

웹 어플리케이션 개요 및 개발 환경 구축

1.1_인터넷 서비스

1.2_프로토콜(Protocol)

1.3_IP와 DNS, Port 번호

1.4_HTTP(HyperText Transfer Protocol)의 개요

1.5_HTTP 클라이언트와 서버

1.6_웹 어플리케이션의 개요

1.7_웹 컴포넌트(Web Component)의 개요

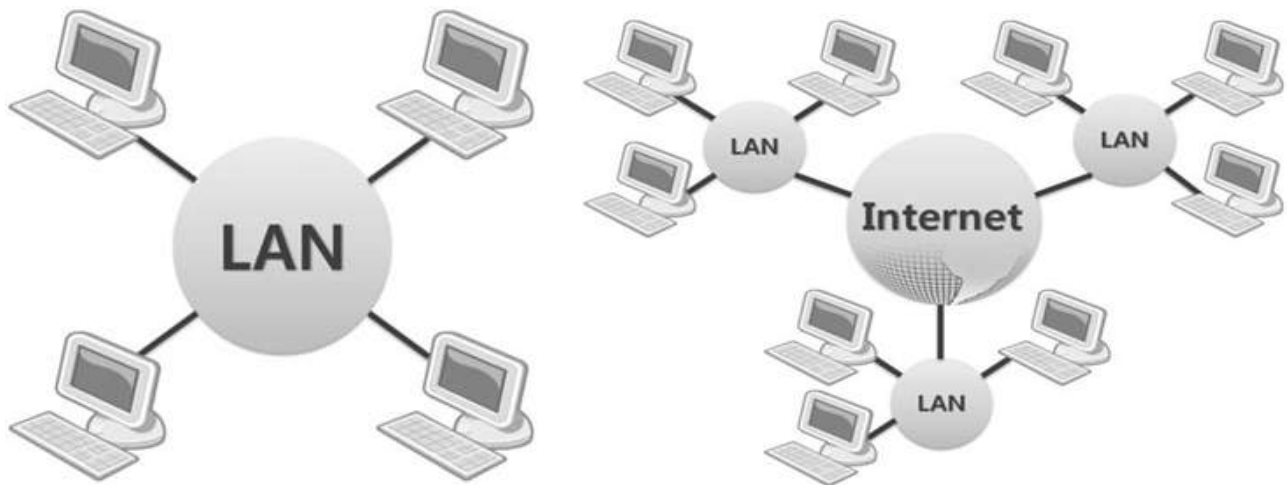
1.8_웹 어플리케이션 개발 환경 구축

1.1 인터넷 서비스

-인터넷의 정의를 내리면 '네트워크의 네트워크'

-초창기의 네트워크는 특정 자원을 LAN(Local Area Network)을 통하여 소수의 사람들이 공유

-네트워크가 전 세계적으로 확대되어 사용 → 인터넷(Internet)



1.2 프로토콜(Protocol) - 1

프로토콜이란 컴퓨터간의 원활한 통신을 위해서 필요한 통신규약

통신을 원하는 두 개체간에 무엇을, 어떻게, 언제 통신할 것인가를 서로 약속한 규약이다.

보다 쉽게, 그리고 정확하고 안전하게 서로의 의사를 확인하기 위해서 필요

1) Telnet 프로토콜

-주로 유닉스 시스템의 네트워크로 연결된 원격 터미널에서 호스트의 셸 모드로 접속 가능한

프로그램

-원격지에서 호스트의 명령어 모드를 그대로 흉내 낼 수 있음

2) FTP 프로토콜

-File Transfer Protocol로서 인터넷과 같은 광범위한 지역의 네트워크에서 각 네트워크 사이의 자료 교환을 목적으로 만든 프로토콜

-파일 전송만을 위한 프로토콜

1.2 프로토콜(Protocol) - 2

3) SMTP 프로토콜

-Simple Mail Transfer Protocol로서 단순한 메일 전송 프로토콜

-일반적으로 인터넷 서비스 업체에서 SMTP 서버를 메일 송신 서버로 쓰고 POP 서버를 메일 수신 서버로 사용

4) POP 프로토콜

-Post Office Protocol로서 메일의 수신을 위한 프로토콜

-최근에는 POP3를 주로 사용

5) DHCP 프로토콜

-Dynamic Host Control Protocol로서 서버가 동적으로 클라이언트나 터미널에게 IP를 부여하는 프로토콜

-일반적으로 각 가정에서 인터넷 서비스를 사용할 때 주로 사용

1.2 프로토콜(Protocol) - 3

6) HTTP 프로토콜

-HyperText Transfer Protocol로서 FTP와 비슷하게 서버로부터 클라이언트로 데이터를 전송하기 위한 프로토콜

-HTML과 밀접한 관련이 있는 프로토콜

-JSP/Servlet 프로그래밍 과정에서 사용되는 프로토콜

-비지속성(Connectionless) 연결 방식으로 동작

-클라이언트가 서버로 요청을 하고 응답을 받으면 연결을 유지하지 않고 즉시 연결을 끊는 방식

-특정 시간에 동시 접속자가 증가하더라도 무리없이 서비스가 가능

1.3 IP와 DNS, Port 번호 - 1

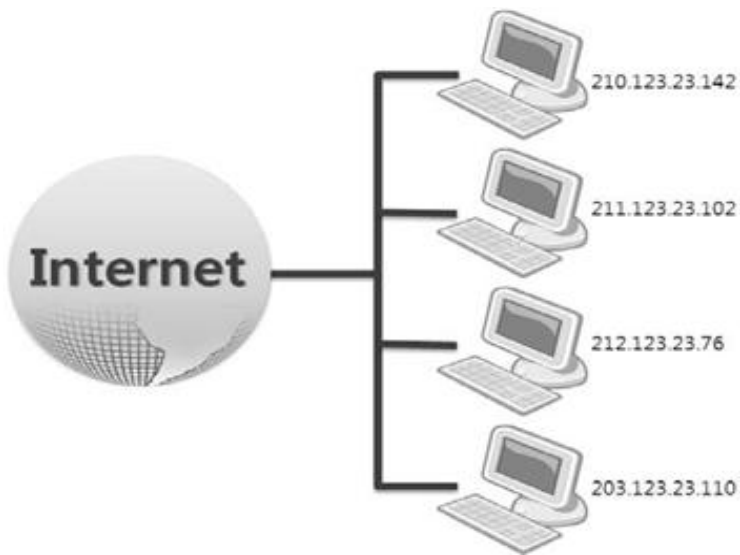
-IP(Internet Protocol)

-인터넷상의 한 컴퓨터에서 다른 컴퓨터로 데이터를 보내는 데 사용되는 프로토콜

-인터넷상의 각 컴퓨터, 즉 호스트들은 다른 컴퓨터와 구별될 수 있도록 적어도 한 개 이상의 고유한 주소를 갖음

-IP 주소를 사용하여 컴퓨터 식별이 가능

- IPv4는 4자리의 최대 12자리의 번호로 구성
- 인터넷 주소의 수요를 충족시킬 수 없어서 IPv6가 등장

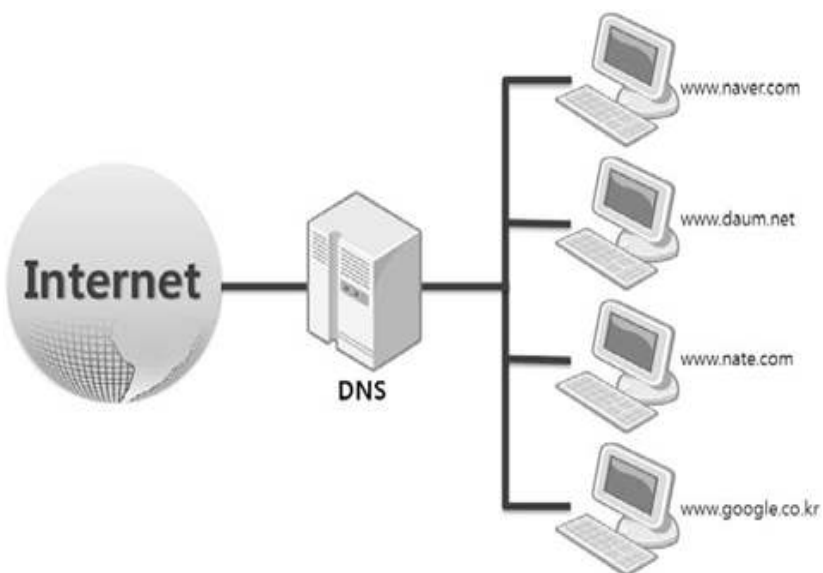


1.3 IP와 DNS, Port 번호 – 2

-DNS(Domain Name System)

-숫자체계의 IP 주소로는 사람들이 기억하기가 어렵기 때문에 쉽고 의미있게 이름 붙인 정보를 가진 시스템 및 서버를 의미

-각 사이트가 운영하는 모든 호스트 서버는 고유한 IP 주소를 갖고 있기 때문에 사용자가 문자로 주소를 입력하면 DNS 서버를 통해서 호스트 서버가 인식할 수 있는 IP로 바뀌어 접속



1.3 IP와 DNS, Port 번호 - 3

-Port 번호

-하나의 컴퓨터에서 네트워크를 사용하는 프로그램이 여러 개가 동시에 실행되는 경우에 IP 주소만 이용해서는 특정 네트워크 프로그램에 접근할 수 없음

-이런 상황에서 해당 프로그램을 구분할 수 있게 해주는 것이 포트

-IP 번호가 아파트의 동이름이라면 Port 번호가 호수

-HTTP 프로토콜의 기본 Port 번호는 80

1.4 HTTP(HyperText Transfer Protocol)의 개요 - 1

-1.4.1 HyperText

-하이퍼텍스트는 사용자의 선택에 따라서 관련된 문서로 옮겨갈 수 있도록 조직화된 정보를 의미

-일반적으로 링크(Link) 또는 하이퍼링크(Hyperlink)라고 함

-하이퍼텍스트는 W3(World Wide Web)의 주요기술로서 웹 브라우저를 통해 웹 서버에서 문서 및 웹 페이지 등의 정보 조각을 읽어 들여 모니터 화면에 출력하는 형태

-사용자는 각 페이지에 있는 하이퍼링크를 따라 다른 문서로 이동하거나 그 페이지를 서비스 하고 있는 서버로 정보를 전송 가능

-1.4.2 웹(World Wide Web, WWW, W3)

-인터넷에 연결된 컴퓨터들을 통해 사람들이 정보를 공유할 수 있는 전 세계적인 정보 공간을 의미

-인터넷과 동일한 의미로 사용되지만 엄밀히 말해 서로 다른 개념

-웹은 인터넷상에서 동작하는 하나의 서비스

1.4 HTTP(HyperText Transfer Protocol)의 개요 - 2

-1.4.3 HTTP(HyperText Transfer Protocol)

-웹상에서 정보를 주고받을 수 있는 프로토콜

-주로 HTML 문서를 주고받는데 사용

-기본적으로 80포트 번호를 사용

-HTTP는 클라이언트와 서버 사이에 이루어지는 요청과 응답 프로토콜

-클라이언트인 웹 브라우저에서 HTTP를 통하여 웹 서버로부터 웹 페이지를 요청하면 웹 서버는 이 요청에 응답하여 필요한 정보를 해당 클라이언트에게 응답 처리

-HTTP는 클라이언트의 데이터 요청에 대한 응답 후 바로 연결을 해제

-필요할 때마다 연결하고 데이터를 받자마자 바로 연결을 해제하는 방식이 HTTP의 메커니즘

-지정된 시간 동안 많은 사용자가 특정 서버에 접속 가능

1.4 HTTP(HyperText Transfer Protocol)의 개요 - 3

-1.4.4 웹 브라우저 및 웹 서버

웹 브라우저

-웹 서버로부터 받은 HTML 문서나 파일을 출력하는 소프트웨어

- 종류 : 인터넷 익스플로러, 모질라 파이어폭스, 오페라, 사파리, 크롬 등
- 웹 서버로부터 웹 페이지를 가져오기 위해서 웹 서버가 사용하는 HTTP 프로토콜을 사용
- HTTP를 이용해 웹 서버에 특정 정보를 전송하는 것도 가능

웹 서버

- 웹 브라우저를 이용하는 클라이언트로부터 HTTP 요청을 받아서 HTML 문서와 같은 웹 페이지를 전송해주는 서버
- URL로 표현된 HTML 문서 및 각종 정보를 수신
- 종류 : 아파치(Apache) 및 IIS 등

1.4 HTTP(HyperText Transfer Protocol)의 개요 - 4

-1.4.5 URL(Uniform Resource Locator)

- 웹 서버가 인터넷상에 존재하는 자원(정보, 파일 등)을 검색하고 해석하는데 필요한 네트워크 서비스의 표현식
- 웹 브라우저 주소란에 다음과 같은 문법으로 입력

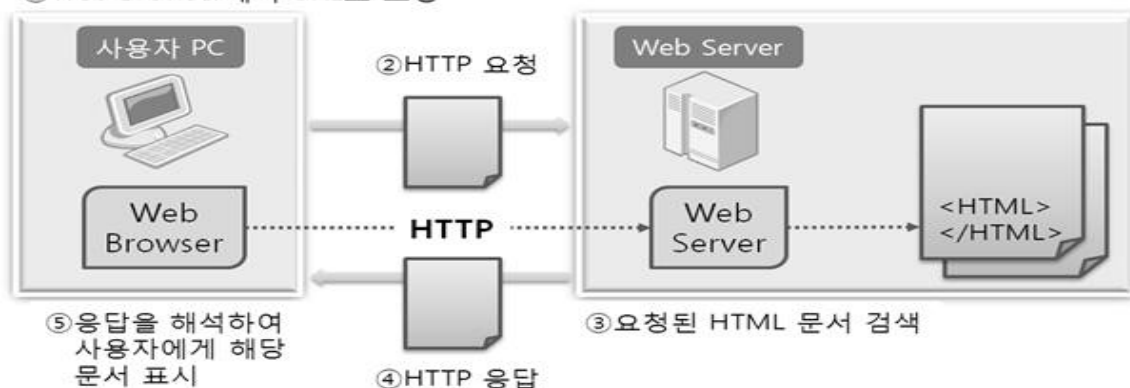
프로토콜://호스트 번호:포트 번호/경로/파일명



1.5 HTTP 클라이언트와 서버 - 1

-1.5.1 Client와 Server 구조

① Web Browser에서 URL로 요청

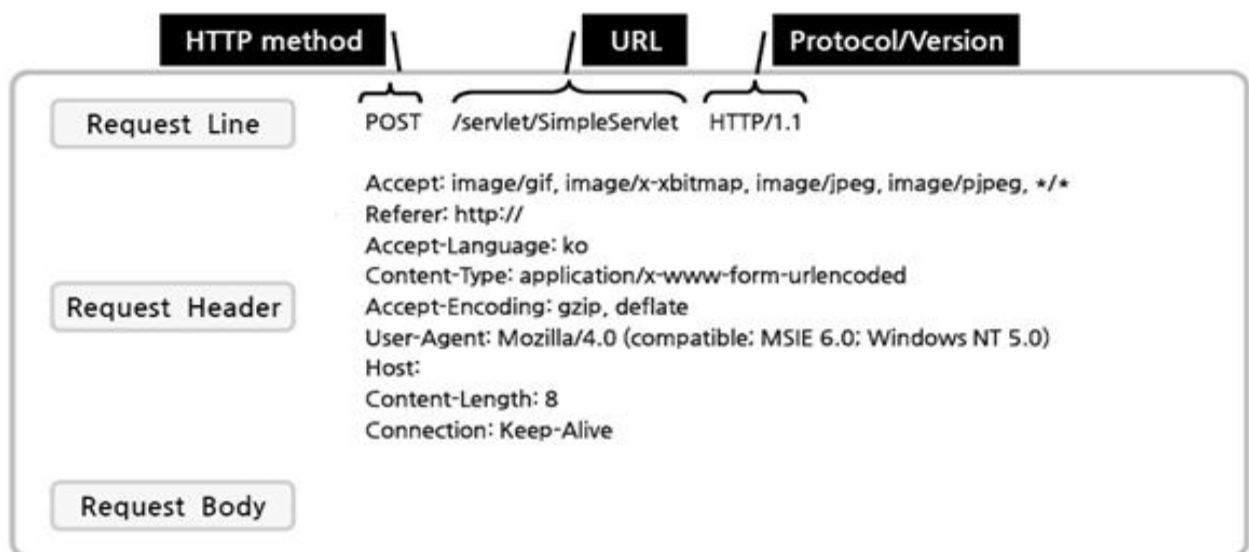


- ① 클라이언트인 웹 브라우저에서 특정 URL로 요청
- ② 요청 라인, 헤더, 몸체로 구성된 요청이 발생
- ③ 웹 서버는 요청된 HTML 문서를 검색. 만약에 요청된 문서가 없으면 File Not Found에러 (404 에러)가 발생.
- ④ 상태 라인, 헤더, 몸체로 구성된 응답이 처리
- ⑤ 웹 브라우저는 응답을 해석하여 사용자에게 응답 결과를 표시

