

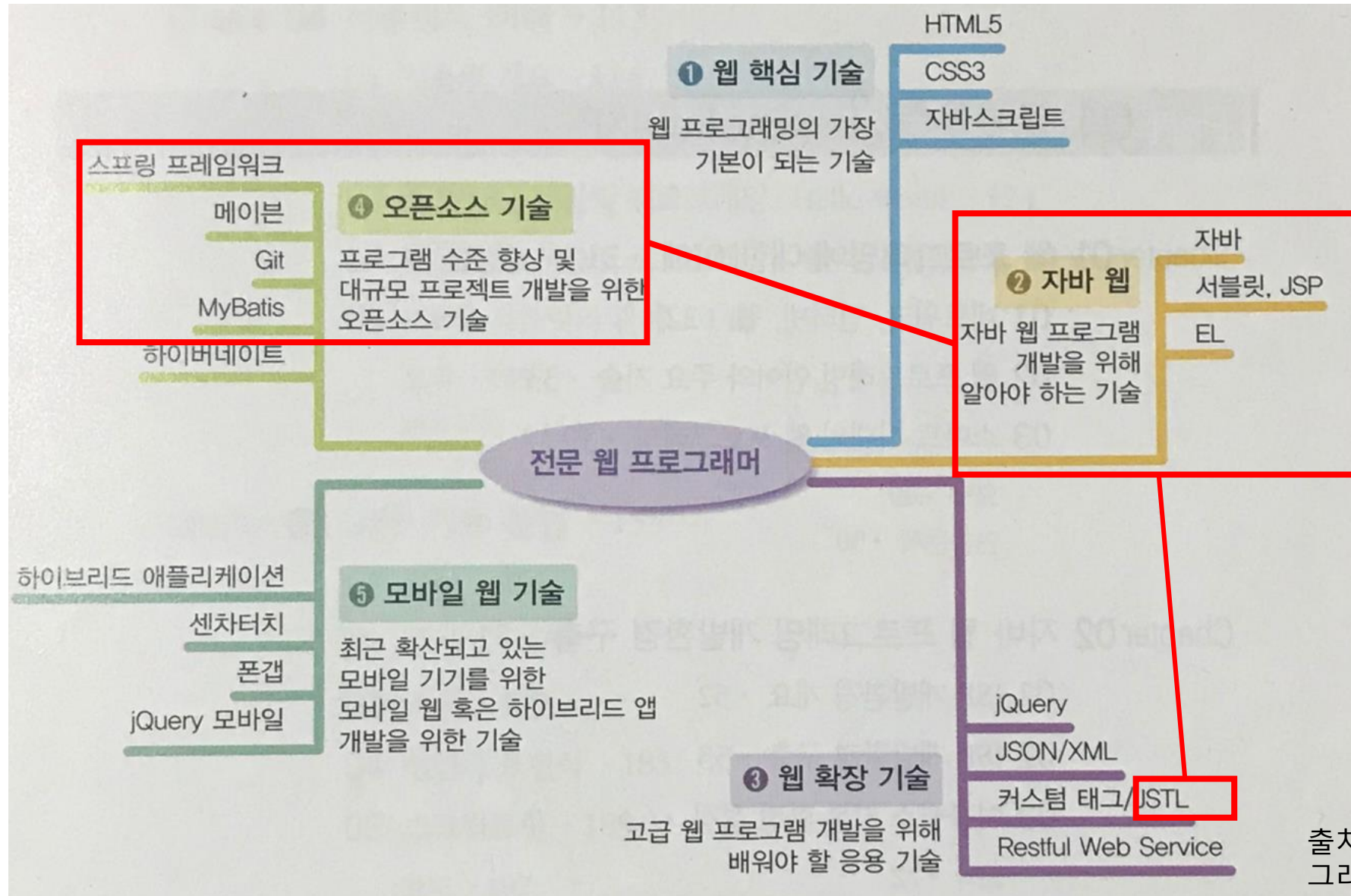
Servlet 기초

동의과학대학교 컴퓨터정보과
김진숙

학습 내용

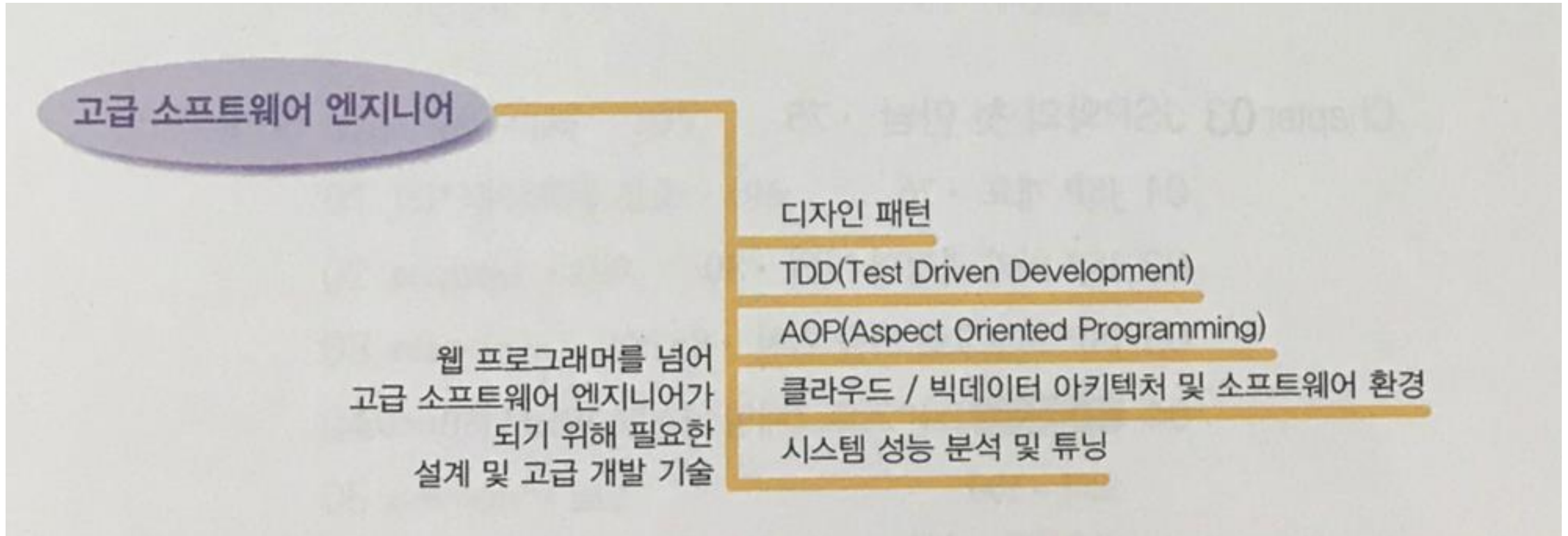
- CGI 프로그램과 servlet
- Servlet 이란?
- Servlet 컨테이너와 웹서버
- Servlet 동작과정
- Servlet 구조와 API
- Servlet / GenericServlet / HttpServlet
- Servlet 생명주기(Life Cycle)

