



Spring Framework

스프링 프레임 워크

김 일한

- ilhank@naver.com
- (주)아이코어이앤씨 연구소장

- 교재(참고용)
- 수업시간 : 08:00 ~17:00
- 점심시간: 12:00 ~13:00
- 쉬는 시간 : 매시 정각 (10~15분:1,5) :
- 과제: 16:00~17:00 (평가X, 반드시 제출)
- 설정- 화질 (720p 권장)

사전 지식

- html
- css
- javascript
- jsp

- 필수지식:java

1일차				2일차				3일차				4일차				5일차			
과정내용	시간	교수법	장비사용유무	과정내용	시간	교수법	장비사용유무	과정내용	시간	교수법	장비사용유무	과정내용	시간	교수법	장비사용유무	과정내용	시간	교수법	장비사용유무
웹 개발환경 구축 및 설정	1	이론 및 실습	X	Servlet API	1	이론 및 실습	X	REST API 구현하기	1	이론 및 실습	X	오라클 데이터베이스 설치 및 설정	1	실습	X	MyBatis를 활용한 Database 연동	1	이론 및 실습	X
웹의 개요	1	이론 및 실습	X	Servlet Parameter 처리	1	이론 및 실습	X	Annotation 사용(@Controller, @RequestMapping)	1	이론 및 실습	X	Spring과 ORM 연동	1	실습	X	MyBatis를 활용한 Database 연동	1	이론 및 실습	X
JSP 이해	1	이론 및 실습	X	JSP Tag, JavaBean,	1	이론 및 실습	X	Annotation 사용(@Controller, @RequestMapping)	1	이론 및 실습	X	DB Pool	1	이론 및 실습	X	MyBatis를 활용한 Database 연동	1	이론 및 실습	X
지시자	1	이론 및 실습	X	Spring Container의 종류와 특징	1	이론 및 실습	X	파일업로드와 예외 처리	1	이론 및 실습	X	DB Pool	1	이론 및 실습	X	실전 게시판 구현	1	이론 및 실습	X
점심식사																			
표현식	1	이론 및 실습	X	Spring 설정파일의 구조	1	이론 및 실습	X	파일업로드와 예외 처리	1	이론 및 실습	X	MyBatis의 개념과 구조	1	이론 및 실습	X	실전 게시판 구현	1	실습	X
스크립트릿	1	이론 및 실습	X	Spring MVC 기초	1	이론 및 실습	X	스프링 AOP	1	이론 및 실습	X	MyBatis를 활용한 Database 연동	1	이론 및 실습	X	과제 시험	1	실습	X
Web Component Overview	1	이론 및 실습	X	Spring MVC 활용	1	이론 및 실습	X	스프링 AOP	1	이론 및 실습	X	MyBatis를 활용한 Database 연동	1	이론 및 실습	X	과제 시험	1	실습	X
jsp 연습문제 및 풀이	1	실습	X	스프링 연습문제 및 풀이	1	실습	X	스프링 연습문제 및 풀이	1	이론 및 실습	X	스프링 연습문제 및 풀이	1	이론 및 실습	X	과정리뷰	1	이론	X
	8				8				8				8				8		

웹의 개요

목차

- 웹의 역사
- 웹 프로토콜의 구조
- 브라우저의 동작 구조
- 프론트엔드 프레임워크의 비교

마크업의 개념

- 웹의 시대
- 국가간 전자문서교환
- 포맷이 맞지않음?

마크업(Markup) 이란 ?

- 전통적인 마크업
 - 전통적인 출판 단계
 - 원고 작성자
 - 원고 작성
 - 서식 설계사
 - 원고의 출판 형태를 정의(서체 종류 및 크기 등)
이러한 수작업을 마크업이라 한다
 - 출판 전문가
 - 마크업된 지시대로 원고 데이터를 표현



마크업(Markup) 이란 ?

