

# Java Web Application

## World Wide Web



World Wide Web(World Wide Web, WWW, W3)은 인터넷에 연결된 컴퓨터들을 통해 사람들이 정보를 공유할 수 있는 전 세계적인 정보 공간을 말한다.

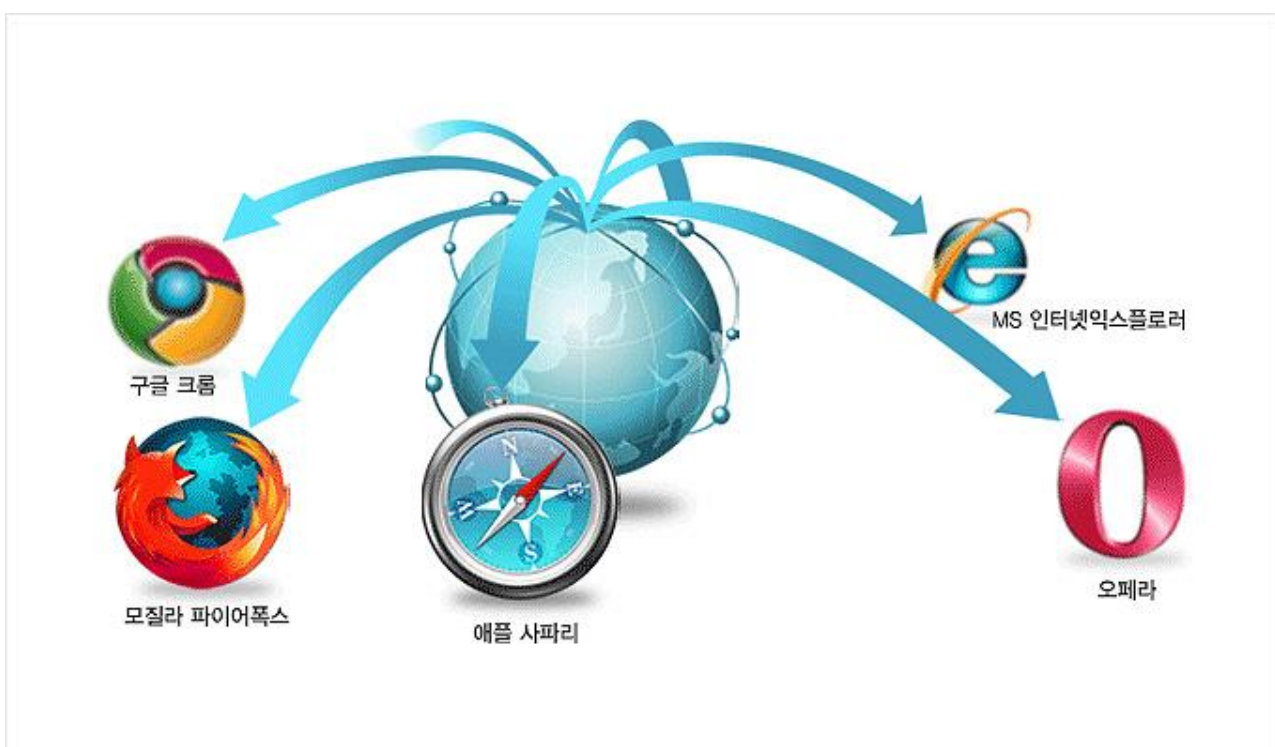
인터넷상의 정보를 하이퍼텍스트 방식과 멀티미디어 환경에서 검색할 수 있게 해주는 정보검색 시스템이다.

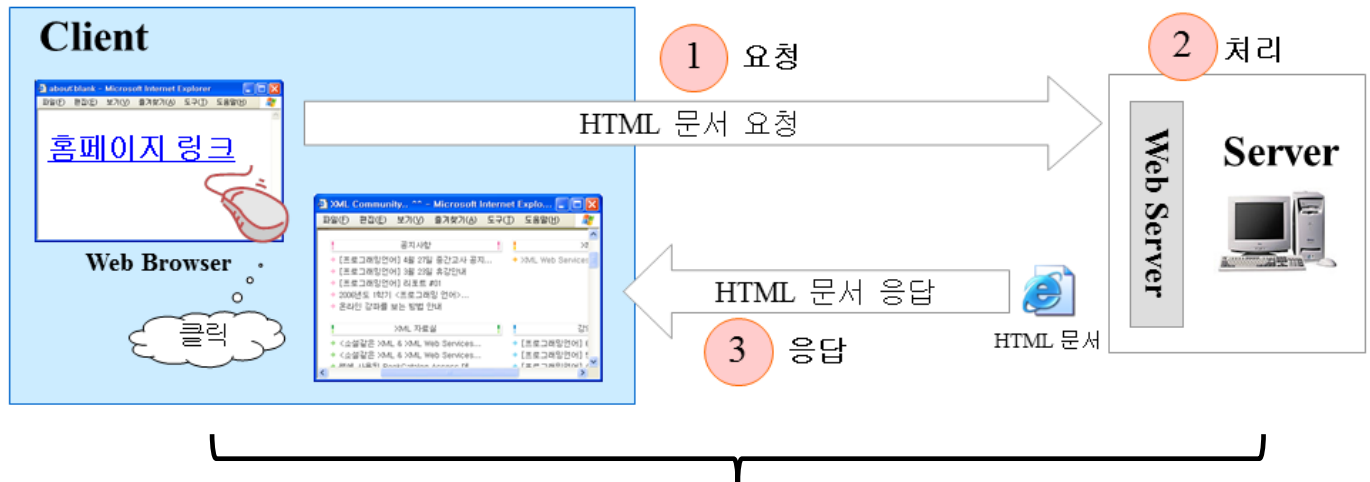
1989년 3월 유럽 입자 물리 연구소(CERN)의 소프트웨어 공학자인 **팀 버너스 리**의 제안으로 시작되어 연구, 개발되었다. 원래는 세계의 여러 대학과 연구기관에서 일하는 물리학자들 상호간의 신속한 정보교환과 공동연구를 위해 고안되었다.

World Wide Web에 관련된 기술은 World Wide Web 컨소시엄(W3C)이 개발하고 있다. W3C는 HTML, HTTP 등의 표준화를 진행하고 있으며, 최근에는 시맨틱 Web에 관련된 표준을 제정하고 있다.



W3C(World Wide Web Consortium)는 World Wide Web을 위한 표준을 개발하고 장려하는 조직으로 팀 버너스 리를 중심으로 1994년 10월에 설립되었다. W3C는 회원기구, 정직원, 공공기관이 협력하여 Web 표준을 개발하는 국제 컨소시엄이다. W3C의 설립취지는 Web의 지속적인 성장을 도모하는 프로토콜과 가이드라인을 개발하여 World Wide Web의 모든 잠재력을 이끌어 내는 것이다.





### HTTP 통신 프로토콜 기반으로 동작

HTTP(HyperText Transfer Protocol)는 Web 상에서 정보를 주고 받을 수 있는 프로토콜이다. **HTTP는 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜**로서 클라이언트인 Web 브라우저가 HTTP를 통하여 서버로부터 Web 페이지나 그림 정보를 요청하면, 서버는 이 요청에 응답하여 필요한 정보를 해당 사용자에게 전달하게 된다.

HTTP 프로토콜은 **Connection Oriented와 Stateless 방식**으로 동작하는 프로토콜로서 신뢰성 있는 통신을 하면서도 처리 효율이라는 부분을 강조하였으므로 인터넷 환경에 가장 적합한 통신구조로 인정 받았다.

[ HTTP 1.1 에서 지원되는 요청 방식들 ]

요청방식	설 명
GET	URI에 지정된 파일을 얻고자 할 때 사용되는 요청 방식으로 디폴트 방식이다. name=value로 구성되는 간단한 데이터(Query 문자열)를 URI 뒤에 추가하여 전달하면서 요청하고자 하는 경우에도 사용된다. http://localhost:8080/test.jsp?productid=00001
HEAD	GET과 동일하나 바디 내용은 받지 않고 HTTP 헤더 정보만 받는다.
POST	원하는 방식으로 인코딩 된 데이터를 요청 바디에 포함하여 전송하면서 파일을 요청하고자 하는 경우 사용된다. Query 문자열 전달시의 GET 방식을 보완한 요청 방식이다.
OPTIONS	요청 URI에 대하여 허용되는 통신 옵션을 알고자 할 때 사용된다.
DELETE	서버에서 요청 URI에 지정된 자원을 지울 수 있다.
PUT	파일을 업로드할 때 사용된다. 그러나, 현재 많이 사용되는 파일 업로드는 POST 방식을 사용하고 있고, PUT은 아직 많이 사용되지 않는다.

## 인터넷 20주년 주요 사건

# World Wide Web

## 20년

월드와이드웹이 등장한지 20년이라는 시간이 흘렀다. 월드와이드웹은 우리의 일상 생활과 연구 및 강의 뿐만 아니라 사회 문화적인 변화를 불러왔다. 지난 20년간 월드와이드웹이 바꿔놓은 우리 사회를 살펴본다.

오른쪽 표에는 지난 시간 동안 세계와 우리나라의 인터넷 주요 사건을 간략히 정리했다.

김지혜 기자 haro@kyosu.net

**1992** World Wide Web(www)의 시대 개막  
스위스 유럽입자물리연구소(CERN)의 연구원 팀 버나스 리가 WWW를 처음 고안했다. 팀 버나스 리는 현재 MIT 교수로 재직 중이다.

**1993** 최초의 인터넷 브라우저 Mosaic 출현

슈퍼컴퓨터 응용 연구소(NCSA)와 일리노이 대학에 근무하던 마크 앤드레센과 에릭 비너 등 8명이 개발했다. 광범위하게 사용된 첫 인터넷 브라우저다.



**1995** 인터넷 쇼핑 사이트 ebay 오픈

미국의 피에르 오미디아르가 설립한 인터넷 경매 사이트. 국내 최초의 경매 전문 사이트 옥션과 오픈 마켓 G마켓을 각각 2001년과 2009년에 인수했다.



**1996** Hotmail 등장

세계 최대의 이메일 서비스 (2011년 5월 기준). 사비어 바티아와 잭 스미스가 선보인 세계 최초의 웹기반 메일 서비스다. 론칭 1년만인 1997년에 Microsoft가 인수했다.



**1997** 한메일(Hanmail) 서비스 시작

국내 최초의 무료 이메일 서비스. 이재웅 전 대표가 설립했다. 1999년 사이트 명칭을 '다음'으로 바꾸고 인터넷 포털사이트로 개편했다.



**1998** Google 서비스 시작

세계 최대의 인터넷 검색 회사. 미국 스탠퍼드대의 대학원생인 래리 페이지와 서지 브린이 공동으로 설립했다.



**1999** Wi-fi 규격화

무선 인터넷 기술인 와이파이가 규격화됐다.



네이버 검색 서비스 시작

1997년 이해진, 권혁일, 김보경 등으로 구성된 삼성SDS의 사내 벤처에서 시작해 분할됐다.



**2001** Wikipedia 등장

누구나 자유롭게 글을 쓸 수 있는 온라인 백과사전으로 지미 웨일스가 설립했다. 집단 지성과 웹 2.0의 대표적인 사례로 꼽힌다.



**2006** twitter, facebook 론칭

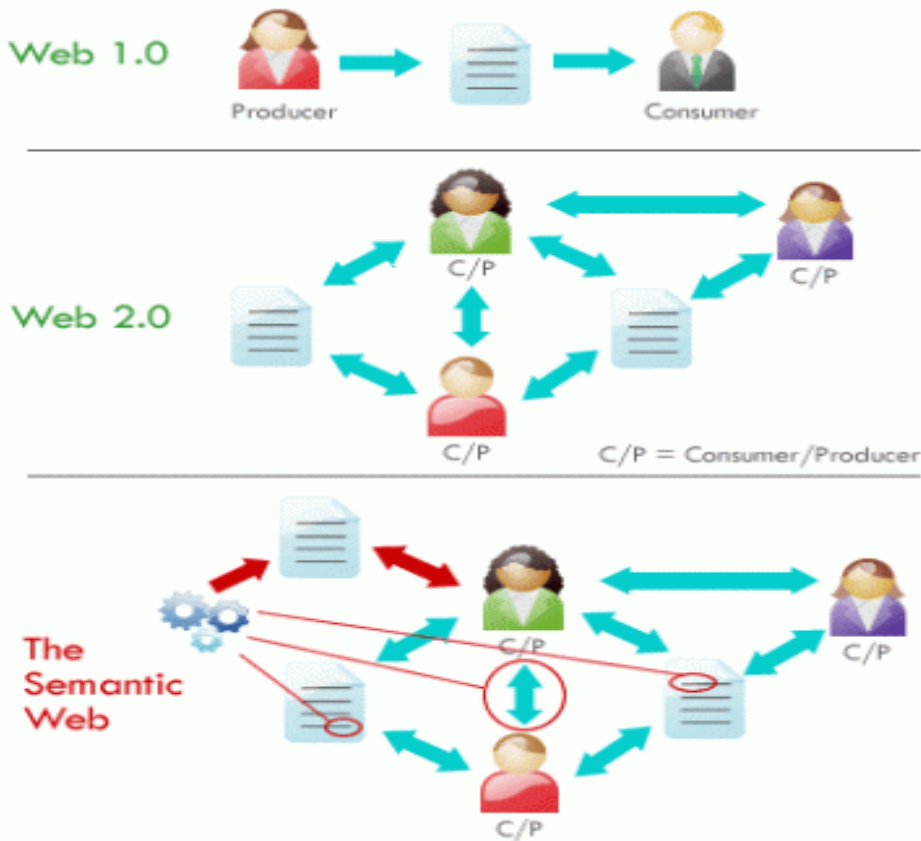
대표적인 소셜 네트워킹(SNS) 서비스.



**2009** 모바일 데이터 전송량 > 음성 전송량

휴대전화 월별 이용에서 모바일 데이터 전송량이 음성 전송량을 초과하기 시작했다.





#### Web1.0과 2.0

Web1.0은 인터넷의 초창기 시절로 단순히 제작자가 소비자에게 필요한 정보를 가진 Web 페이지를 제공하던 시절을 말한다. 여기서 제작자란 기업, 학교, 정부와 소수의 개인 홈페이지 소유자이며 소비자는 Web에서 정보를 검색하는 개별 사용자로 TV나 라디오와 같은 매체와 같이 정보의 흐름은 제작자에서 소비자로의 일방통행이라고 할 수 있다.

2000년도를 지나며 모뎀시대를 지나 고속인터넷의 보급과 함께 전세계 인터넷 사용자가 폭발적으로 늘어나고 그 영향력이 갈수록 커지면서 인터넷의 사용목적이 다양화 되기 시작한다. 단순히 정보의 검색과 열람만이 아닌 소비자의 능동적인 정보입력이 필요한 인터넷 쇼핑과 banking, 게시판 및 미니 홈페이지 서비스 등 쌍방향 참여 비즈니스 모델들이 인터넷상에서 실제와 같은 기능으로 제공 되면서 Web1.0에서 2.0으로 넘어가기 전 과도기 상태인 Web1.5세대로 접어들게 된다. 그리고 머지않아 사용자들에게 '참여'의 맛을 살짝 느끼게 해준 Web1.5세대는 이후 블로그, 위키, UCC 등으로 진화해 나가며 2004년 팀 오라일리과 존 바텔에 의해 'Web2.0'이라는 새로운 인터넷의 개념으로 정의된다.

**"개방형 서비스 구조를 기반으로 사용자의 참여를 통해 핵심가치를 창출하는 인터넷 서비스".** Web2.0을 정의하는 이 문장에서 가장 핵심단어는 역시 **개방과 참여**다. 기존의 Web이 사용자들이 데이터와 서비스를 수동적으로 받는 일방적인 정보제공과 활용의 개념이라면 Web2.0은 개방과 참여를 바탕으로 사용자들이 자유롭게 정보와 네트워크를 활용하는 개념이다. 현재 Web2.0이 자리잡으면서 1세대때 뚜렷했던 제작자와 소비자의 경계가 점차

희미하게 되고 사용자에게 의해 창조된 새로운 데이터 및 가공된 기존의 데이터로 인해 인터넷상에 존재하는 정보의 양이 전과는 비교할 수 없을 만큼 증가하고 정보의 질 자체도 크게 향상 된다.