웹프로그래밍 학습 로드 맵



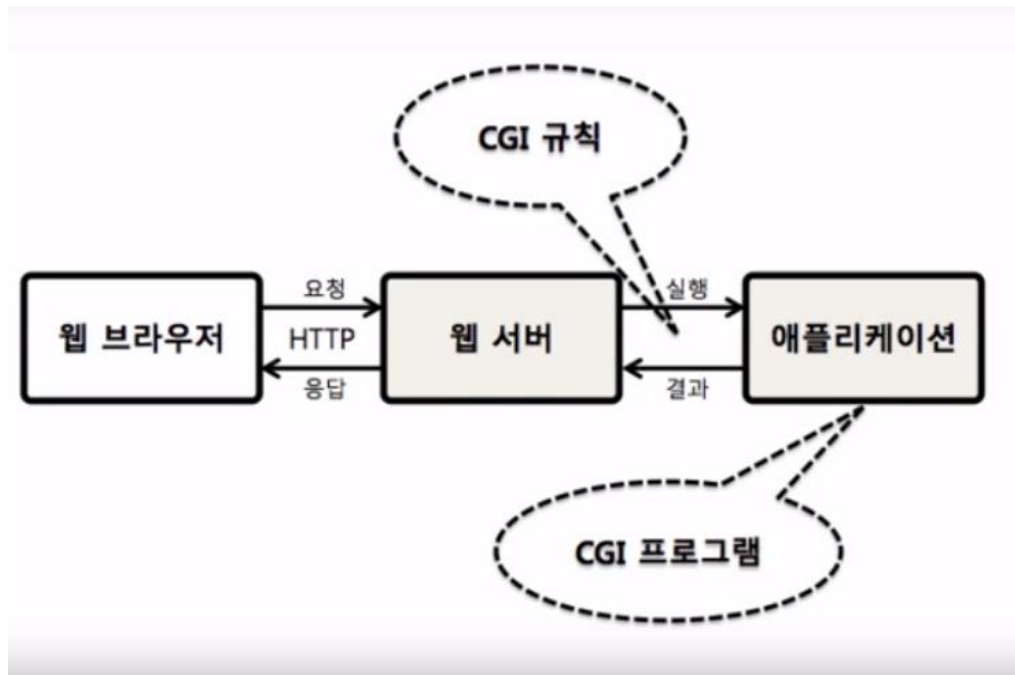
출처 : 프로젝트로 배우는 자바 웹 프로그래밍, 황희정, 한빛아카데미, 2016

웹프로그래밍 학습 로드 맵



CGI(Common Gateway Interface)

- CGI 규약
 - 웹 서버와 애플리케이션 간의 데이터를 주고 받는 규약
 - 특정 플랫폼에 의존하지 않고, 웹 서버로부터 외부 프로그램을 호출하는 규칙
- CGI 프로그램(스크립트) : CGI 규칙으로 작성한 웹 애플리케이션
 - 웹 애플리케이션 : 사용자가 웹 서버를 통해 간접적으로 실행시키는 프로그램