1.5 HTTP 클라이언트와 서버 - 2

-1.5.2 HTTP Request

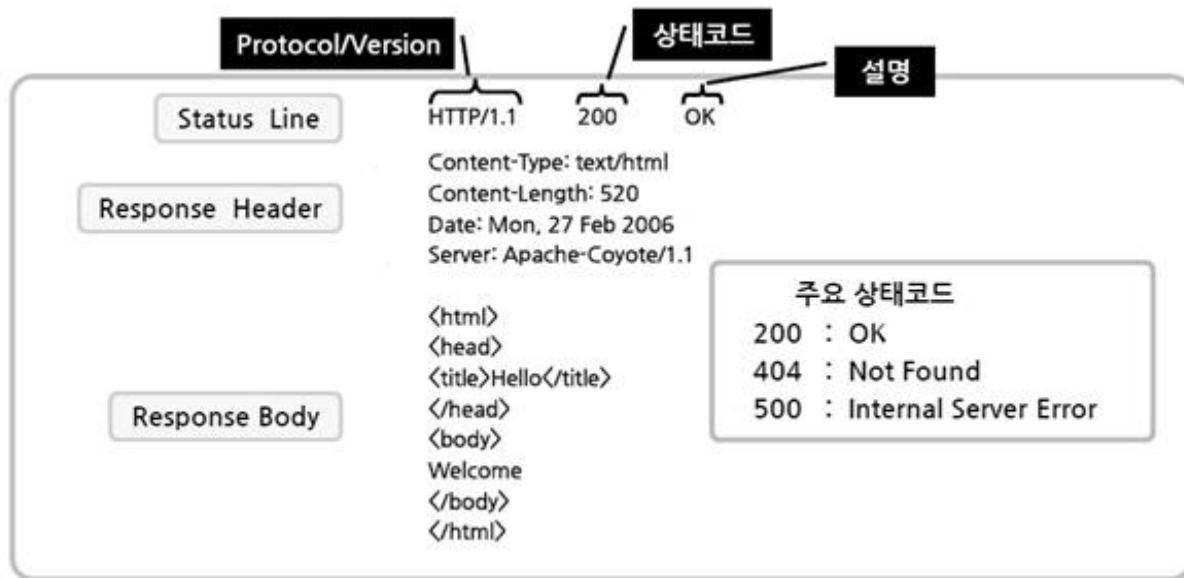
- 요청 라인(Request Line) : HTTP 메서드 방식 및 요청 URL과 프로토콜 정보를 갖음
- 요청 헤더(Request Header) : 웹 브라우저 정보, 언어, 인코딩 방식, 요청 서버 정보 등과 같은 추가 정보를 갖음
- 요청 본체(Request Body) : 요청에 필요한 내용을 갖는다. 일반적으로 HTML 폼 태그 안에 입력된 값들인 파라미터 정보를 의미



1.5 HTTP 클라이언트와 서버 - 3

-1.5.3 HTTP Response

- 상태 라인(Status Line) : 응답 상태 코드 및 프로토콜 정보를 갖음
- 응답 헤더(Response Header) : 응답처리 날짜, 인코딩 방식, 요청 서버 정보 등과 같은 추가 정보를 갖음
- 응답 본체(Response Body) : 응답에 필요한 내용을 갖으며 일반적으로 HTML 문서



1.6 웹 어플리케이션의 개요 - 1

-1.6.1 웹 어플리케이션 정의: 웹상에서 웹 브라우저를 이용한 클라이언트와 HTML(정적 웹 컴포넌트), JSP/Servlet(동적 웹 컴포넌트) 같은 자원을 가진 웹 서버 간에 동적으로 요청/응답 처리하는 프로세스를 의미

-초창기의 웹은 정적인 특징을 갖는 HTML만으로 서비스가 가능했으나 점차 다양한 사용자들의 요구를 만족시키기 위해 매우 부족

-정적인 특징이란, 이미 만들어진 HTML 문서는 요청 시점과 장소에 무관하게 항상 고정된 웹 페이지로 처리된다는 것

-HTML과 같은 정적인 특징이 아닌, 사용자의 요청 시 동적으로 실행되어 원하는 결과 값을 응답 처리하는 새로운 방법이 필요

→ CGI(Common Gateway Interface)

1.6 웹 어플리케이션의 개요 - 2

1.6.2 CGI(Common Gateway Interface)

-CGI의 특징

-정적인 HTML 문서 서비스의 한계를 극복 가능

-서버 쪽에서 동적인 프로그래밍으로 실행되는 스크립트의 시초

-일반적으로 Server-side 프로그래밍이라고 함

-CGI 등장으로 인터넷 시장이 비약적으로 활성화

-클라이언트의 요청만큼 서버에 프로세스가 생성되어 처리하기 때문에 서버 부하가 높아 질수 있음(JSP/Servlet는 Thread로 구현되어 있기 때문에 제외)

-CGI 명세를 구현한 실제적인 동적 웹 컴포넌트로서 제작된 주요 기술

-ASP(Active Server Page)

▶MS 계열로서 VisualBasic 언어를 기반으로 제작

▶장점 : 쉬운 문법과 MS 기술 등을 쉽게 사용 가능하기 때문에 초창기에 많이 사용

- ▶단점 : MS의 IIS 서버만을 사용할 수 있기 때문에 플랫폼에 의존적인 문제와 JSP와 비교해서 자원의 효율성이 떨어짐

1.6 웹 어플리케이션의 개요 - 3

-PHP(Personal HyperText Preprocessor)

- ▶장점 : C 언어를 기반으로 제작되었으며 쉬운 문법과 빠른 속도 및 다양한 플랫폼, 무료로 사용 가능
- ▶단점 : Enterprise급 웹 개발 시에는 적용하기가 쉽지 않고 보안에 취약

-JSP/Servlet

- ▶자바언어를 기반으로 제작되었기 때문에 객체지향적인 특징과 플랫폼에 독립적인 장점을 얻을 수 있음
- ▶다른 CGI 프로그래밍과 비교해서 Thread 기반으로 구현되었기 때문에 다수의 사용자 처리에 적합
- ▶최근의 대부분의 웹 프로젝트는 JSP를 이용하여 개발

1.6 웹 어플리케이션의 개요 - 4

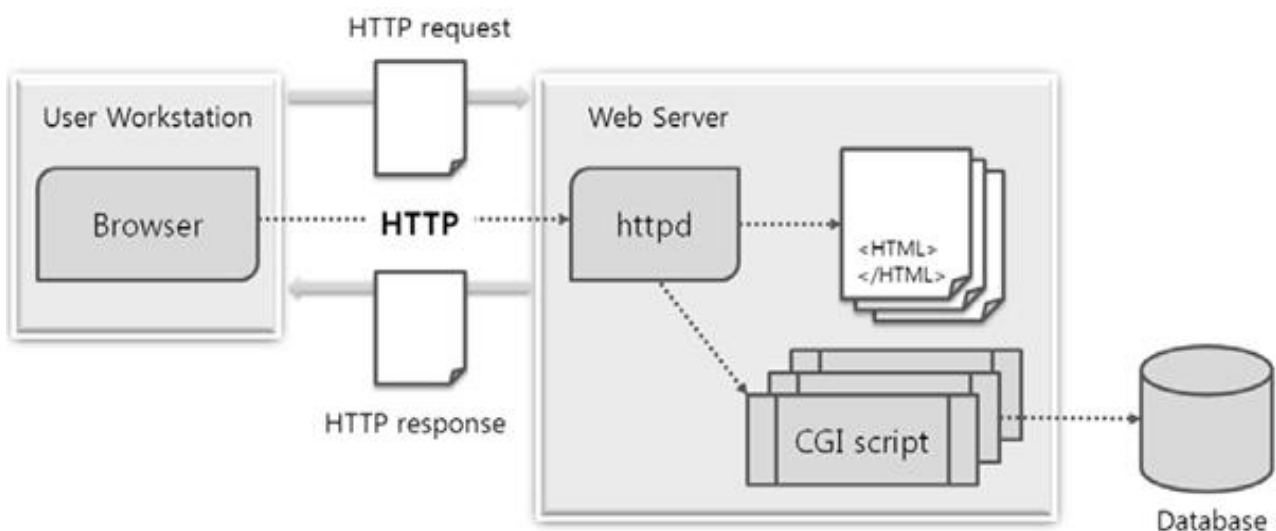
-1.6.3 CGI vs JSP/Servlet

-CGI는 다양한 언어로 작성이 가능하기 때문에 구현이 쉬움

-서버 부하가 높아질 수 있으며 비 객체지향적이고 플랫폼에 의존적인 CGI도 있기 때문에 확장성이 떨어짐

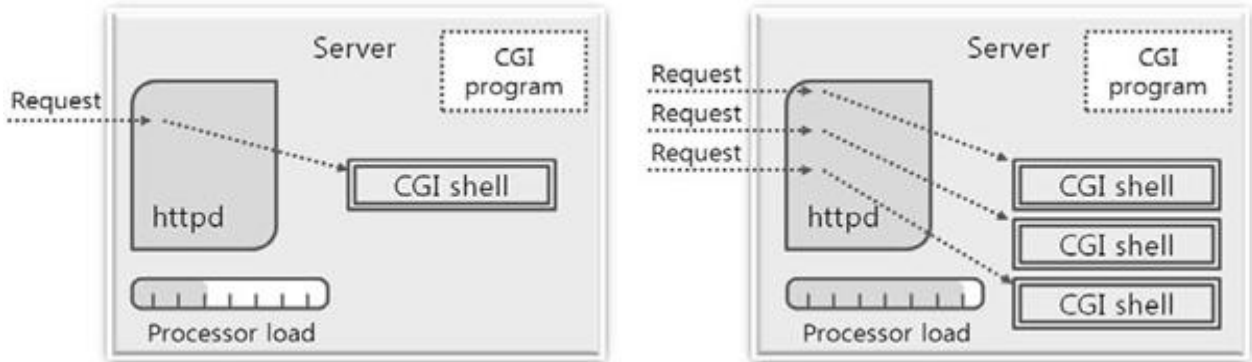
-JSP/Servlet는 성능이 좋고 확장성이 좋으며 객체지향적으로 프로그래밍이 가능하고 플랫폼에 독립적

-CGI를 이용한 웹 어플리케이션의 구조

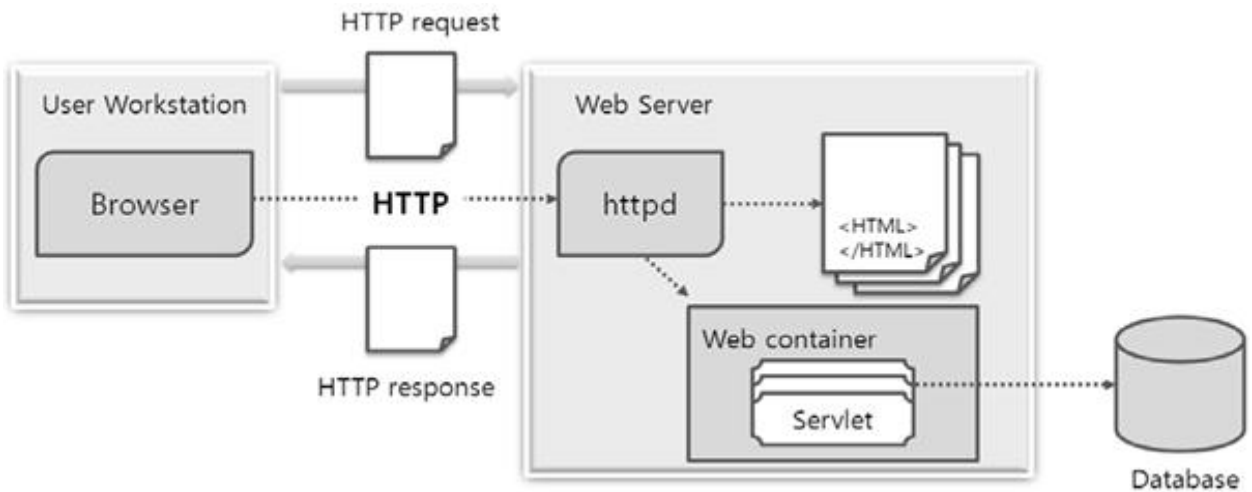


1.6 웹 어플리케이션의 개요 - 5

-하나의 클라이언트와 여러 클라이언트 요청 시의 CGI 서버 구조

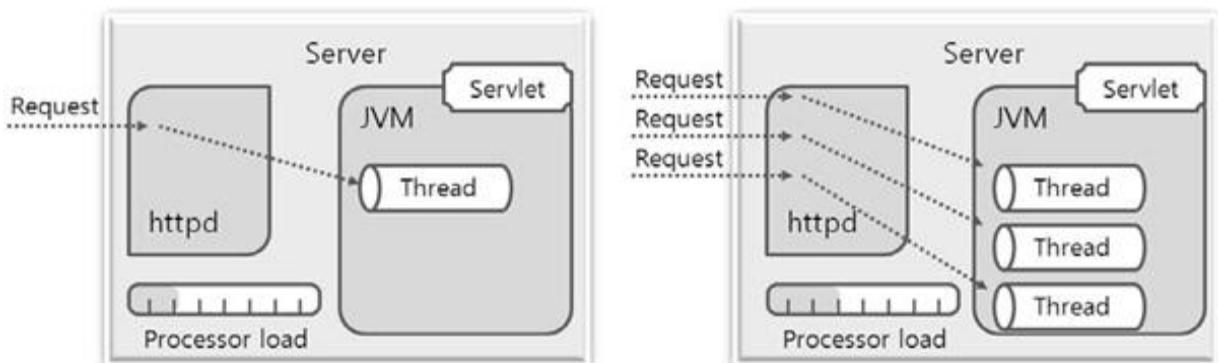


-JSP/Servlet을 이용한 웹 어플리케이션의 구조



1.6 웹 어플리케이션의 개요 - 6

-하나의 클라이언트와 여러 클라이언트 요청 시의 JSP/Servlet 서버 구조



1.7 웹 컴포넌트(Web Component)의 개요 - 1

-1.7.1 웹 컴포넌트의 이해와 종류

- 웹 어플리케이션 개발 시 사용되는 HTML, JSP, Servlet 및 이미지 파일
- 똑같은 페이지는 HTML인 정적 컴포넌트로 작성,
- 변경되는 페이지 및 데이터베이스 연동같은 프로그래밍이 필요한 페이지는 JSP/Servlet 같은 동적 컴포넌트로 작성