## Web3.0

Web2.0이란 단어가 처음 사용되었을 때와 마찬가지로 Web3.0 역시 시간의 흐름과 직접적인 사용으로 다듬어지기 전까진 그것을 정확하게 정의 내릴 수는 없겠지만, 과거와 현재의 인터넷 환경과 변화를 주시하고 분석한 전문가들이 말하는 Web3.0은 다음과 같다.

### 속도와 플랫폼의 변화

Web의 세대간 변화를 이끌어낸 가장 큰 원동력 중 하나는 바로 네트워크의 속도였다. 간단한 정보검색이 주를 이뤘던 Web의 첫 번째 세대 때는 50kbps의 모뎀이, 참여와 공유를 중심으로 하는 Web의 두 번째 세대는 1메가를 넘는 고속인터넷이 바탕이 되었다면 Web3.0은 10메가에서 1기가까지의 초고속 인터넷 환경에서 구현될 것이다. 네트워크의 고속화는 3D, 비디오, 멀티미디어가 본격적으로 Web에 진출함을 뜻하며 이는 기존에 우리가 알고 있던 인터넷의 '얼굴'자체를 바꿀 수도 있다.

속도의 변화는 무선(wireless)시장에서 더욱 활발하게 진행될 것이며 현재 노트북에 종속 되어있던 주류 인터넷이 휴대폰, PDA 등 각종 무선기기들이나 기존 전자용품들과 결합되어 장소와 시간에 상관없는 생활 속 인터넷이 구현될 것이다. 이와 같은 플랫폼의 다양화는 인터넷 시장에 새로운 비즈니스 모델을 파생시키고 사용자에게 대한 개인정보 수집을 원활하게 하여 보다 다양한 서비스와 개별화된 정보제공이 원활해질 것이다.

### 똑똑한 데이터와 인공지능의 향상

위의 두 개의 그림에서도 나와있듯이 Web3.0을 대표하는 키워드는 시맨틱(Semantic)이다. 시맨틱 Web이란 기존의 Web페이지에서 진화한 개념으로 각각의 페이지가 사용자(인간)에게만 이해되고 읽혀지는 정보가 아닌 기계에게도 이해될 수 있는 "데이터를 설명하는 데이터", 즉 메타데이터를 포함한 Web 환경을 말한다. 현재와 같이 사용자가 일일이 읽고 정보의 가치를 판단해야 하는 환경과 달리 Web에 존재하는 모든 정보가 의미있는 메타데이터로 연결되며 향상 된 인공지능을 갖고 있는 기계에 의해 개인에 요구에 맞게 논리적으로 분석되어 가장 값진 정보가 사용자에게 전달될 수 있게 된다.

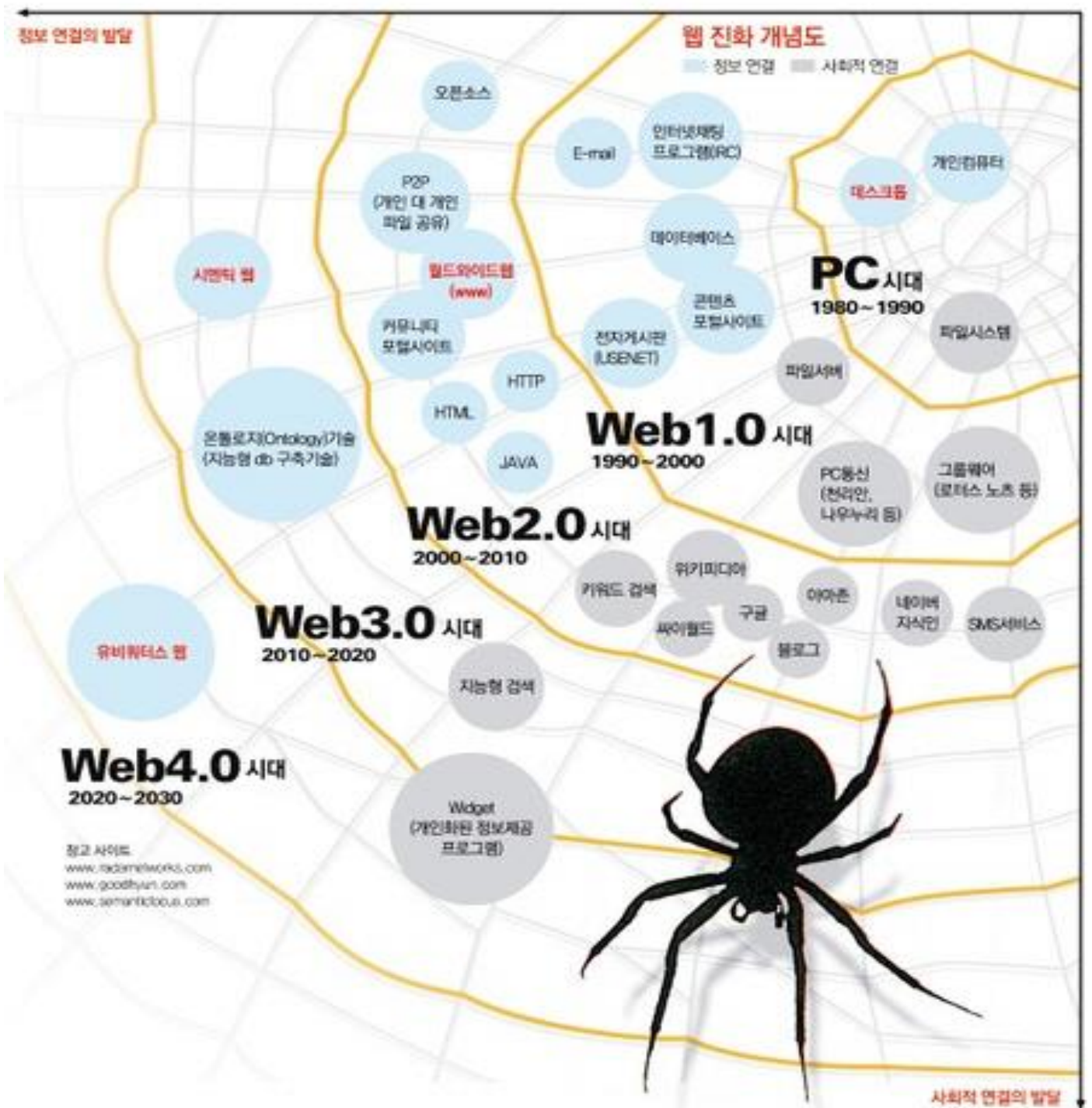
Web3.0의 가장 큰 변화 중 하나인 "똑똑한 데이터"와 "인공지능"은 무한한 정보의 생산과 검색에 기반을 둔 Web1.0과 2.0의 최대 약점인 올바른 정보를 얻기 위한 사용자의 막대한 시간과 노력을 혁신적으로 절약해 줄 것이다.



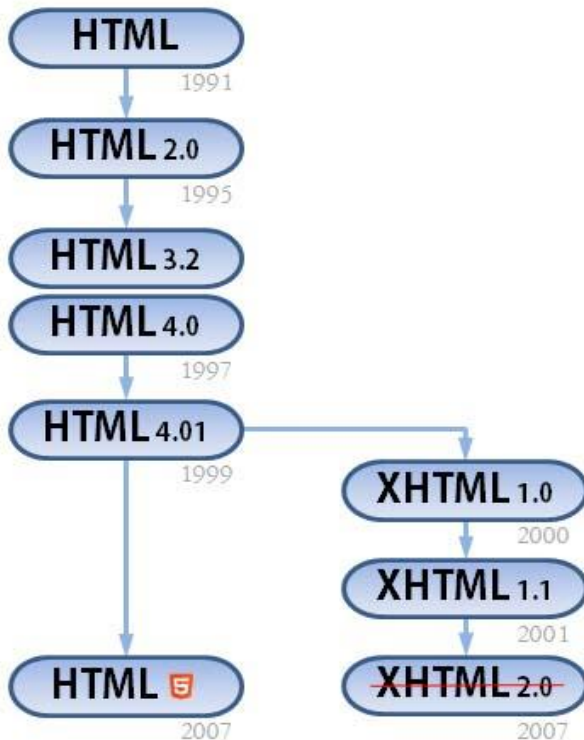
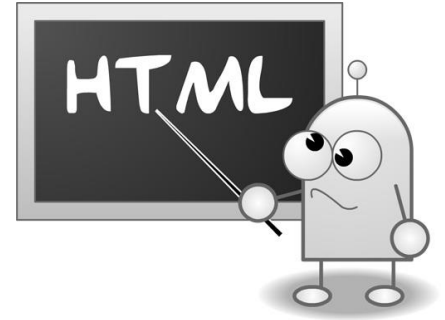
## 애플리케이션의 진화 (메쉬업)

빠른 네트워크와 의미있는 데이터의 집합은 그것을 사용하는 애플리케이션에도 영향을 미친다. Web2.0에서 시도되었던 오픈 API와 SOA 및 메쉬업은 그 영역을 넓혀 수많은 애플리케이션에 적용될 것이다. 메쉬업이란 컴포넌트화 된 애플리케이션의 부분 부분을 조합해 개인이나 그룹의 용도에 맞게 만든 파생애플리케이션의 구성을 말한다. 기존의 애플리케이션과는 달리 프로그래밍에 대한 많은 지식이 없어도 누구나 쉽게 만들 수 있으며 안정성 및 효율성이 보장된다.

기존의 데스크톱 애플리케이션은 물론 기업용 소프트웨어까지 컴포넌트화 될 경우 효율성은 극대화하고 가격은 최소화 할 수 있을 뿐만 아니라 인터넷에 접속 가능한 모든 플랫폼에서 사용이 가능하다.



HTML은 Hypertext Markup Language의 약자로 웹 페이지 만들기 위한 문서 규약이다. 통상 링크(link)라고 부르는 하이퍼텍스트(hypertext)를 통해 다른 웹 페이지로 이동할 수 있다. 이러한 링크 덕분에 사용자는 Web 을 통해 쉽고 빠르게 다양한 정보를 얻을 수 있고, World Wide 웹(World Wide Web, WWW) 이 인터넷 서비스의 대표 주자로 성장할 수 있게 되었다.



HTML이라는 용어는 1991년에 처음 사용되었고, 1995년 World Wide Web Consortium(이하 W3C) 에서 정식으로 HTML 2.0 표준안을 발표했다. 그 후, 계속해서 단점을 보완하고 업데이트하여 지금 발표된 최신 버전은 1999년에 제정된 HTML 4.01 이다. HTML을 XML 형태로 사용할 수 있도록 한 XHTML 또한 별도로 제정되어 최근까지도 많이 사용하고 있는 버전은 2001년 발표된 XHTML 1.1 과 HTML 4.01 이다.

표준이 10년간 머물러 있는 동안에도 웹 기술은 급속하게 발달했다. 2000년대 중반에는 CSS, AJAX, RSS 등의 최신 기술을 사용해 웹이 콘텐츠 제공 뿐만 아니라 애플리케이션 플랫폼의 역할까지 할 수 있다는 것을 보여주는 Web 2.0 개념이 출현하였다.

문제는, 웹브라우저 제조업체들이 자기 브라우저에서만 지원하는 확장 HTML과 부가기능 들을 경쟁적으로 개발하고 이를 독립 기술로 분류해 도입하면서 발생했다. 브라우저의 특색을 살리기 위해 표준을 지키지 않은 편법까지 마다하지 않은 것이다.

이러한 상황 때문에, 개발자들은 크로스브라우징(cross-browsing)이라는 취지 하에 각각의 브라우저가 가진 기능에 대응하는 별도의 코드를 작성하는 비효율적인 작업을 해야 했고, 사용자들은 크로스브라우징이 고려되지 않은 웹 서비스 때문에 여러 브라우저를 설치해 사용하는 불편을 겪고 있다. 지금도 여전히 일부 기능 때문에 특정 브라우저를 사용하기도 한다.

2005년, 당시 웹 브라우저 점유율이 극히 낮았던 오페라(Opera), 모질라(지금의 Firefox), 사파리(Safari)와 같은 브라우저 업체 들은 웹 표준을 제정하는 W3C와 Web의 차후 방향



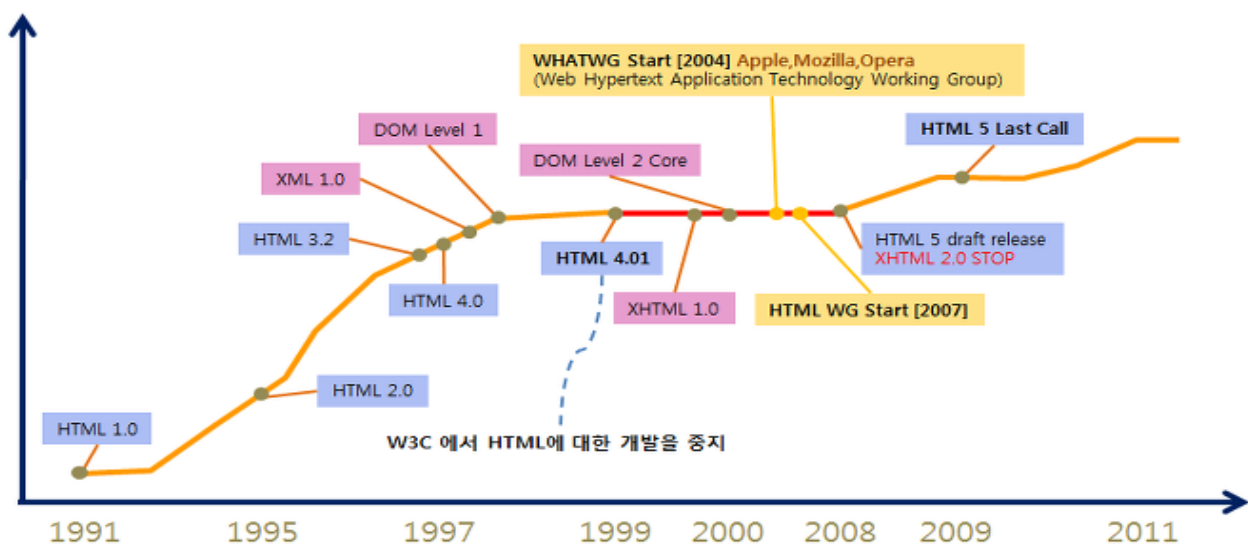


에 대해 논의를 진행했다. 하지만, W3C는 XML 기반의 XHTML을 보완해 표준으로 제정하려고 했고, 웹 브라우저 업체들은 Web 2.0에서 사용된 최신 기술들에 대한 표준 정의와 기존에 브라우저 간에 호환되지 않는 기능들에 대한 해결을 필요로 했기 때문에 서로 지향점이 달라 합의점을 도출하는 것은 실패했다.

결국, W3C는 XHTML 2.0 이라는 매우 엄격한 표준을 새로 만들어 냈고, 웹브라우저 업체들은 W3C와 별개로 WHATWG(Web Hypertext Application Technology Working Group)이라는 표준화 기구를 만들어 Web Application 1.0이라는 이름으로 별도의 표준안 제정을 진행하며 독립기구로서의 역할을 단행했다.

W3C의 XHTML 2.0은 형식상으로는 완벽하지만 실제 사용하기는 매우 어려웠던 반면, WHATWG의 Web Application 1.0은 진행과정에서 공개적으로 여러 개발자의 의견을 수렴하는 등 현실적인 사안을 반영했기 때문에, 기존 HTML 과 하위 호환성을 갖는 유연한 표준안을 만들어 낼 수 있다.