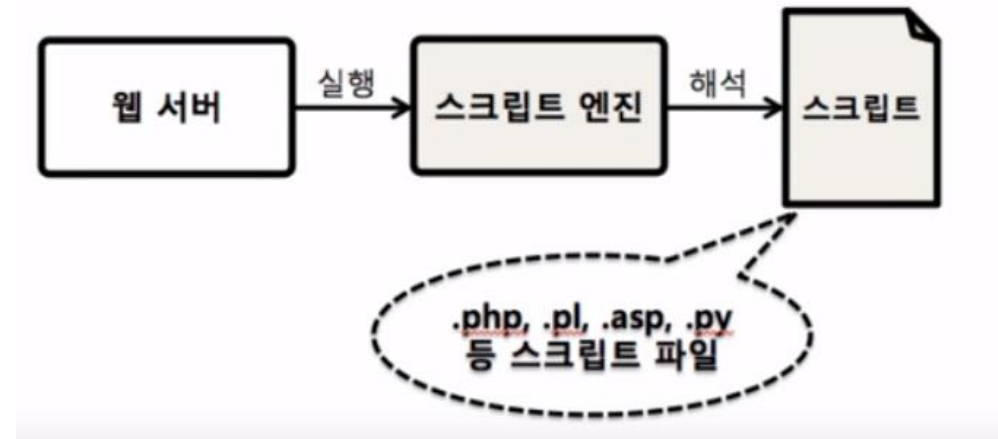


웹 서버가 애플리케이션을 실행

CGI 프로그램 유형

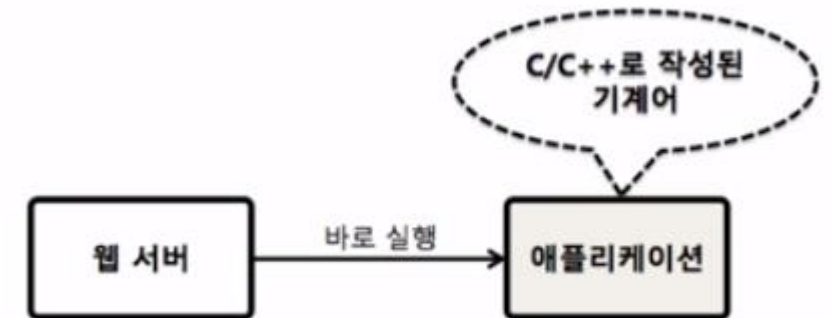
- 인터프리터 방식

- 프로그램 코드 자체로 애플리케이션을 실행
- 코드를 한 라인 씩 해석하고 실행
- PHP, ASP, Python 등



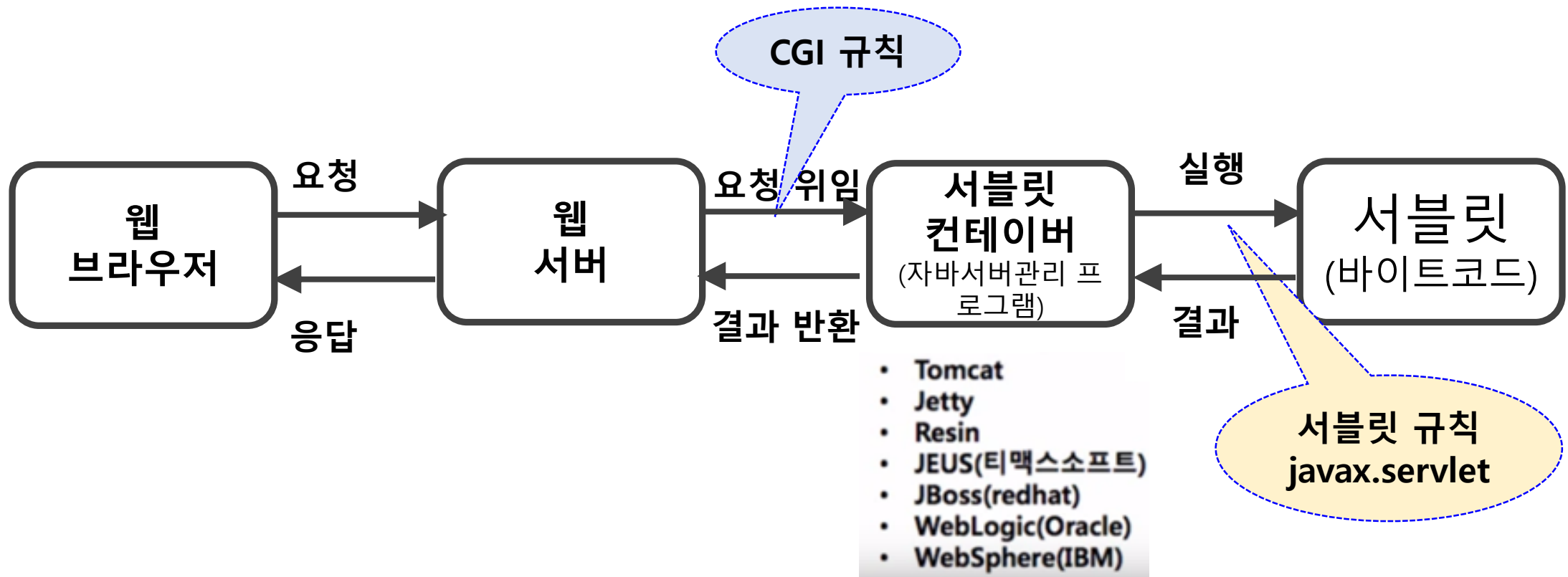
- 컴파일 방식

- 컴파일하여 기계어로 변환하여 실행
- 웹 서버가 기계어로 만들어진 애플리케이션을 실행



자바 웹 애플리케이션 실행

- Servlet 컨테이너(WAS)
 - CGI 규칙이 적용된 자바 서버 관리 프로그램
 - Servlet 컨테이너는 라이프 사이클을 통해 Servlet 을 포함하고 관리
 - 개발자는 **Servlet 규칙**에 따라 자바 CGI 프로그램을 작성



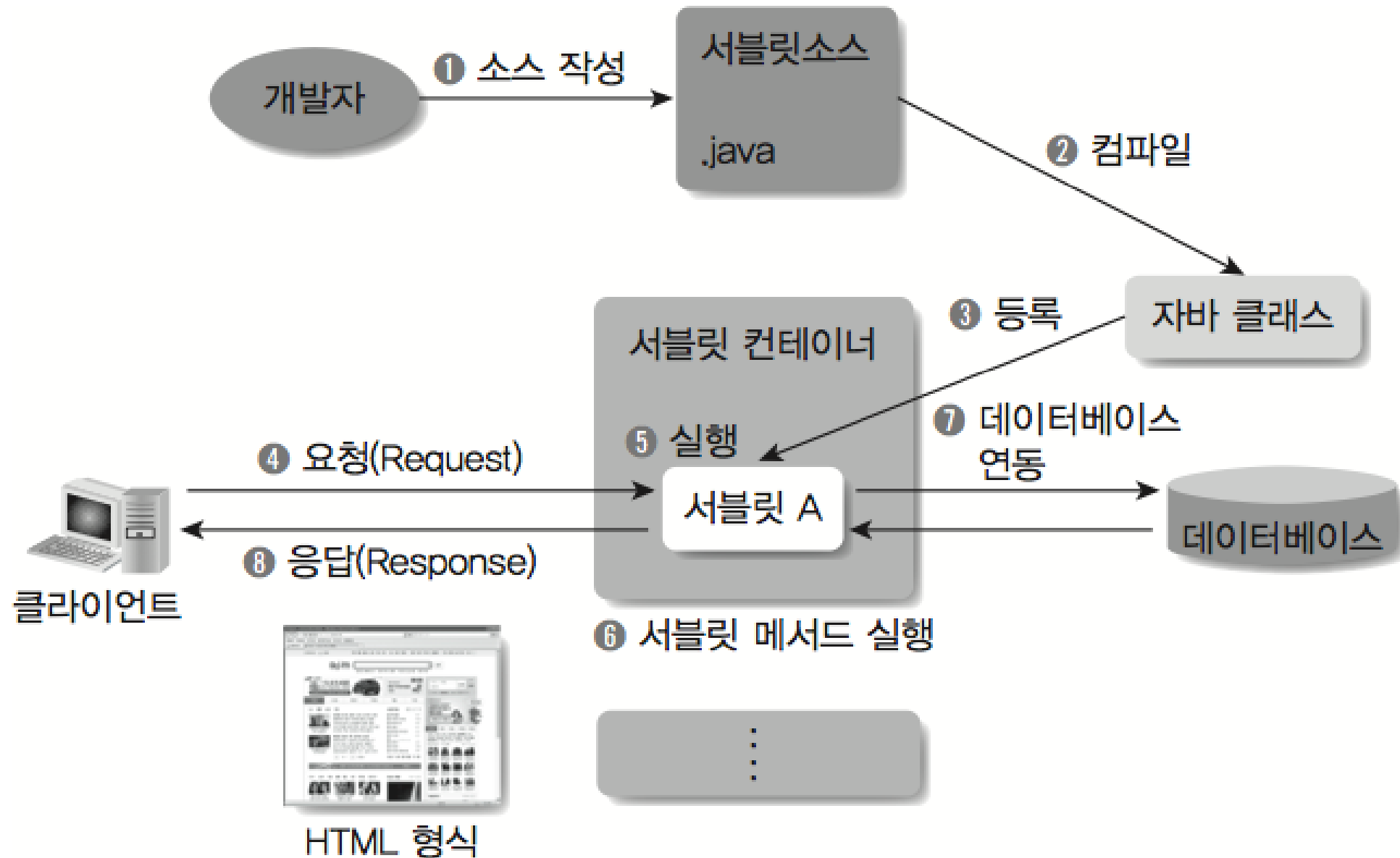
Servlet 이란?