1) HTML 컴포넌트

- 파일 확장자는 .html로 작성되며 웹 브라우저에게 웹 페이지의 구조와 내용을 전달할 목적으로 작성
- 태그로 구성되어 있으며 브라우저는 HTML을 읽으면서 텍스트 주위에 있는 모든 태그들을 해석
- 태그는 작성한 텍스트의 구조와 의미에 대해 브라우저에게 알려주는 역할 수행

```
<html>
  <head>
    <title>HTML 타이틀입니다.</title>
  </head>
  <body>
    <h1>이곳은 제목입니다.</h1>
    <p>이곳은 문단입니다.</p>
  </body>
</html>
```

1.7 웹 컴포넌트(Web Component)의 개요 - 2

2) JSP 컴포넌트

- 파일 확장자는 .jsp로 작성되며 동적인 웹 페이지를 작성하기 위한 컴포넌트
- 정적인 HTML 태그와 동적인 JSP 태그가 혼합된 구조
- JSP 태그 안에 자바 코드를 삽입하여 자바 프로그래밍으로 실행
- JSP는 자바를 실행 시킬 수 있는 웹 컨테이너라는 특별한 환경에서 동작

-Model 1 Architecture

- ▶JSP만을 사용하여 웹 어플리케이션을 개발하는 모델 방법
- ▶발이 비교적 쉽기 때문에 경량의 웹 사이트 개발시 주로 사용
- ▶화면을 구현하는 코드(Presentation Logic)와 자바 프로그래밍 코드(Business Logic)가 혼합된 형태이기 때문에 유지 보수가 어려우며 확장성이 떨어짐
- ▶HTML 코드를 사용하기 쉽기 때문에 usiness Logic 보다는 Presentation Logic 처리에 적합

```
<%@ page contentType="text/html; charset=EUC-KR" %>
<html>
  <head>
    <title>JSP 타이틀입니다.</title>
```

```

</head>
<body>
<%
    String name = "홍길동";
%>
    이름은<%= name %>
</body>
</html>

```

1.7 웹 컴포넌트(Web Component)의 개요 - 3

3) Servlet 컴포넌트

-파일 확장자는 .java로 작성되며 동적인 웹 페이지를 작성하기 위한 컴포넌트
 -JSP와는 다르게 모든 코드가 자바 코드로 구성되며 웹 컨테이너가 필요

-Model 2 Architecture

- ▶서블릿과 JSP를 모듈화하여 개발하는 모델 방법
- ▶일반적으로 MVC 패턴이라고 알려진 모델 방법론으로서, 서블릿은 Business Logic 처리를 담당하고 JSP는 Presentation Logic 처리를 담당

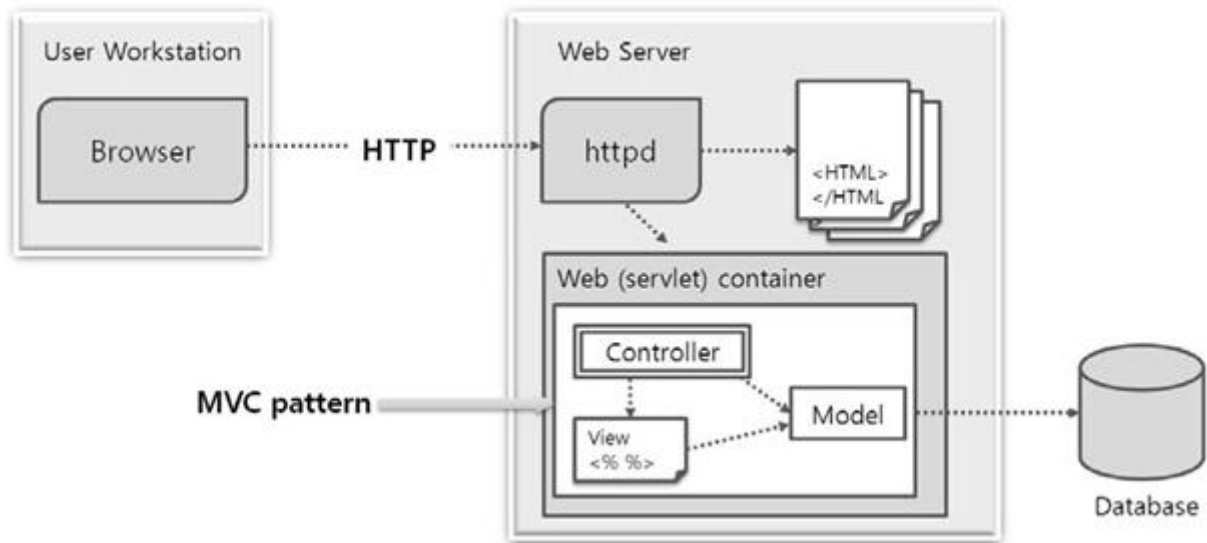
-1.7.2 웹 컨테이너(Web Container)

- JSP와 Servlet 컴포넌트를 관리하는 역할을 담당
- 웹 컴포넌트는 Java SE 환경처럼 main 메서드 역할을 담당하는 시작점(starting point)이 없으며, 단지 이벤트 방식처럼 동작
- 이벤트는 웹 브라우저를 통해서 사용자가 웹 서버에 요청을 보내는 상황을 의미
- 클라이언트 요청에 의해서 웹 컴포넌트들을 생성하고 제거하는 작업을 웹 컨테이너가 담당
- 대표적인 웹 컨테이너로 아파치그룹의 Tomcat 및 resin 등이 있으며 일반적으로 Tomcat 컨테이너를 가장 많이 사용

1.7 웹 컴포넌트(Web Component)의 개요 - 4

-1.7.3 Model 2 Architecture(MVC 패턴)

- Servlet과 JSP를 각 역할에 맞게 모듈화하여 개발하는 웹 어플리케이션 개발 방법론
- 모듈화하는 방법을 사용하기 때문에 개발 초기에는 복잡하고 어렵지만 유지 보수가 향상되며 표준 프레임워크인 Spring도 MVC 기반으로 구성
- Model 2 Architecture인 MVC 패턴의 구조도



1.7 웹 컴포넌트(Web Component)의 개요 - 5

-Model

- ▶ 웹 어플리케이션의 Business Logic을 담당하는 모듈로서 일반 자바 클래스로 구현
- ▶ 데이터베이스 연동 시 필요한 DAO, DTO 클래스 파일들에 해당

-View

- ▶ 사용자와의 직접적인 화면 구성을 담당하는 모듈로서 JSP로 구현
- ▶ Model의 데이터를 HTML로 보여주거나, 사용자가 입력하는 화면 구성을 담당

-Controller

- ▶ Model과 View는 논리적으로 밀접한 관련을 갖지만, 물리적으로는 관련이 없음
- ▶ 서로간의 의존성을 감소시킴으로써 유지 보수를 쉽게 할 목적으로 분리시켜 작업이 이루어짐
- ▶ Model과 View를 관리하는 역할의 Controller가 필요하며, Servlet로 구현
- ▶ Controller는 여러 개의 Model 클래스 파일과 여러 개의 View들 중에서 적절하게 선택하여 실행시키는 핵심 역할을 담당

1.7 웹 컴포넌트(Web Component)의 개요 - 6

-1.7.4 JSP 및 Servlet 스펙 버전

-현재 JSP 2.2와 Servlet 3.0이 가장 최신 스펙이고, 이 스펙에 맞추어진 웹 컴포넌트를 관리하는 톰캣의 최신 컨테이너는 Tomcat 7.X

<http://tomcat.apache.org/tomcat-7.0-doc/index.html>



1.7 웹 컴포넌트(Web Component)의 개요 - 7

이전 버전의 스펙과 비교

- 가장 큰 특징은 어노테이션 설정이라고 할 수 있으며 JDK 5.X부터 추가된 기능
- XML 파일 등을 이용하여 환경 설정 및 추가 정보를 등록하는 방법 대신에 자바 코드에 직접 설정하는 기술이며 '@'으로 시작
- 이전 버전에서는 웹 어플리케이션에서 필요한 주요 설정 작업들을 배치 지시자(Deployment Descriptor)라고 하는 web.xml 파일에 설정해서 사용
- Servlet 3.0 부터는 @으로 시작하는 어노테이션을 직접 서블릿에 사용하여, web.xml 파일 없이 주요 설정 작업 처리 가능
- 설정 환경정보를 알기 위해서 web.xml 파일을 살펴보지 않고도 서블릿 코드만을 이용하여 설정 정보를 쉽게 파악할 수 있으며, 개발 작업도 훨씬 간편해 질수 있음

1.8 웹 어플리케이션 개발 환경 구축 - 1

-웹 어플리케이션의 개발 환경 구축 작업 순서

- ① JDK 설치
- ② 이클립스 개발 툴 설치
- ③ 톰캣(웹 컨테이너) 설치
- ④ 이클립스와 톰캣 연동
- ⑤ 이클립스 글꼴 변경 및 소스 코드 라인 추가

-1.8.1 JDK(Java Development Kit) 설치

-자바 및 JSP/Servlet를 이용한 웹 어플리케이션 프로그램을 개발하기 위한 개발 툴킷

-'http://java.sun.com'에 접속하여 무료로 다운로드 가능

-JDK와 JRE의 설치 경로 지정

-1.8.2 이클립스 개발 툴 설치

- 'http://www.eclipse.org'에 접속하여 무료로 다운로드 가능
- Java EE Developers 다운로드
- 미러링 사이트 선택, 설치 경로 지정, 작업 공간 지정

1.8 웹 어플리케이션 개발 환경 구축 - 2

- 1.8.3 톰캣 (웹 컨테이너) 설치
 - 'http://tomcat.apache.org'에서 무료로 다운로드 받아서 설치
 - 다운로드 경로 지정 후 압축 해제
- 1.8.4 이클립스와 톰캣 연동
 - 이클립스를 실행
 - 사용할 서버를 생성
 - 톰캣의 홈 디렉토리를 지정
- 톰캣 컨테이너의 환경 설정
 - ① Server Locations 설정
 - ② webapps 경로(웹 컴포넌트 파일들을 저장할 주요 디렉터리) 설정
 - ③ 개발자가 작성한 웹 어플리케이션 설정 파일을 외부 XML 파일로 관리토록 설정
 - ④ 톰캣 컨테이너가 기본으로 사용하는 Port 변경
- 서버 컨테이너를 구동시키기 위해서 Start 작업과 종료하기 위한 Stop 작업 수행

1.8 웹 어플리케이션 개발 환경 구축 - 3

- 1.8.5 이클립스 글꼴 변경 및 소스 코드 라인 추가
 - [Window] 메뉴의 [Preferences]에서 설정
 - General-[Appearance]-[Colors and Fonts]를 선택
 - [Basic]-[Text Font]를 선택하고 오른쪽 상단의 [Edit] 버튼을 클릭하여 원하는 글꼴로 변경
 - 소스 코드의 라인 번호 추가는 임의의 파일을 선택하고, 소스 코드의 왼쪽 회색 바에서 마우스 오른쪽 버튼을 클릭하여 바로 가기 메뉴에서 [Show Line Numbers]를 선택

서블릿(Servlet)의 이해

2.1_서블릿(Servlet)의 개요

2.2_서블릿 맵핑

2.3_서블릿 아키텍처 및 핵심 API

2.4_서블릿 LifeCycle 메서드

2.5_서블릿 응답 처리

2.6_어노테이션을 이용한 서블릿의 선처리 및 후처리 작업

2.1 서블릿(Servlet)의 개요 - 1

-서블릿은 웹 컨테이너에 의해서 관리

-다양한 클라이언트 요청에 의해서 동적인 콘텐츠(content)로 응답 가능한 자바 기반의 웹 컴포넌트

-서블릿 웹 컴포넌트의 특징

- ▶자바기반의 웹 컴포넌트로서 java 확장자를 가짐
- ▶클라이언트의 요청에 의해서 동적으로 실행(다양한 클라이언트 요구 사항을 처리 가능)
- ▶클라이언트는 브라우저를 이용한 URL 지정을 통해 서블릿에 요청 가능
- ▶서블릿의 응답 결과는 일반적으로 HTML 형식으로 서비스(자바 코드를 이용해서 클라이언트에 HTML 코드로 전송하는 추가 작업이 필요)
- ▶MVC 패턴을 적용하여 웹 어플리케이션을 개발한다면, 서블릿이 아닌 JSP에서 HTML 코드를 작성
- ▶서블릿은 반드시 웹 컨테이너에 의해서 관리되며, 자바 스레드로 동작되기 때문에 효율적으로 사용이 가능
- ▶MVC 패턴의 Controller 역할로서 서블릿이 사용

2.1 서블릿(Servlet)의 개요 - 2

2.1.1 HelloServlet 작성하기

-이클립스를 실행시키고 먼저 Java EE 퍼스펙티브를 선택

-이클립스 메뉴에서 [File]-[New]-[Dynamic Web Project]를 선택

-새로운 웹 프로젝트를 생성하는 화면에서 값 설정

-Project name에는 ServletTest문자열을 지정

-Target runtime에는 Apache Tomcat 7.0으로 지정

-Dynamic web module version에는 3.0으로 지정

-자바 소스 및 클래스 파일의 저장 폴더 경로 정보 확인

-물리적 파일인 웹 컴포넌트를 저장하는 디렉터리와 웹 컨테이너가 논리적으로 관리하는 이름인 Context를 지정

-이클립스의 Project Explorer 창에 ServletTest 프로젝트 추가 확인

-서블릿을 추가하기 위해서 ServletTest 프로젝트를 선택하고 메뉴에서 [File]-[New]-[Servlet]을 선택

2.1 서블릿(Servlet)의 개요 - 3

-서블릿 이름과 패키지명 및 부모 클래스를 지정

- ▶java package에는 'com.test'로 지정
- ▶class name에는 'HelloServlet'로 지정
- ▶Super class에는 자동으로 'javax.servlet.http.HttpServlet'로 지정