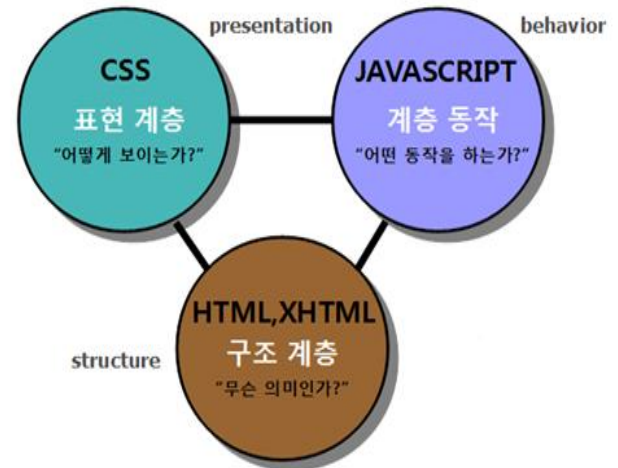
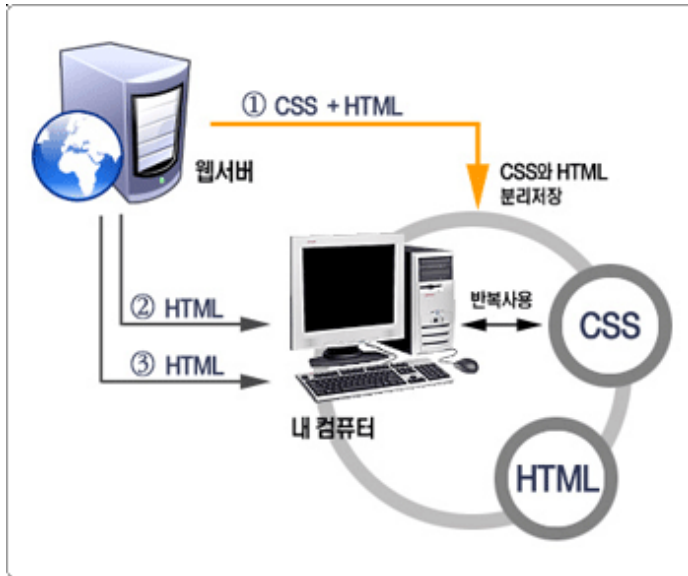
결국, W3C는 2007년에 XHTML 2.0을 완전히 포기하고 WHATWG의 표준안을 수용하여 HTML5 라는 이름으로 새롭게 표준안을 제정하고 있다.



[ Web 표준 ]

특정 브라우저에서만 사용하는 비 표준화된 기술을 배제하고, W3C라는 조직에서 정한 표준 기술만을 사용하여, 웹 문서의 구조와 표현 그리고 동작을 구분해서 사용하는 것으로 웹 문서의 구조를 담당하는 것은 **HTML** 이고 표현을 담당하는 것은 **CSS**이며 동작을 담당하는 것은 **자바스크립트**이다. 이 3가지 요소가 유기적으로 결합하여 작동하게 되면, 웹 문서가 가벼워지며, 유지보수 및 수정 시에도 간편하고 빨리 처리할 수 있게 된다.

웹표준의 세가지 계층

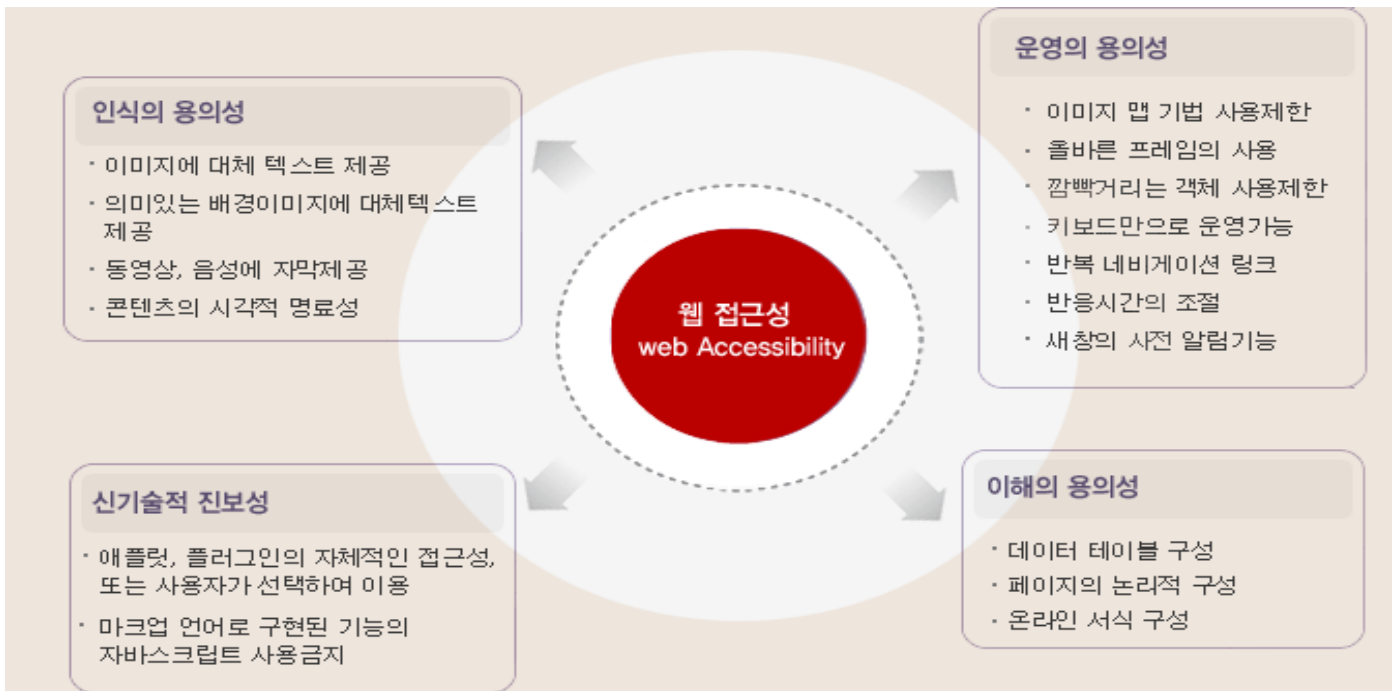


[ Web 접근성 ]



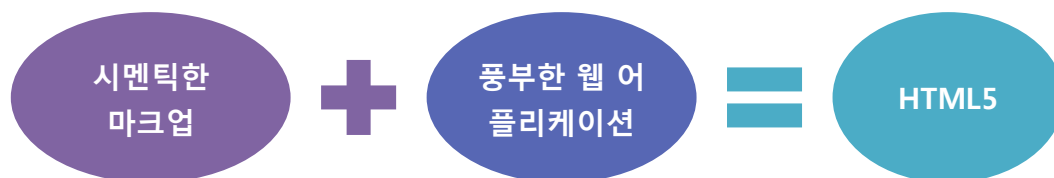
World Wide 웹(World Wide Web)을 창시한 팀 버너스 리 (Tim Berners-Lee)는 웹이란 '장애에 구애 없이 모든 사람들이 손쉽게 정보를 공유할 수 있는 공간'이라고 정의하였으며, 웹 콘텐츠를 제작할 때에는 장애에 구애됨이 없이 누구나 접근할 수 있도록 제작하여야 한다고 하였다.

누구든지 신체적·기술적 여건과 관계없이 웹사이트를 통하여 원하는 서비스를 이용할 수 있도록 접근성을 보장하는 것으로, 장애인, 고령자 등 정보이용접근이 어려운 사용자뿐만 아니라 일반적인 사용자들도 편리하게 웹서비스를 이용할 수 있도록 견고한 콘텐츠를 만드는 것이다. 웹 접근성을 만족하려면 다음의 사항들을 반영해야 한다.

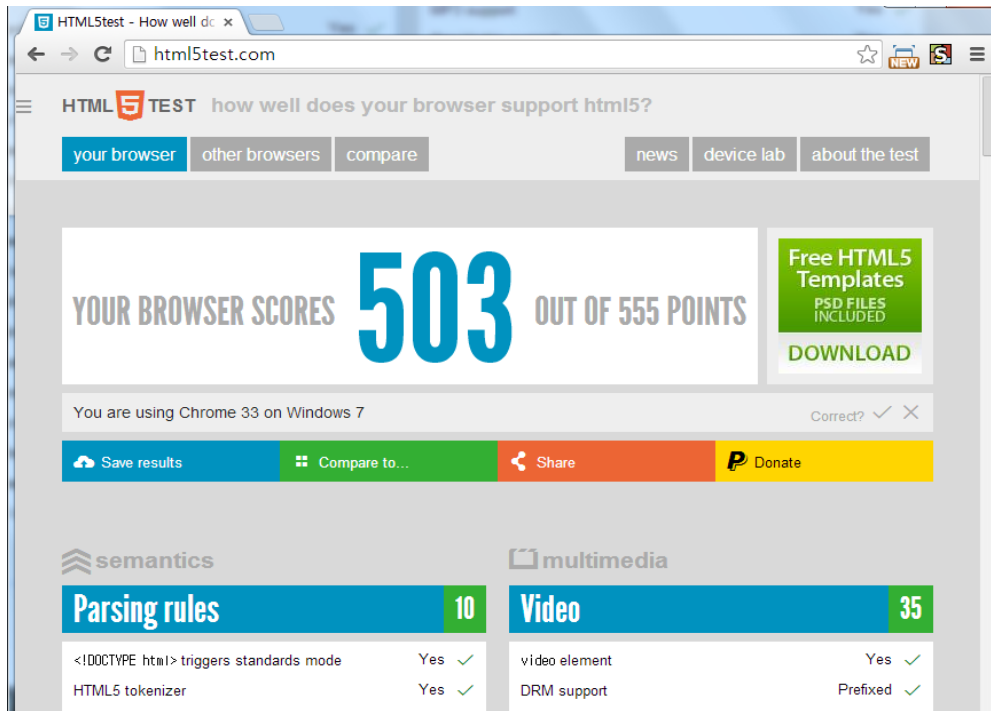


## [ HTML5 ]

HTML5는 W3C(월드와이드웹콘소시엄)의 HTML WG(Working Group)을 통해서 만들어지고 있는 차세대 마크업 언어 표준으로서 2014년 최종 완성을 목표로 하고 있지만, 다양한 스마트 기기에 효과적인 대응을 위한 유일한 해결책으로 인식하여 이미 시장에서 급속한 확산이 이루어지고 있는 실정이다. 이것은 웹이 가지고 있는 중요한 특징인 플랫폼 중립적이며 특정 디바이스에 종속되지 않게 활용할 수 있 수 있다는 점 때문이며, 기존 유선 환경을 넘어 스마트폰/태블릿 서비스 환경으로 확산되고 있고 향후에는 스마트TV 등을 포함한 대부분의 스마트기기 환경에서의 공통 콘텐츠 플랫폼으로 활용될 것으로 전망되고 있다.



\* 크롬의 HTML5 지원 스코어



Forms 108/110	
<b>Field types</b>	
▶ input type=text	Yes ✓
▶ input type=search	Yes ✓
▶ input type=tel	Yes ✓
▶ input type=url	Yes ✓
▶ input type=email	Yes ✓
▶ input type=date	Yes ✓
▶ input type=month	Yes ✓
▶ input type=week	Yes ✓
▶ input type=time	Yes ✓
▶ input type=datetime	No ✗
▶ input type=datetime-local	Yes ✓
▶ input type=number	Yes ✓
▶ input type=range	Yes ✓
▶ input type=color	Yes ✓
▶ input type=checkbox	Yes ✓
▶ input type=image	Yes ✓
▶ input type=file	Yes ✓
▶ textarea	Yes ✓
▶ select	Yes ✓

HTML5에서 지원하는 <FORM> 태그의 타입들과 서브 태그들

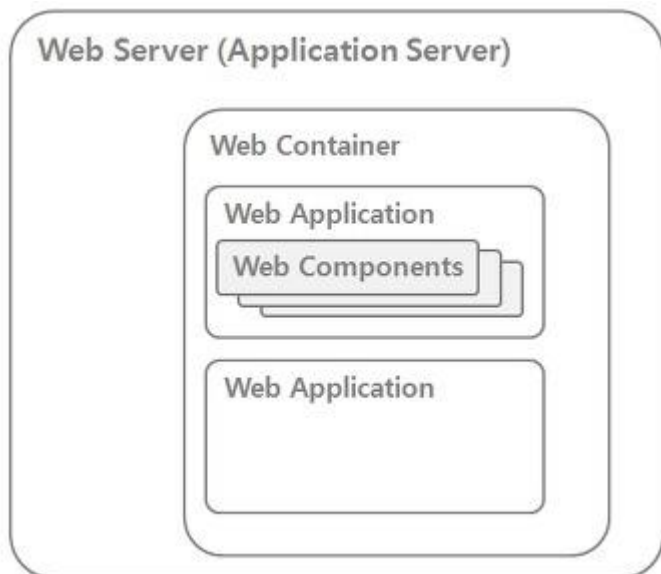
## Java EE 기반의 Web 어플리케이션

Web 어플리케이션이란 Web 또는 Application Server의 동적 확장 기능으로서 다음과 같이 두 가지 타입이 있다.

- Presentation-oriented : 프리젠테이션 지향 Web 어플리케이션은 Request에 대한 Response에 다양한 타입 (HTML, XML 등)의 **Markup 언어와 동적 콘텐츠를 포함하는 대화형 Web 페이지를 생성**한다.
- Service-oriented : 서비스 지향 Web 어플리케이션은 Web Service endpoint를 구현한다. 프리젠테이션 지향 Web 어플리케이션에서 전달되는 **서비스 로직에 대한 요청을 수행하고 제어**한다.

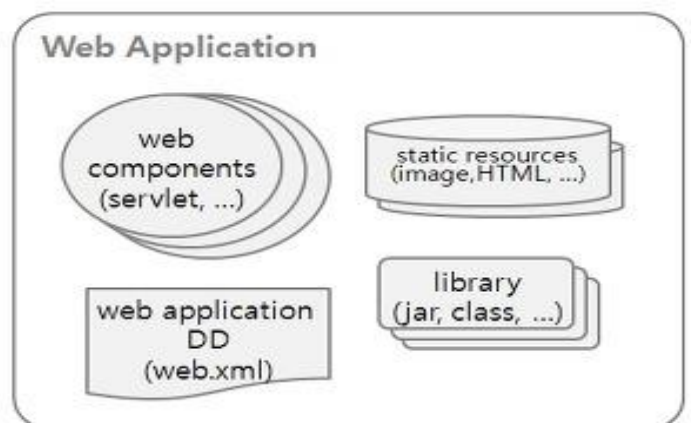
Servlet은 Service-oriented application의 여러 서비스 로직의 수행과 , Presentation-oriented application의 제어 기능에 적합하다. Servlet은 동적으로 요청을 처리하고 응답할 결과를 생성하는 Java 클래스이다.

JSP page는 텍스트 기반 마크업 생성(HTML, SVG, WML, XML)에 더 적합하다. JSP page는 텍스트기반 document로 Servlet 처럼 동작하지만 web 페이지의 콘텐츠를 생성하는데 있어 보다 쉽게 해준다.



Java EE platform에서 **Web Components**는 Java Servlet 이거나, JSP page를 의미하며 **Web Container**는 Web 어플리케이션 서버에 포함되는 기능으로서 정적인 콘텐츠(static HTML, image등)와 동적인 콘텐츠 (Servlet, JSP page 등에 의한)로 나누어 서비스를 처리하는 프로그램(엔진이라고도 한다)이다.

**Web Application**은 임의의 기능을 서비스하기 위한 Web Components 들의 묶음으로서 Web Container 내에서 동작한다. 다음은 하나의 Web Application 을 구성하는 요소들을 소개하는 그림이다.