- 서버에서 실행되면서 **클라이언트**의 요청에 따라 동적으로 서비스를 제공하는 **자바 클래스**
 - 톰캣과 같은 **Servlet 컨테이너**에서 실행됨
 - 일반적으로 **HTTP** (HyperText Transfer Protocol)를 통해 웹 클라이언트의 요청을 수신하고 응답
- Servlet 관련 객체
 - javax.servlet.**Servlet**
 - javax.servlet.**GenericServlet**
 - javax.servlet.http.**HttpServlet**

관련 자료

<https://jcp.org/aboutJava/communityprocess/final/jsr369/index.html>

Servlet 동작과정



Servlet 컨테이너(WAS)

- Servlet 을 실행하기 위한 **실행 환경**
 - 아파치 톰캣이 대표적임
- 웹서버와 **Servlet** 컨테이너(WAS) 비교

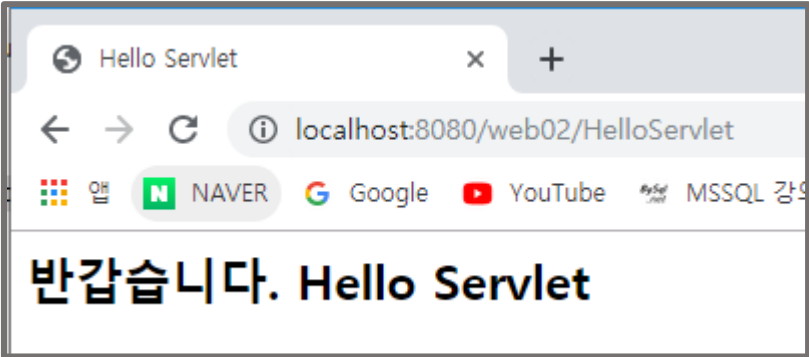
구분	웹 서버	서블릿 컨테이너 웹애플리케이션서버 (WAS)
사용목적	웹 서비스를 제공하기 위해 필요한 서버 기반의 소프트웨어다.	서블릿으로 개발된 자바 프로그램을 실행하고 처리하기 위한 서버 기반의 소프트웨어다.
처리 콘텐츠	HTML, CSS, 자바스크립트, 이미지 파일 등이다.	서블릿 클래스다.
실행 방법	콘텐츠가 위치한 URL 요청에 의해 실행하며 요청할 때마다 매번 디스크에서 읽어 처리한다.	서블릿 클래스 정보에 따라 서버에 매핑된 URL 정보에 따라 실행하며 컨테이너에 적재된 상태에서 처리한다.
JSP 실행	자체로 처리할 수 없다. <u>서블릿 컨테이너로 처리를 넘긴다.</u>	JSP 자체로 처리할 수 있다.
특징	웹 서비스 제공을 위한 다양한 설정을 제공하기 때문에 서버를 유연하게 운영하려면 웹 서버를 사용해야 한다.	컨테이너에 따라 기본적인 웹 서버 기능을 내장하고 있으나 고급 설정이나 성능이 떨어지기 때문에 웹 서버와 병행해서 사용할 것을 권장한다.

실습1 – HelloServlet 작성하기

- 'Hello, World!'를 출력하는 서블릿 생성

실습1 – HelloServlet 작성하기

```
1 package cs.web02;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 /**
11  * Servlet implementation class HelloServlet
12  */
13 @WebServlet("/HelloServlet")
14 public class HelloServlet extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
19      */
20     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
21         //컨텐츠 타입 선언 및 한글 설정
22         response.setContentType("text/html;charset=UTF-8");
23         //웹브라우저 출력을 위한 PrintWriter 객체 정의
24         PrintWriter out = response.getWriter();
25     /**
26      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
27      */
28     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29         out.println("<html>");
30         out.println("<head><title>Hello Servlet</title></head>");
31         out.println("<body><h2>반갑습니다. Hello Servlet</h2></body>");
32         out.println("</html>");
33     }
34 }
```



