

Hanbit eBook

Realtime 18

# JSP 바이블

STEP 01

## JSP 시작과 개발환경 구축

조효은 지음



# JSP 바이블

STEP 01: JSP 시작과 개발환경 구축

지은이\_ **조효은**

알고리즘/최적화를 이용한 게임 엔진, 증권/주식 분석 엔진, SOA 서비스 구현 경험을 바탕으로 프로그램 언어, 아키텍처, CBD, SOA, 웹 프로그래밍, 안드로이드 등 실무 위주의 강의를 진행하고 있다.

(현) 프리랜서

(전) 한경닷컴 수석 강사/차장

(전) 한국정보기술연구원 OOP/CBD 선임 연구원

(전) ORACLE SOA 강사

(전) 씬마이크로 시스템즈 공인 강사

## JSP 바이블 STEP 01: JSP 시작과 개발환경 구축

---

초판발행 2013년 2월 28일

지은이 조효은 / 펴낸이 김태현

펴낸곳 한빛미디어(주) / 주소 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부

전화 02-325-5544 / 팩스 02-336-7124

등록 1999년 6월 24일 제10-1779호

ISBN 978-89-6848-601-2 15560 / 비매품

책임편집 배용석 / 기획·편집 스마트미디어팀

디자인 표지 여동일, 내지 스튜디오 [맘], 조판 김현미

마케팅 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오타 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.

한빛미디어 홈페이지 [www.hanb.co.kr](http://www.hanb.co.kr) / 이메일 [ask@hanb.co.kr](mailto:ask@hanb.co.kr)

---

Published by HANBIT Media, Inc. Printed in Korea

Copyright © 2013 조효은 & HANBIT Media, Inc.

이 책의 저작권은 조효은과 한빛미디어(주)에 있습니다.

저작권법에 의해 보호를 받는 저작물이므로 무단 복제 및 무단 전재를 금함

---

지금 하지 않으면 할 수 없는 일이 있습니다.

책으로 펴내고 싶은 아이디어나 원고를 메일([ebookwriter@hanb.co.kr](mailto:ebookwriter@hanb.co.kr))로 보내주세요.

한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

# 대상 독자 및 사용 틀

초급

초중급

중급

중고급

고급

---

- JDK 7.X : <http://www.oracle.com/technetwork/java/index.html>
- Apache Tomcat : <http://tomcat.apache.org/>
- 이클립스 : <http://www.eclipse.org>

# 한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook 입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

## 1. eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.

500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

## 2. 무료로 업데이트되는, 전자책 전용 서비스입니다.

종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

### 3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 내가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

### 4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

지금 보시는 전자책에 소유권한을 표시한 문구가 없거나 타인의 소유권한을 표시한 문구가 있다면 위법하게 사용하고 계실 가능성이 높습니다. 이 경우 저작권법에 의해 불이익을 받으실 수 있습니다.

다양한 기기에 사용할 수 있습니다. 또한 한빛미디어 사이트에서 구입하신 후에는 횡수에 관계없이 다운받으실 수 있습니다.

한빛미디어 전자책은 인쇄, 검색, 복사하여 붙이기가 가능합니다.

전자책은 오타자 교정이나 내용의 수정보완이 이뤄지면 업데이트 관련 공지를 이메일로 알려드리며, 구매하신 전자책의 수정본은 무료로 내려받으실 수 있습니다.

이런 특별한 권한은 한빛미디어 사이트에서 구입하신 독자에게만 제공되며, 다른 사람에게 양도나 이전되지 않습니다.

# 차례

01	<b>JSP 학습 범위와 개념 정리</b>	1
<hr/>		
	1.1 이 책에서 다룰 JSP 범위.....	1
	1.2 이 책에서 다루는 JSP 수준.....	2
	1.3 이 책에서 다루는 JSP 내용.....	3
	1.3.1 JSP 구성요소.....	4
	1.3.2 JSP 관련 기술.....	5
	1.3.3 JSP에서 사용되는 기본 문법.....	6
	1.4 웹 용어.....	7
02	<b>JSP 실행환경</b>	32
<hr/>		
	2.1 JEE 버전과 발표연도.....	32
	2.1.1 연도별 개발 유형.....	34
	2.2 설치와 환경설정.....	37
	2.2.1 JDK 7.X 설치와 환경설정.....	37
	2.2.2 Tomcat 7.X 설치와 환경설정.....	45
	2.2.3 이클립스 4.X 설치와 환경설정.....	49
	2.2.4 Tomcat과 이클립스 연동.....	52
	2.3 이클립스 활용방법.....	60
	2.3.1 JSP 작성하기.....	60
	2.3.2 JSP의 useBean 액션 태그 만들기.....	6

---

3.1 컨텍스트 .....	68
3.2 경로 .....	70
3.3 정적문서와 동적문서 .....	71
3.4 JSP 스크립트 요소 .....	79

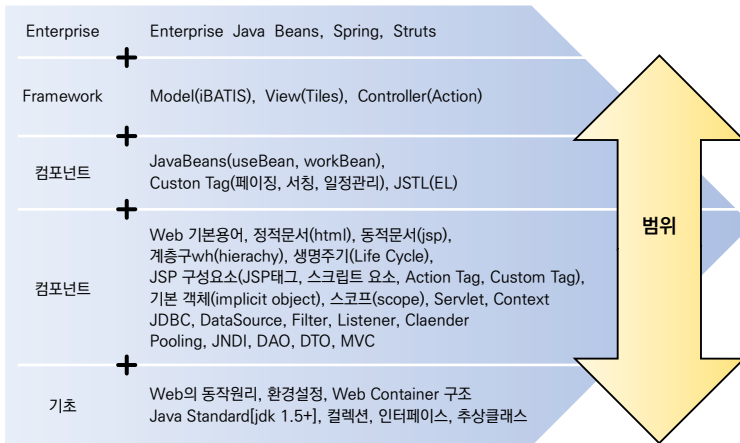


# 1 | JSP 학습 범위와 개념 정리

## 1.1 이 책에서 다룰 JSP 범위

JSP는 기초부터 JSP 기본, 컴포넌트, 프레임워크, 그리고 엔터프라이즈 수준까지 폭 넓게 활용할 수 있지만 이 책에서는 기초, JSP 기본, 컴포넌트, 프레임워크 (iBatis, Tiles, Log4j)까지 다룬다. JSP 역시 기초가 중요하므로 웹의 동작원리, 환경설정, 컨테이너 구조, 반드시 알아야 할 웹 용어부터 시작한다. 그리고 이를 기반으로 JSP 구성요소와 JDBC를 익혀서 자주 사용하는 부분을 웹 컴포넌트화하여 개발 속도를 높이는 방법도 익힌다. 또한 기존 프레임워크를 이용하거나 프레임워크를 만들어서 개발 방법을 통일화하고 의도한 대로 개발하는 방법도 배워보자.

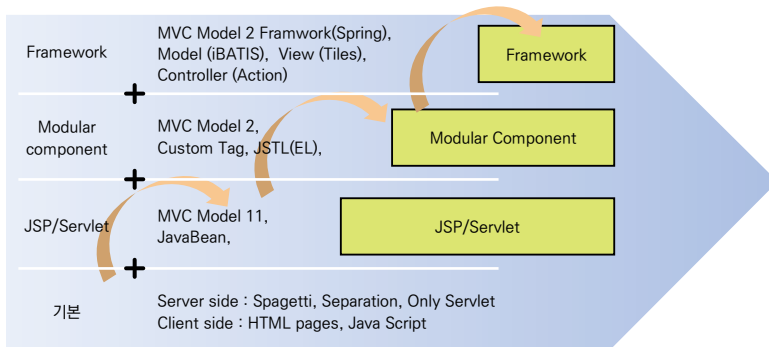
그림 1-1 JSP 범위



## 1.2 이 책에서 다루는 JSP 수준

JSP 사용범위가 넓어질수록 복잡도도 높아진다. 처음에는 HTML 소스와 JAVA 코드가 섞여있는 스파게티 소스로 만들어볼 수 있다. 그 다음에는 세퍼레이션 소스(자바빈 또는 유즈빈)를 사용(MVC Model 1)한 예제소스와 프로젝트도 만나게 된다. 여기에서 그치지 않고 MVC Model 2에서 JSP 중심(센트릭)으로 작성하는 방법, 서블릿 중심으로 작성하는 방법, 멀티 서블릿, 액션 서블릿으로 발전시켜 MVC Model 2 기반 프레임워크(스피링, 스트러츠) 원리까지도 자연스럽게 알 수 있다. 모듈화(화면 분할) 방법은 프레임셋(Frameset), 액션 인클루드(Action Include), 타일즈(Tiles 프레임워크)도 다룬다. 이 책으로 학습하면 JDBC를 직접 사용하는 스파게티부터 DAO까지 사용할 수 있을 뿐 아니라 iBATIS 프레임워크도 활용할 수 있다. 일반적인 JSP 책에서는 잘 다루지 않는 서비스 객체도 익힐 수 있으므로 기초부터 고급 기술로 발전하면서 다양한 기술을 터득할 수 있다.