- 절차적(Procedural) 마크업
 - (예) troff 절차적 마크업

```
.bp.  
.ps 20.  
.ft I  
.ce.  
문서 기술 언어 SGML.  
.sp 2.  
.ps 10.  
.ft B  
SGML 은 "Standard generalized Markup Language"의 약자로, "문서 기  
술 언어"로 번역할 수 있다.  
.sp 2.
```

문서 기술 언어 SGML

SGML 은 "Standard generalized Markup Language" 의 약자로, "문서 기술 언어"로 번역할 수 있을 것이다.

- 인쇄 제어 명령 마크업들이 특정 분리된 라인에 위치
- 라인 맨 앞에 "."으로 시작하여 순수 텍스트와 구분
- .bp : 새로운 페이지로 개행
- .ps 20 :폰트 크기를 20
- .ft I : 글자를 이탤릭체
- .ce :텍스트를 중앙정렬
- .sp 2 : 두 행 띄우기

마크업(Markup) 이란 ?

HTML(HyperText Markup Language)

W3C의 명세

웹상에서 Hypertext 문서를 생성할 수 있는 간단한 마크업 언어

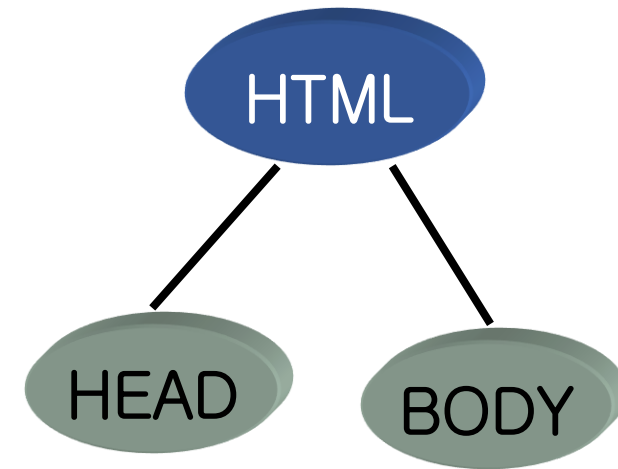
SGML의 subset

ASCII Text 양식의 문서

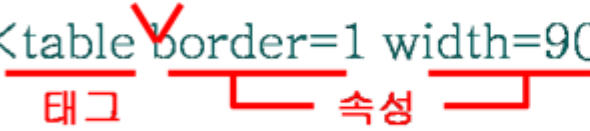
(일반 텍스트 편집기 작성 가능)