반갑습니다. Hello Servlet

서블릿을 변경했을 경우에는 반드시 서버를 다시 실행해야 함

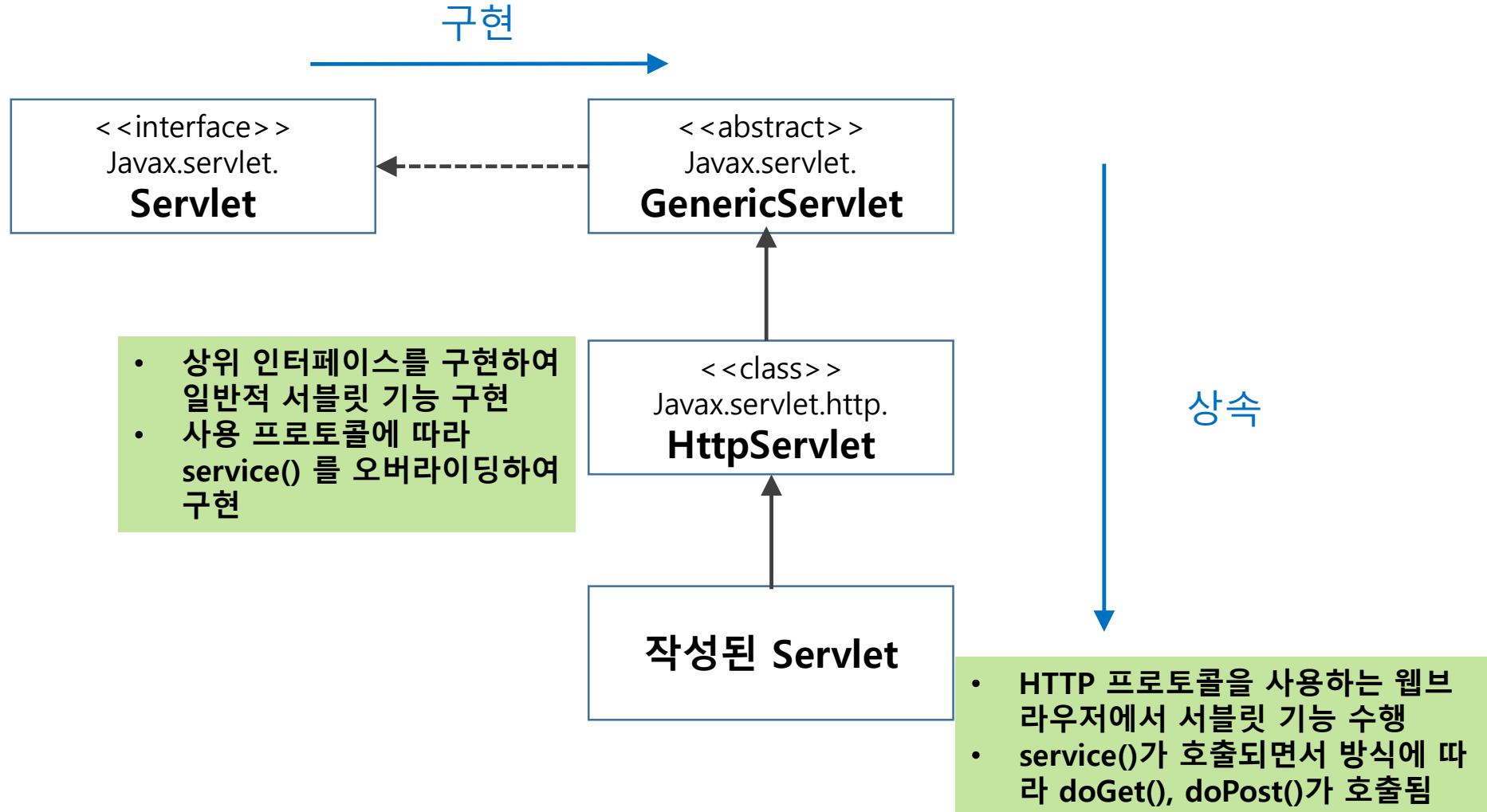
Servlet 특징

- 일반적인 자바 클래스 구조
- 컴파일 과정 필요
- Servlet 컨테이너에 의해 실행
- 특정 클래스(**Servlet**)를 상속 받아야만 구현할 수 있는 구조
- Servlet 클래스의 상관 관계나 **API의 기본 구조**를 이해해야 함
- 일반적으로 Servlet 은 javax.servlet.http.**HttpServlet** 클래스를 상속하여 구현

API(Application Programming Interface) : 특정 클래스를 다른 프로그램에서 사용하기 위해 필요한 정보를 규격화 해놓은 것(.jar 파일)

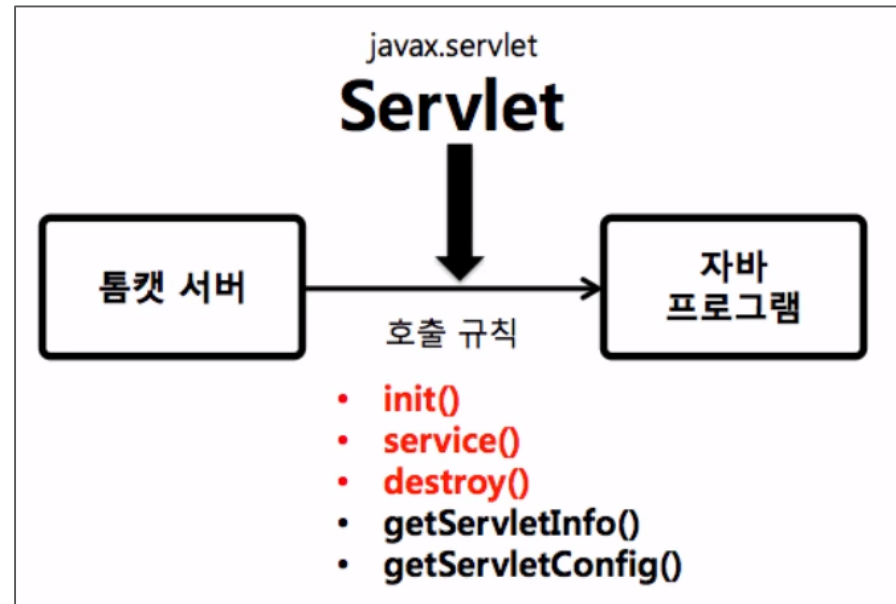
Servlet 클래스의 상속/구현 관계도

- tomcat9/lib/servlet-api.jar 패키지에 있음



Servlet 인터페이스(최상위 클래스)

- Servlet 인터페이스의 메소드
 - **init()** : 서블릿 초기화
 - **service()** : 클라이언트의 모든 호출 처리
 - **destroy()** : 서블릿 서비스가 중단 된 후 Garbage 수집 등을 완료
 - **getServletConfig()** : 서블릿의 시작 정보를 가져 오는 데 사용
 - **getServletInfo()** : 서블릿의 작성자, 버전 및 서버와 같은 자체에 대한 기본 정보 제공

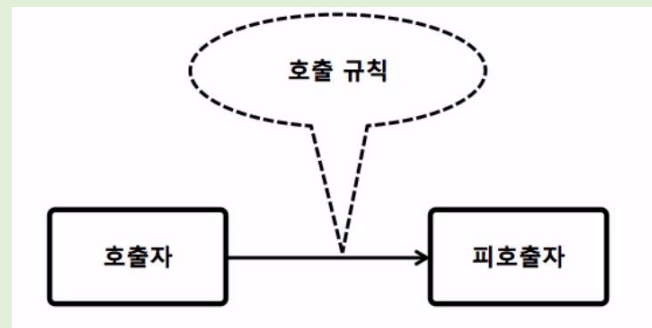


Servlet 인터페이스

```
1 package cs.web02;
2
3 import java.io.IOException;
4
5 import javax.servlet.Servlet;
6 import javax.servlet.ServletConfig;
7 import javax.servlet.ServletException;
8 import javax.servlet.ServletRequest;
9 import javax.servlet.ServletResponse;
10
11 public class MyServlet1 implements Servlet{
12
13     @Override
14     public void destroy() {
15         // TODO Auto-generated method stub
16     }
17
18     @Override
19     public ServletConfig getServletConfig() {
20         // TODO Auto-generated method stub
21         return null;
22     }
23
24     @Override
25     public String getServletInfo() {
26         // TODO Auto-generated method stub
27         return null;
28     }
29
30     @Override
31     public void init(ServletConfig arg0) throws ServletException {
32         // TODO Auto-generated method stub
33     }
34
35     @Override
36     public void service(ServletRequest arg0, ServletResponse arg1) throws ServletException, IOException {
37         // TODO Auto-generated method stub
38     }
39 }
```