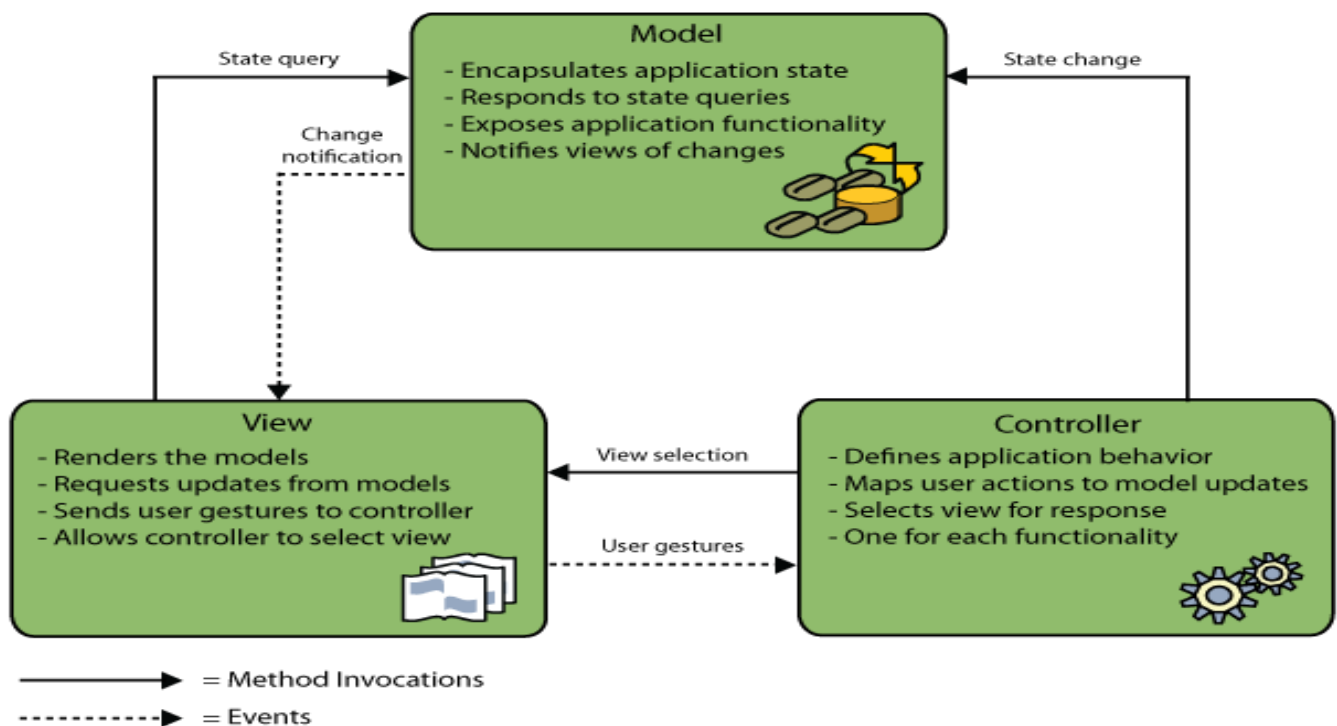




## MVC

MVC 패턴 (Model-View-Controller)은 컴퓨터 소프트웨어 개발의 구조적 패턴이다.

모델-뷰-컨트롤러(Model-View-Controller, MVC)는 소프트웨어 공학에서 사용되는 아키텍처 패턴이다. 이 패턴을 성공적으로 사용하면, 사용자 인터페이스로부터 비즈니스 로직을 분리하여 애플리케이션의 시각적 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 애플리케이션을 만들 수 있다. MVC에서 모델은 애플리케이션의 정보(데이터)를 나타내며, 뷰는 텍스트, 체크박스 항목 등과 같은 사용자 인터페이스 요소를 나타내고, 컨트롤러는 데이터와 비즈니스 로직 사이의 상호동작을 관리한다.



### [ 다층 구조 ]

다층 구조(Multi-tier Architecture 또는 n-tier Architecture)는 비즈니스 로직을 완전히 분리하여 데이터베이스 시스템과 클라이언트의 사이에 배치한 클라이언트 서버 시스템의 일종이다. 예를 들어 사용자와 데이터베이스간의 데이터 요구 서비스에 미들웨어를 이용하는 것을 들 수 있다. 일반적으로는 3층 구조가 널리 쓰인다.

### [ 3층 구조 (3-Tier) ]

세 층이란 사용자 인터페이스, 비즈니스 로직, 데이터베이스를 말하며, 이들을 각각 독립된 모듈로 개발하고 유지하는 구조로, 일반적으로 이들은 각각 다른 플랫폼 상에서 구동된다.

3층 구조는 2층 구조의 제한을 극복하기 위해서 탄생한 구조로, 사용자 인터페이스 환경과 데이터베이스 관리 서버 환경 사이의 중간층이 추가된 구조이다. 중간층의 구현은 트랜잭션 처리 모니터, 메시지 서버, 응용 서버 등 여러 가지 방법으로 구축될 수 있다. 이러한 중간층은 데이터베이스의 다단계나 응용프로그램의 실행 또는 사용자 요구 분산을 위한 큐잉을 수행할 수 있다. 예를 들어, 중간층이 큐로써 역할을 한다면 클라이언트는 자신의 요청을 중간층에 전달만 하고 중간층이 서버에 접속해서 클라이언트가 남기고 간 요청에 대한 응답을 받아 클라이언트에 돌려줄 것이다. 이러한 중간층의 역할은 스케줄링을 가능하게 할 뿐만 아니라, 다수 사용자 요구 처리에 대한 우선 순위를 정할 수 있게 해주어 서버의 부하를 줄여준다. 3층 구조는 아래의 세 층으로 나뉘어 있다.

#### - 프레젠테이션 계층

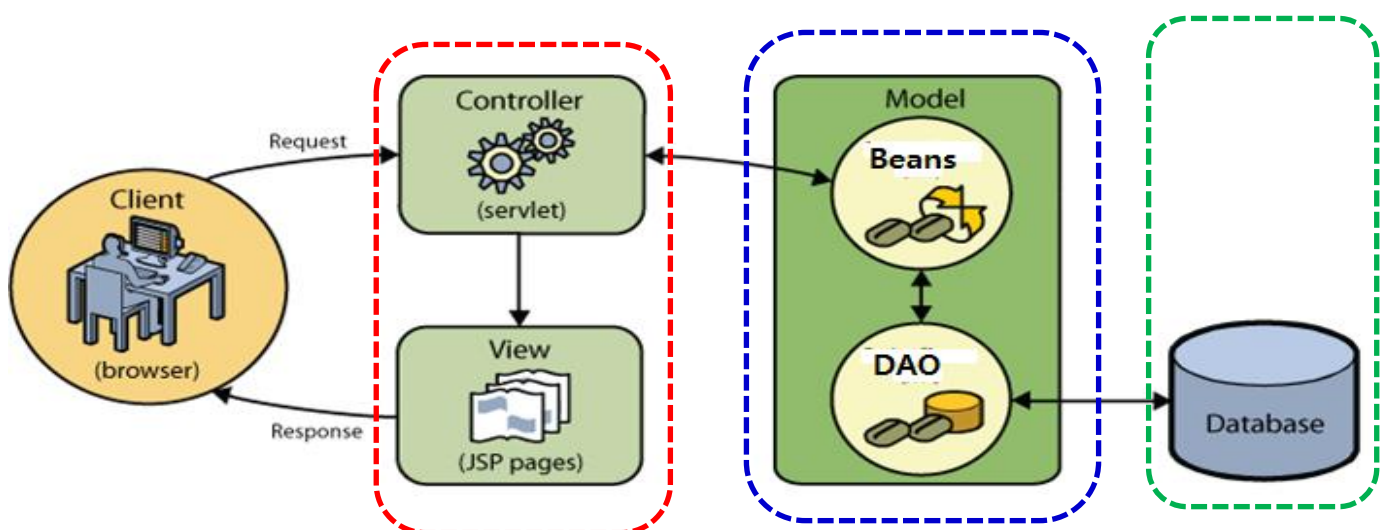
프레젠테이션 계층은 응용 프로그램의 최상위에 위치하고 있는데 이는 서로 다른 층에 있는 데이터 등과 커뮤니케이션을 한다.

#### - 애플리케이션 계층

이 계층은 비즈니스 로직 계층 또는 트랜잭션 계층이라고도 하는데, 비즈니스 로직은 워크스테이션으로부터의 클라이언트 요청에 대해 마치 서버처럼 행동한다. 그것은 차례로 어떤 데이터가 필요한지를 결정하고, 메인프레임 컴퓨터 상에 위치하고 있을 세 번째 계층의 프로그램에 대해서는 마치 클라이언트처럼 행동한다.

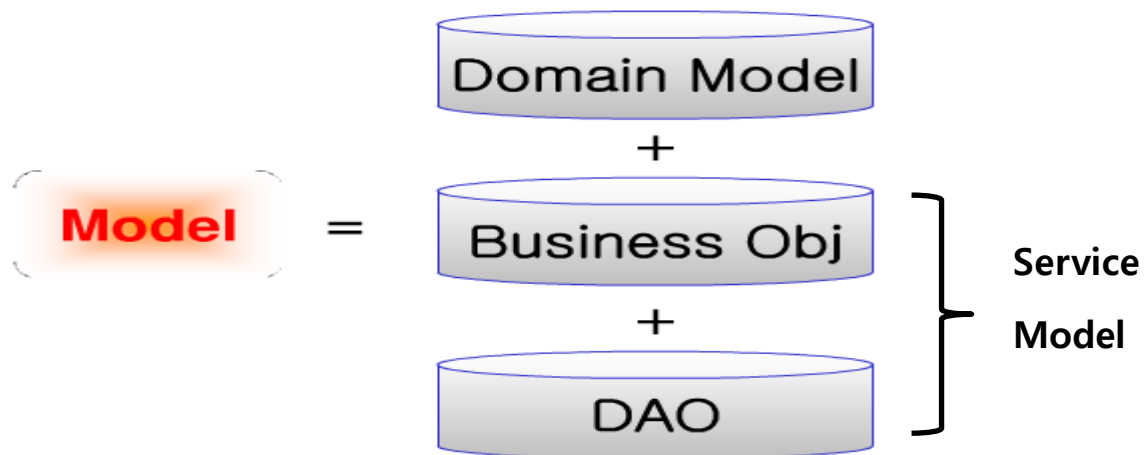
#### - 데이터 계층

데이터 계층은 데이터베이스와 그것에 액세스해서 읽거나 쓰는 것을 관리하는 프로그램을 포함한다. 애플리케이션의 조직은 이것보다 더욱 복잡해질 수 있지만, 3계층 관점은 대규모 프로그램에서 일부분에 관해 생각하기에 편리한 방법이다.



## [ 모델 ]

다양한 비즈니스 로직, DB 연동 로직 그리고 처리 결과를 저장하는 기능을 지원하는 Java 객체이다.



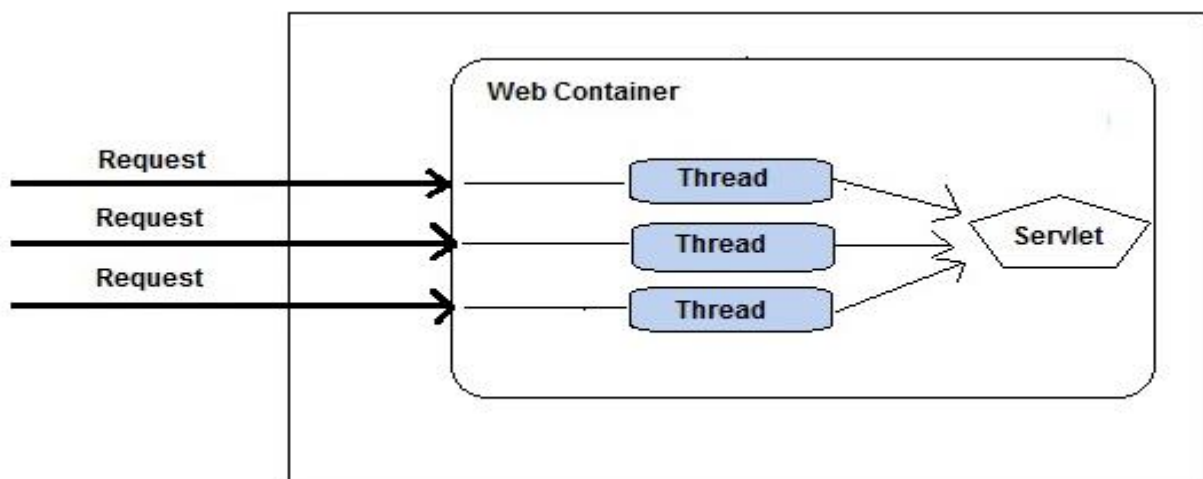
## [ 서블릿 ]

Java 서블릿(Java Servlet)은 Java를 사용하여 Web페이지를 동적으로 생성하는 서버측 프로그램 혹은 그 사양을 말하며, 흔히 "서블릿"이라 불린다. Java 서블릿은 Java EE 사양의 일부분으로, 주로 이 기능을 이용하여 쇼핑몰이나 온라인 뱅킹 등의 다양한 Web 시스템이 구현되고 있다.

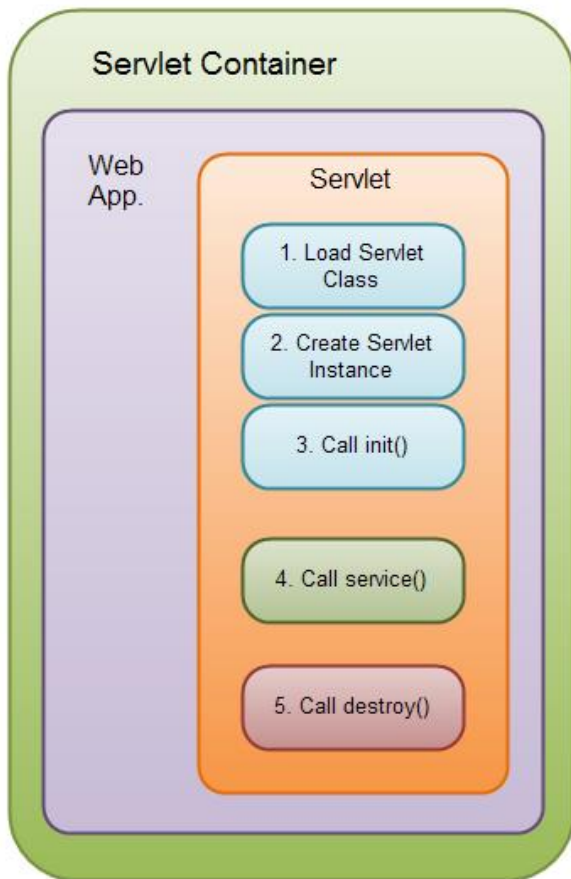


비슷한 기술로는 펄 등을 이용한 CGI, PHP를 아파치 Web 서버 프로세스에서 동작하게 하는 mod\_php, 마이크로소프트사의 IIS에서 동작하는 ASP 등이 있다. CGI는 요청이 있을 때마다 새로운 프로세스가 생성되어 응답하는 데 비해, Java 서블릿은 외부 요청마다 프로세스보다 가벼운 스레드 기반으로 응답하므로 보다 가볍다. 또한, Java 서블릿은 Java로 구현되므로 다양한 플랫폼에서 동작한다.

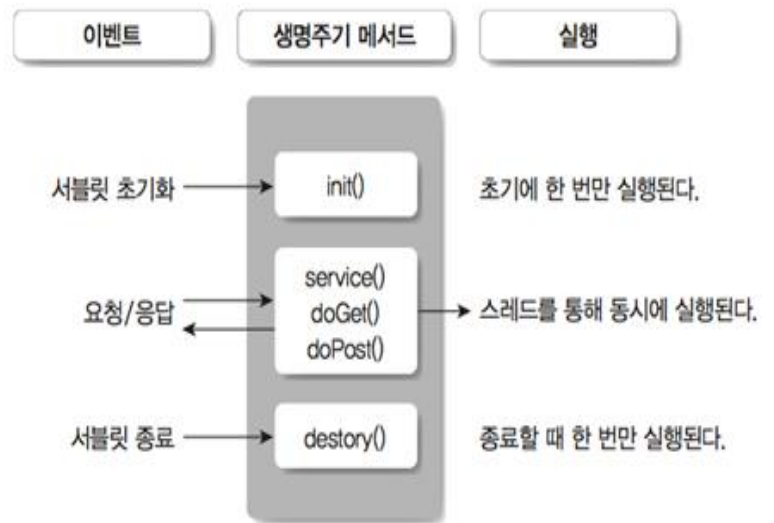
Web 클라이언트로부터의 수행 요청으로 생성된 서블릿의 객체는 객체 상태를 계속 유지하면서 다음 요청에 대하여 바로 수행될 수 있는 상태를 유지한다. 또한 하나의 서블릿을 여러 클라이언트가 동시 요청했을 때 하나의 서블릿 객체를 공유하여 다중 스레드 기반에서 처리되므로 응답 성능을 향상시킬 수 있다.



