

1. 첫 번째 요청일 경우, 객체를 생성하며
   init()메소드를 호출함
2. 이 후 작업이 실행될 때마다 service()메소드가
   요청한 HTTP Type에 따른 doGet(), doPost() 호출
3. 최종적으로 서블릿이 서비스 되지 않았을 때
   destroy() 호출

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



▶ 서블릿 구동


http프로토콜
Request
Client
Web Server
Request
WAS
http프로토콜
Response
결과전송
컨테이너
호출 / 생성
서블릿 객체
생성
request
response
init 메소드
객체초기화
service 메소드
전달
doGet, doPost
request, response
객체 데이터로 응답
페이지 작성
destroy메소드
스레드(Thread) 1
스레드(Thread) 2

▶ 사용자 데이터 전송 방식

✓ get방식

URL창에 “?” 뒤에 데이터를 입력하는 방법(쿼리스트링)

데이터가 여러 개일 경우 &로 묶어서 보냄

데이터 검색에 많이 사용하고 데이터 크기에 한계가 있으며 보안 취약

✓ post방식

BODY에 내용을 보내는 방식으로 데이터 크기에 제한이 없고

보안이 뛰어남

* Servlet이 두 방식 중 하나로 전달 받으면 해당하는 메소드 호출
  html의 <form>에서 method속성을 이용해 방식 결정(default : get)

▶ 서블릿 메소드

✓ doGet

client에서 데이터 전송 방식을 get방식으로 전송하면 호출되는 메소드

✓ doPost

client에서 데이터 전송 방식을 post방식으로 전송하면 호출되는 메소드

* 반드시 ServletException 처리 해야 함

▶ 서블릿 매개변수 객체

✓ HttpServletRequest

HTTP Serlvet을 위한 요청 정보(request information) 제공

* 인터페이스나 인터페이스 구현은 컨테이너가 알아서 설정하므로 메소드만 이용

상속 : javax.servlet.ServletRequest

▶ 서블릿 매개변수 객체

✓ HttpServletRequest

메소드 명	내 용
getParameter(String)	client가 보내준 값이 저장된 명칭이 매개변수와 같은 명칭에 저장된 값을 불러오는 메소드
getParameterNames()	client가 보내준 값을 저장한 명칭을 불러오는 메소드
getParameterValues(String)	client가 보내준 값이 여러 개일 경우 그 값을 배열로 불러오는 메소드
getParameterMap()	client가 보내준 값이 전체를 Map방식으로 불러오는 메소드
setAttribute(String, object)	request객체에 전달하고 싶은 값을 String 이름으로 Object저장하는 메소드
getAttribute(String)	매개변수와 동일한 객체 속성값 불러오는 메소드
removeAttribute(String)	request객체에 저장되어 매개변수와 동일한 속성값 삭제하는 메소드
setCharacterEncoding(String)	전송 받은 request객체의 값들의 CharaterSet을 설정해주는 메소드
getRequestDispatcher(String)	컨테이너 내에서 request, response객체를 전송하여 처리할 컴포넌트(jsp파일 등)를 불러오는 메소드 forward()메소드와 같이 사용

▶ 서블릿 매개변수 객체

✓ HttpServletResponse

요청에 대한 처리 결과를 작성하기 위해 사용하는 객체

* 인터페이스나 인터페이스 구현은 컨테이너가 알아서 설정하므로 메소드만 이용

상속 : javax.servlet.ServletResponse

메소드 명	내 용
setContentType(String)	응답으로 작성하는 페이지의 MIME type을 정하는 메소드
setCharacterEncoding(String)	응답하는 데이터의 CharacterSet을 지정해주는 메소드
getWriter()	문자를 페이지에 전송을 위한 Stream을 가져오는 메소드
getOutputStream()	byte단위로 페이지에 전송을 위한 Stream을 가져오는 메소드
sendRedirect(String)	client가 매개변수의 페이지를 다시 서버에 요청하게 하는 메소드