그림 1-2 JSP 수준



## 1.3 이 책에서 다루는 JSP 내용

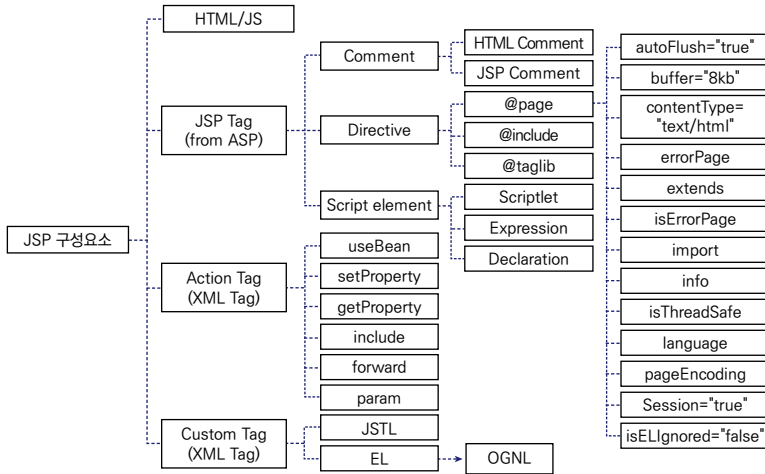
이 책에서 다루는 JSP 내용은 다음과 같다.

- 웹 기본 용어
- HTTP/HTML/JS
- 웹 서버Web Server/웹 컨테이너
- 요청/응답
- 기본 객체/구성요소/스크립트요소
- 주석
- 디렉티브
- 파라미터parameter와 파라미터 처리
- 서블릿(Servlet)/생명주기(라이프사이클life cycle)
- 서블릿 리스너/서블릿 필터
- XML
- Custom Tag/JSTL/EL
- 유즈빈useBean(자바빈JavaBean)/workBean
- 흐름제어/모듈
- DAOdata access object/DTOdata transfer object/iBATIS
- MVC(Model, View, Controller), MVC Model 1, Model 2
- 팩토리 패턴/싱글톤 패턴/헬퍼뷰 패턴
- 액션 서블릿/멀티 서블릿Multi Servlet
- JSP 웹 역사(Only Servlet, Spagetti, Separation, JSP Centric, Servlet Centric, Multi Servlet, Action Servlet, Framework)
- 일반게시판, 답변형 게시판, 페이징, 서칭, 일정관리, 입사지원관리, 인사관리

### 1.3.1 JSP 구성요소

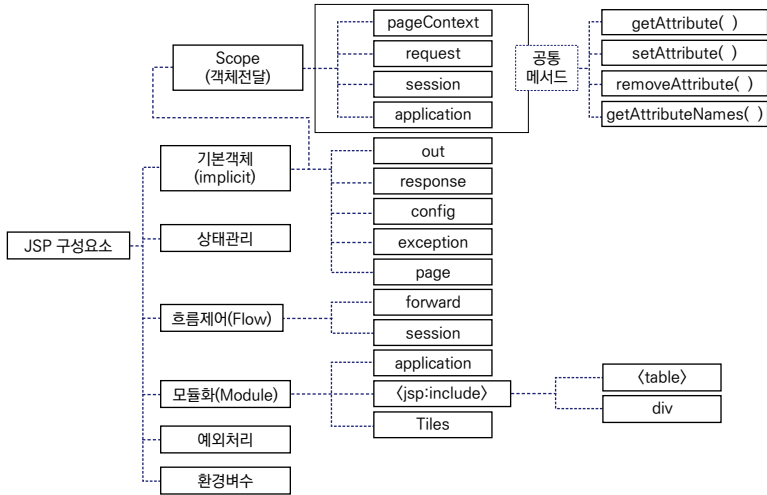
JSP 작성 시 사용하는 태그와 클래스 등을 JSP 구성요소라고 한다. 한 페이지의 특징을 설정하는 디렉티브directive, HTML사이에 자바코드를 넣어서 원하는 화면을 만드는 스크립트script요소, 스크립트요소를 줄이고 자바코드를 숨기면서 웹 프로그래밍 기능을 하는 액션action 태그 등이 JSP 기본 구성요소다.

그림 1-3 JSP 기본 구성요소



원래 한 페이지에서 다른 페이지로 객체를 전달할 수 없지만 전달하게 만드는 스코프scope 객체, 상태가 없는stateless JSP에서 상태를 만들어statefull 사용하는 세션session, 여러 화면을 보여주기 위한 모듈화, 객체를 만들지 않고도 이미 준비된 객체를 사용하는 기본 객체implicit object, 한 화면에서 다른 화면으로 이동하는 흐름 제어flow control, 예외 발생 시 처리 방법, 저장되어 있는 환경변수를 사용하는 방법 모두가 중요한 JSP 구성요소다.

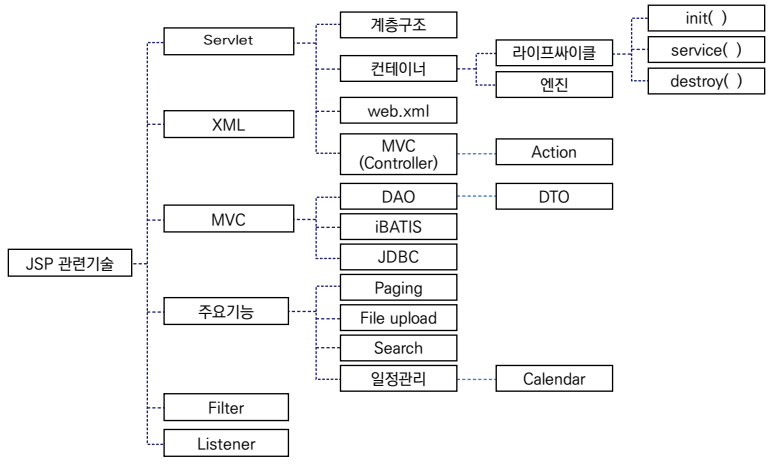
그림 1-4 JSP 중요 구성요소



### 1.3.2 JSP 관련 기술

JSP는 실행될 때 서블릿Servlet으로 변환되므로 서블릿을 아는 만큼 JSP를 알 수 있다. 서블릿은 웹 서버의 컨테이너에서 실행되기 때문에 컨테이너, 엔진, 생명주기 life cycle도 파악해야 한다. 또한 환경변수와 서블릿을 매핑시키는 web.xml도 필요하다. 세션을 추적하고 관리하는 세션 리스너session listener, 한 페이지에서 다른 페이지로 이동할 때 가로챌 수 있는 서블릿 필터servlet filter 등도 살펴본다. 한 페이지에 화면, DB관련 작업, 제어 로직이 섞인 스파게티를 각 용도에 맞게 MVCModel, View, Controller로 분할하는 방법과 Model의 발전 순서인 JDBC, DAO, iBATIS, 서비스 객체도 배운다.

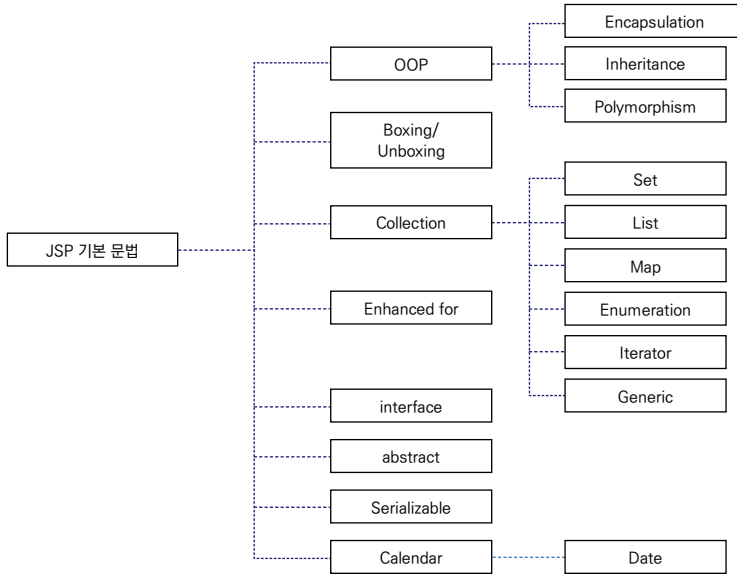
그림 1-5 JSP 관련 기술



### 1.3.3 JSP에서 사용되는 기본 문법

JSP는 적절하게 작동하도록 잘 만들어야 하는데, Java 기본 문법을 알면 많은 도움이 된다. JSP/Servlet은 인터페이스와 추상클래스로 이루어지므로 인터페이스와 추상클래스를 반드시 이해해야 한다. DAO의 반환타입은 주로 DTO와 List<DTO> 타입이므로 컬렉션을 사용할 수 있어야 한다. 이 책에서는 JSP에서 사용하는 기본 문법도 함께 설명한다.