-서블릿 정보를 배치 지시자(web.xml)에 설정

-URL mappings의 '/HelloServlet'을 선택하고 [Edit] 버튼을 클릭

-Pattern 입력란에 /Hello를 입력하고 [OK] 버튼을 클릭

-URL Mappings 값을 원하는 값으로 변경한 후에 [Next] 버튼을 클릭

-서블릿에서 구현해야 되는 메서드를 설정하는 화면에서 doGet 메서드만 체크한 후에 [Finish] 버튼을 클릭

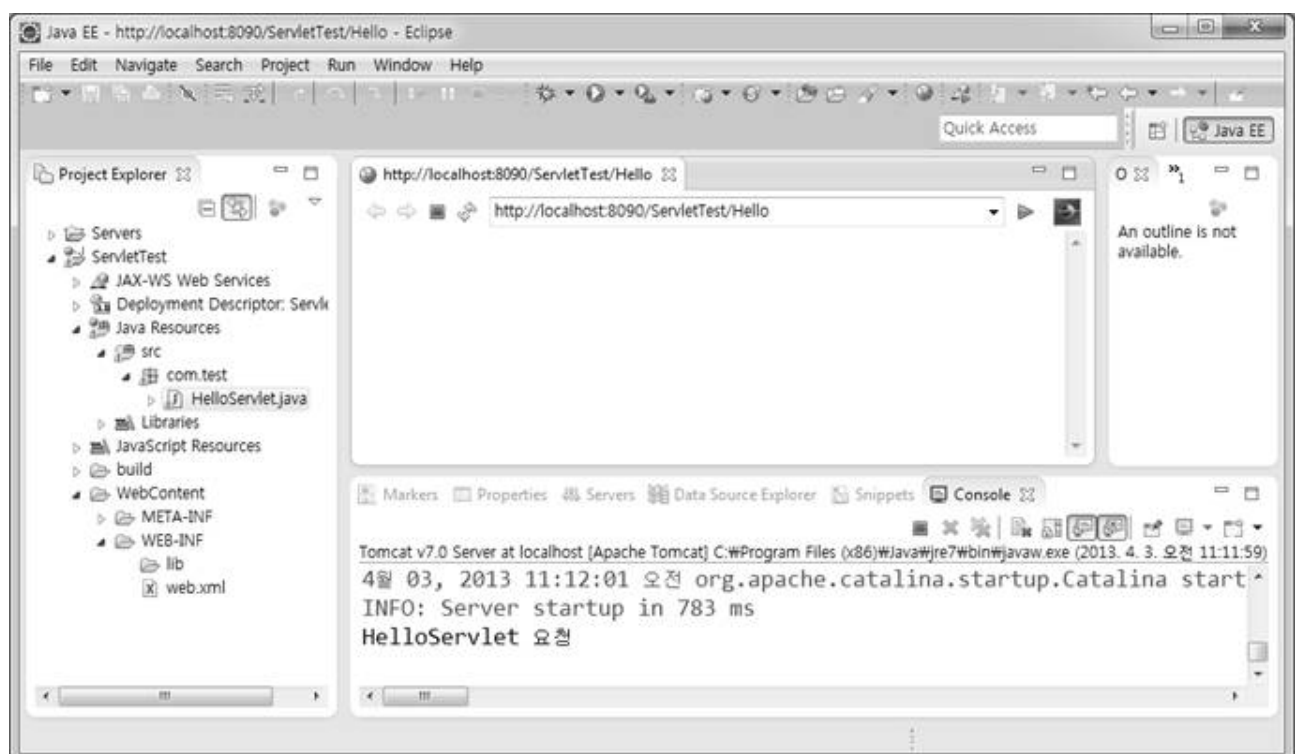
-요청 처리가 제대로 실행되었는지를 확인하기 위해서 doGet 메서드 내에 "HelloServlet요청"이라는 문자열을 출력하는 코드를 추가

-src 폴더에 있는 HelloServlet.java 파일을 선택하고 마우스 오른쪽 버튼을 클릭하여 바로 가기 메뉴에서 [Run As]-[Run on Server]를 선택 Tomcat 7.0 Server를 선택하고 [Next] 버튼을 클릭

2.1 서블릿(Servlet)의 개요 - 4

-작성한 ServletTest 컨텍스트가 톰캣 컨테이너에 등록이 되었으며 실행을 위해서 [Finish] 버튼을 클릭

-실행된 최종 결과 화면에서 자동으로 이클립스 안에서 브라우저가 실행되고 URL 입력란에 'http://localhost:8090/ServletTest/Hello'



2.2 서블릿 맵핑 - 1

-2.2.1 web.xml에 등록하는 방법

-Servlet 2.5까지 사용하던 방법이며 Servlet 3.0에서도 사용 가능

-WEB-INF 폴더안의 web.xml 파일에 <servlet> 태그와 <servlet-mapping> 태그를 사용하여 설정

-여러 개의 서블릿 등록이 가능하며 주의할 점

▶url-pattern 값에는 임의의 값으로 지정 가능하지만, 반드시 '/'를 사용

▶<servlet> 태그의 <servlet-name> 값과<servlet-mapping> 태그의 <servlet-name> 값도 임의의 값으로 지정 가능하지만, 반드시 일치해야 함

2.2 서블릿 맵핑 - 2

-2.2.2 @WebServlet 어노테이션(annotation) 이용하는 방법

-어노테이션은 JDK 5.X부터 추가된 기능

-XML 파일 등을 이용하여 환경 설정 및 추가 정보를 등록하는 방법 대신에 자바 코드에 직접 설정하는 기술로서 '@'으로 시작

-어노테이션을 사용하지 않으려면 web.xml 파일의 <web-app> 태그에 다음 속성 값을 지정

metadata-complete="true"

[예] <web-app metadata-complete="true"

Servlet 3.0에서 추가된 어노테이션 종류

@WebServlet	@WebFilter	@WebInitParam	@WebListener	@MultipartConfig
@DeclareRoles	@EJB	@EJBs	@Resource	@Resources
@PersistenceContext	@PersistenceContexts	@PersistenceUnit	@PersistenceUnits	@PostConstruct
@PreDestroy	@RunAs			

2.2 서블릿 맵핑 - 3

1) 서블릿 맵핑명만 지정하는 방식

@WebServlet("/맵핑명")

```
public class MyServlet extends HttpServlet{ .. }
```

2) 추가 속성을 이용하는 방식

서블릿 별명과 urlPatterns 속성을 사용하여 여러 개의 맵핑명을 지정할 수 있는 방식

@WebServlet(name="서블릿별명", urlPatterns= { "/맵핑명", "/맵핑명2" })

```
public class MyServlet extends HttpServlet{ .. }
```

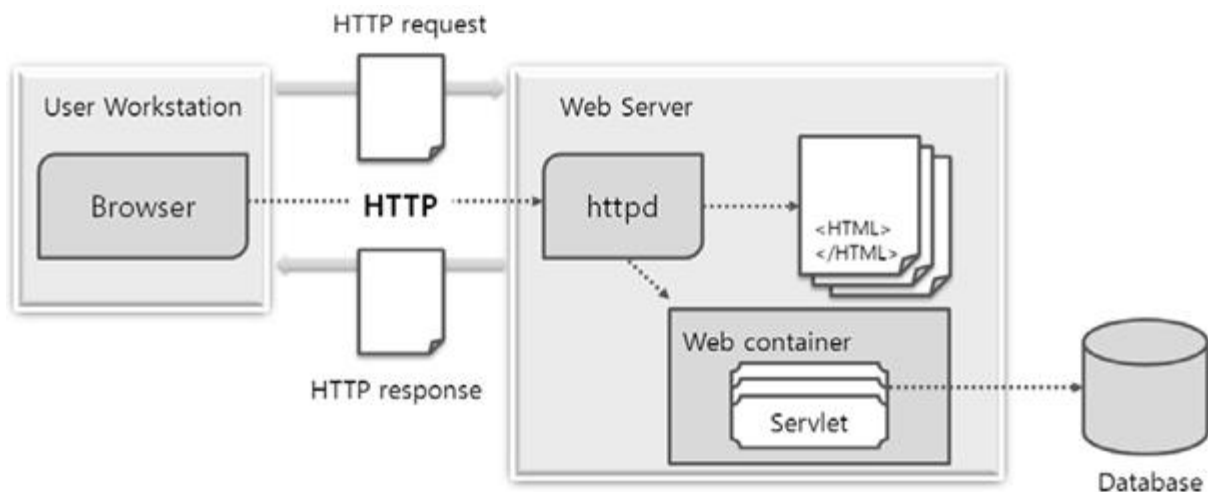
여러 개의 맵핑명을 지정할 수 있는 다른 방식은 value 속성을 사용하는 것

```
@WebServlet( name="/서블릿별명", value= { "/맵핑명", "/맵핑명2" } )
```

```
public class MyServlet extends HttpServlet{ .. }
```

2.3 서블릿 아키텍처 및 핵심 API – 1

서블릿을 사용하는 경우의 웹 아키텍처



클라이언트에서 웹 브라우저를 이용하여 적절한 URL 형식으로 서블릿에 요청하면, 웹 컨테이너에서 서블릿을 실행하고 결과 값을 html로 구성하여 클라이언트로 응답 처리

javax.servlet.http.HttpServletRequest : HTTP Request인 요청과 관련된 핵심 API

javax.servlet.http.HttpServletResponse : HTTP Response인 응답과 관련된 핵심 API

2.3 서블릿 아키텍처 및 핵심 API – 2

-2.3.1 HttpServletRequest API

-HTTP Request 관련 작업을 처리하는 핵심 API

-클래스가 아닌 인터페이스로 제공

- | | |
|-----------------------------------|---|
| • getHeader(String name):String | • setCharacterEncoding(String encoding) |
| • getHeaderNames():Enumeration | • setAttribute(String name, Object obj) |
| • getCookies():Cookie[] | • getAttribute(String name):Object |
| • getRequestURI():String | • removeAttribute(String name) |
| • getServletPath():String | • getParameter(String name) |
| • getSession(boolean):HttpSession | • getParameterNames():Enumeration |
| • getSession():HttpSession | • getParameterValues(String name):String[] |

-2.3.2 HttpServletResponse API

-HTTP Response 관련 작업을 처리하는 핵심 API

-클래스가 아닌 인터페이스로 제공

- addCookie(Cookie c)
- addHeader(String name, String value)
- encodeURL(String url)
- getStatus()
- sendRedirect(String loc)
- getWriter():PrintWriter
- getOutputStream():ServletOutputStream
- setContentType(String type)

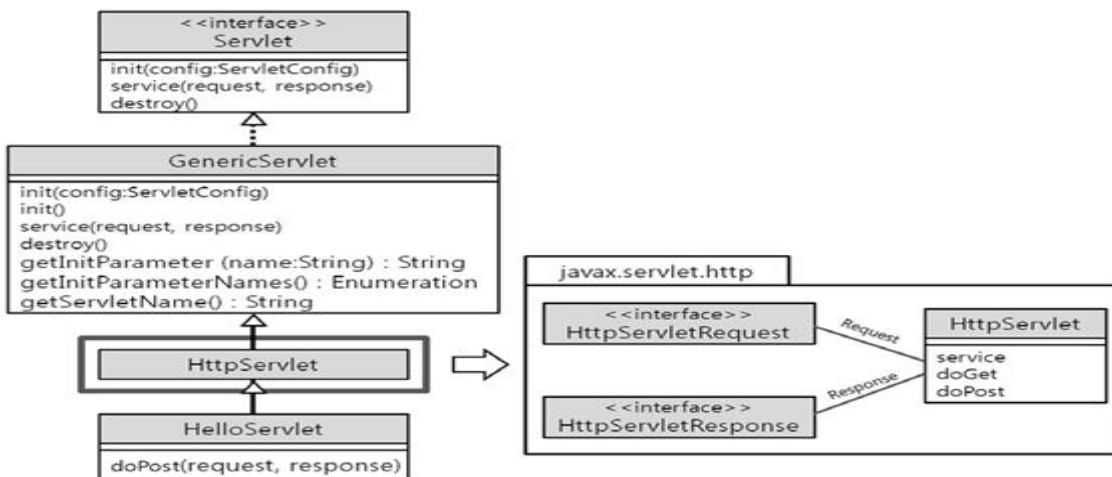
2.3 서블릿 아키텍처 및 핵심 API - 3

-2.3.3 HttpServlet API

-서블릿을 구현하기 위한 핵심 API

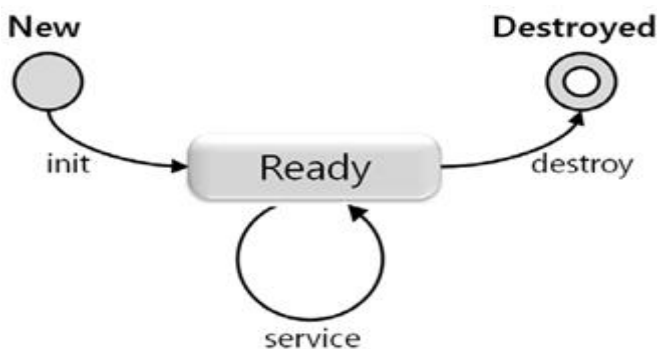
-일반 클래스가 아닌 추상 클래스로 제공

-HttpServlet의 계층 구조



2.4 서블릿 LifeCycle 메서드

서블릿의 LifeCycle 상태도



1) init 메서드

웹 컨테이너에 의해서 서블릿 인스턴스가 처음 생성될 때, 단 한번 호출
서블릿에서 필요한 초기화 작업 시 주로 사용

2) service 메서드

클라이언트가 요청할 때마다 호출

클라이언트가 원하는 동적인 처리 작업 시 필요

일반적으로 service 메서드보다는 doGet 또는 doPost 메서드를 사용

3) destroy 메서드

서블릿 인스턴스가 웹 컨테이너에서 제거될 때 호출

init 메서드에서 구현했던 초기화 작업을 반납 처리하는 작업 시 주로 사용

2.5 서블릿 응답 처리

-클라이언트에서 서블릿으로 요청을 하면 서블릿은 처리한 결과를 html 형식으로 응답 처리

-실제로 MVC 패턴을 적용한 웹 어플리케이션 개발에서는 jsp에서 응답 처리를 담당

-서블릿에서 응답 처리와 관련된 API는 'HttpServletResponse'

-response.setContentType("text/html;charset=EUC-KR")

▶클라이언트인 웹 브라우저에게 처리할 데이터의 MIME 타입을 알려주는 메서드

▶기본은 일반 텍스트를 의미하는 'text/plain'

▶실습에서는 html 형식의 전송이므로 'text/html'로 지정

▶한글 처리를 위해서 'charset=EUC-KR'을 추가 지정

-response.getWriter()