HTML = SGML 선언 + a DTD

HTML SGML



HTML

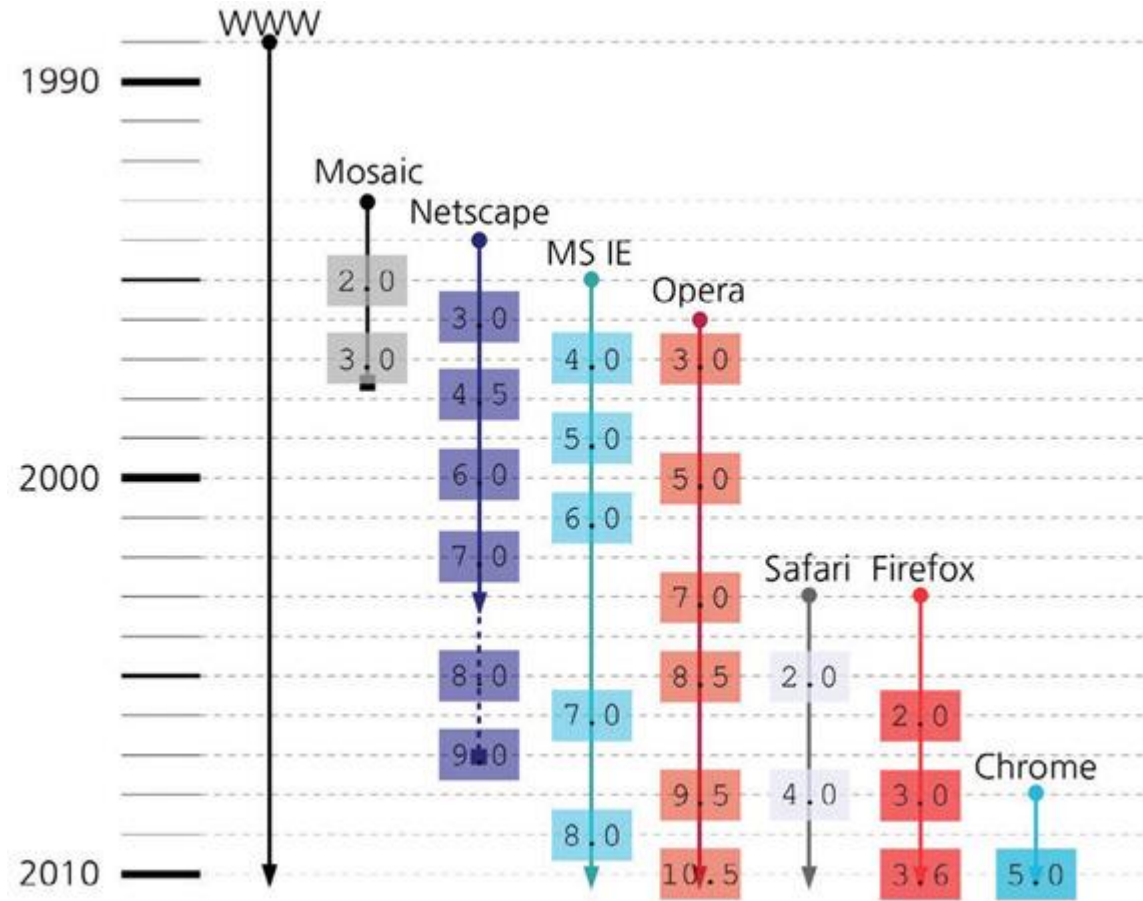
- Hyper Text Markup Language
- 웹(www)문서를 작성하기 위해 사용되는 언어
- Http 프로토콜 사용
- 한 쌍의 태그(tag)명령으로 이루어짐 – 한쌍이 아닌경우도 있음
 - `...`, `
`
- 태그 – 속성(attribute)로 옵션을 줌
 - `<table border=1 width=900>.....</table>`

- 대소문자 구별하지 않음
- 2 이상의 공백 혹은 enter 무시 – 별도의 특수문자 기호 필요
 - Enter-`
`, 공백-` `

HTML문서의 기본 구성

머릿글	<HTML>	:HTML문서의 시작
	<HEAD>	:문서의 헤더 시작
	<TITLE>	:문서의 제목 시작
	문서의 제목...	:문서의 제목 내용
	</TITLE>	:문서의 제목 끝
본문	</HEAD>	:문서의 헤더 끝
	<BODY>	:문서의 본문 시작
	본문 내용	:문서의 본문 내용
	</BODY>	:문서의 본문 끝
	</HTML>	:HTML문서의 끝