그림 1-6 JSP 기본 문법



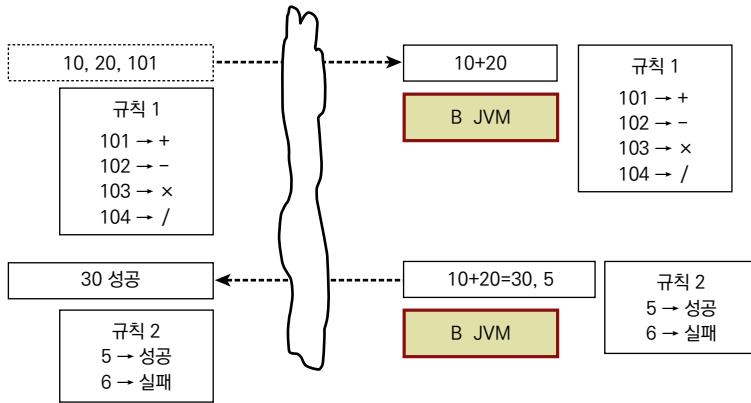
## 1.4 웹 용어

### ■ 서버Server와 클라이언트(Client)

내 컴퓨터를 이용하지 않고 다른 컴퓨터(B JVM)를 이용하여 문제를 해결한다고 가정해보자. 그리고 '10+20'을 구한다고 할 때 숫자와 '+', '만 사용하는 규칙을 만들었다고 해보자. 10+20을 10, 20, 101로 만들어 다른 컴퓨터(B JVM)에 보내면, B에서는 규칙1을 보고 10+20의 값을 구하게 된다. 10+20=30을 해결한 후 규칙 2를 이용하여 30, 5를 A에게 돌려보내면 10+20은 30이라는 값이 도출되어 문제가 해결되었다는 사실을 알 수 있다.

문제를 해결해 달라고 요청하는 쪽이 클라이언트, 문제를 해결해서 돌려보내는 쪽이 서버다.

그림 1-7 서버와 클라이언트 구성



### ■ 요청(Request)/응답(Response)

클라이언트가 서버에게 문제를 해결해달라고 요구하는 행위는 요청(Request)이라고 한다. 반대로 서버가 문제를 해결해서 클라이언트에게 보여주는 행위는 응답(Response)이다. 따라서 웹 브라우저가 클라이언트이고, 웹 서버는 서버라 할 수 있다. 웹 브라우저를 이용하여 서버에게 한빛미디어 홈페이지를 요청하면 서버는 웹 브라우저에게 한빛미디어 홈페이지를 보여주도록 응답한다.

### ■ 프로토콜(Protocol, 규약, 규칙, 약속)

10+20을 10, 20, 101로 바꿔서 보내는 것과 10, 20, 101을 10+20으로 다시 바꾼 후 계산해 30을 얻는 과정은 전송 시의 문제를 제거할 수 있다. 같은 방법으로 10+20의 결과를 돌려보낼 때 30, 5를 사용하면 클라이언트 쪽에서는 규약을 지킨 연산결과가 30이고, 제대로 도출되었다는 점을 알 수 있다. 이처럼 클라이언트와 서버 간 전송을 위한 규약이 프로토콜이다. 또한 30, 5의 5처럼 서버 쪽에서 문제를 제대로 해결했는지 실패했는지에 따라 약속된 값을 클라이언트로 보낼 수 있는데 이를 응답 상태코드(status code)라고 한다.

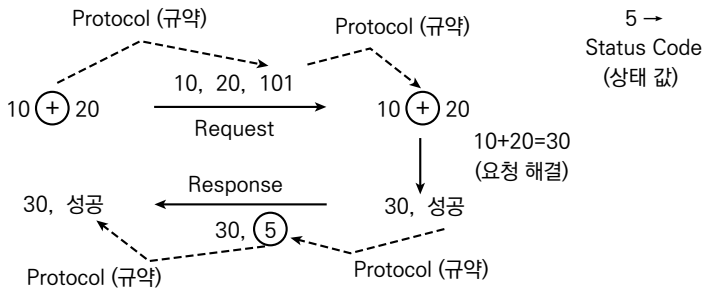


## ■ 웹 서버(Web Server)

웹 서버는 서버쪽 컴퓨터에 있는 소프트웨어다. 클라이언트의 요청을 받아서 웹 페이지(HTML, 그림파일, CSS, 자바스크립트 등으로 구성된 문서)를 클라이언트인 웹 브라우저에 응답하는 역할을 한다.

웹Web은 World Wide Web의 줄임말로 문서들이 인터넷으로 연결된 컴퓨터 세계를 말한다. 하이퍼텍스트hypertext는 글자뿐 아니라 그림처럼 보여줄 수 있는 내용물을 컴퓨터 사용자가 마우스나 키보드입력으로 요청할 때 바로 접근하고 사용할 수 있는 문서다. 웹 페이지는 웹에서 인터넷으로 연결되어 다른 웹 페이지로 자유롭게 이동navigate하고 연결되어야 하는 문서이므로 하이퍼텍스트로 만들어진다.

그림 1-8 자유로운 이동이 가능한 웹 페이지

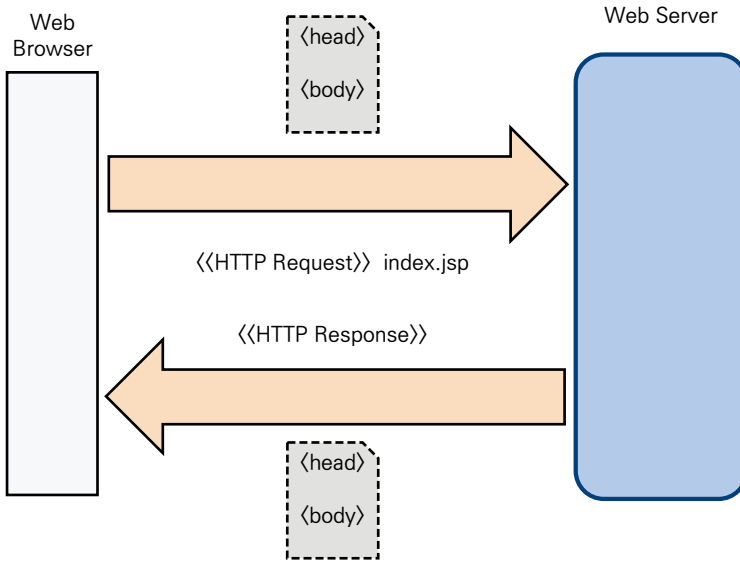


## ■ HTTP(Hypertext Transfer Protocol)

웹 서버에서 서버-클라이언트 사이에 대화(Request/Response)를 할 수 있도록 만든 규약이다. HTTP는 헤더header와 바디body로 구성된다. HTTP의 요청과 응답에는 메시지가 포함되어 요청과 응답에 대한 상태를 알 수 있게 한다. 서버는 요청 헤더 메시지를 읽고 클라이언트의 요청 사항을 파악한 다음 클라이언트에게 응답을 보낸다. 응답헤더에는 요청이 제대로 처리되었는지, 응답해주는 서버의 간단한 정보, 응답 내용의 타입 및 인코딩, 응답 크기 등이 포함된다. 응답헤더가 브라우저

에게 전달되면 응답바디 내용이 브라우저에 출력된다. 헤드와 바디 사이에는 한 줄이 비어있으므로 쉽게 구분할 수 있다.