▶서블릿 및 JSP를 이용한 응답 처리는 기본적으로 자바 I/O 기술을 이용

▶자바 출력을 위한 OutputStream 또는 Writer 클래스를 사용

▶서블릿에서는 getWriter() 메서드를 이용한 PrintWriter와 getOutputStream() 메서드를 이용한

▶ServletOutputStream 클래스를 사용

▶문자 데이터를 처리하기 위해서는 PrintWriter를 이용

▶바이너리(binary) 데이터를 위해서는 ServletOutputStream 클래스를 사용

2.6 어노테이션을 이용한 서블릿의 선처리 및 후처리 작업

-2.6.1 @PostConstruct를 이용한 선처리 작업

-서블릿의 LifeCycle 메서드인 init 메서드가 호출되기 전에 수행되는 선처리 작업 메서드에 지정 가능

-반드시 리턴 타입은 void로 지정하고, 예외 클래스를 throws 할 수 없음

@PostConstruct

```
public void postConstruct( ) {
```

```
    ...
```

```
}
```

2.6.2 @PreDestroy를 이용한 후처리 작업

서블릿의 LifeCycle 메서드인 destroy 메서드가 호출된 후에 수행되는 후처리 작업 메서드에 지정 가능

반드시 리턴 타입은 void로 지정하고, 예외 클래스를 throws 할 수 없음

@PreDestroy

```
public void cleanup( ) {  
    ...  
}
```

HTML5 Form 태그와 서블릿

3.1_HTML5의 개요

3.2_HTML5의 Form

3.3_서블릿에서 파라미터 처리

3.1 HTML5의 개요 - 1

- HTML5는 기존의 HTML, XHTML, HTML DOM 기술을 포괄하는 웹 표준 기술
- 잘 사용하지 않는 XHTML 보다 기존에 많이 사용되어진 HTML을 업그레이드하는 방향으로 애플, 모질라, 오페라 벤더들이 WHATWG (Web HyperText Application Technology Working Group)을 구성하여 진행하고 있는 프로젝트
- 아직까지 표준안으로 확정되지 않고 진행 중
- 모든 브라우저가 HTML5를 지원하지는 못하지만, 기존에 사용되던 HTML, CSS, DOM, Javascript 기술이 기반이 되고 플랫폼에 독립적이며 플래시(Flash)와 같은 외부 플러그인을 대체 할 수 있는 기술
- 현재 최신 브라우저에서 거의 대부분이 지원되는 추세이며, 앞으로 모든 브라우저가 지원할 것으로 예상

3.1 HTML5의 개요 - 2

-HTML5의 주요 특징

- ▶자유로운 2D 그래픽 처리가 가능
- ▶동영상이나 음성 재생이 쉽게 처리 가능
- ▶클라이언트 측에 데이터 저장 기능이 가능
- ▶offline에서도 동작하는 애플리케이션이 가능
- ▶백그라운드 처리 수행이 가능
- ▶서버로부터의 데이터 푸시나 서버와의 쌍방향 통신 수행이 가능

- ▶로컬 파일의 내용 읽기가 가능
- ▶웹 접근성이 향상
- ▶calendar, time, date, url, search 같은 새로운 폼 관련 태그 추가가 제공

3.2 HTML5의 Form – 1

- HTML 폼은 CGI 및 JSP/Servlet을 포함한 웹 어플리케이션을 개발할 때 반드시 필요한 기본이 되는 요소
- 클라이언트와 웹 어플리케이션 간에 데이터 전달을 가능하게 하는 다양한 태그들을 제공
- 회원 가입, 로그인 등과 같이 사용자가 직접 데이터를 입력할 수 있도록 지원
- 폼 필드의 컴포넌트

The image displays a variety of HTML5 form elements arranged in two columns. The left column includes a text input with the placeholder '홍길동', a password input with masked dots, a search input with a '찾아보기...' button, a group of three checkboxes labeled 'value1', 'value2', and 'value3' (where 'value2' is checked), and a group of three radio buttons labeled 'value1', 'value2', and 'value3' (where 'value2' is selected). The right column features a '일반단추' (General Button), a multi-line text area labeled '멀티라인' with a vertical scrollbar, a '보내기단추' (Send Button), and a '지우기단추' (Delete Button).

3.2 HTML5의 Form – 2

- input 태그의 공통되는 3가지 속성
- type(필수 속성)
 - ▶input 태그의 종류를 지정할 때 사용 가능
 - ▶text, submit, reset, checkbox, radio, password, hidden, number, date, color, range 값들을 지정할 수 있음
- name(필수 속성)
 - ▶사용자가 입력한 데이터는 폼 데이터 또는 파라미터(parameter)라고 부르며, 웹 서버로 전송이 가능
 - ▶파라미터는 'name=value' 형식으로 전송
 - ▶value는 사용자가 입력한 데이터이고 name은 input 태그의 name에 해당
 - ▶만약 파라미터가 여러 개인 경우에는 &를 사용하여 여러 파라미터를 구분
 - ▶서버로 전송된 파라미터는 JSP 및 Servlet에서 name 값을 이용하여 value를 얻을 수 있음
- value(선택 속성)
 - ▶사용자가 지정한 설정 값을 저장하며, 입력되는 모든 데이터는 문자열로 처리

3.2 HTML5의 Form – 3

-3.2.1 textfield 태그

-한 줄 데이터를 입력하는 용도로 사용되는 태그

형 식	<input type="text" name="username" value="홍길동" />
결 과	홍길동

-폼 데이터가 웹 서버에 파라미터로 전송되는 경우에는 'username=홍길동' 형식으로 전송

-일반적으로 value는 사용자가 지정한 값으로 설정하기 때문에 생략

3.2 HTML5의 Form – 4

-HTML5의 textfield 설정이 가능한 속성

-autofocus

▶화면이 보일 때 자동으로 포커스가 지정

-placeholder :

▶사용자가 입력할 데이터의 종류를 쉽게 알 수 있도록 흐리게 문자로 표시

▶사용자가 데이터를 입력하면 자동으로 제거(최신 크롬 브라우저 환경)

실습 폼

- 이름
-

-required

▶필수 입력 속성에 사용 가능데이터를 입력하지 않고 전송 버튼을 클릭하면, 데이터가 서버에 전송되지 않고 데이터를 입력하라는 경고문이 출력(최신 크롬 브라우저 환경)

실습 폼

- 이름
-

이 입력란을 작성하세요.

3.2 HTML5의 Form – 5

-3.2.2 checkbox 태그

여러 데이터 값을 체크하는 용도의 태그

일반적으로 NAME 값은 모두 동일한 값으로 지정하여 그룹으로 사용

형 식	<input name="fruit" type="checkbox" value="apple"/> 사과 <input name="fruit" type="checkbox" value="orange"/> 오렌지 <input name="fruit" type="checkbox" value="banana"/> 바나나
결 과	<input type="checkbox"/> 사과 <input checked="" type="checkbox"/> 오렌지 <input checked="" type="checkbox"/> 바나나

- 폼 데이터가 웹 서버에 파라미터로 전송되는 경우에는 'fruit=orange& fruit=banana' 형식으로 전송
- 체크하지 않은 apple은 전송되지 않음

3.2 HTML5의 Form – 6

-3.2.3 radio 태그

- 여러 데이터 값 중에서 하나만을 체크하는 용도의 태그
- checkbox와 마찬가지로 NAME 값은 모두 동일한 값으로 지정하여 그룹으로 사용

형 식	<input name="sex" type="radio" value="F"/> 여성 <input checked="" name="sex" type="radio" value="M"/> 남성
결 과	<input type="radio"/> 여성 <input checked="" type="radio"/> 남성

checked="checked" 속성 값을 이용하여 기본 값으로 설정이 가능

폼 데이터가 웹 서버에 파라미터로 전송되는 경우에는 'sex=M' 형식으로 전송

3.2 HTML5의 Form – 7

-3.2.4 password 태그

- 한 줄 데이터를 입력하는 Textfield 용도와 유사
- 입력시킨 데이터는 다음과 같이 암호화 되어 표시

형 식	<input ><="" name="passwd" td="" type="password" value="secret"/>
결 과	●●●●●●

폼 데이터가 웹 서버에 파라미터로 전송되는 경우에는 'passwd=secret' 형식으로, 실제 입력한 데이터가 전송

3.2 HTML5의 Form – 8

-3.2.5 hidden 태그

- 실제로는 존재하지만 웹 브라우저에서 hidden 형태로 사용하기 위한 태그

형 식	<input ><="" name="passwd" td="" type="password" value="secret"/>
-----	--


- 폼 데이터가 웹 서버에 파라미터로 전송되는 경우에는 'action=hidden Data' 형식으로 전송
- 일반적으로 사용자가 입력한 데이터가 아닌 직접 특정 값을 웹 서버로 전송하기 위해서 사용(상수 값 형태로 사용되는 경우)

3.2 HTML5의 Form – 9

-3.2.6 select 태그

-리스트 형태의 데이터를 사용하기 위한 태그

-select 태그에 NAME 값을 지정하고, option 태그에 VALUE 값을 설정

형 식	<pre><select name="smartphone"> <option value="iPhone5">아이폰5</option> <option value="Galaxy3" selected="selected">갤럭시3</option> <option value="Galaxy4">갤럭시4</option> </select></pre>
결 과	

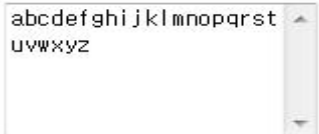
-selected="selected" 속성 값을 이용하여 기본 값으로 설정이 가능

-폼 데이터가 웹 서버에 파라미터로 전송되는 경우에는 'smartphone= Galaxy3' 형식으로 전송

3.2 HTML5의 Form – 10

-3.2.7 textarea 태그

-여러 줄의 많은 데이터를 입력하는 용도로 사용되는 태그

형 식	<pre><textarea name="comment" rows="5" columns="20"> abcdefghijklmnopqrstuvwxyz </textarea></pre>
결 과	

-rows 속성을 이용하여 행의 개수를 지정하고 columns 속성을 이용하여 열의 개수를 지정

-VALUE 속성이 아닌 <textarea>와 </textarea> 태그 사이에 설정 값을 지정

-폼 데이터가 웹 서버에 파라미터로 전송되는 경우에는 'comment= abcdef.yz' 형식으로 전송

3.2 HTML5의 Form – 11

-3.2.8 number 태그

-HTML5에서 추가된 input 태그

-정수 값을 입력하는 경우에 사용

-설정 가능한 속성

▶min : 지정 가능한 최소 정수 값

▶max : 지정 가능한 최대 정수 값

▶step : 선택 시 증가 및 감소되는 값. 지정하지 않으면 자동으로 1씩 증가 및 감소



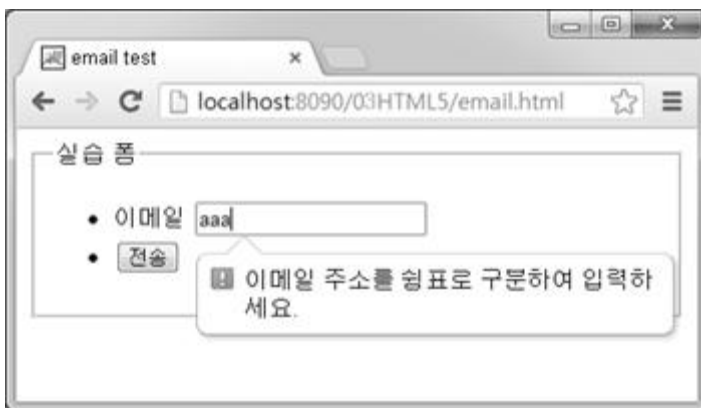
-3.2.9 email 태그

-HTML5에서 추가된 input 태그

-email을 입력하는 경우에 사용

-email 형식에 위배되면, 서버에 전송되지 않고 에러 문구를 표시

-multiple 속성을 지정하여 email을 복수로 지정이 가능



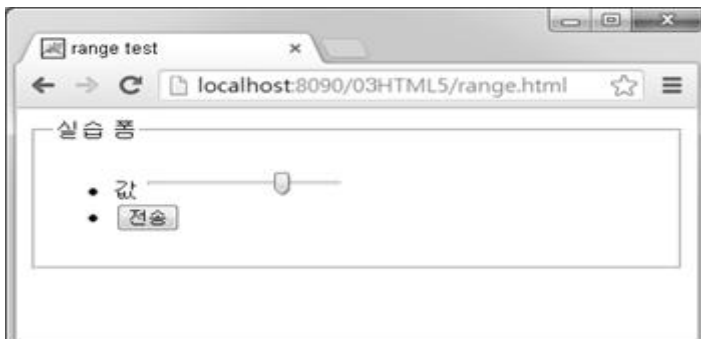
3.2 HTML5의 Form – 12

3.2.10 range 태그

-HTML5에서 추가된 input 태그

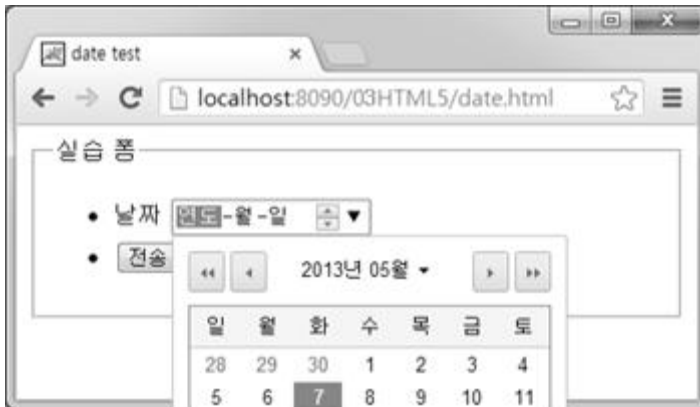
-특정 값을 범위로 지정하는 경우에 사용

-number 태그와 마찬가지로 min, max, step 속성 값을 지정



3.2.11 date 태그

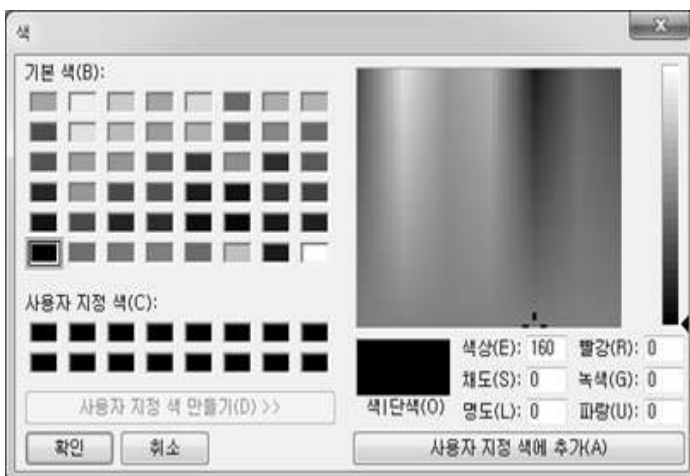
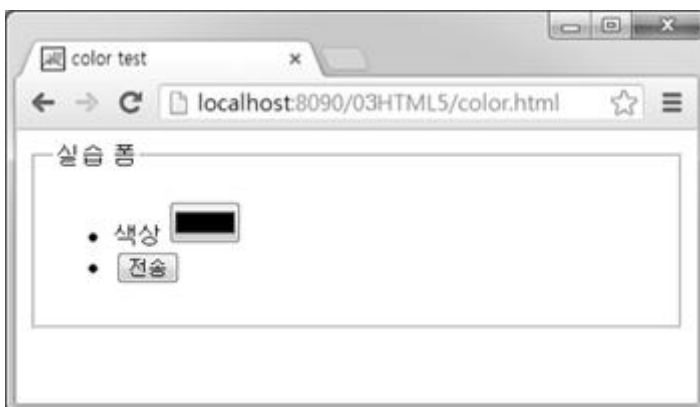
- HTML5에서 추가된 input 태그
- 날짜를 지정하는 경우에 사용