웹의 역사

웹브라우저 발달역사



출처: <http://evolutionofweb.appspot.com/>

웹브라우저 역사

- 최초의 웹브라우저는 1990년 12월 25일 영국의 소프트웨어 공학자인 팀 버너스리(Tim Berners-Lee)가 정보 공유 시스템용으로 개발한 월드와이드웹(WorldWideWeb)이었다. 처음에는 웹브라우저의 이름과 정보 공유 방법을 모두 월드와이드웹이라고 했으나, 후에 두 기능의 혼동을 방지하기 위하여 웹브라우저의 이름은 넥서스(Nexus)로, 정보 공유 방법은 월드 와이드 웹(World Wide Web) 또는 간단히 웹(web)으로 구분하였다.
- 넥서스는 단순한 웹브라우저로 그 기능이 텍스트를 처리하는 정도였다
- 모자이크(Mosaic)은 미국 일리노이 대학 슈퍼컴퓨터 응용 연구소(National Center for Supercomputing Applications: NCSC)의 마크 앤드리슨에 의하여 개발되었다.
- 윈도 환경에서 사용할 수 있도록 멀티미디어 그래픽 사용자 인터페이스를 채용하였다.
- 마크 앤드리슨이 회사를 설립하고 모자이크을 개량하여 네스케이프 내비게이터(Netscape Navigator)를 개발
- 1995년에 MS사는 스파이글래스 모자이크을 인터넷 익스플로러(Internet Explorer: IE)라는 이름으로 출시하였다. 이것이 바로 윈도95에 끼워 제공하던 IE이다.
- 윈도98과 함께 배포될 때에는 대폭적으로 성능이 개선되었다. 이 시점 이후로 IE는 네스케이프를 밀어내고 웹브라우저의 절대 강자로 군림하게 되었다.
- 1996년에 개발된 오페라(Opera)는 작고 빠른 웹브라우저라는 모토 하에 꾸준히 명맥을 이어왔다.
- 2003년 네스케이프의 몰락과 함께 모질라(Mozilla) 재단은 독자적으로 파이어폭스(FireFox)를 개발하여 지금에 이르고 있다.
- 2003년 애플사는 자체적으로 사파리(Safari)를 개발하여 매킨토시에서 아이폰에 이르는 모든 제품의 웹브라우저로 제공해오고 있다.
- 2008년에 들어와 구글 크롬(Chrome)이 출현하였다. 크롬의 가장 큰 특징은 페이지 탭별로 멀티태스킹이 가능하고 빠르다는 점이다. 이에 힘입어 크롬은 지금까지 꾸준히 점유율을 높여오고 있다

html역사

연월	사양	설명
1993년 6월	HTML 1.0	IETF Internet Draft
1995년 11월	HTML 2.0	RFC 1866
1997년 1월	HTML 3.2	W3C 권고
1997년 12월	HTML 4.0	W3C 권고
1998년 2월	XML 1.0	W3C 권고
1998년 12월	HTML 4.01	W3C 권고
2000년 1월	XHTML 1.0	W3C 권고

HTML 의 발달사

넷스케이프 → W3C → WHATWG →

W3C: 팀버너스 리가 1994년 설립한 비영리 단체로
HTML4.01을 권고한 것을 1999년 일로 10년간 HTML 최
신버전으로 있었음

WHATWG(2004년) :모질라 재단과 오페라 소프트웨어
(모든 웹사이트에 플러그인(액티브엑스) 이 들어가면서 웹
사이트가 점점 무거워짐
익스플로러를 제외한 독자적으로 새로운 웹 표준기관을 설
립함

2004년 6월 HTML5 표준을 제정하는 WHATWG가 설립됨



HTML5 배경

- 웹이 정적 문서에서 동적 프로그램으로의 변화
- XHTML2.0의 기존 웹과의 비호환성과, 엄격한 XML 규칙의 적용으로 인한 제어의 어려움

WHATWG (Web Hypertext Application Technology Working Group)

모질라재단, 애플, 오페라 소프트웨어의 세 회사가 모여 현재의 HTML4.01 기술과 호환되면서 웹의 기능과 표현 범위를 확장하고자 하는 기술 표준을 작성