그림 1-9 HTTP를 통한 통신



요청 메시지의 첫 줄에 요청라인request line이 있고(GET /pagecentric01\_comp01/index.jsp HTTP/1.1), 2번째 줄부터 다음과 같은 헤더내용이 있다.

```
Accept:image/jpeg, */*
Accept-Language: ko-KR
```

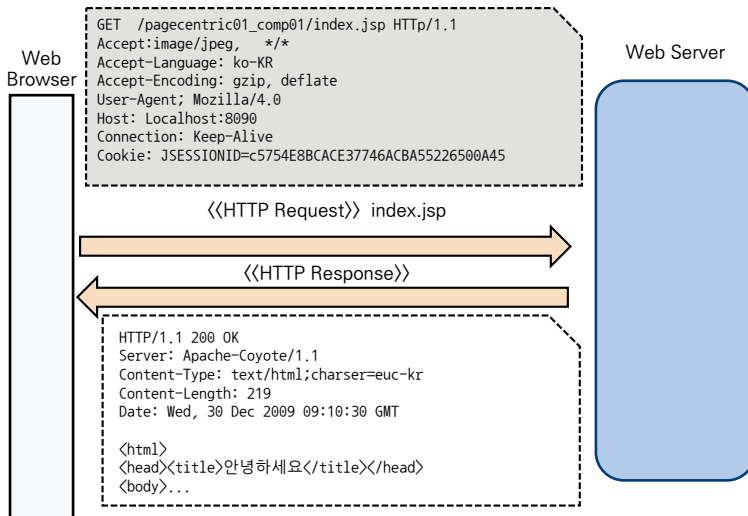
헤더가 끝나면 한 줄이 비어있다. 요청의 GET 방식은 바디가 없어서 한 줄이 비어있는 상태로 끝나지만, 포스트POST방식일 때 파라미터가 있다면 빈줄 밑에 파라미터가 들어간다. 응답 메시지는 'HTTP/버전 응답상태코드 상태'를 반환하고 다음 줄은 서버, 콘텐츠타입, 응답크기, 응답시간 등을 보여준다.

```

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=euc-kr
Content-Length: 219
Date: Wed, 30 Dec 2009 09:10:30 GMT

```

그림 1-10 응답 메시지



## ■ HTML(HyperText Markup Language)

HTML은 웹 페이지를 구성하는 핵심요소며, 마크업 언어 Markup Language는 문자를 이용하여 다른 문자의 특성을 표시하는 방법이다.

<b>안녕하세요</b>

<b></b>를 태그tag라 한다. 앞의 <b>는 시작태그, 뒤의 </b>는 ‘끝태그’라고 부르며, 태그사이의 ‘안녕하세요’는 콘텐츠다. <b>안녕하세요</b>와 같은 ‘시작태그-

컨텐츠-끝태그'를 엘리먼트<sup>element</sup>라고 한다. <b> 태그는 '안녕하세요'를 볼드체 boldface로 표현하라는 의미다. 다시 말하면 <b> 태그는 컨텐츠의 특성을 표시한다.

안녕하세요

웹 브라우저에서는 위와 같이 <b> 태그가 없어지고 볼드체 '안녕하세요'만 보여 준다. 웹 브라우저에서 컨텐츠를 표현(출력)하려고 태그를 이용하는 방법이 바로 HTML이다.

### ■ 상태코드(Status Code)<sup>01</sup>

클라이언트가 서버에게 요청하면, 서버는 요청을 처리한 다음 그 결과를 3자리 숫자로 된 상태코드와 함께 클라이언트에게 보내준다. 성공 200이외에는 서버 쪽에서 오류가 발생했다고 볼 수 있어 오류코드<sup>Error Code</sup>라고도 한다. '200 성공', '404 경로가 잘못됨', '500 서버쪽에서 문법적으로 예외발생'은 꼭 기억하자.

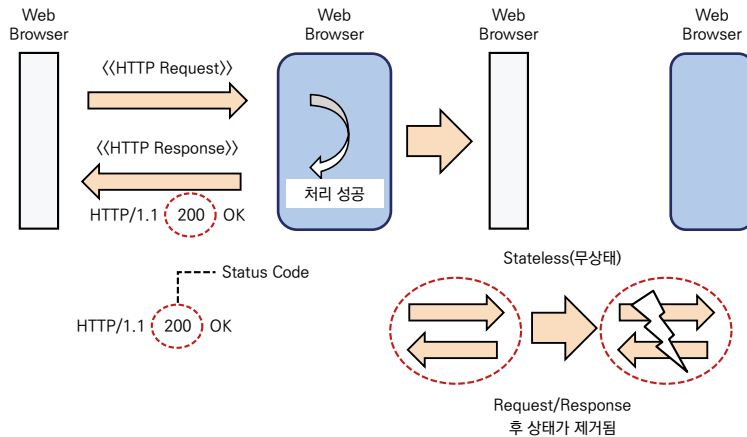
상태 코드	상태 메시지	상태 내용	발생 예
100	Continue	클라이언트로부터 일부분만 받았으니 나머지 요청정보를 요청	
200	OK	오류 없이 클라이언트로 전송 성공	성공
300	MultipleChoices	최근에 옮겨진 데이터를 요청	
404	Not Found	서버가 요청한 파일이나 스크립트를 찾지 못함	경로가 잘못되거나 없는 jsp/servlet을 호출
405	Method Not Allowed	메서드 허용 안됨	doGet()이 없는데 GET 방식으로 호출
500	Internal Server Error	서버 내부 오류	문법오류 등 서버코드에서 문제 발생

01 · 상태코드 관련 웹 사이트: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

## ■ 상태(State) - 무상태/유상태(Stateless/Statefull)

HTTP를 통한 클라이언트와 서버 사이의 대화는 방금 전 대화를 기억하지 못하는 ‘무상태stateless’다. 서버는 웹 브라우저를 통해 받은 요청에 응답한 다음 연결을 끊는다. 이 상태에서 클라이언트가 서버에게 요청을 하려면 서버에 다시 연결해야 한다. 반면 DB 서버는 클라이언트와 나눈 대화를 기억(유상태, statefull)한다. 대화가 끝난 후 다시 연결하지 않아도 요청을 할 수 있다. DB 서버가 대화를 끝내려면 명시적(close 호출)으로 끝내야 한다. HTTP Session을 무상태로 한 이유는 많은 클라이언트들의 요청에 의한 웹 서버의 과부하를 방지하기 위한 것이다. 그러나 DB 서버는 사용자가 제한적이므로 연결을 유지하는 편이 좋다.

그림 1-11 상태 연결



## ■ HTTP 요청 메서드

웹 서버나 웹 애플리케이션 서버는 서버-클라이언트 사이의 요청/응답용 프로토콜로 HTTP를 지원한다. HTTP는 헤드와 바디로 나뉘며, 전송데이터를 보내는 방법에 따라 요청 메서드 종류가 달라진다. JSP/Servlet은 GET, POST를 주로 사용하고 HEAD는 가끔 사용한다.

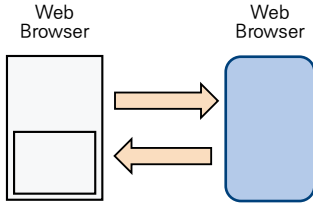
표 1-1 HTTP 주요 요청 메서드

요청 메서드	설 명
GET	서버에 요청 메서드를 보낼 때 헤드에만 내용이 있고 바디에는 전송데이터가 없다. 요청을 간단하게 하기 위한 것으로 헤드에 포함되는 내용의 크기도 제한적이다. 또한 URL에 전송데이터가 노출될 수 있다. HTTP Server는 GET과 HEAD를 반드시 지원해야 한다.
POST	서버에 보낼 전송데이터를 바디에 넣어서 요청한다. <FORM>을 이용하여 전송데이터를 서버로 보낼 때 사용한다. 전송데이터 크기에 제약이 없다.
HEAD	응답용으로 사용되며 헤드에만 내용이 있다. 바디 없이 헤드정보, 캐시정보를 클라이언트에 보낼 때 사용한다.

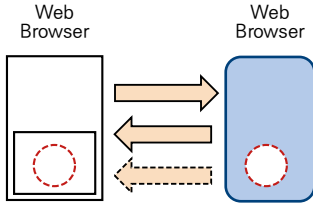
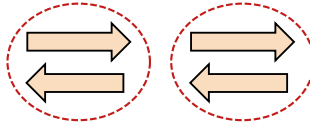
### ■ 동기 / 비동기(Synchronous / Asynchronous) 전송

요청 후 응답이 올 때까지 다른 요청을 할 수 없는 전송을 동기라고 한다. 새로운 요청을 하면 한 페이지 전체를 다시 불러온다. 응답할 때마다 페이지 전체를 다시 불러오고 프로그레스바가 작동한다. 요청한 다음 응답을 받아 화면을 보여주면서 백그라운드에서 서버로부터 데이터를 받는 등 다른 작업을 할 수 있는 전송상태를 비동기라고 한다. 비동기 상태에서 서버는 필요한 데이터만 전송되도록 응답한다. 1991년부터 비동기를 사용하기 시작했고, 1995년 이후 AJAX<sup>A</sup>Asynchronous Javascript And XML라는 이름으로 많이 사용하고 있다.