3.2 HTML5의 Form – 13

-3.2.12 color 태그

- HTML5에서 추가된 input 태그
- 색상을 지정하는 경우에 사용



3.2 HTML5의 Form – 14

-3.2.13 form 태그와 submit 태그

-form 태그

- ▶여러 가지 input 태그들을 포함하는 컨테이너(container) 역할을 담당하는 태그
- ▶웹 서버에 사용자가 입력한 데이터를 전송하기 위해서는 반드시 form 태그 내에 input 태그들을 지정해야 함
- ▶form 태그 내에 지정하지 않은 input 태그는 웹 서버로 전송되지 않음

-submit 태그

- ▶웹 서버에 사용자가 입력한 폼 데이터를 전송하기 위한 버튼 태그
- ▶사용자는 form 태그안의 input 태그에 값을 설정하고 submit 버튼을 선택함으로써, 원하는 데이터를 웹 서버에 전송
- ▶form 태그와 submit 태그는 일반적으로 같이 사용

3.2 HTML5의 Form – 15

-form 태그의 주요 속성들

-action

- ▶submit 버튼을 선택했을 때, 웹 서버에서 요청을 처리할 웹 컴포넌트를 지정
- ▶html 및 jsp와 servlet 설정이 모두 가능

<form action="login.html"> // login.html로 요청

<form action="login.jsp"> // login.jsp로 요청

<form action="login"> // login로 매핑된 서블릿으로 요청

-method

- ▶웹 브라우저에서 웹 서버로 요청하는 처리 방법을 명시

① GET(get)

- ▶기본처리 방법으로서, 요청 파라미터 값이 웹 브라우저 URL에 명시되어 웹 서버로 전송 (HTTP Request의 요청 라인(Request Line)에 포함되어 전송)
- ▶파라미터 구분은 '&'로 처리하고, 웹 컴포넌트인 login과 파라미터 구분은 '?'를 사용

http://서버 IP 번호:포트번호/컨텍스트명/경로명/login?name=홍길동&age=20

3.2 HTML5의 Form – 16

- 웹 클라이언트에서 GET 방식으로 요청하면, 서버에서는 doGet 메서드가 처리
- GET 방식은 파라미터 길이에 제한이 있으며 URL에 명시적으로 파라미터 값이 보이기 때문에 보안에 취약
- 웹 사이트의 링크(link) 및 명시적인 URL 요청 등은 모두 기본 방식인 GET 방식으로 요청하는 형태

② POST(post)

- ▶ 요청 파라미터 값이 웹 브라우저 URL이 아닌 HTTP Request의 요청 본체(Request Body)에 포함되어 전송되기 때문에 웹 브라우저 URL에서 확인이 불가능
- ▶ 파라미터가 전송된다면 URL에 서버의 맵핑명만 표시되어 전송

<http://서버 IP 번호:포트 번호/컨텍스트명/경로명/login>

- ▶ 웹 클라이언트에서 POST 방식으로 요청하면, 서버에서는 doPost 메서드가 처리
- ▶ GET 방식과 비교해서 보안에 덜 취약
- ▶ 새로 고침을 선택하면 재요청되지 않고 사용자에게 재요청 의사를 묻는 정보창이 실행

`<form action="login.html" >` //method를 지정하지 않으면 기본 방식인 get 방식으로 처리

`<form action="login.jsp" method="get" >` // get 방식으로 처리

`<form action="login" method="post">` // post 방식으로 처리

3.3 서버에서 파라미터 처리 – 1

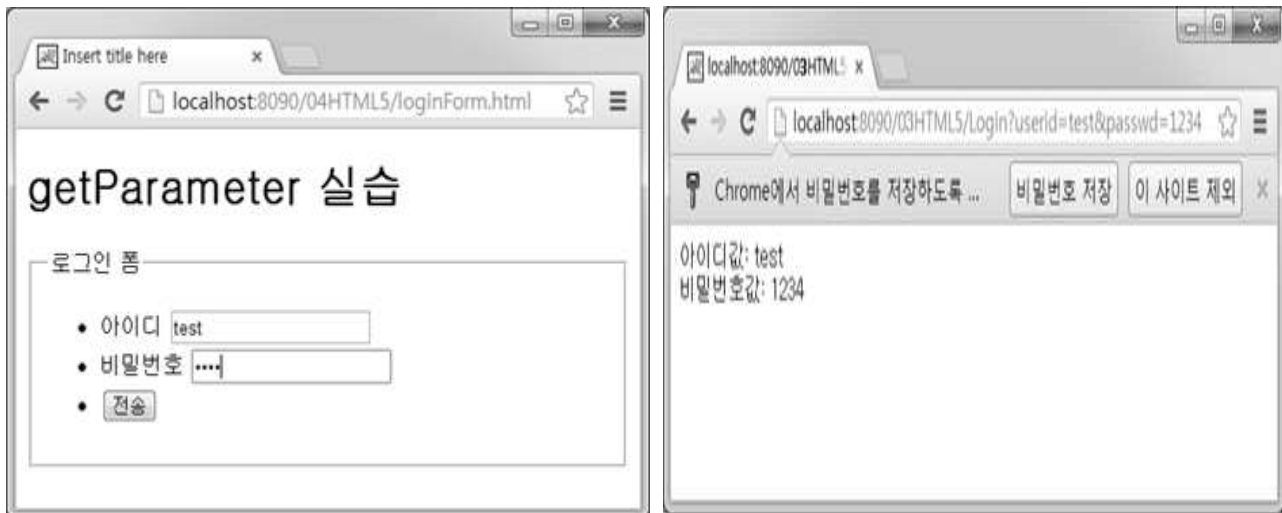
- HTML의 input 태그에 값을 지정하고 submit 버튼을 선택하면, 서버에 파라미터(폼 데이터)가 전송
- 요청받은 서버는 HttpServletRequest 객체의 3가지 메서드를 사용하여 파라미터 값을 얻을 수 있음

리턴 타입	메서드명	내용
String	getParameter(name)	name에 해당되는 파라미터 값을 리턴 만약 지정된 name의 파라미터 값이 없으면 null을 리턴
String []	getParameterValues(name)	checkbox, radio 태그와 같이 하나의 name에 여러 값을 가지는 경우에 주로 사용 Name에 해당되는 파라미터 값을 배열로 리턴
Enumeration	getParameterNames()	폼 태그 안에 여러 개의 input 태그가 있는 경우에 주로 사용 모든 name 값을 Enumeration 타입으로 리턴 얻은 name 값을 이용하여 파라미터 값을 얻음

3.3 서블릿에서 파라미터 처리 - 2

-3.3.1 getParameter(name) 메서드

-아이디와 비밀번호를 입력하고 전송버튼을 클릭하면, 파라미터 값을 서블릿에서 getParameter(name) 메서드를 사용하여 얻어서 결과 값을 브라우저에 출력

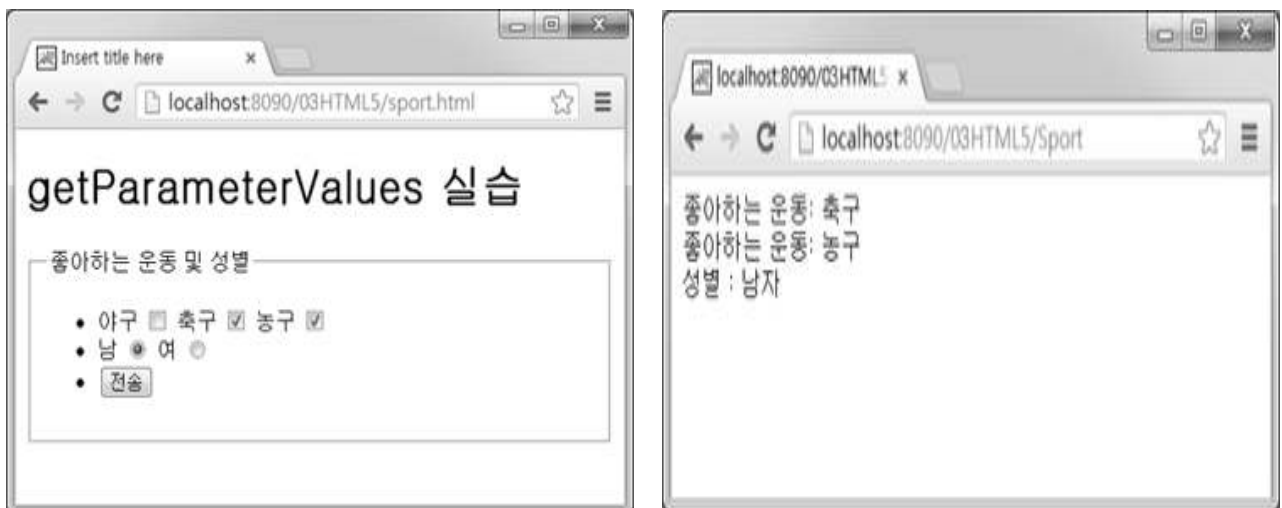


3.3 서블릿에서 파라미터 처리 - 3

-3.3.2 getParameterValues(name) 메서드

-하나의 name에 여러 값을 가지는 checkbox와 radio 태그를 이용

-파라미터 값을 서블릿에서 getParameterValues(name) 메서드를 사용하여 얻고 결과 값을 브라우저에 출력



3.3 서블릿에서 파라미터 처리 - 4

-3.3.3 서블릿의 한글 처리

- HTML에서 입력한 파라미터 값에 한글이 포함되어 있다면 반드시 한글 인코딩 작업 필요
- 기본적으로 브라우저에서 문자를 처리하는 방식과 서버에서 문자를 처리하는 방식이 일치해야 한글이 깨지지 않고 처리가 가능
- Tomcat 서버의 기본 문자 처리 방식은 ISO-8859-1 방식으로 처리하기 때문에 기본적으로 한글이 제대로 인식 안됨

1) POST 처리 방식

- 파라미터 값을 얻기 전에 서블릿의 doGet 또는 doPost 메서드내에서 request.setCharacterEncoding("EUC-KR") 메서드를 사용하여 한글 인코딩 처리

2) GET 처리 방식

- Tomcat의 설정 파일인 server.xml을 수정해야 함
- Servers 항목의 server.xml 파일을 선택하고, port="8090" 값을 가진 Connector 태그에 URLEncoder="EUC-KR" 값을 지정

서블릿 핵심 클래스

4.1_ServletConfig API를 활용한 초기화 파라미터 사용

4.2_ServletContext API

4.3_ServletContextListener API

4.4_Filter API

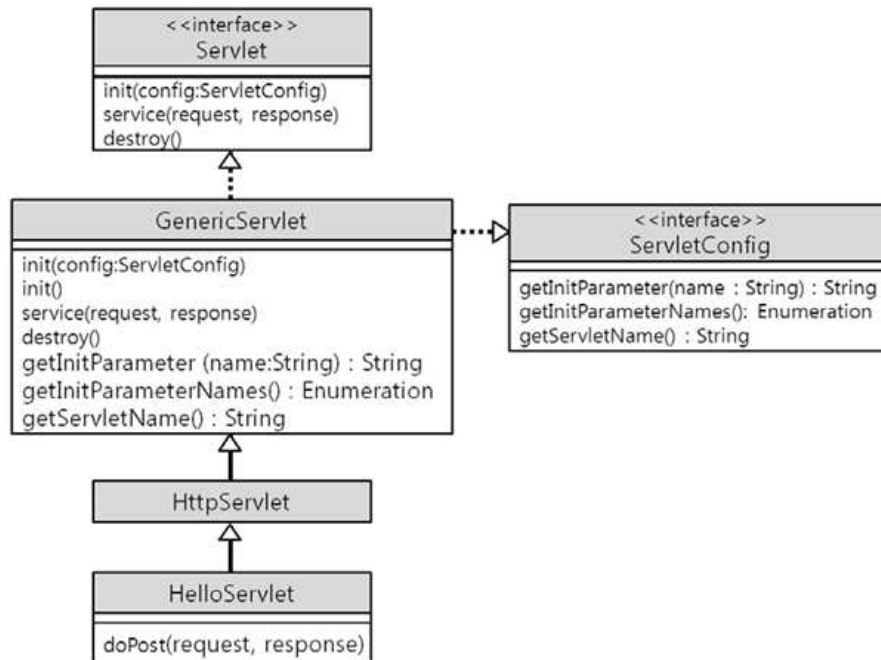
4.5_url-pattern의 개요

4.1 ServletConfig API를 활용한 초기화 파라미터 사용 - 1

- 서블릿이 초기화될 때, 공통적으로 적용해야 되는 작업들이 필요 경우(예:외부 파일 및 디렉터리 경로, JDBC에서 사용하기 위한 데이터베이스 경로, 계정 및 비밀번호와 같은 정보들)
- 이런 정보들을 서블릿에서 설정하지 않고 web.xml에서 설정한 후 서블릿에서 접근해서 사용
- 서블릿에서 설정하는 경우에는 정보가 변경되면 반드시 서블릿을 재컴파일 시켜야 함
- web.xml에서 설정하면 재컴파일 없이 변경된 정보를 참조할 수 있기 때문에 유지보수가 쉬워짐
- web.xml에 설정된 설정 값을 '초기화 파라미터(Initialization Parameter)라고 하며 ServletConfig API를 이용해서 접근 가능
- 여러 서블릿에서 공유해서 사용하지 못하고 <init-param>으로 등록된 서블릿에서만 사용 가능
- 서블릿 코드 내에서 @WebInitParam 어노테이션을 이용하여 초기화 파라미터를 등록할 수도 있음

4.1 ServletConfig API를 활용한 초기화 파라미터 사용 - 2

-ServletConfig API의 계층 구조

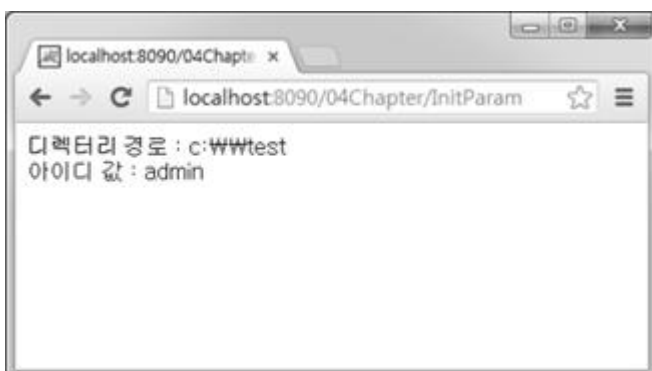


4.1 ServletConfig API를 활용한 초기화 파라미터 사용 - 3

-4.1.1 web.xml에 초기화 파라미터 등록

▶ServletConfig의 핵심 메서드

리턴 타입	메서드명	내용
String	getInitParameter(name)	name에 해당되는 파라미터 값을 리턴 만약 지정된 name의 파라미터 값이 없으면 null을 리턴
Enumeration	getInitParameterNames()	모든 초기화 파라미터 name 값을 Enumeration 타입으로 리턴 얻은 name 값을 이용하여 초기화 파라미터 값을 얻음
String	getServletName()	요청한 서블릿의 이름을 리턴한다.