html5 의 역사



W3C

VS

WHATWG

xhtml을 제안

html5을 제안
apple, mozilla, opera

html5 의 역사



W3C



WHATWG

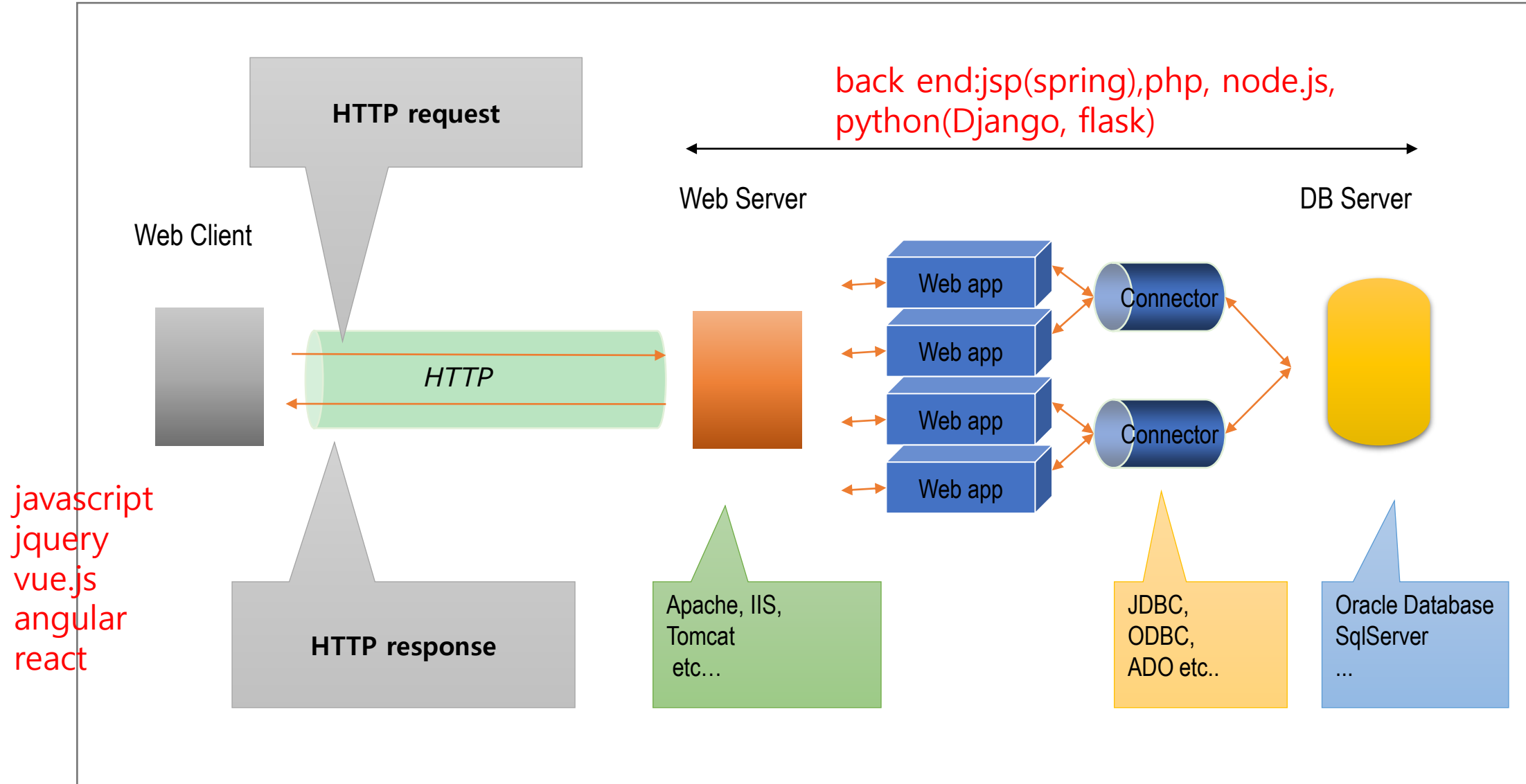
XHTML 사양 폐기
HTML5를 표준으로 채택
2009년 10월 WHATWG 모든 사양을 W3C 에 이관

html5 역사

연월	사양	설명
2011년 5월	HTML5 최종 초안	IETF Internet Draft
1995년 11월	HTML5 후보 표준안	RFC 1866
1997년 1월	HTML5 제안 표준안	W3C 권고
1997년 12월	HTML5 최종 표준안	W3C 권고