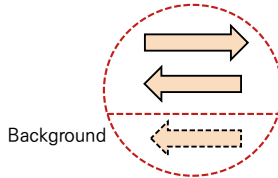
그림 1-12 새로운 요청에 대한 응답



Request 후 Response가 올때까지 다른 요청을 할 수 없다. : Synchronous  
 새로운 요청을 하면 응답으로 전체 페이지가 불러진다.



Request 후 Response가 오는 도중에도 다른 요청을 할 수 있다. : Asynchronous  
 현재 보여주는 화면(응답)을 방해하지 않고 서버로부터 데이터를 받을 수 있다.



### ■ 파라미터(parameter, 쿼리스트링)

클라이언트에서 서버로 요청을 할 때 요청경로 이외에도 데이터를 보낼 수 있다. 서버로 보내는 데이터는 '키=밸류' 형태로 전송된다. 예를 들어 'command=detail&id=cust090' 과 같이 서버로 보내는 데이터를 파라미터라고 한다. GET 방식은 헤드에 파라미터가 포함되어 있다. 헤드부분은 웹 경로에 노출되므로 웹 브라우저의 경로 부분에서도 볼 수 있다. 파라미터는 '?'로 시작을 표시하고 파라미터가 여러 개면 '&'를 이용하여 연결한다. 헤드 길이는 제한적이기 때문에 파라미터 크기도 제한이 있다. POST 방식은 바디에 파라미터가 포함되므로 경로에 파라미터가 노출되지 않는다. 파라미터의 크기도 제한이 없다.

그림 1-13 GET방식과 파라미터

```
GET/ pagecentric01_comp01/custusercontrol.jsp?command=detail&id=cust090 HTTP/1.1
Accept:image/jpeg, */*
Accept-Language: ko-KR
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: localhost:8090
Connection: Keep-Alive
Cookie: JSESSIONID=C5754E8BCACE37746ACBA55226500A45
```

GET 방식

```
<html>
<body>
<a href= ' custusercontrol.jsp?command=detail&id=cust090 ' >상세보기</a>
</body>
</html>
```

Browser

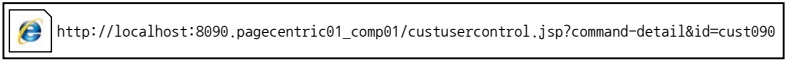


그림 1-14 POST방식과 파라미터

```
post/ pagecentric01_comp01/custusercontrol.jsp HTTP/1.1
Accept:image/jpeg, */*
Accept-Language: ko-KR
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: localhost:8090
Content-Length: 27
Connection: Keep-Alive
Cookie: JSESSIONID=C5754E8BCACE37746ACBA55226500A45
```

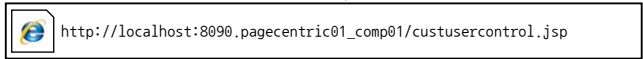
command=bfupdate&id=cust002

빈 한 줄로 HEAD와 BODY구분  
파라미터가 바디에 위치한다.

POST 방식

```
<form action="custusercontrol.jsp" method='post' ?
  <input type='hidden' name='command' value='bfupdate' />
  <input type='hidden' name='id' value='cust002' />
  <input type='hidden' value='고객정보변경' />
</form>
```

Browser





## ■ 정적문서/동적문서

서버에서 클라이언트로 응답할 때 바디 부분은 HTML로 구성된다. HTML은 클라이언트인 웹 브라우저에 전달되어 화면에 출력된다. 서버에서는 변환이나 실행이 되지 않으며, 그대로 웹 브라우저에 전달된다. HTML과 같은 문서를 정적문서(static document)라고 한다. 서버에서는 변환이나 실행을 하지 않다가 웹 브라우저에서 정상문자 입력 검증(validate-null인지 아닌지 등)이나, 변환 혹은 실행을 하는 스크립트(자바스크립트)를 클라이언트 동적문서(dynamic document)라 한다. HTML이나 스크립트 언어는 클라이언트 문서다. 스크립트는 브라우저에서 한 줄씩 인터프리터로 전달되어 실행된다. JSP/Servlet은 서버 문서이고, 컴파일되어 실행된 결과를 HTML로 만든다. JSP/Servlet는 서버에서 실행되며 클라이언트에게 응답하는 문서를 만들므로 동적문서다. 웹 프로그래밍은 클라이언트의 요청을 받아 웹 서버에서 JSP/Servlet을 실행하고 결과를 웹 브라우저로 응답하는 것이다.

클라이언트사이드 문서의 종류

- 정적문서 : HTML, CSS
- 동적문서 : 스크립트

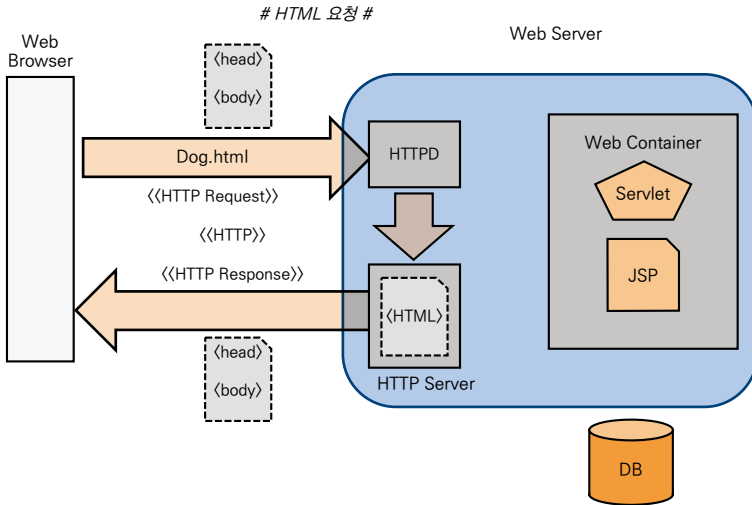
서버사이드 문서의 종류는

- 동적문서: JSP/Servlet

## ■ HTTP Server

HTML, CSS, 자바스크립트로 구성된 문서는 서버에서 변환되거나 실행되지 않는다. 이런 클라이언트사이드 문서를 요청한 웹 브라우저로 응답해주는 서버가 HTTP Server다. HTTP Server는 요청에 대한 응답만 한다.

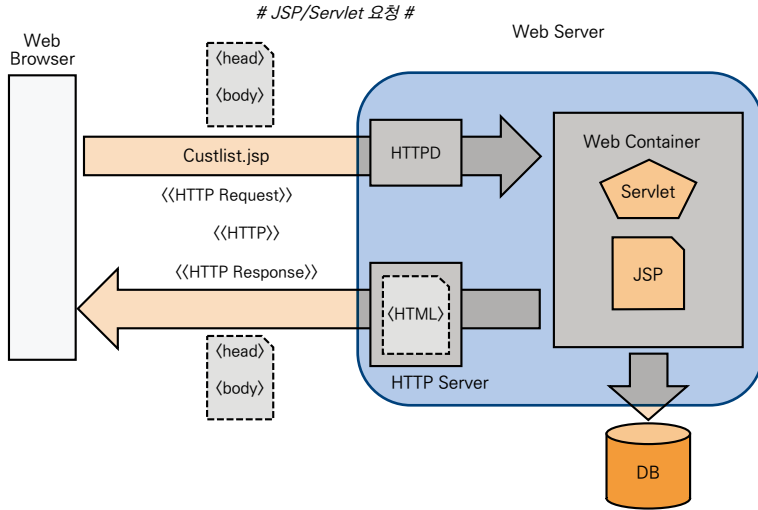
그림 1-15 HTTP Server



### ■ 웹 서버 구조

요청이 들어오면 확장자(.jsp)등을 확인하여 서버사이드 문서를 요청했는지 판단한다. HTTPD는 요청된 문서가 클라이언트 사이드인지 서버사이드인지 판단한다. 서버사이드라면 웹 컨테이너에서 JSP/Servlet을 실행시킨 다음, 요청 결과를 클라이언트 사이드 문서(HTML)로 만들어 HTTP Server로 보내면 HTTP Server가 웹 브라우저로 보낸다.