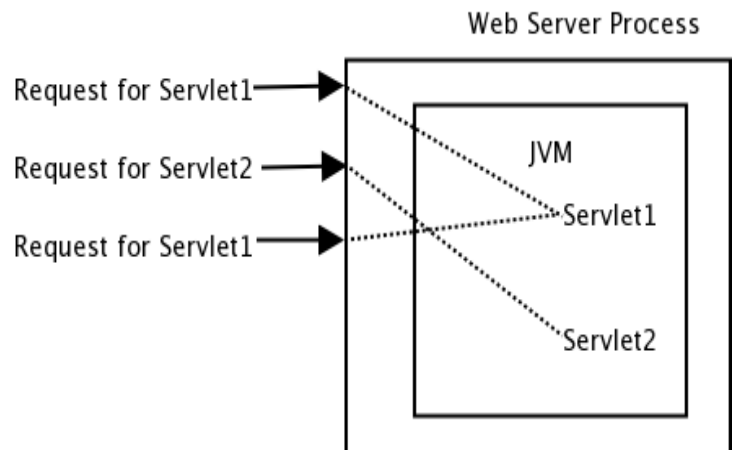
\* 서블릿의 최초 요청시 수행 흐름



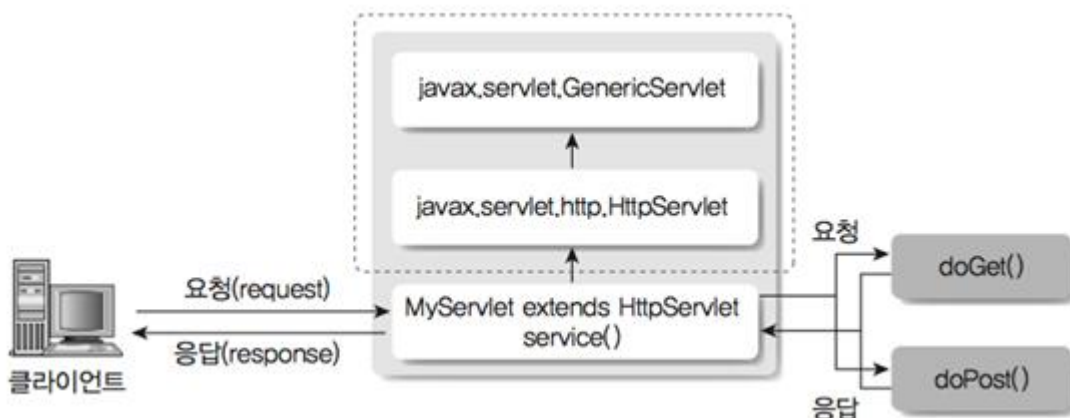
\* 서블릿 수행의 라이프사이클



\* 요청에 따른 서블릿 객체 공유



\* 요청 방식(GET, POST)에 따른 메서드 호출



## ■ Annotation 을 이용한 서블릿 등록

Servlet 3.0(JSR-315) 부터는 개발 편의성의 증대라는 목표로 Annotation을 통한 선언적 프로그래밍 그리고 손쉬운 웹 어플리케이션 설정 등을 지원한다. web.xml 에 대한 작성과 배포 기술자 파일 없이도 Web Application 의 개발이 가능해졌다.

지원되는 Annotation 의 종류 :

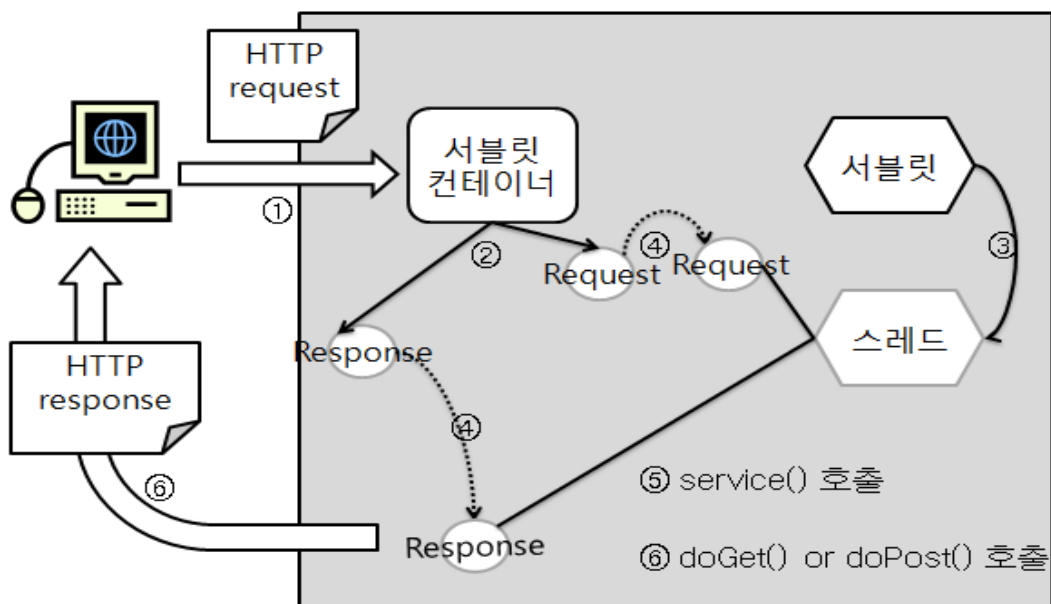
- **@WebServlet** – Define a Servlet
- **@WebFilter** – Define a Filter
- **@WebListener** – Define a Listener
- **@WebInitParam** – Define init params
- **@MultipartConfig** – Define fileupload

```
@WebServlet("/test")
public class TestServlet extends HttpServlet {
    ...
}

@WebServlet(name="testServlet", urlPatterns={"/test", "/test.do"})
public class TestServlet extends HttpServlet {
    ...
}
```

## ■ 요청 및 응답 객체 생성

Web 서버로부터 서블릿에 대한 수행 요청이 전달되면, 서블릿 컨테이너는 요청 정보를 이용해서 HttpServletRequest 객체와 HttpServletResponse 객체를 생성하게 된다. HttpServletRequest 객체는 클라이언트의 요청 정보를 담고 있으며, HttpServletResponse 객체는 클라이언트로 전달될 응답 객체를 나타낸다.



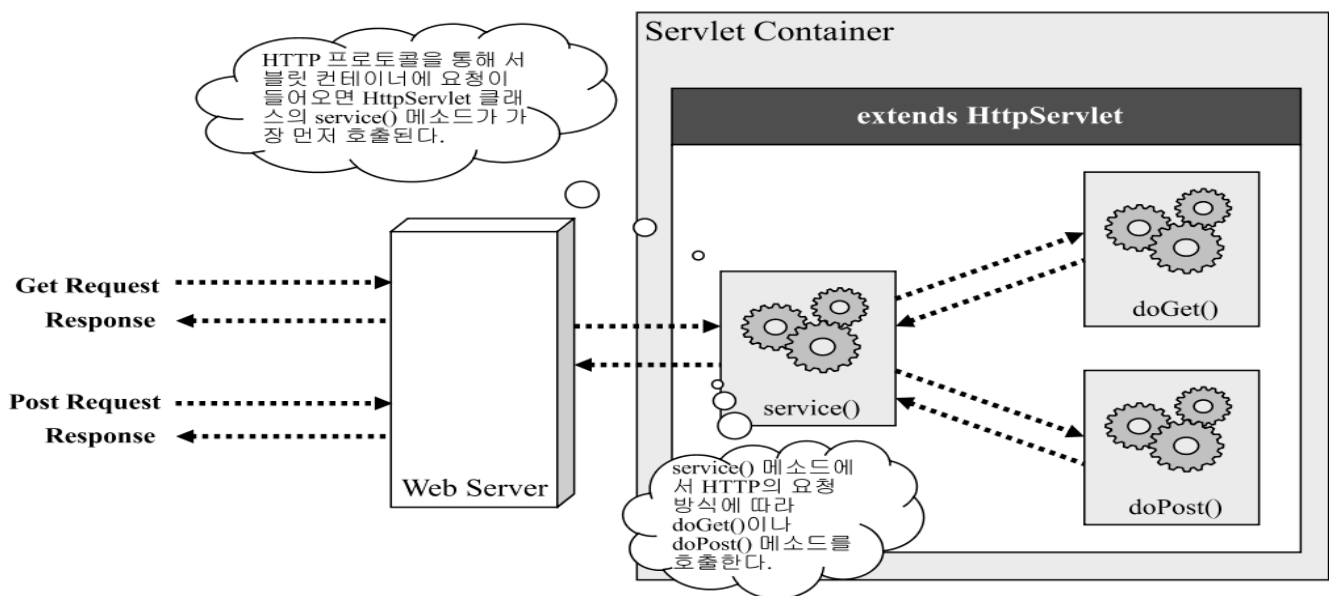


## ■ 서블릿 객체 생성

HttpServletRequest 와 HttpServletResponse 객체를 생성한 후 서블릿 컨테이너는 요청된 서블릿이 서블릿 컨테이너에 존재하는지 검사하게 된다. 만약 서블릿 객체가 서블릿 컨테이너에 로딩되어 있다면 해당 서블릿 객체를 생성하고, 로딩되어 있지 않다면 해당 서블릿의 클래스 파일을 로딩한 후 객체를 생성하게 된다.

## ■ 서블릿의 실행

서블릿은 HttpServlet 클래스를 상속받아 어떠한 요청 방식을 지원하는 서블릿인가에 따라서 doGet() 또는 doPost() 메소드를 재정의하여 구현한다. Web 클라이언트로부터 서블릿이 요청된 방식에 따라서 doGet() 또는 doPost() 메소드가 서블릿 컨테이너에 의해 호출되어 서블릿의 기능을 처리하게 된다.



## ■ Query 문자열(요청 파라미터)

Query 문자열이란 Web 클라이언트에서 Web 서버에 요청을 보낼 때 추가로 전달하는 name 과 value 로 구성되는 문자열로서 전달방식은 GET 방식과 POST 방식으로 나뉜다.

**name=value&name=value&name=valu&.....**

영문대소문자와 숫자는 그대로 전달되지만 그외의 문자들은 % 기호와 함께 16 진수 코드값으로 전달, 공백은 + 로 전달

- GET

전달되는 Query 문자열의 길이에 제한이 있고 내용이 브라우저의 주소 필드에 보여진다.

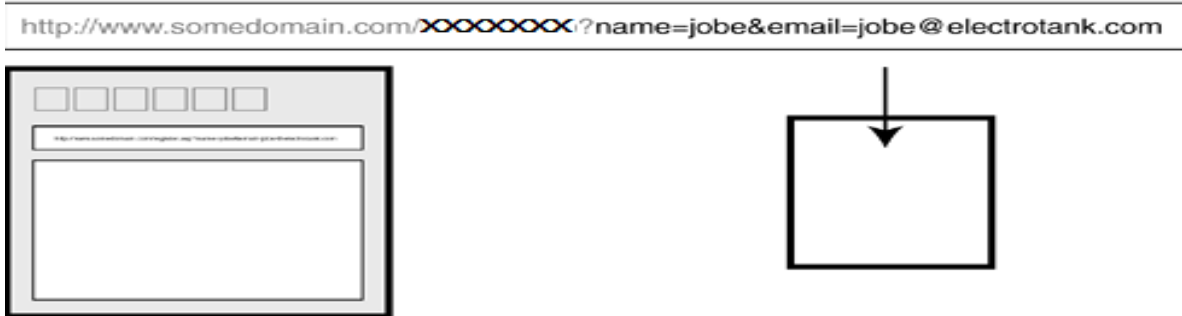
<FORM> 태그를 사용해도 되고 요청 URL 에 ? 기호와 함께 직접 Query 문자열을 붙여서 전달하는 것도 가능하다.

- POST

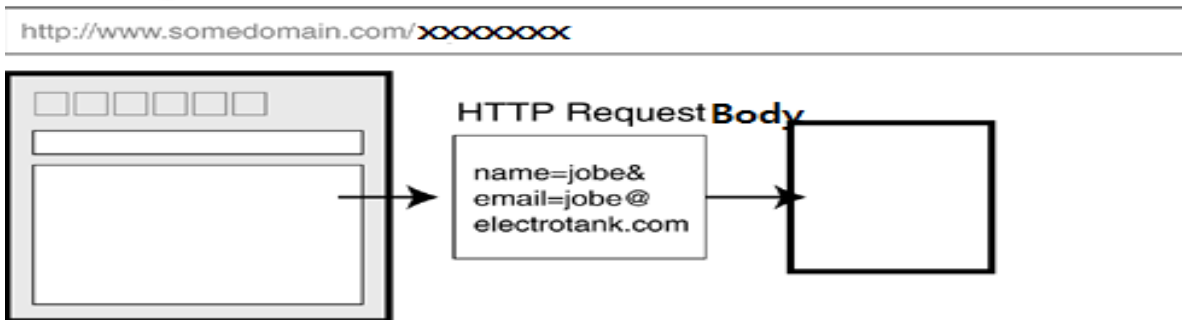
전달되는 Query 문자열의 길이에 제한이 없고 내용이 브라우저의 주소 필드에 보여지지 않는다.

요청 바디안에 담겨져서 전달된다. <FORM> 태그를 통해서만 사용할 수 있다.

## Using GET



## Using POST



- Query 문자열의 추출

주어진 name 으로 하나의 value 가 전달될 때

```
String address = request.getParameter("address");
```

주어진 name 으로 여러 개의 value 가 전달될 때

```
String hobby[ ] = request.getParameterValues("hobby");
```

request.getParameter("name") 는 전달된 Query 문자열에서 name 이름으로 전달된 name=value 쌍이 없는 경우 null을 리턴하고 name 은 있는데 value 가 비어 있는 경우에는 널 문자열("")을 리턴한다.

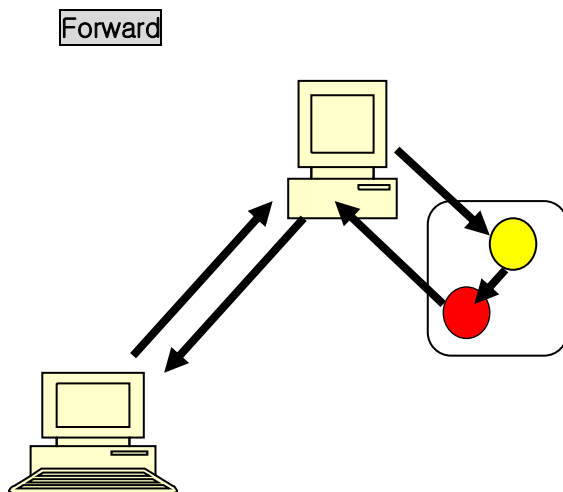
request.getParameterValues("name")는 주로 멀티체크박스나 다중 선택 리스트의 경우 Query 문자열을 추출하기 위해 사용하며 클라이언트에서 선택된 것이 없는 경우에는 null 을 리턴한다.