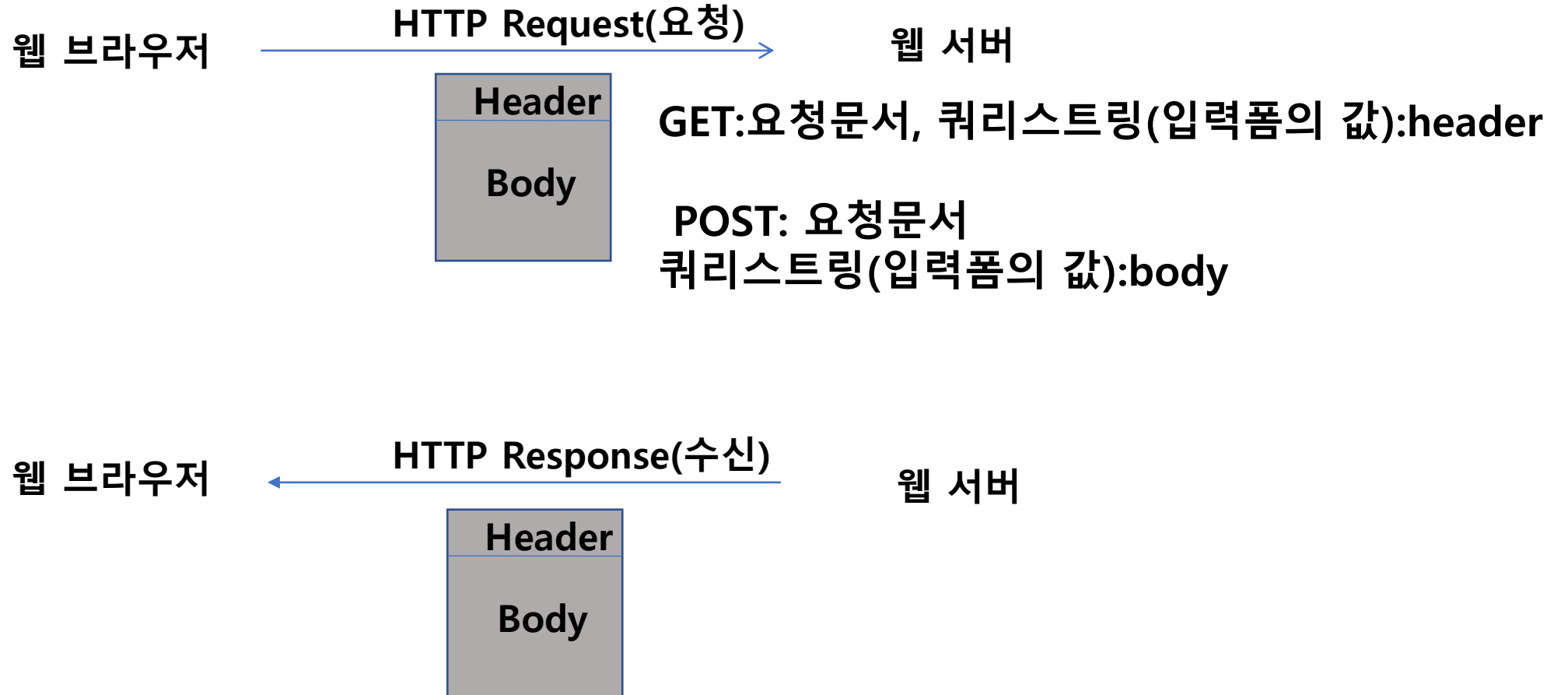
- html5
- css
- javascript
- ECMA

웹어플리케이션 아키텍처



웹프로토콜의 구조

웹 프로토콜 – HTTP Request



HTTP Request

○ HTTP Request에 포함된 상세 정보

- | | |
|--|----------------------------------|
| ① GET /index.html HTTP/1.1 | // 요청 URL 정보(메소드, 페이지) 및 HTTP 버전 |
| ② user-agent: MSIE 6.0; Windows NT 5.0 | // 사용자 웹 브라우저 종류 |
| ③ accept: text/html; */* | // 요청 데이터 타입 |