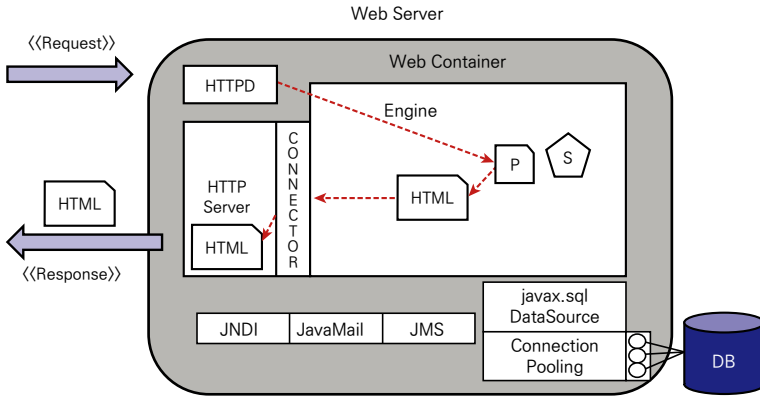
그림 1-16 웹 서버 구조



### ■ 웹 컨테이너(Web Container)

웹 서버에는 HTML을 클라이언트로 보내는 HTTP Server, JSP/Servlet을 실행하는 엔진, 엔진과 JSP/Servlet을 실행하는데 필요한 라이브러리 등을 포함한 웹 컨테이너가 있다. 엔진은 JSP/Servlet을 실행해 결과물인 HTML을 만든다. 커넥터 connector는 HTML을 HTTP Server로 보내 웹 브라우저가 응답하게 한다. Tomcat 웹 서버 Tomcat web server에는 카탈리나 catalina 컨테이너, 자스퍼 jasper 엔진, 컨테이너와 HTTP Server를 연결하는 코요테 coyote 커넥터가 있다. 웹 프로그래밍에는 웹 컨테이너가 반드시 있어야 한다. HTTP Server는 자바로, Apache HTTP Server는 C로 만들어졌다는 점도 알아두자.

그림 1-17 웹 컨테이너



### ■ 파라미터 처리

클라이언트에서 서버로 요청할 때 요청경로와 데이터를 함께 보낼 수 있다. 이 데이터는 '키=값' 형태로 전송되며 파라미터라고 한다. 웹 서버에서 파라미터를 받을 때는

```
String 값 = request.getParameter("키");
```

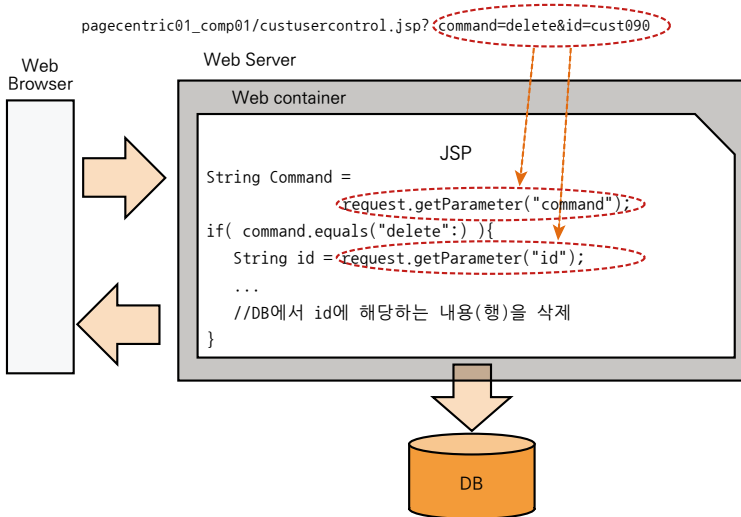
로 받는다. '키'의 이름으로 '값'을 받아 요청을 처리한다. 파라미터의 '키'와 '값'은 모두 String이다. 데이터베이스에서 id가 'cust003'인 행을 삭제하고 싶다면 다음 처럼 요청에 파라미터를 함께 보낸다.

```
pagecentric01_comp01/custusercontrol.jsp?command=delete&id=cust003
```

서버는 id를 이용하여 요청한 행을 삭제한다. 요청 중 상세내용을 확인하기 위해 command를 받고, if문에서 command가 삭제, 삽입, 변경인지 확인하고 요청별로 처리한다.

```
String command = request.getParameter( " command " );
if( command.equals( " delete " ) )
    String id = request.getParameter( " id " );
    ...
//DB에서 id에 해당하는 내용(행)을 삭제한다.
```

그림 1-18 파라미터 처리



## ■ 서블릿(Servlet)

서블릿은 “Servlet = Server + let”이며, 'let'는 컨테이너에서 실행되는 프로그램이다. 다시 말해 서블릿은 웹 서버(Tomcat)의 컨테이너(카탈리나)에서 실행되는 웹 프로그래밍이다. 요청 또는 요청과 동반하는 파라미터를 서블릿에서 받아 요청을 처리하고 결과를 HTML로 만든다. 웹 브라우저는 응답으로 보내진 HTML을 받아서 화면에 출력한다.

'let'가 붙은 다른 프로그램도 있다. 애플릿<sup>applet</sup>은 applet = application + let로 컴파일된 프로그래밍이 웹 브라우저의 컨테이너에서 실행되는 애플리케이션이며, 애플릿은 클라이언트사이드 프로그래밍이다. 미들릿<sup>middlelet</sup>은 mid<sup>Mobile Information Device Profile</sup>+let를 말하며 모바일 정보기기 프로파일을 사용하는 애플리케이션으로 핸드폰 컨테이너에서 실행된다.

## ■ 라이프사이클(Life Cycle)

JSP/Servlet은 웹 프로그래밍이다. 웹 프로그래밍은 컨테이너가 반드시 필요하며, 프로그래밍을 엔진에서 실행하여 원하는 결과를 생성한다. 결과물은 HTML로 만들어서 웹 브라우저에 응답한다. 서블릿의 'let'는 서블릿이 컨테이너에서 실행되는 프로그래밍이라는 점을 알려준다. 'let'가 붙은 애플리케이션은 컨테이너의 명령에 따라 생성, 초기화, 실행, 소멸 등 정해진 행동을 한다. 이렇게 하는 것을 라이프사이클이라고 한다. 컨테이너는 요청에 따라 정해진 메서드를 호출한다. 요청을 처음으로 받으면 서블릿을 생성한다. 초기화(`init()`) 메서드를 호출하여 생성된 서블릿을 실행하는 데 필요한 데이터나 값을 얻는다. 실행(`service()`) 메서드를 호출하여 요청을 처리하고 결과를 응답해준다. 두 번째 요청부터는 실행(`service()`) 메서드만 호출한다. 더 이상 요청이 없다면 소멸(`destroy()`) 메서드를 호출하여 자원을 회수하고, 서블릿의 객체<sup>instance</sup>를 제거한다. 컨테이너는 초기화와 소멸 메서드를 한 번만 호출한다.

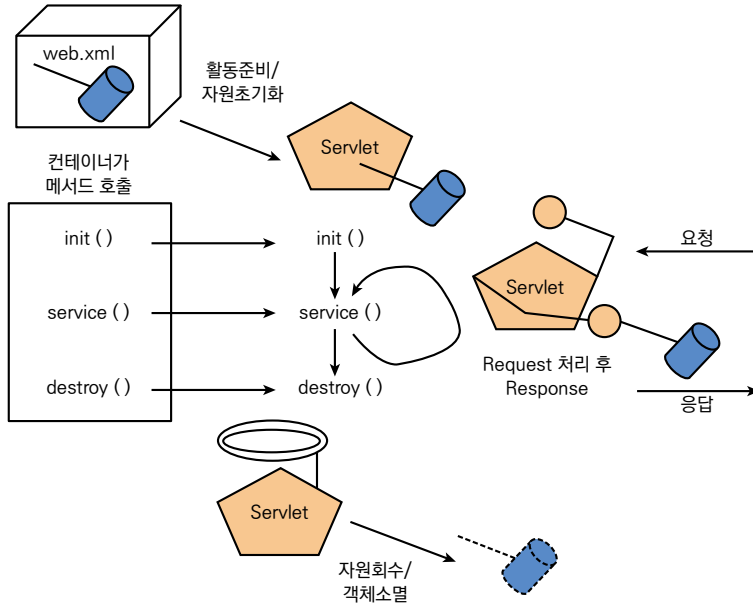
요청에 따라 필요한 로직을 실행하려면 아래와 같이 `service()` 메서드 바디()에 프로그래밍하면 된다.

```
public void service(request, response)
    //실행로직
```

요청을 받으면 컨테이너는 서블릿의 `service()` 메서드를 호출하여 로직을 실행한

다.