인터페이스(interface)

- 호출자와 호출당하는 것 사이의 호출 규칙 정의에 사용되는 문법
- Implements 키워드로 구현
- 인터페이스를 구현하는 클래스는 반드시 인터페이스에 선언된 모든 메서드를 구현해야 함



GenericServlet

- Servlet 인터페이스를 구현한 **추상 클래스**
 - GenericServlet은 **service()**를 제외한 4개의 메소드를 미리 구현해 두어 서브 클래스가 **service()**만을 구현할 수 있도록 함
 - GenericServlet 클래스는 **프로토콜에 무관한** 기본 서비스만 제공하는 클래스
 - **HttpServlet**과 같은 프로토콜 특정 서브 클래스를 확장하는 것이 더 일반적

추상 클래스(abstract class)

- 하나 이상의 추상 메소드를 포함하는 클래스
- 직접 인스턴스를 만들어 사용하는 것이 아니라 서브 클래스에게 상속해주는 용도의 클래스
- **공통 속성**이나 **공통 메소드**를 가지고 있는 클래스로서 자체가 사용되기 보다는 서브 클래스에서 상속해주는 용도로 사용되는 클래스로 정의하고자 할 때 추상클래스를 정의

GenericServlet

- GenericServlet 을 상속하는 MyGenericServlet 클래스

```
1 package cs.web02;
2
3 import java.io.IOException;
4
5 import javax.servlet.GenericServlet;
6 import javax.servlet.ServletException;
7 import javax.servlet.ServletRequest;
8 import javax.servlet.ServletResponse;
9
10 public class MyGenericServlet1 extends GenericServlet {
11
12     @Override
13     public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
14         // TODO Auto-generated method stub
15
16     }
17 }
```

@Override 애노테이션

- 상위 클래스의 메소드를 override 하는 메소드임을 지정하여 잘못되지 않도록 방지

HttpServlet 클래스

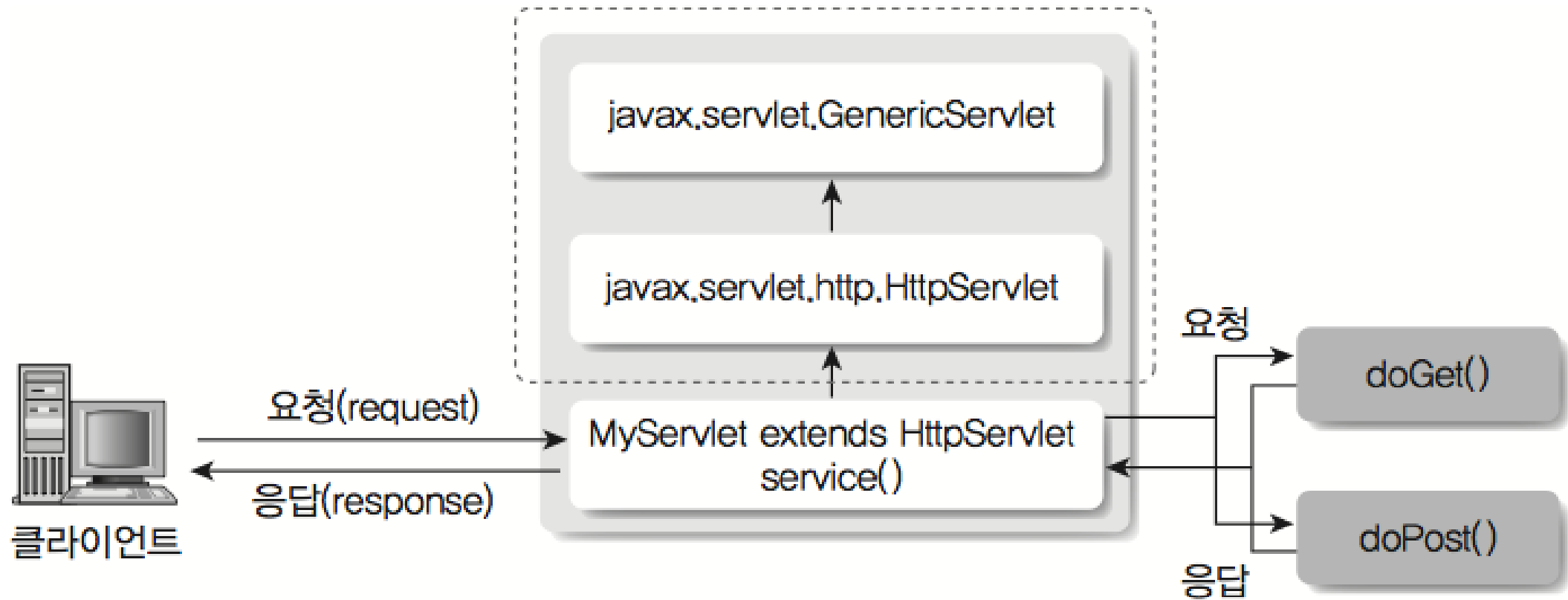
- GenericServlet 클래스 보다는 HttpServlet이 HTTP 프로토콜 지원이 포함되어 일반적 **웹 프로그램에 적합**
- HttpServlet의 서브클래스는 적어도 다음 중 하나의 메소드를 작성해야 함
 - request의 종류에 따라 doXXX() 메서드를 호출
 - **doGet() : get 요청 시**
 - **doPost() : post 요청 시**
 - **service() : 클라이언트의 요청 시**
 - doPut()
 - doDelete()
 - doHead()
 - doPut()
 - doOptions()
 - doTrace()
 - getLastModified()

Servlet 작성 시 규칙

- 서블릿 클래스는 javax.servlet.http.**HttpServlet** 클래스를 상속
- doGet 또는 doPost 메서드 내에 요청 처리 내용을 기술
- HTML 문서는 doGet, doPost 메소드의 response 파라미터를 이용하여 출력

Servlet 동작구조

- javax.servlet.http.HttpServlet 을 상속 받은 MyServlet의 동작 구조



파라미터는 항상 request와 response

클라이언트 요청 방식

• GET 방식

- 서버정보를 클라이언트로 가져오기 위한 방법
- HTML, 이미지 등을 웹브라우저에서보기 위해 요청하는 방식
- 서버에 최대 240Byte 전달
- QUERY_STRING 환경변수를 통해 서버로 전달
 - QUERY_STRING : 파일명?속성1=속성값1&속성2=속성값2...
- URL 노출로 보안 문제가 생길 수 있다.