| ④ cookie: name = value | // 쿠키(인증 정보) |
| ⑤ referer: http://www.bbb.com | // 경유지 URL |
| ⑥ host: www.aaa.kr | // 요청 도메인 |

웹 프로토콜 – HTTP Request

GET Method

2,083 정도의 길이 데이터만을 처리(게시판 글 입력 처리 불가 등)

Method	구조	설 명
GET	GET [request-uri]?query_string HTTP/1.1 Host:[Hostname] 혹은 [IP]	GET 요청 방식은 요청 URI(URL)가 가지고 있는 정보를 검색하기 위해 서버 측에 요청하는 형태

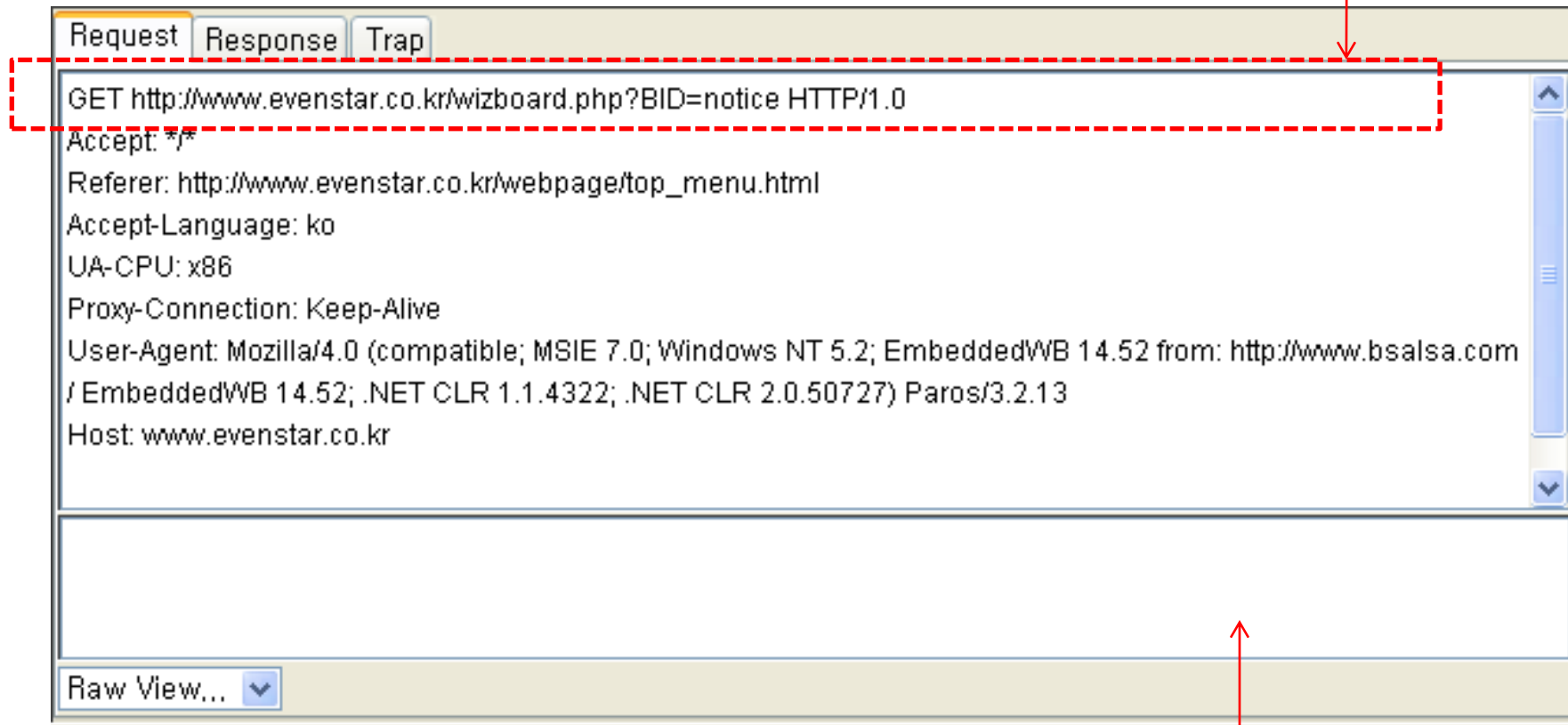
○ HTTP GET 구조 (URI + Query String)