그림 1-19 웹 라이프 사이클



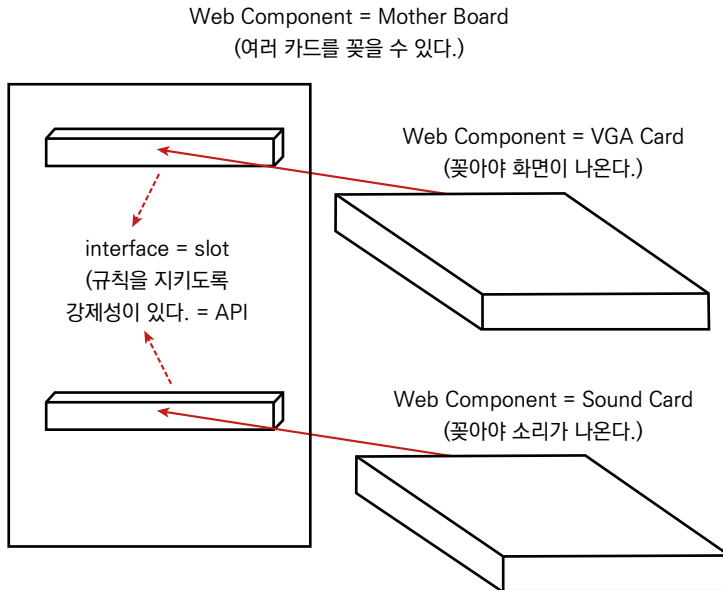
### ■ JSP(Java Server Page)

JSP 중 Server Page는 서버에서 실행되는 라이프사이클이 있는 웹 프로그래밍용 클래스를 말한다. Java는 개발 언어를 가리킨다. Tomcat 웹 서버에서 자스퍼(jasper) 엔진이 JSP를 서블릿으로 변환한다. 라이프사이클을 고려하지 않고 개발해도 서블릿으로 변환될 때 라이프사이클 메서드가 자동으로 만들어져 서블릿처럼 실행된다. hello.jsp를 만들면 org.apache.jsp.hello\_jsp.java와 같이 서블릿으로 변환되고 org.apache.jsp.hello\_jsp.class로 컴파일된 다음 실행된다. JSP도 서블릿이므로 서블릿에서 사용하는 객체나 메서드를 JSP에서도 사용할 수 있다.

## ■ 웹 컴포넌트

컨테이너에서 실행될 수 있도록 필요한 규칙을 준수한 웹 프로그램 묶음을 웹 컴포넌트라고 한다. 예를 들어 컴퓨터의 마더보드는 여러 카드를 꽂을 수 있는 구멍(슬롯)이 있고, 각 카드는 정해진 목적이 있다. 목적을 이루기 위해 여러 부속품으로 여러 카드를 만든다. JSP나 서블릿은 부속품에, 웹 컴포넌트는 각 카드에 해당한다. 각 카드는 마더보드에 꽂혀야 실행된다. VGA 카드를 꽂아야 화면이 나오며, 사운드 카드를 꽂아야 소리가 나온다. 카드를 마더보드에 꽂을 때는 슬롯의 암수가 맞아야한다. 슬롯 설치 시 반드시 이 규칙을 지켜야 하는데, 웹 컴포넌트에서도 API라는 규칙을 반드시 지켜야한다. 따라서 웹 프로그래밍은 웹 컴포넌트를 만드는 방법, 웹 컴포넌트와 웹 컨테이너 사이의 규칙을 배우는 것이라 할 수 있다.

그림 1-20 웹 컴포넌트





## ■ 컨텍스트(Context)

배포단위, 실행단위의 디렉터리를 컨텍스트라고 한다. 개발 후 JSP나 서블릿, 그림, HTML, web.xml을 압축하여 묶어서 (아카이브) war를 만든다. 이 war를 웹 컨테이너에 놓으면 압축이 풀려 실행을 위한 디렉터리가 된다. 이 디렉터리가 바로 컨텍스트이며 모든 파일을 각 용도에 맞게 컨텍스트 안의 디렉터리에 놓아야 한다. 예를 들어 hello.war를 웹 컨테이너에 놓으면 hello.war가 풀어져 hello 디렉터리가 된다. 컨텍스트 바로 아래에는 HTML, CSS, 자바스크립트, 이미지, JSP, WEB-INF 디렉터리가 있다. WEB-INF 디렉터리 안에는 web.xml이 있고, 서블릿과 컴파일된 클래스는 classes 디렉터리에 있어야 하며, .jar로 압축된 라이브러리는 lib에 안에 있어야 한다. 컨테이너는 컨텍스트의 web.xml을 가장 먼저 한 번 읽어들인다.

그림 1-21 컨텍스트

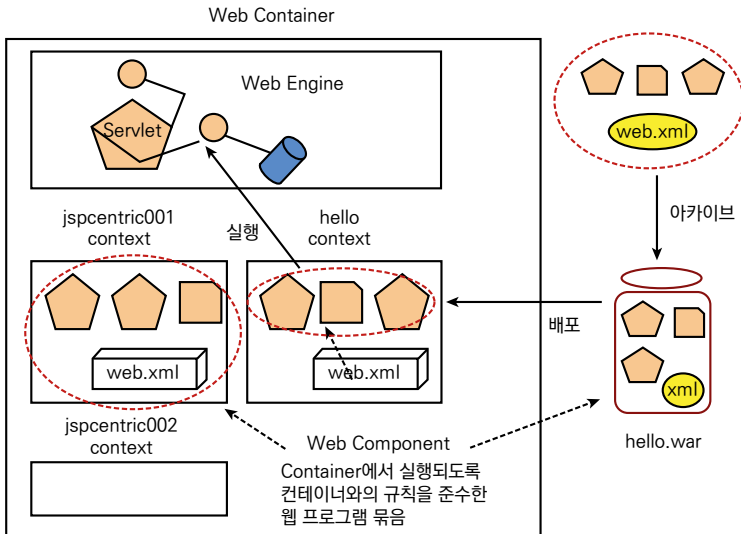
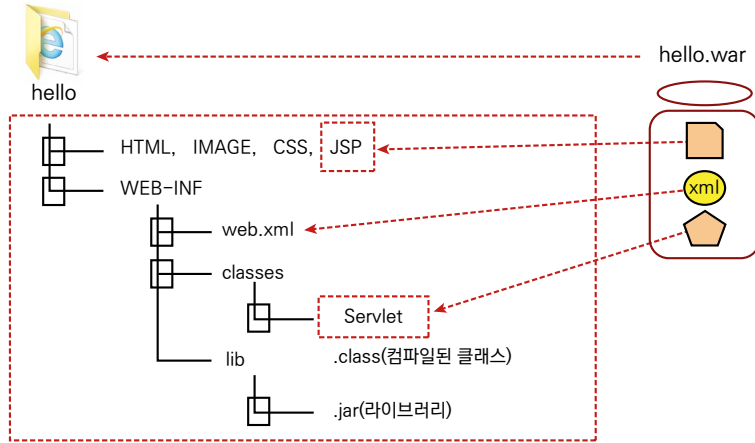


그림 1-22 컨텍스트

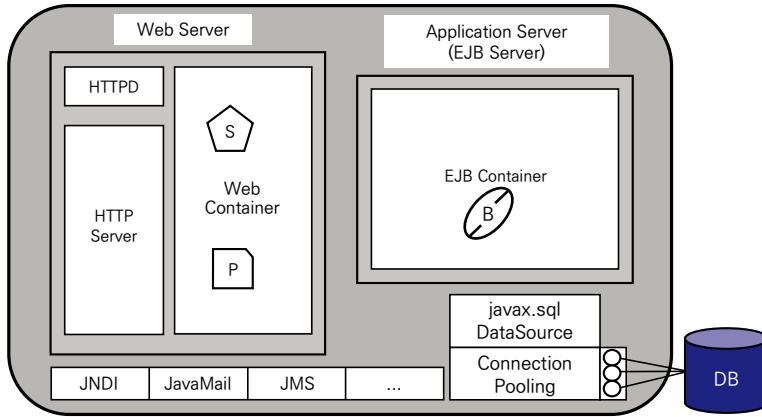


### ■ WAS(Web Application Server-Java Enterprise Edition Server)

WAS(JEE)는 “Web Server + Application Server + Service”의 의미다. 웹 서버는 JSP/Servlet를 위한 웹 컨테이너와 HTML을 서비스하는 HTTP Server가 있다. Application Server에서는 비즈니스 로직을 수행하는 EJB 컨테이너가 있다. 서비스는 JNDI, JMS, JTA, JavaMail을 제공하여 다른 기능의 서버와 연결하여 사용하게 하거나 서버 안에서 편리하게 사용할 수 있게 한다. Enterprise Edition의 필수요소를 모두 구현한 서버를 Enterprise Edition Server 또는 WAS라 부른다. WAS에는 WebLogic, WebSphere, JBoss, Zeus 등 수많은 제품이 있다. Tomcat을 WAS로 부르기도 한다. 이 책에서는 Tomcat 웹 서버를 Tomcat 애플리케이션 서버와 같은 의미로 사용한다.