## ■ 요청 재지정

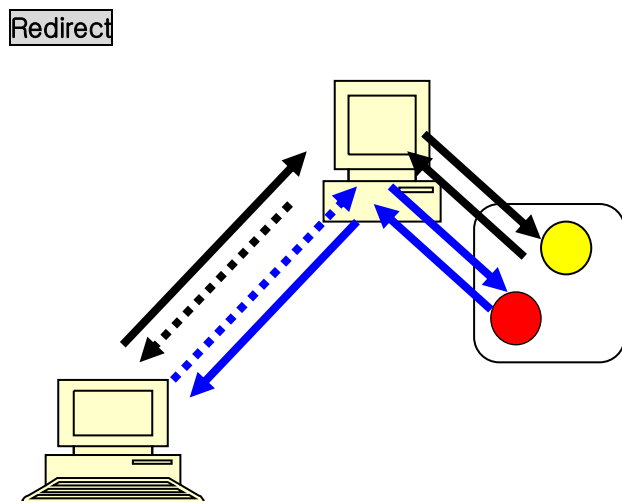
요청재지정이란 클라이언트에서 요청한 페이지 대신 다른 페이지를 클라이언트가 보게 되는 기능으로서 redirect 방법과 forward 방법으로 나뉜다.

- **redirect** : `HttpServletResponse` 의 `sendRedirect()` 메서드를 사용한다.

- **forward** : `RequestDispatcher` 의 `forward()` 메서드를 사용한다.



- 동일한 요청상에서 다른 자원에 요청을 넘겨서 대신 응답하게 함
- 동일한 서버의 동일 Web 어플리케이션에 존재하는 대상만 가능
- 브라우저의 주소필드의 URL 이 바뀌지 않음
- 두 자원이 `HttpServletRequest` 객체 공유

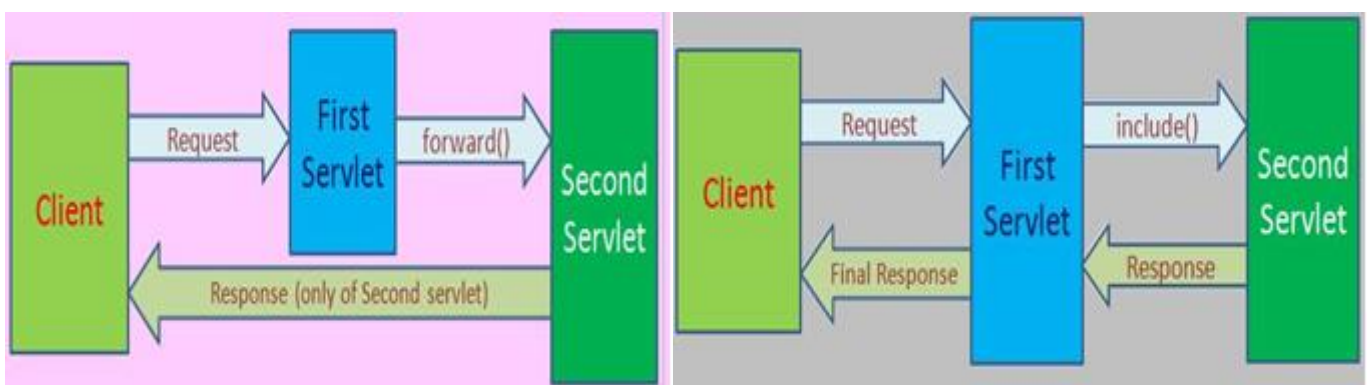


- 다른 자원을 다시 요청하여 응답하게 함
- Web 상의 모든 페이지로 요청재지정 가능
- 브라우저의 주소필드의 URL 이 바뀜
- 재지정되는 대상에 대한 요청 자체를 브라우저가 하게 됨
- 두 자원이 `HttpServletRequest` 객체를 공유하지 않음

**RequestDispatcher** 를 사용하는 요청 재지정 방법은 `forward()` 메서드 외에도 `include()` 메서드가 사용될 수 있다.

`forward()` : 요청 페이지 대신 다른 페이지가 대신 응답하게 한다.

`include()` : 요청 페이지 안에 다른 페이지의 처리 내용이 포함되어 같이 응답하게 된다.



## ■ 상태정보 유지 기술

Web 브라우저에서 Web 서버에 정보를 요청할 때 이전 접속시의 어떠한 결과(상태정보)를 일정시간 동안 유지하는 것을 상태정보 유지라고 한다. 상태정보 유지 방법은 여러 가지가 있으며




이 중에서 HttpSession 객체를 이용한 상태 정보 유지를 학습한다.

### [ HttpSession 객체를 이용한 상태정보 유지 ]

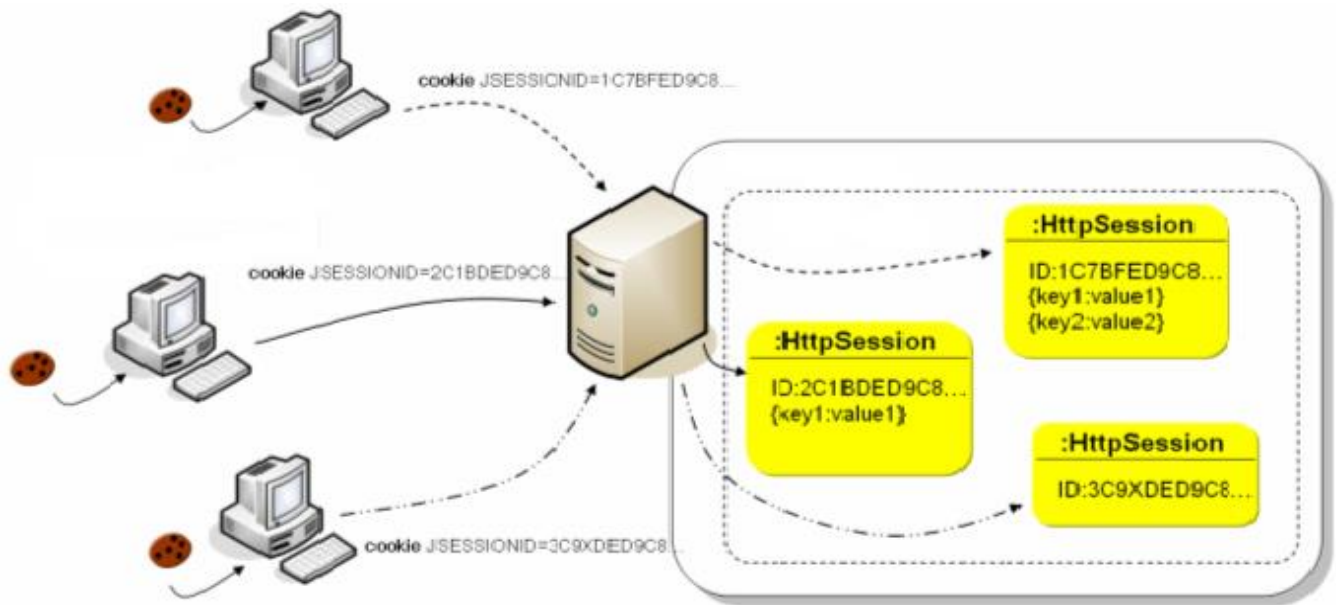
HttpSession 객체를 이용하는 상태 정보 유지는 다음과 같은 특징을 지원한다.

- 상태 정보를 서버에 보관한다.
- 상태 정보가 유지되는 최대 시간은 요청을 보내온 브라우저가 기동되어 있는 동안이다.
- 상태정보는 객체로 만들어서 보관한다.
- 구현 방법
  - (1) HttpSession 객체를 생성하거나 추출한다.
  - (2) HttpSession 객체에 상태정보를 보관할 객체를 등록한다. (한번만 등록하면 된다.)
  - (3) HttpSession 객체에 등록되어 있는 상태정보 객체의 참조 값을 얻어서 사용한다.(읽기, 변경)
  - (4) HttpSession 객체에 등록되어 있는 상태정보 객체가 더 이상 필요 없으면 삭제할 수도 있다.



```
(1) HttpSession session = request.getSession();  
(2) session.setAttribute("xxx", new Data());  
(3) Data ref = (Data)session.getAttribute("xxx");  
(4) session.removeAttribute("xxx");
```

HttpSession 객체는 Web 클라이언트별로 하나씩 만들어진다. HttpSession 객체가 생성될 때 세션ID 가 하나 부여되며 이 세션ID 는 요청을 보내온 클라이언트 브라우저에 쿠키 기술로 저장된다. 브라우저에 저장되는 세션ID 에 대한 쿠키는 최대 유지 시간이 브라우저가 기동되어 있는 동안이다.



#### ■ 객체 공유

객체의 스코프란 객체가 생성되어 유지되는 기간을 의미하며 **Page Scope, Request Scope, Session Scope** 그리고 **Application Scope** 로 구성된다.

**Page Scope** : 서블릿 또는 JSP가 수행되는 동안만 유효한 객체가 된다.

**Request Scope** : Web 클라이언트로 부터의 요청이 끝날 때까지 유효한 객체가 된다.

**Session Scope** : 요청을 보내온 Web 클라이언트가 기동되어 있는 동안 유효한 객체가 된다.

**Application Scope** : 서버가 기동되어 있는 동안 유효한 객체가 된다.

**Request Scope** → HttpServletRequest 객체에 객체를 보관한다.

**Session Scope** → HttpSession 객체에 객체를 보관한다.

**Application Scope** → ServletContext 객체에 객체를 보관한다.

HttpServletRequest, HttpSession 그리고 ServletContext 는 모두 객체를 저장하는 방으로 사용하는 것이 가능하며 다음과 같은 객체의 저장, 추출, 삭제 기능의 메서드들을 지원한다.

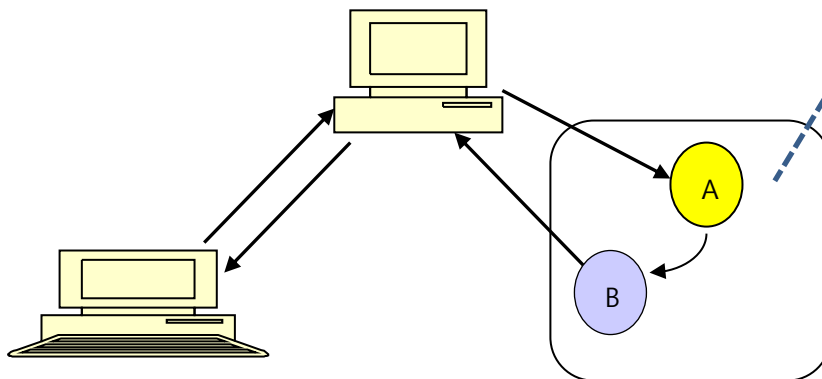


```
public void setAttribute(String key, Object value)
```

```
public Object getAttribute(String key)
```

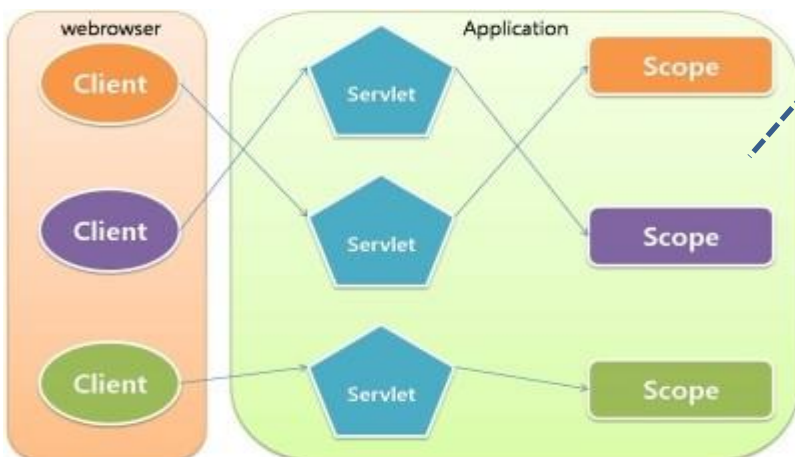
```
public void removeAttribute(String key)
```

[ 요청 동안의 객체 공유 ]



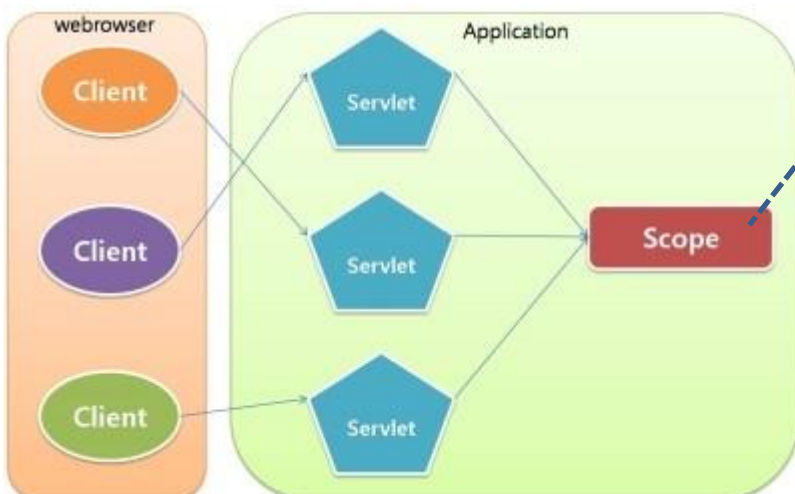
A와 B가 forward 또는 include 관계에 있는 경우 A가 생성하는 객체를 HttpServletRequest 객체에 보관하면 B에서 추출할 수 있다. 요청이 끝나면 사라진다.

[ 세션이 유지되는 동안 각 클라이언트별 객체 공유 ]



각 클라이언트별로 서버에 하나씩 만들어지는 HttpSession 객체에 객체를 보관하면 세션이 유지되는 한 계속해서 클라이언트별로 이 객체를 꺼내서 사용할 수 있다.

[ 서버가 기동되어 있는 동안 모든 클라이언트에 의한 객체 공유 ]



서버에 등록되는 Web Application 당 하나씩 만들어지는 ServletContext 객체에 객체를 보관하면 서버 종료시까지 이 객체를 꺼내서 사용할 수 있다. 이 객체는 모든 클라이언트에 의해 공유된다.

## [ JSP ]

Java 서버 페이지(JavaServer Pages, JSP)는 HTML내에 Java 코드를 삽입하여 Web 서버에서 동적으로 Web 페이지를 생성하여 Web 브라우저에 돌려주는 언어이다. Java EE 스펙 중 일부로 Web 애플리케이션 서버에서 동작한다.



Java 서버 페이지는 실행 시에 Java 서블릿으로 변환된 후 실행되므로 서블릿과 거의 유사하다고 볼 수 있다. 하지만, 서블릿과는 달리 HTML 문서에 작성되므로 Web 디자인하기에 편리하다. 이와 비슷한 구조인 것인 PHP, ASP, ASP.NET 등도 있다.

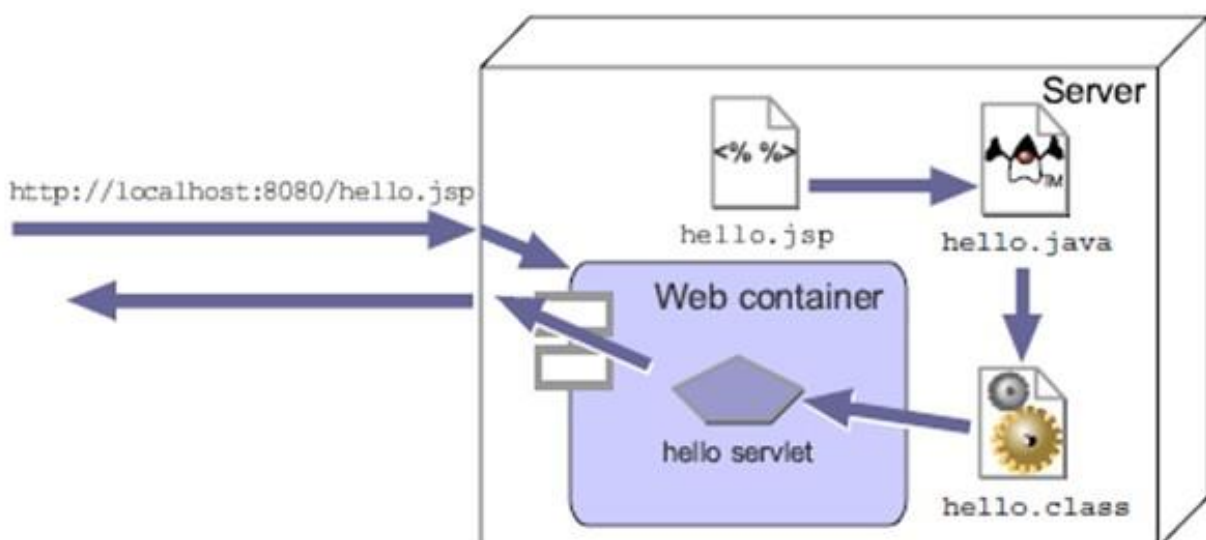
Web 페이지로 표현하고자 하는 HTML에서 동적인 처리 내용을 포함하고자 하는 부분에 JSP 태그와 Java 코드를 사용하여 구현한다.