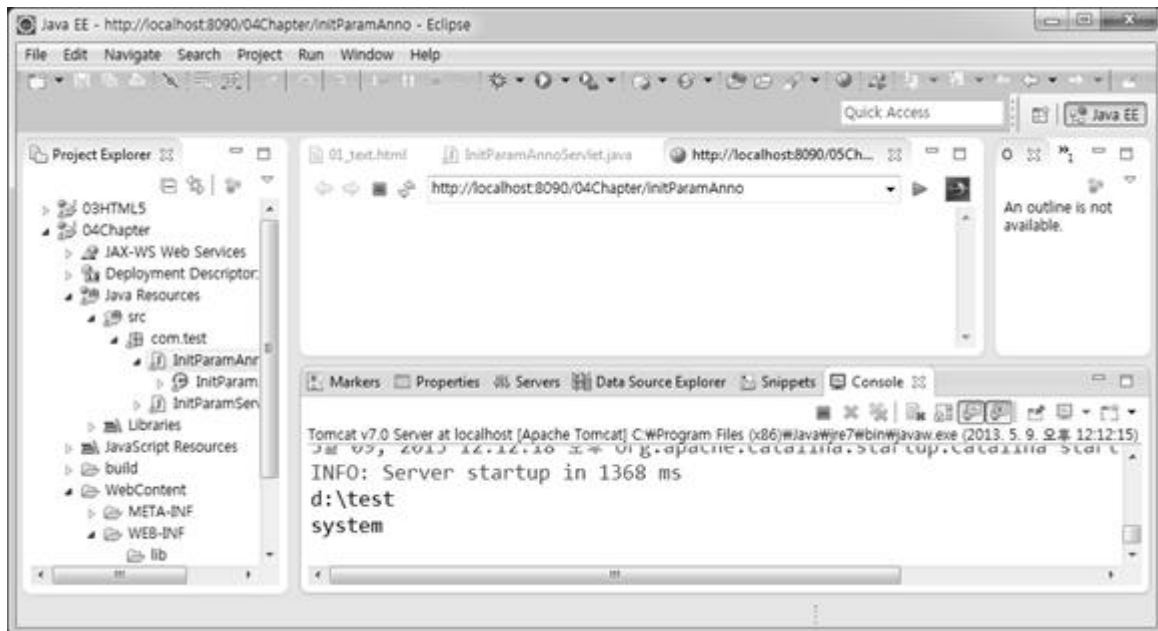


4.1 ServletConfig API를 활용한 초기화 파라미터 사용 - 4

-4.1.2 @WebInitParam 어노테이션을 이용한 초기화 파라미터 등록

- 서블릿 코드내에서 @WebInitParam 어노테이션을 사용하여 초기화 파라미터를 등록할 수 있음
- ServletConfig의 getInitParameter(name) 메서드를 사용하여 초기화 파라미터 값을 얻음

<http://localhost:8090/04Chapter/InitParamAnno>

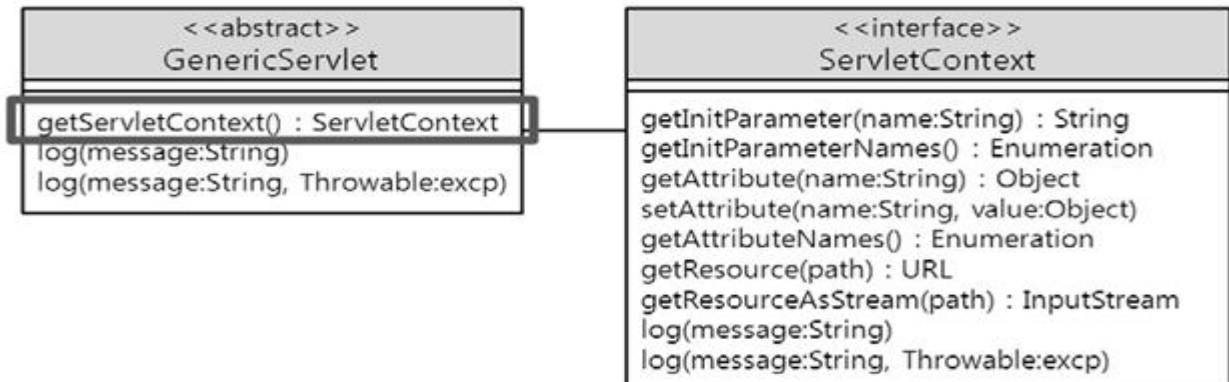


4.2 ServletContext API - 1

- 웹 어플리케이션에는 여러 가지 자원을 포함 가능
- html 파일, 미디어 파일, 이미지 파일, 다수의 JSP 파일과 서블릿 등이 유기적으로 동작
- ServletContext는 웹 어플리케이션(Context)마다 하나씩 생성되는 객체로서, 다수의 JSP 파일과 서블릿에서 공유해서 사용 가능
- ServletContext 객체는 웹 어플리케이션의 LifeCycle과 일치하기 때문에, 웹 어플리케이션이 Tomcat 컨테이너에 존재한다면 계속 사용 가능 → 'application scope'
- ServletContext 객체를 이용한 핵심 기능
- 여러 서블릿에서 사용 가능한 초기화 파라미터 사용가능
- 일반적으로 '컨텍스트 파라미터(Context Parameter)'라고 함
- 서블릿에서 파일 접근 가능(읽기 모드만 가능)
- application scope에 해당되는 속성(Attribute)을 저장하고 조회 가능

4.2 ServletContext API – 2

ServletContext API의 계층 구조



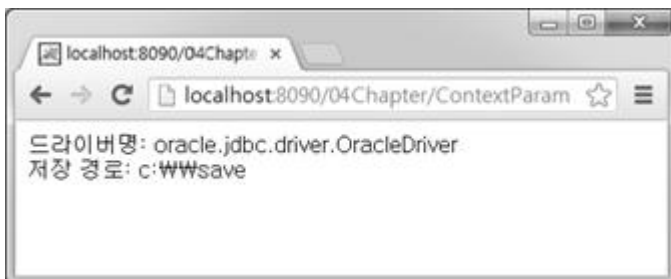
▶ServletContext의 핵심 메서드

리턴 타입	메서드명	내용
String	getInitParameter(name)	name에 해당되는 컨텍스트 파라미터 값을 리턴 만약 지정된 name의 파라미터 값이 없으면 null을 리턴
InputStream	getResourceAsStream(path)	웹 어플리케이션의 path 경로에 해당되는 파일을 읽기모드로 접근 가능
Void	setAttribute(name,value)	application scope 해당되는 속성 값을 저장할 때 사용 브라우저를 종료해도 속성 값을 사용 가능
Object	getAttribute(name)	name에 해당되는 속성 값을 리턴

4.2 ServletContext API – 3

-4.2.1 컨텍스트 파라미터(context parameter) 설정

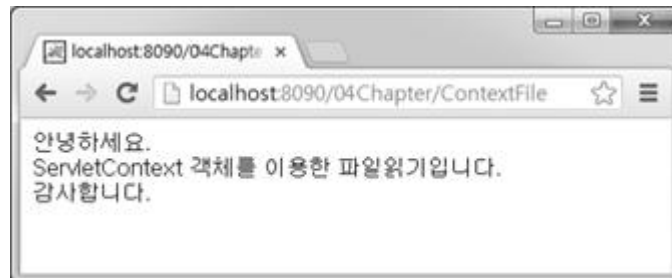
- 다수의 서블릿이 공통적으로 사용되는 특정 데이터가 필요하다면, 컨텍스트 파라미터를 사용하는 것이 바람직
- 초기화 파라미터는 특정 서블릿만이 파라미터 값을 사용 가능
- 컨텍스트 파라미터는 application scope이기 때문에 어플리케이션의 모든 서블릿이 공유해서 사용 가능
- 초기화 파라미터와 마찬가지로 web.xml에 등록하여 사용
- ServletConfig 대신에 ServletContext 객체의 getInitParameter(name) 메서드를 사용해서 컨텍스트 파라미터 값을 얻음



4.2 ServletContext API – 4

-4.2.2 서블릿에서 파일 접근(읽기 모드)

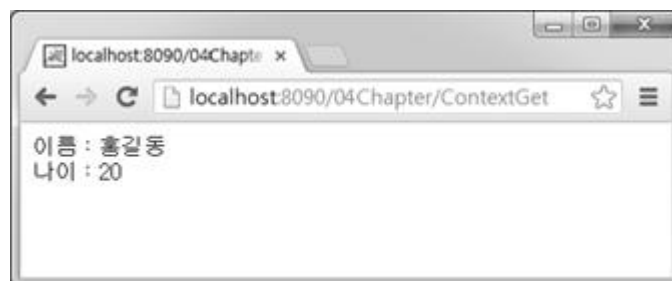
- 서블릿에서 웹 어플리케이션내의 특정 파일을 접근하기 위해서 ServletContext 객체를 사용가능
- 읽기 모드만 가능하고 쓰기는 불가능



4.2 ServletContext API – 5

-4.2.3 서블릿에서 속성(attribute) 설정 및 참조

- 웹 어플리케이션에서 브라우저를 종료해도 지속적으로 사용해야 되는 데이터가 필요하다면, application scope에 해당되는 속성(attribute)을 사용
- 대표적인 예로 '웹 사이트의 방문자수 조회' 형태로써, 웹 어플리케이션을 종료할 때까지 속성 값은 유지

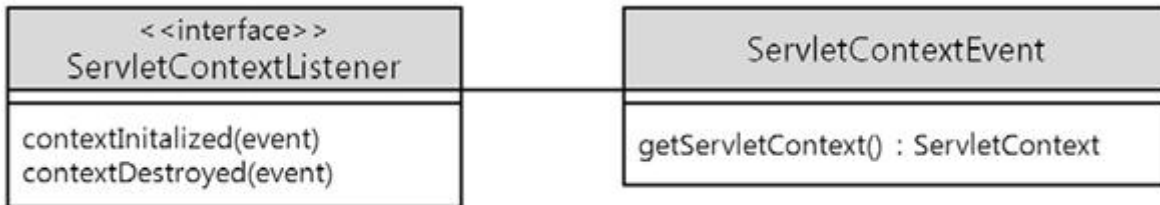


4.3 ServletContextListener API – 1

- 서블릿이 LifeCycle를 가지고 있는 것처럼, 웹 어플리케이션도 LifeCycle를 갖음
- Tomcat 컨테이너가 시작될 때 웹 어플리케이션도 초기화되고, Tomcat 컨테이너가 종료될 때 웹 어플리케이션도 제거
- 웹 어플리케이션이 초기화되고 제거되는 이벤트를 감지하는 ServletContextListener API를 사용하면, 언제 초기화되고 제거되었는지를 쉽게 알 수 있음
- 이 이벤트는 JDBC의 Pooling 기법에 적용 가능
- 웹 어플리케이션이 초기화될 때 Pooling을 활성화하고 제거될 때 Pooling을 비활성화 시키면 효율적으로 Connection을 관리 가능

4.3 ServletContextListener API – 2

-ServletContextListener API 계층 구조



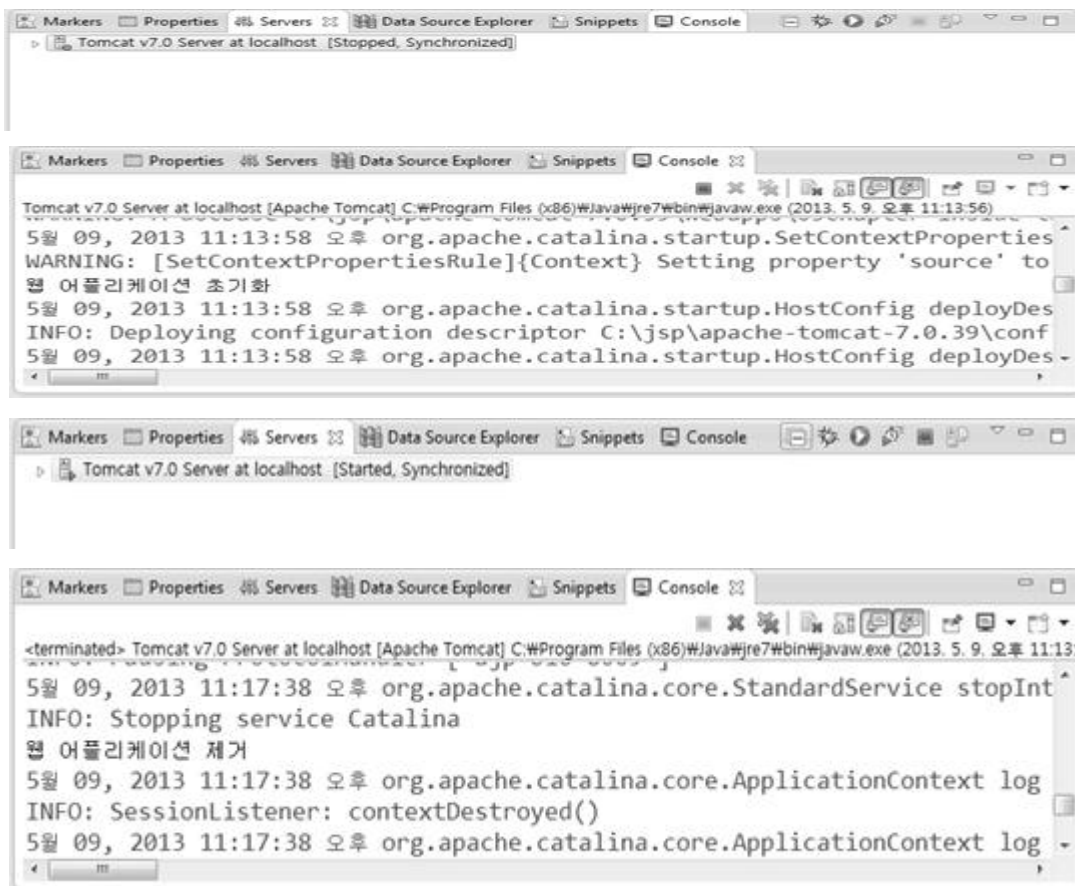
실습 순서

- (1) ServletContextListener 인터페이스를 구현하는 클래스를 작성
- (2) web.xml에 구현한 클래스를 <listener> 태그로 등록 또는 @WebListener 어노테이션을 이용
- (3) Tomcat 컨테이너를 시작하고 종료하는 작업을 실행하여 이벤트 감지를 확인