그림 1-23 웹 애플리케이션 서버

Web Application Server (WAS, JEE Server)



### ■ 레이어/티어

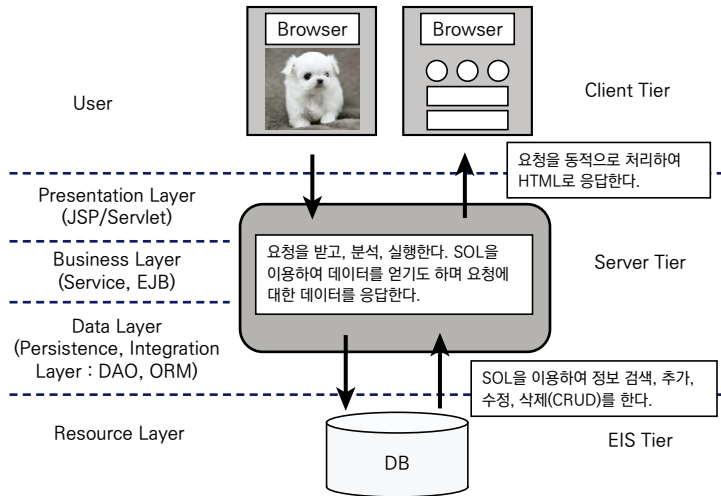
레이어는 프로그램의 역할에 따라 논리적으로, 티어는 시스템의 역할에 따라 물리적으로 나눈 것이다. 혼용되기도 하지만 논리적인 것과 물리적인 것으로 구분되는 점을 기억하자. 데이터베이스에서 DVD 대출정보를 찾아 웹 브라우저에 출력하는 웹 프로그래밍을 한다고 가정해보자. DVD 대출 프로그램은 데이터베이스에 접근해 정보를 가져오는 부분Data Layer, 고객정보인지 비디오 정보인지 요청을 판단하고 실행하는 부분Business Layer, HTML로 결과를 만들고 응답하는 부분Presentation Layer으로 나눌 수 있다. DVD 대출 프로그램은 역할에 따라 나눌 수 있으며 이를 레이어layer라 한다. 데이터베이스 서버, 웹 서버, 클라이언트인 웹 브라우저등 시스템을 물리적으로 나눌 수 있는데 이것을 티어tier라 한다. 웹 프로그래밍은 기본적으로 3티어다. 레이어는 보통 4개로 나눈다. 이 책에서는 3티어와 4레이어를 기본으로 사용한다.

- 화면 레이어Presentation Layer : JSP와 서블릿이 담당한다. 요청을 받아 비즈니스 레이어에 정보를 요청한다. 결과를 받아 요청한 화면을 동적으로 만들고, 요청한

화면으로 옮긴다.

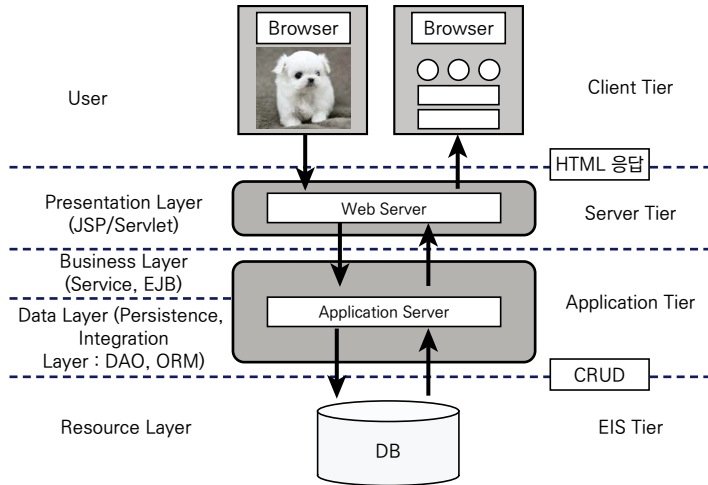
- 데이터 레이어Data Layer: 영속성 레이어Persistence Layer로도 알려진 레이어로, 데이터베이스에 쿼리(CRUD)를 실행한다. 데이터베이스에 접근하여 쿼리를 실행하기 위해 JDBC를 이용하거나, 해당 쿼리에 대해 JDBC 작업을 자동으로 실행해 쉽게 사용할 수 있도록 만든 프레임워크(ORM-iBatis, Hibernate)를 사용한다.
- 비즈니스 레이어Business Layer: 화면 레이어는 데이터 레이어를 이용한다. 비즈니스 레이어에서 얻은 객체를 비즈니스 객체라고 하며, 데이터베이스의 테이블(엔터티)과 관련이 있다. 데이터 레이어에서 얻은 결과로 화면 레이어에서 필요한 비즈니스 객체를 만들어도 된다.

그림 1-24 3티어 구성



애플리케이션 서버를 사용하는 경우는 4티어가 된다. 4티어일 때는 웹 서버와 애플리케이션 서버가 물리적으로 나누어진 티어가 되므로 레이어를 티어라고 부른다.

그림 1-25 4티어 구성



### ■ DAO(Data Access Object-데이터 접근 객체)

데이터베이스에 관련된 작업(CRUD-Create, Retrieve, Update, Delete: SQL DML+Select)을 전문적으로 담당하는 객체다. DAO 안의 메서드는 모두 데이터 베이스와 관련된 작업을 한다. 아래와 같이 CRUD를 실행하는 메서드는 JDBC등을 이용하여 데이터베이스에 접근해서 쿼리를 실행한다. 다른 개발자도 해당 메서드를 호출하면 해당 쿼리를 실행하여 결과를 얻을 수 있다.

```
public CustUserDto getCustUser(String id) 해당 id에 해당하는 결과를 얻는 쿼리 실행
public int addCustUser(CustUserDto uDto)insert 쿼리 실행
public int updateCustUser(CustUserDto uDto)update 쿼리 실행
public int deleteCustUser(String id)delete 쿼리 실행
```

### ■ DTO(Data Transfer Object, 데이터 전송 객체)

데이터베이스의 테이블(엔티티)에 해당하는 객체로 테이블의 컬럼들을 일대일로

저장할 수 있는 멤버필드가 있고 get/set 메서드를 갖는다. 연필들(테이블의 컬럼들)을 안전하게 옮길 수 있도록 만들어진 필통<sup>DTO</sup>이라고 생각하자. DTO는 로직이 없으며 일반적으로 하나의 DTO가 하나의 행<sup>ROW</sup>에 해당되고, 대부분 DAO와 같이 사용된다. DAO의 메서드는 DTO를 반환하거나 DTO를 대입한다. 비즈니스 레이어에서 반환하는 Business 객체를 DTO로 보면 된다. 아래 소스에서 CustUserDto가 DTO다. getCustUser() 메서드를 호출하면 테이블에서 해당 행에 해당하는 데이터를 DTO에 담아서 반환한다.

```
public CustUserDto getCustUser(String id) 해당 id에 해당하는 결과를 얻는 쿼리 실행
public int addCustUser(CustUserDto uDto)insert 쿼리 실행
public int updateCustUser(CustUserDto uDto)update 쿼리 실행
```

### ■ MVC 패턴(Model-View-Controller)

MVC 패턴은 요청을 처리하는 과정에서 발생하는 처리순서, 데이터 전송, 관리 작업과 데이터 출력에 대한 웹애플리케이션 작업을 간단하게 도식화한다. 요청을 받은 컨트롤러는 요청을 분석한다. 요청에 해당하는 모델을 이용하여 비즈니스 로직(데이터 베이스에 관련된 작업)을 실행하고 비즈니스 객체를 얻는다. 해당 뷰(화면)로 제어권을 넘긴다(포워드). 뷰는 받은 비즈니스 객체를 동적으로 처리하고 HTML로 화면을 만들어(렌더링) 웹 브라우저에 응답한다. 아래 그림은 MVC 패턴을 웹애플리케이션에서 구현한 MVC Model 2의 처리과정이다. 과정은 다음과 같다.

- ① 컨트롤에 요청을 한다.
- ② 컨트롤은 요청을 분석하고 해결할 모델을 찾아서 실행한다.
- ③ 모델은 쿼리를 실행하고 결과를 얻는다.
- ④, ⑤ 해당화면으로 이동하여 모델에서 얻은 데이터를 동적으로 처리한다(HTML로 렌더링).
- ⑥ 브라우저로 요청했던 화면을 보내 응답한다.