아파치 스트럿츠나 자카르타 프로젝트의 JSTL 등의 JSP 태그 라이브러리를 사용하는 경우에는 Java 코딩없이 태그만으로 간략히 기술이 가능하므로 생산성을 높일 수 있다.

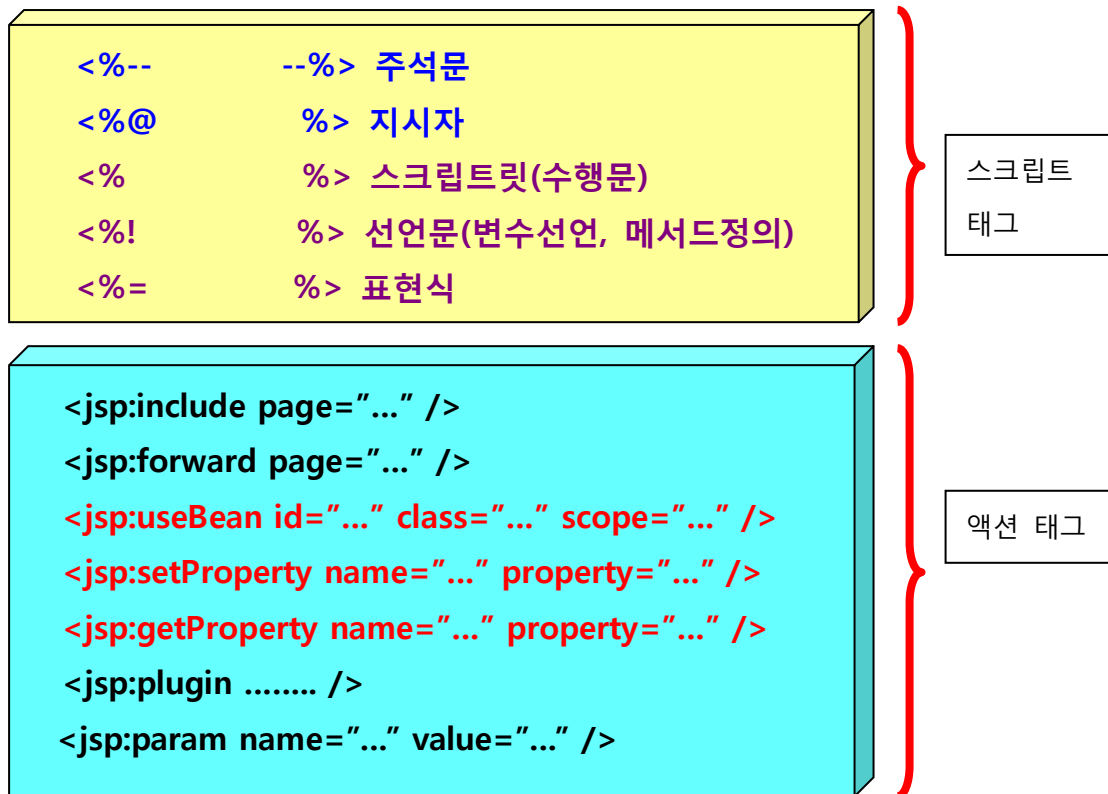
### - 동작구조

클라이언트에서 서비스가 요청되면, JSP의 실행을 요구하고, JSP는 Web 애플리케이션 서버의 서블릿 컨테이너에서 컨버터 프로그램에 의해 서블릿 원시코드로 변환된다. 그 후에 서블릿 원시코드는 바로 컴파일된 후 실행되어 결과를 HTML 형태로 클라이언트에 돌려준다.

서블릿 원시 코드로의 변환은 JSP 가 작성 또는 수정된 후 최초 요청 시에만 수행된다.



## ■ JSP 태그의 종류



### Commenting

`<%--calculate the sum of x and z here --%>`

### Directives – Specify set up in about JSP page

`<%@ page ..... %>`

e.g. `<%@page contentType ="text/html" %>`

`<%@ include ..... %>`

e.g. `<%@ include file="relativeURL" %>`

`<%@ taglib ..... %>`

e.g. `<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>`

### Scripting Elements – Enable Java to be imbedded in JSP page

Expressions `<%=....%>` e.g. `<%= a + b + c %>`

Scriptlets `<%....%>` e.g. `<% for(int i=1;i<=10;i++) { %> <%=i%> <br/> <%}%>`

Declarations `<%!....%>` e.g. `<%! int a, b, c; %>`

### Action Elements – These allow developers to code in tags rather than scriptlet programming

`<jsp:forward>` e.g. `<jsp:forward page="errorpage.jsp"/>`

`<jsp:include>` e.g. `<jsp:include page="hello.jsp"/>`

`<jsp:param>` e.g. `<jsp:param name="username" value="jsmith" />`

## ■ JSP의 내장 객체

JSP는 표현식(expression)과 스크립트릿 (scriptlets)에서 코드를 심플하게 만들어 주기 위해 내장 객체라는 것을 지원하고 있다. 내장 객체를 선언하고 초기화 해주는 일은 JSP컨테이너가 JSP 페이지에 대한 구현 서블릿 소스를 생성하는 과정에서 자동적으로 하게 된다.

JSP의 스크립팅 요소에는 Java에서 제공하는 표준 API를 이용해서 객체를 생성하거나 활용할 수 있다. 또한 개발자가 만든 클래스도 JSP페이지내에서 생성하여 사용할 수도 있다. JSP페이지내에서 이런 표준 API 및 클래스를 활용하는 방법은, 지시자를 사용해서 이용할 패키지를 import하고 해당 객체를 생성 혹은 참조하는 프로그램을 작성하는 것이다. 내장 객체는 내부적으로 정의된 객체이므로 이런 과정을 필요로 하지 않는다.

JSP에서는 자주 사용되는 API 들을 내장 객체라는 것으로 지원한다.

**request, response, out, session, application**  
config, exception, pageContext, page

객체변수	클래스 및 인터페이스	설 명
request	<b>http.HttpServletRequest</b>	요청데이터
response	<b>http.HttpServletResponse</b>	응답데이터
pageContext	<b>jsp.PageContext</b>	페이지가 처리되는 시점에서의 외부 환경 데이터
session	<b>http.HttpSession</b>	사용자마다의 세션 데이터
application	<b>ServletContext</b>	모든 어플리케이션 페이지가 공유하는 데이터
config	<b>ServletConfig</b>	서블릿 구성 데이터
out	<b>jsp.JspWriter</b>	페이지 콘텐츠 출력용 스트림
page	<b>jsp.HttpJspPage</b>	페이지의 서블릿 인스턴스
exception	java.lang.Throwable	처리되지 않은 에러나 예외

javax.servlet 의  
서브패키지들임

## ■ EL(Expression Language)

특정 스코프 영역에 보관되어 있는 객체를 추출하여 이 객체의 값 또는 속성 값을 추출하여 표현하고 하는 경우 적절한 Java 코드와 함께 표현식 태그를 사용해도 되지만 JSP 가 추가로 지원하는 Expression Language 라는 구문으로 좀 더 간단하게 구현하는 것이 가능하다.

EL 은 \$ 와 블록({ })으로 구현하는 것으로 표현하는 것과 관련된 연산자, 내장 객체 등을 지원한다. Query 문자열을 추출하여 표현하는 경우에도 다음과 같이 스크립트 태그를 사용하는 것보다 간단하게 구현할 수 있다.

```
<% out.println(request.getParameter("q")); %>
```

```
<%= request.getParameter("q") %>
```

**`${param.q}` 또는 `${param["q"]}`**

- EL(Expression Language)의 내장 객체

pageContext - PageContext 객체

pageScope - page 스코프에 포함된 객체들

requestScope - request 스코프에 포함된 객체들

sessionScope - session 스코프에 포함된 객체들

applicationScope - application 스코프에 포함된 객체들

param - HTTP의 파라미터들

paramValues - 한 파라미터의 값들

header - 헤더 정보들

headerValues - 한 헤더의 값들

cookie - 쿠키들

initParam - 컨텍스트의 초기화 파라미터들

HttpSession 객체에 cart 라는 명칭으로 저장된 객체의 getApple() 을 호출하여 리턴된 결과를 표현하려면 다음과 같이 구현한다.

```
${ sessionScope.cart.apple } 또는 ${ cart.apple }
```

- EL 에서의 . 연산자

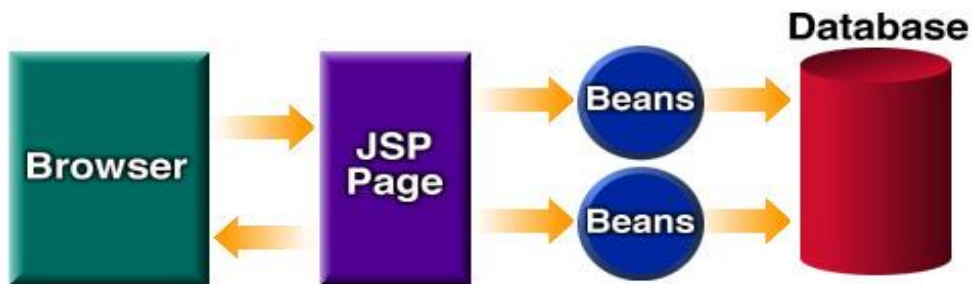
**변수명.xxx**

(1) 변수의 참조 대상이 일반 Java 객체이면 **getXxx()** 를 호출한 결과가 된다.

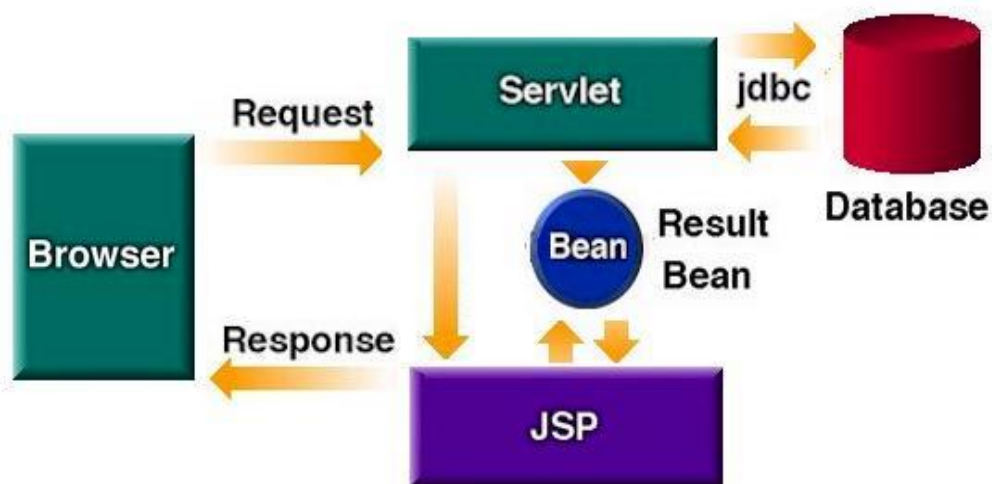
(2) 변수의 참조 대상이 Map 객체이면 **get("xxx")** 을 호출한 결과가 된다.



\* 모델 1



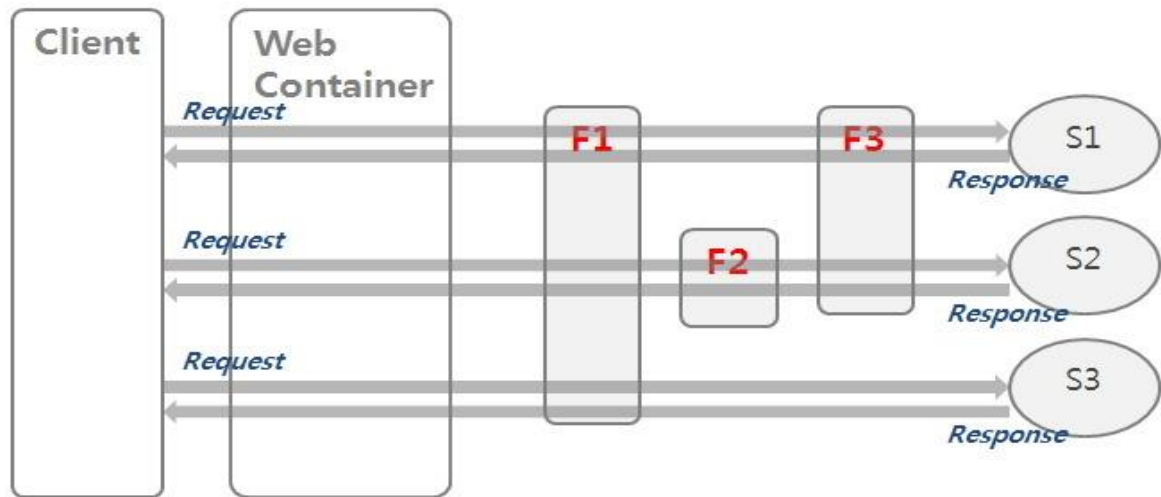
\* 모델 2 → MVC



[ Filter ]

**Filter** 란 Web 클라이언트에서 요청한 Web 자원들(서블릿 또는 JSP)이 수행되기 전 또는 후에 수행되는 객체로서 request 또는 response에 영향을 주거나 또는 특정 처리를 할 수 있다. Filter의 응용 예로 인증, 로깅, 이미지 변환, 데이터 압축, 암호화, 스트림 토큰화, XML 변환 등이 있다.

Web 자원이 순서대로 하나 또는 두 개 이상의 Filter들의 chain에 의해 필터링 되도록 설정 할 수 있다.



Filter 구현 시에는 javax.servlet.Filter 라는 인터페이스를 상속하여 init(), doFilter(), destroy() 를 오버라이딩 한다.