4.3 ServletContextListener API – 3

-4.3.1 web.xml에 <listener> 태그로 등록하는 방법

-ServletContextListener 인터페이스를 사용하여, 웹 어플리케이션의 초기화 이벤트와 삭제 이벤트를 감지하여 처리



4.3 ServletContextListener API – 4

-4.3.2 @WebListener 어노테이션으로 등록하는 방법

-web.xml에 등록한 <listener> 태그를 제거하거나 주석 처리

-Context ListenerImpl 클래스에 @WebListener 어노테이션을 추가하고 실행

4.4 Filter API – 1

-클라이언트인 웹 브라우저에서 서블릿으로 요청하면, 웹 컴포넌트인 서블릿이 요청을 받아서 작업을 처리하고 결과를 HTML 형식으로 작성하여 웹 브라우저에게 응답 처리

-서블릿이 요청 받기 전과 결과를 웹 브라우저에게 응답하기 전에 특정 작업을 수행할 수 있도록 Filter API를 사용 가능

-웹 컴포넌트가 실행되기 전의 선처리(pre-processing) 작업과 응답되기 전의 후처리(post-processing) 작업을 수행하는 API

-다수의 Filter를 체인(Chain)처럼 묶어서 적용시킬 수도 있음

▶선처리 작업의 필터 → 요청 필터(Request Filter)

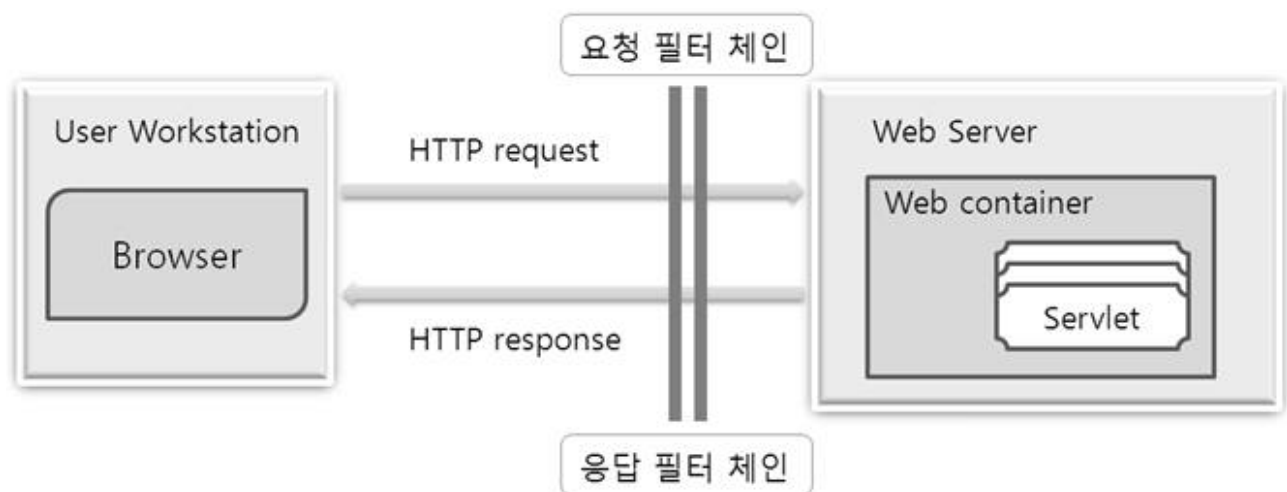
▶후처리 작업의 필터 → 응답 필터(Response Filter)

-선처리 작업의 적용 예 : 한글인코딩 및 보안관련 작업 등

-후처리 작업의 적용 예 : 압축 및 데이터 변환 작업 등

4.4 Filter API – 2

-Filter를 적용한 아키텍처



-request 요청이 요청 필터 체인을 거쳐서 서블릿으로 전송

- 서블릿이 처리하기 전에 실행되어야 하는 선처리 작업 수행 가능
- response 응답이 응답 필터 체인을 거쳐서 전송
- 요청과 마찬가지로 웹 브라우저로 응답되기 전에 실행되어야 하는 후처리 작업 수행 가능

4.4 Filter API – 3

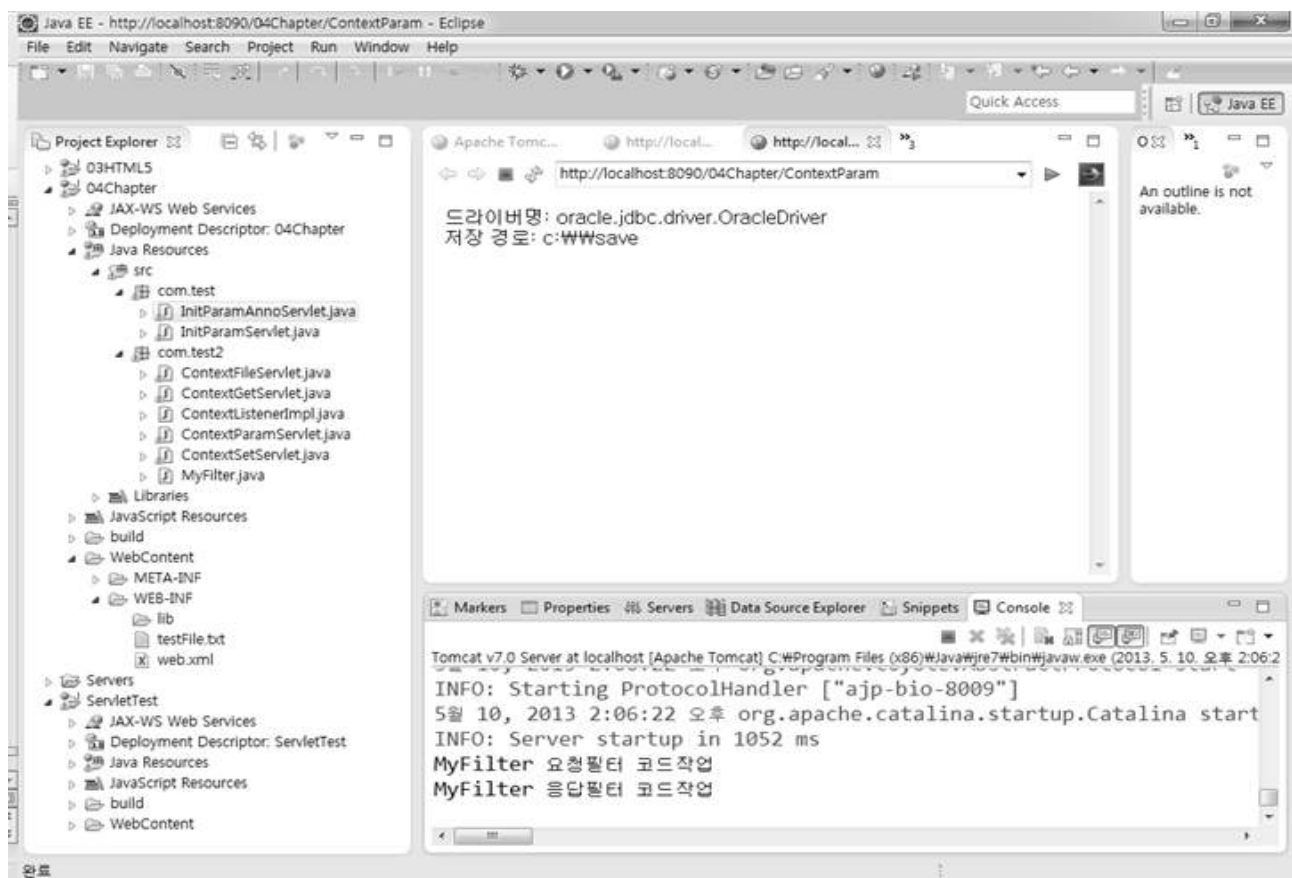
-Filter를 적용하기 위한 실습 순서

- (1) Filter 인터페이스를 구현한 클래스를 작성
- (2) web.xml에 <filter> 태그를 사용하여 등록하거나, @WebFilter 어노테이션을 사용하여 등록
- (3) 웹 서버에 요청하여 Filter가 적용되었는지를 확인

-4.4.1 web.xml에 <filter> 태그로 등록하는 방법

- 웹 어플리케이션의 서블릿이 요청을 처리하기 전의 선처리(pre-processing) 작업과 후처리(post-processing) 작업을 처리하는 Filter 예제
- 코드를 간단하게 하기 위해서 console 창에 문자열을 출력하는 형태로 실습

4.4 Filter API – 4



4.5 url-pattern의 개요 - 1

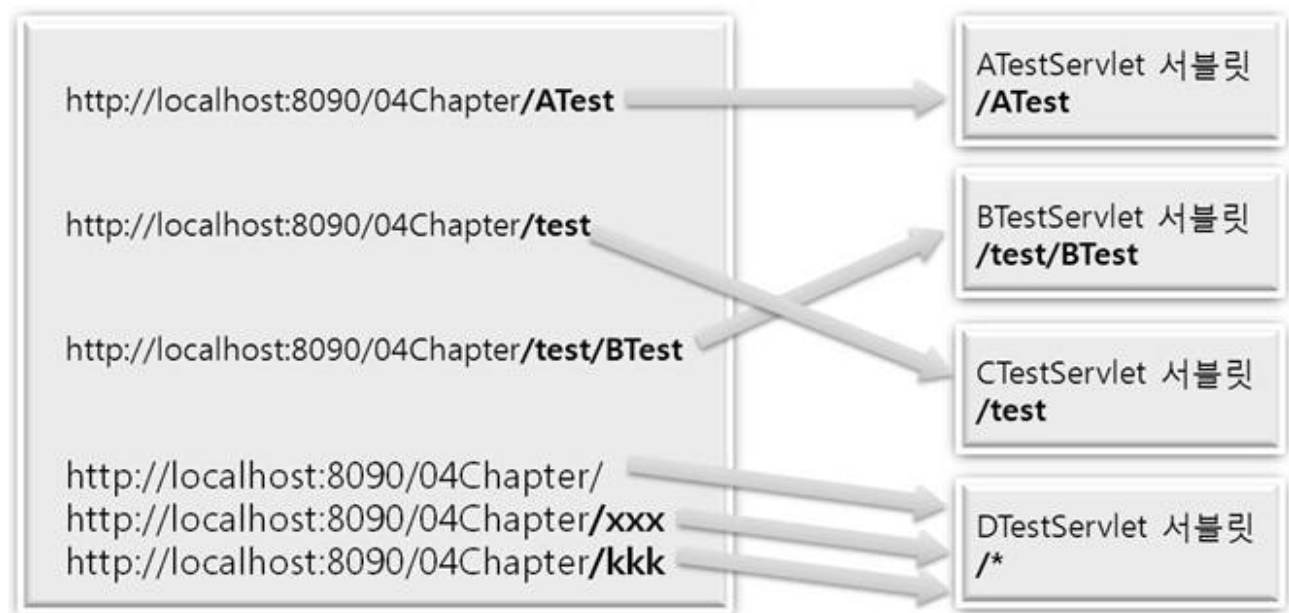
- 서블릿 맵핑과 필터 맵핑은 web.xml 파일 및 어노테이션을 사용하여 설정
- 이때 공통적으로 사용하는 값이 url-pattern
- url-pattern은 웹 브라우저에서 클라이언트가 요청하는 URL 값의 패턴에 따라서, 서버의 어떤 웹 컴포넌트가 실행될지를 결정하는 방법

1) 디렉터리 패턴

- ▶ 디렉터리 패턴과 일치하는 형태의 url-pattern을 지정한 웹 컴포넌트가 수행
- ▶ 이때, url-pattern 값은 반드시 '/디렉터리 패턴 값' 형식으로 지정해야 되며, 계층 구조를 가질 수 있음
- ▶ 만약에 '/' 형식으로 지정하면, 클라이언트가 요청하는 URL 값의 패턴과 무관하게 항상 수행

4.5 url-pattern의 개요 - 2

- 다음은 웹 컨테이너에 4개의 서블릿을 작성하고 각각 url-pattern 값을 지정한 후에, 웹 브라우저에서 특정 패턴을 사용하여 요청한 경우

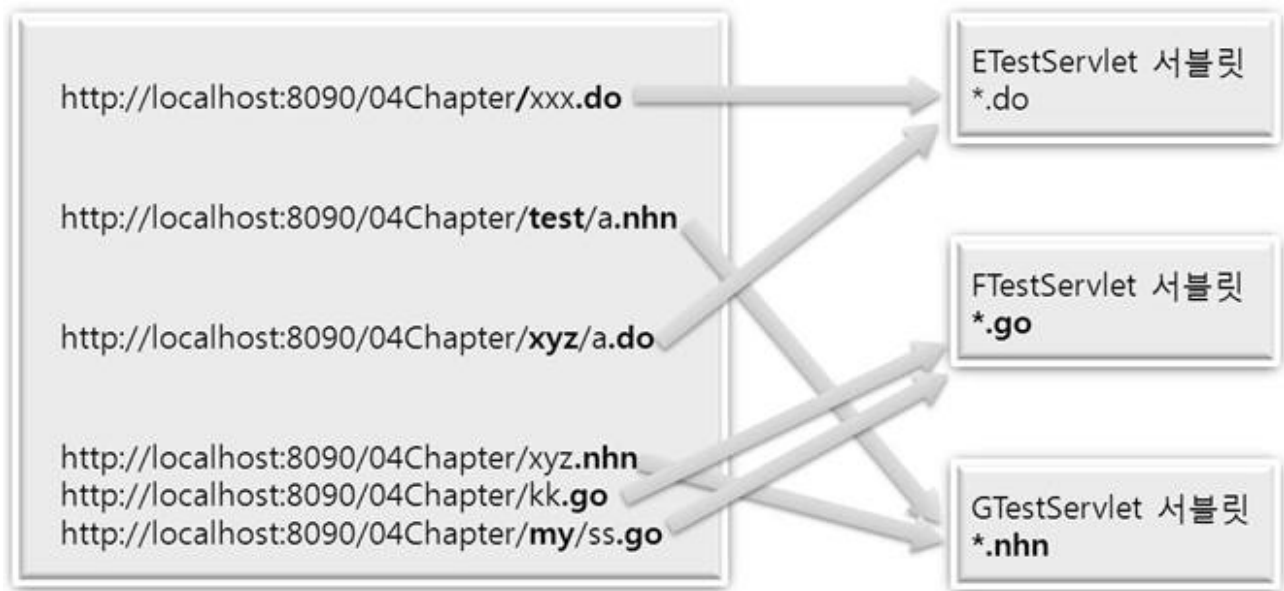


4.5 url-pattern의 개요 - 3

2) 확장자 패턴

- ▶ 일치하는 확장자 형태의 url-pattern을 지정한 웹 컴포넌트가 수행
- ▶ url-pattern 값은 반드시 '*.확장자' 형식으로 지정해야 됨
- ▶ 주의할 점은 '/'를 사용하면 안되며 Spring, Struts, Struts2 프레임워크 같은 다양한 웹 프레임워크에서 많이 사용되는 패턴

- ▶다음은 웹 컨테이너에 3개의 서블릿을 작성하고 각각 url-pattern 값을 지정한 후에, 웹 브라우저에서 특정 패턴을 사용하여 요청한 경우이며, /test, /xyz 또는 /my 같은 경로 값 설정과 무관하게 요청



-이 자료는 북스홀릭의 JSP 2.2 & Servlet 3.0 정복하기(PPT) 자료를
참조하였으므로 무단 배포할 경우 법적인 책임이 있음을 알려드립니다.-