• POST 방식

- 서버로 정보를 보내기 위해 설계된 방법
- HTML 폼에 입력한 내용을 서버로 전달
- 서버 전달 데이터크기에 제한 없음
- URL에 매개변수가 표시되지 않음

테스트

- HttpServlet 클래스를 상속한 MyHttpServlet 클래스를 eclipse에서 작성하기

```
1 package cs.web02;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 /**
11  * Servlet implementation class MyHttpServlet
12  */
13 @WebServlet("/MyHttpServlet")
14 public class MyHttpServlet extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
19      */
20     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
21         // TODO Auto-generated method stub
22         response.getWriter().append("Served at: ").append(request.getContextPath());
23     }
24
25     /**
26      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
27      */
28     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29         // TODO Auto-generated method stub
30         doGet(request, response);
31     }
32
33 }
```

doPost()로 요청이 와도 doGet()호출

```

1 package cs.test;
2
3 import java.io.IOException;
10
11 @WebServlet("/Calculator")
12 public class Calculator extends HttpServlet{
13
14     private static final long serialVersionUID = 1L;
15
16 @Override
17     protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
18
19         if(req.getMethod().equals("GET")) {
20             System.out.println("GET 요청입니다.");
21         }
22         else if(req.getMethod().equals("POST")) {
23             System.out.println("POST 요청입니다.");
24         }
25         super.service(req, resp); //부모의 service() 함수를 호출
26                                     //부모 service() 함수는 사용자 요청에 따른 doGet 또는 doPost 함수를 호출한다.
27     }
28 @Override
29     protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
30         System.out.println("POST 요청입니다.-doPost");
31     }
32 @Override
33     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
34         System.out.println("GET 요청입니다.-doget");
35     }
36 }

```


Servlet 생명 주기(Life Cycle)

- Servlet 생명주기 메서드 : Servlet 실행 단계마다 호출되어 기능을 수행하는 메서드

- Servlet 초기화 : **init()** 메서드

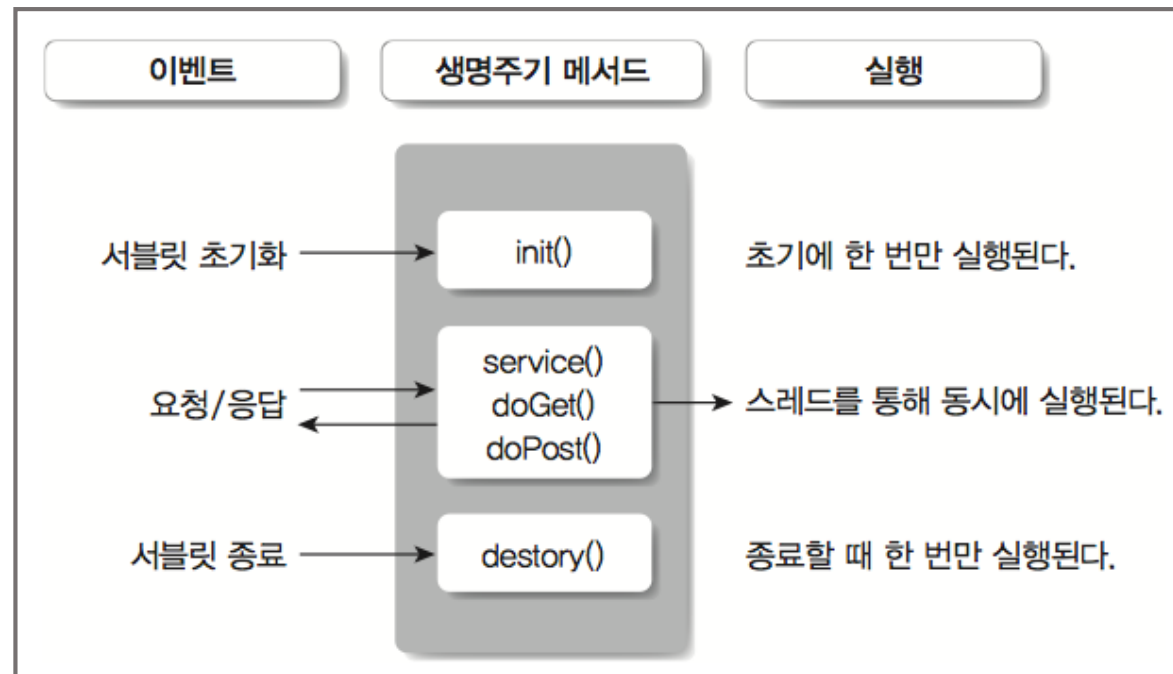
- Servlet 실행 시 호출되는 메서드
- 초기에 한 번만 실행(Java의 main())와 같음

- 요청/응답 : **service()/doGet()/doPost()** 메서드

- 사용자 요청에 따라 스레드로 실행되는 메서드
- service() 를 통해 doGet() 혹은 doPost() 메서드 호출
- 매개변수 HttpServletRequest 와 HttpServletResponse 를 통해 사용자 요청을 처리

- Servlet 종료 : **destroy()** 메서드

- 컨테이너로부터 Servlet 종료 요청이 있을 때 호출
- init()와 마찬가지로 한 번만 실행
- Servlet 종료 시 정리 작업을 destroy() 를 재정의해 구현



브라우저 출력

- 클라이언트에서 출력될 수 있도록 **response** 객체에 정보 저장
- 출력 스트림 객체 반환

PrintWriter **out** = **response.getWriter()**:

```
//웹브라우저 출력을 위한 PrintWriter 객체 정의
PrintWriter out = response.getWriter();

out.println("<html>");
out.println("<head><title>");
out.println("Hello Servlet");
out.println("</title></head>");
out.println("<body>");
out.println("<h2>반갑습니다. Hello Servlet</h2>");
out.println("</body></html>");
```