**@WebFilter("/")**

```

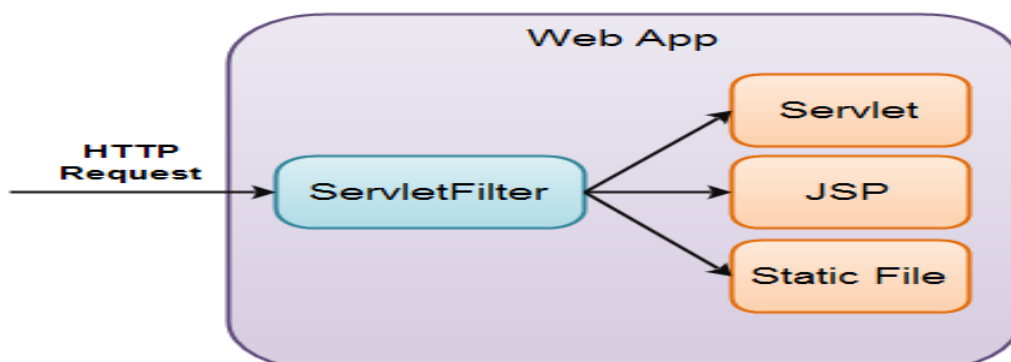
public class MyFilter implements Filter {
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
        throws IOException, ServletException {

        // 클라이언트에서 요청한 Web 자원이 수행하기 전에 처리할 기능
        chain.doFilter(req, res);

        // 클라이언트에서 요청한 Web 자원이 수행한 후에 처리할 기능
    }

    public void destroy() {
    }

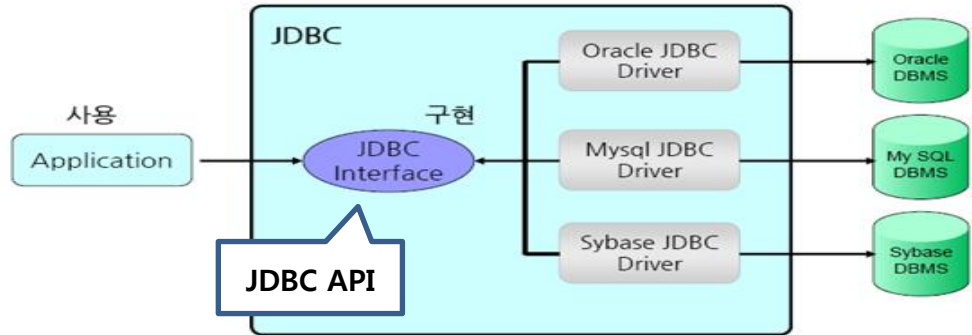
    public void init(FilterConfig fConfig) throws ServletException {
    }
}
  
```



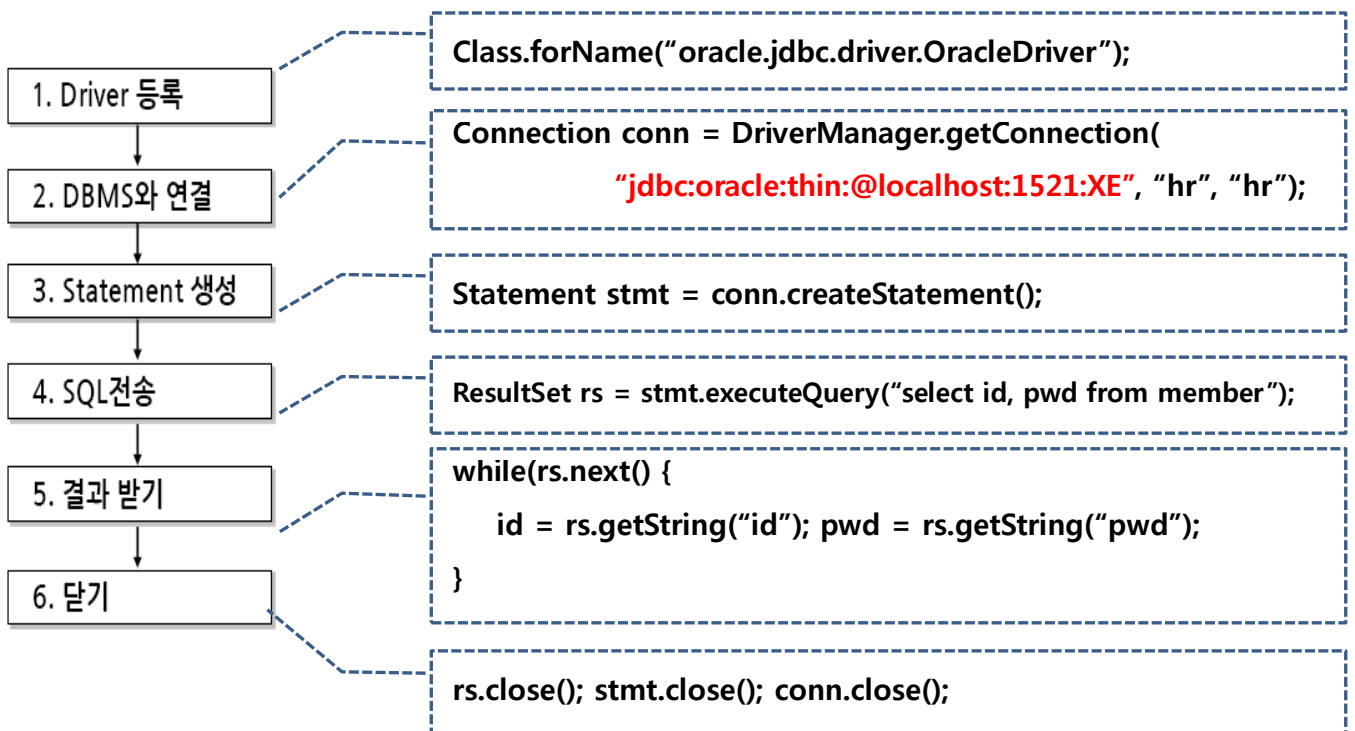
# JDBC



JDBC(Java Database Connectivity)는 Java에서 데이터베이스에 접속하여 기능을 처리할 수 있도록 지원하는 Java API이다. JDBC는 데이터베이스에서 자료를 쿼리하거나 업데이트하는 방법을 제공한다. JDBC API 는 java.sql, javax.sql 에서 Java의 표준 API 로 제공되지만 JDBC 드라이버는 연동하려는 DB 서버에 알맞은 것을 추가로 준비해야 한다.



## ■ JDBC 프로그래밍 절차

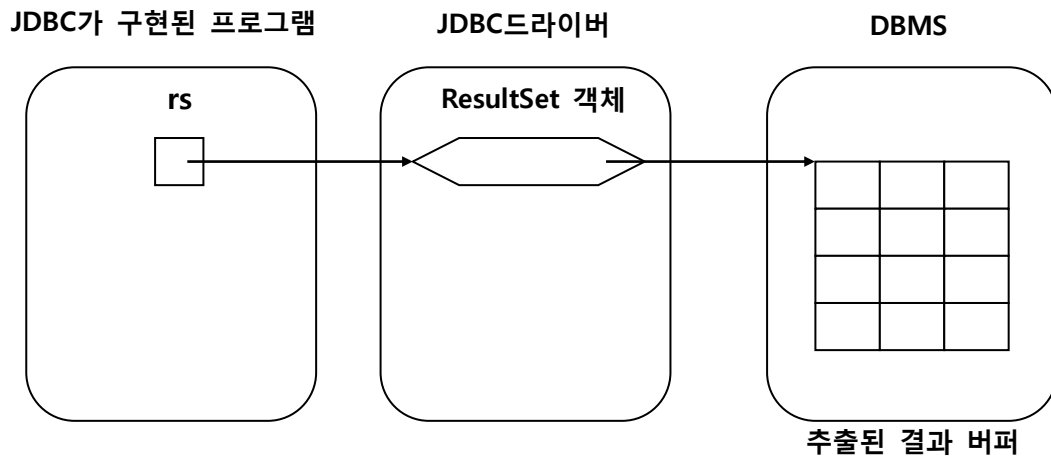


## - Statement 와 PreparedStatement

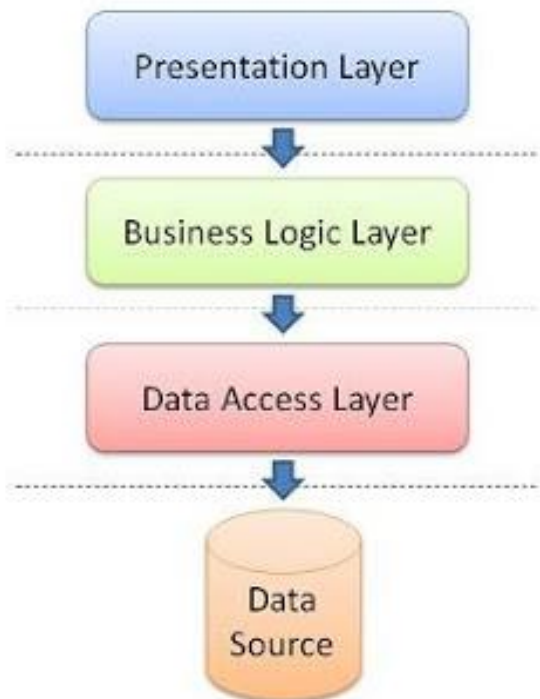
```
Statement stmt = conn.createStatement();
stmt.executeUpdate("insert into account values('"+name+"','"+ passwd+"')");

PreparedStatement pstmt = conn.prepareStatement("insert into account values (?, ?)");
pstmt.setString(1, name);
pstmt.setString(2, passwd);
pstmt.executeUpdate();
```

- SELECT 수행 결과를 처리하는 ResultSet

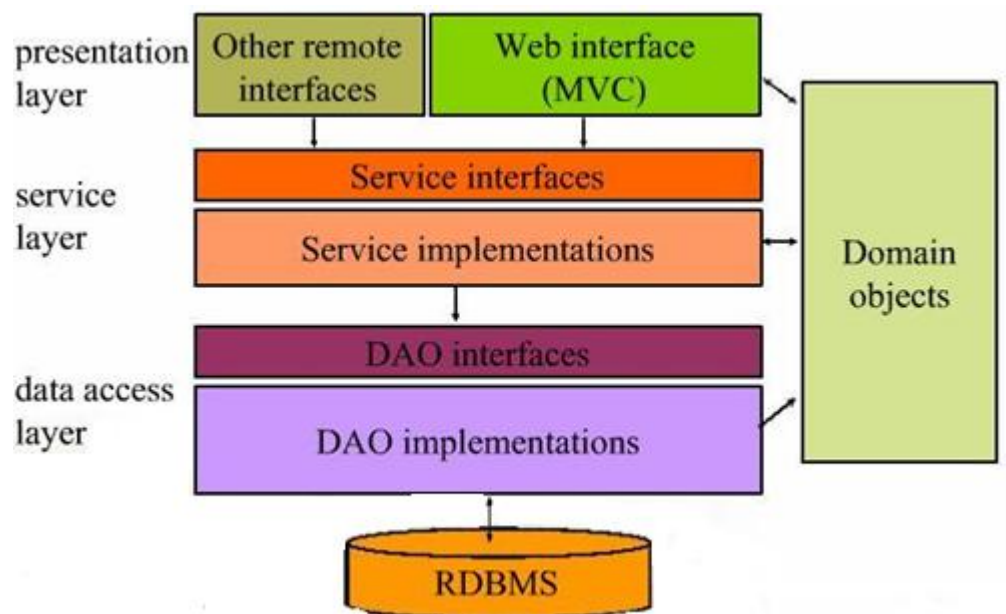


- DAL(Data Access Layer)의 구성



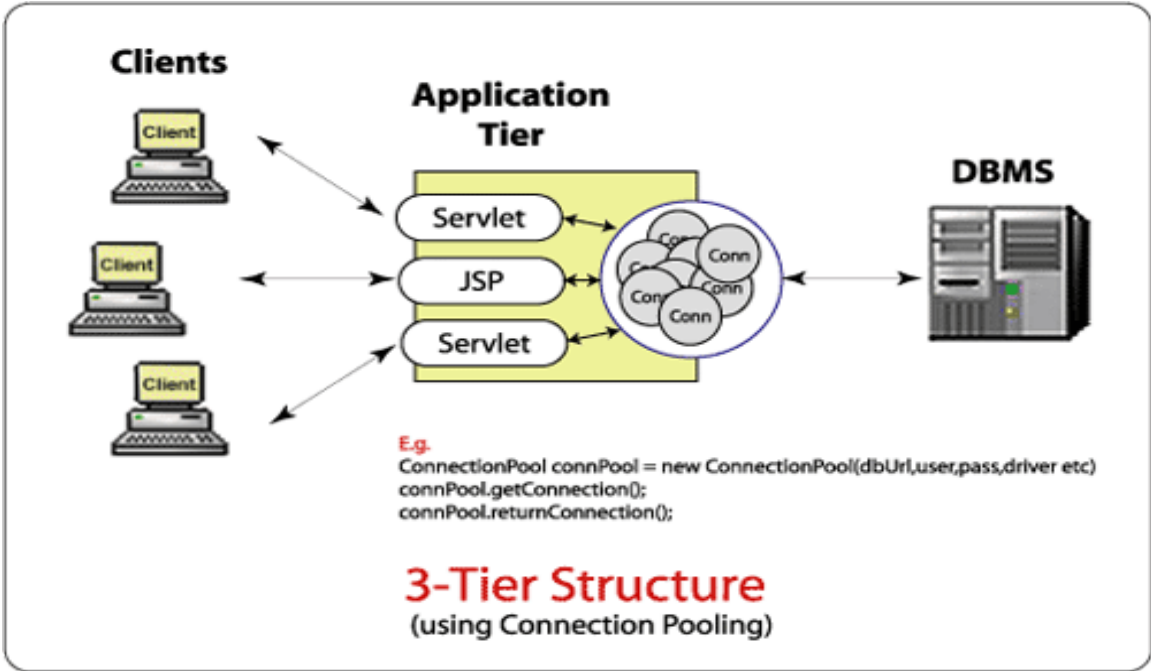
DB 연동 기능을 전담하는 객체들이 모여있는 계층으로 응용 프로그램과 DB 사이에 존재하여 DB 에 대한 세부 정보를 노출하지 않고 DB 연동에 대한 모든 역할을 처리한다.

Data Access Layer 의 기능을 구현하는데 있어서 주로 DAO(Data Access Object)라고 불리는 객체를 순수하게 JDBC 기술을 사용하여 구현할 수도 있으며 JPA, Spring JDBC, iBatis 그리고 Hibernate 와 같은 프레임워크 기술을 사용할 수도 있다.

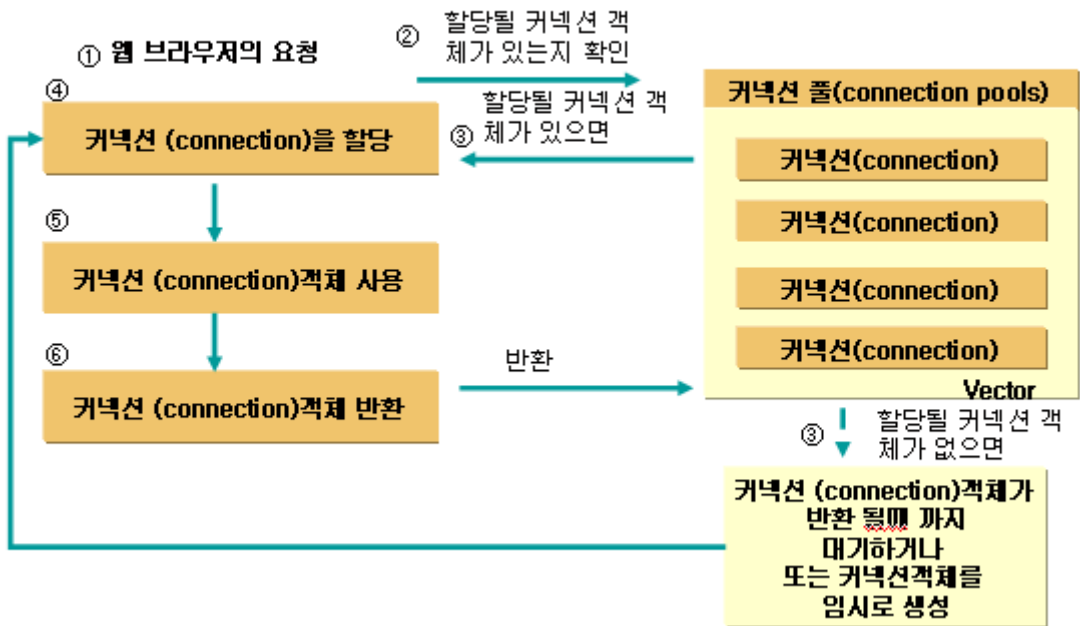


## ■ Connection Pool

Connection Pool 이란 데이터베이스와 연결된 커넥션을 미리 만들어서 풀(pool)이라는 저장소에 저장해 두고 필요할 때에 커넥션을 풀에서 가져다 쓰고 다시 풀에 반환하는 구현 방법이다. 클라이언트로부터 요청이 올 때마다 DB 서버에 접속하고 Connection 객체를 생성하는 부분에서 발생하는 대기 시간을 줄여서 효율적으로 DB 연동을 수행할 수 있도록 지원하는 기술이므로 Web 서버 프로그램 구현 시 필수적으로 사용되는 기술이다.



[ Connection Pool 의 처리 과정 ]



■ MVC 패턴을 적용한 공지사항 처리 프로그램의 구조