Servlet 배포(Deployment) 방법

- Servlet 매핑 정보 설정 이유
 - 서블릿 저장 위치 (톰캣 환경 기준) : /WEB-INF/classes/
 - WEB-INF 파일은 외부에서 접근 불가능
 - 클라이언트에서 WEB-INF 내에 존재하는 파일을 호출하기 위해서는 매핑 정보를 이용
- Servlet 매핑 방법
 1. **web.xml** 파일에 배치 정보 설정
 2. **Annotation** 으로 배치 정보 설정(Servlet 3.0 부터 적용, tomcat7.0 이상)

1. web.xml 에 서블릿 배포 정보 설정

- 프로젝트의 설정 파일인 web.xml 에 서블릿 관련 정보 저장

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://ja
3   <display-name>web02</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12 </web-app>
```

Welcome 파일 정보 설명하기

```
5 <servlet>
6   <servlet-name>hello</servlet-name>
7   <servlet-class>cs.web02.HelloServlet</servlet-class>
8 </servlet>
9
10 <servlet-mapping>
11   <servlet-name>hello</servlet-name>
12   <url-pattern>/hello</url-pattern>
13 </servlet-mapping>
```

1. web.xml 에 서블릿 배포 정보 설정

- 여러 개의 Servlet 을 등록할 경우 Servlet 등록과 Servlet 매핑에 각각 정리한다.
- <servlet-name>은 프로젝트 내에서 유일 해야 한다.

```
<servlet>
  <servlet-name>hello1</servlet-name>
  <servlet-class>cs.dit.controller.HelloServlet1</servlet-class>
</servlet>

<servlet>
  <servlet-name>hello2</servlet-name>
  <servlet-class>cs.dit.controller.HelloServlet2</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>hello1</servlet-name>
  <url-pattern>/h1</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>hello2</servlet-name>
  <url-pattern>/h2</url-pattern>
</servlet-mapping>
```

2. Annotation - @WebServlet()

- 클래스나 메소드에 붙여지는 주석으로 **메타 데이터**라고도 함
- servlet-api.jar 버전이 3.0 이상에서 사용 가능
- @WebServlet의 속성 값을 통해 해당 Servlet과 매핑 될 URL 패턴을 지정
- 가독성, 소스코드 구성에 도움 줌
- 코드가 실행되는 것은 아니나 연결 방법, 소스 구조를 변경할 수 있음

• @WebServlet()의 주요 속성

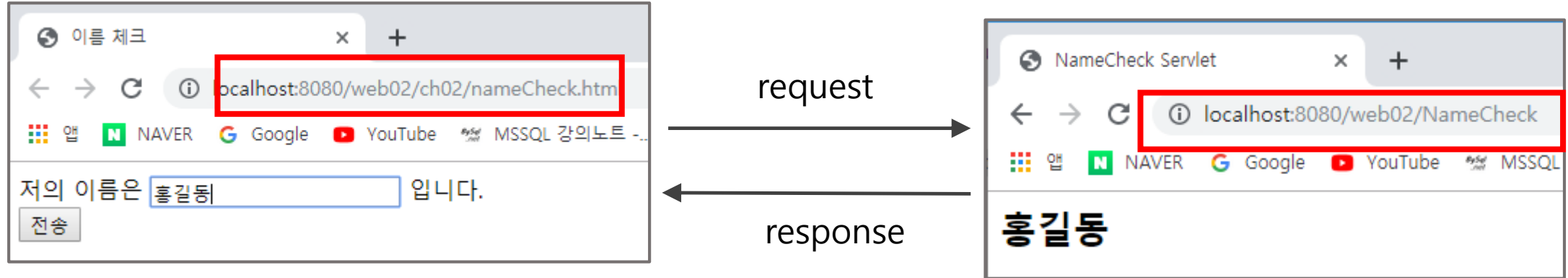
속성	기능	사용 예
name	<ul style="list-style-type: none"> 서블릿 이름 설정 기본 값은 ""(빈문자열) 	@WebServlet(name = "서블릿 이름")
urlPatterns	<ul style="list-style-type: none"> 서블릿 URL 목록 설정 속성값으로 String 배열 	@WebServlet(urlPatterns = "/url") @WebServlet(urlPatterns = {"/url", "/url2", "/url3"}) @WebServlet(urlPatterns = {"/", "/home", "/webcome"})
value	<ul style="list-style-type: none"> urlPatterns와 동일한 기능 속성이름 없이 값만 설정 가능 value 속성 외에 다른 속성의 값도 필요한 경우는 속성의 이름을 생략할 수 없음 	@WebServlet(value = "/calc") @WebServlet("/calc") @WebServlet(urlPatterns="/url", name="hello")

실습1

- annotation @WebServlet()으로 실행하기
- web.xml로 실행하기

과제1 – NameCheck 작성하기

- 서블릿 NameCheck를 화면과 실행으로 분리하여 작성

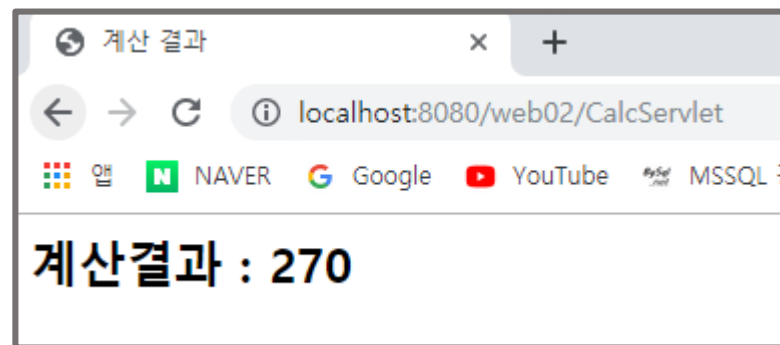


과제2- 간단한 계산기 만들기

- calculator.html 작성
- CalcServlet 작성
- 배포
 - web.xml 사용
- 실행

계산기

첫 번째 값 : + 두 번째 값 : 계산 재입력



과제3

- CGI 규칙과 CGI 프로그램을 설명하시오.
- 서블릿 API에서 서블릿을 작성하기 위해 필요한 객체 3개를 열거하고 구조를 설명하시오.
- 서블릿을 배치하고 매핑하는 방법 두가지를 설명하시오.
- 서블릿을 작성하고 배치한 뒤 실행하는 예제를 구현하시오.

도전 과제(옵션)

- 실습3의 내용에서 계산 부분만 다른 클래스에서 구현하기
 - calculator2.html : 화면
 - CalcServlet2 : 서블릿으로 Calc 객체 사용
 - Calc.java : 계산 코드
 - Web.xml 또는 annotation 으로 서블릿 배치