<http://www.test.co.kr/a.html> (http header에 포함)
URI

<http://www.test.co.kr/a.jsp?name=kim> (http header에 포함)
URI Query String

○ GET Method

GET 요청



메시지(Body)는 없음

GET /home/index.html HTTP/1.1

Host:www.evenstar.co.kr

Accept:text/html, text/plain

Accept-Encoding:gzip, compress

Accept-Language:ko

If-Modified-Since:Sat,31 Jan 2004 12:00:00 GMT

User-Agent:Internet Explorer6.0

본문 (GET인 경우는 빈 공백임)

HTTP Header

요청 Header에는 메소드 종류 및 서버에 전달하는 클라이언트의 정보가 포함된다.

헤더와 본문의 경계 (1줄 빈 공백)

HTTP Body

웹 프로토콜 – HTTP Request

○ POST Method

길이 제한이 없어 많은 입력 데이터를 처리(게시판 입력 글 처리 가능)

Method	전송 형태	설 명
POST	POST [request-uri] HTTP/1.1 Host:[Hostname] 혹은 [IP] Content-Length:[Bytes] Content-Type:[Content Type]	게시판 등과 같은 폼 데이터 페이지를 위해 처리하기 위해 POST 방식으로 전송하게 되며, 웹 브라우저와 시스템 간 데이터 처리로 웹 브라우저에는 페이지 정보만을 확인할 수 있다.
	[query-string] 혹은 [데이터]	

○ HTTP POST 구조 (URI + Query String)

http://www.test.co.kr/a.jsp (http header에 포함)
URI

name=kim (http Body에 포함)
Query String

웹 프로토콜 – HTTP Response

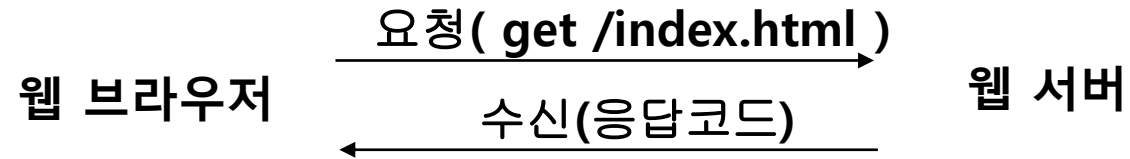
○ HTTP Response에 포함된 상세 정보

① HTTP/1.1 OK 200	// 프로토콜 버전 및 응답코드	Header
② Server: NCSA/1.4.2	// 웹 서버 배너 정보	
③ Content-type: text/html	// MIME타입	
④ Content-length: 107	// HTTP Body 사이즈	
빈 공백 1줄		
⑤ <html> <head> </head> <Title>test</Title> <body> hello web </body> </html>	// 페이지 구성 정보(HTML태그 등)	Body

- HTTP Header 포함 범위 : (1), (2), (3), (4)
- HTTP Body 포함 범위 : (5)

웹 프로토콜 – HTTP Response

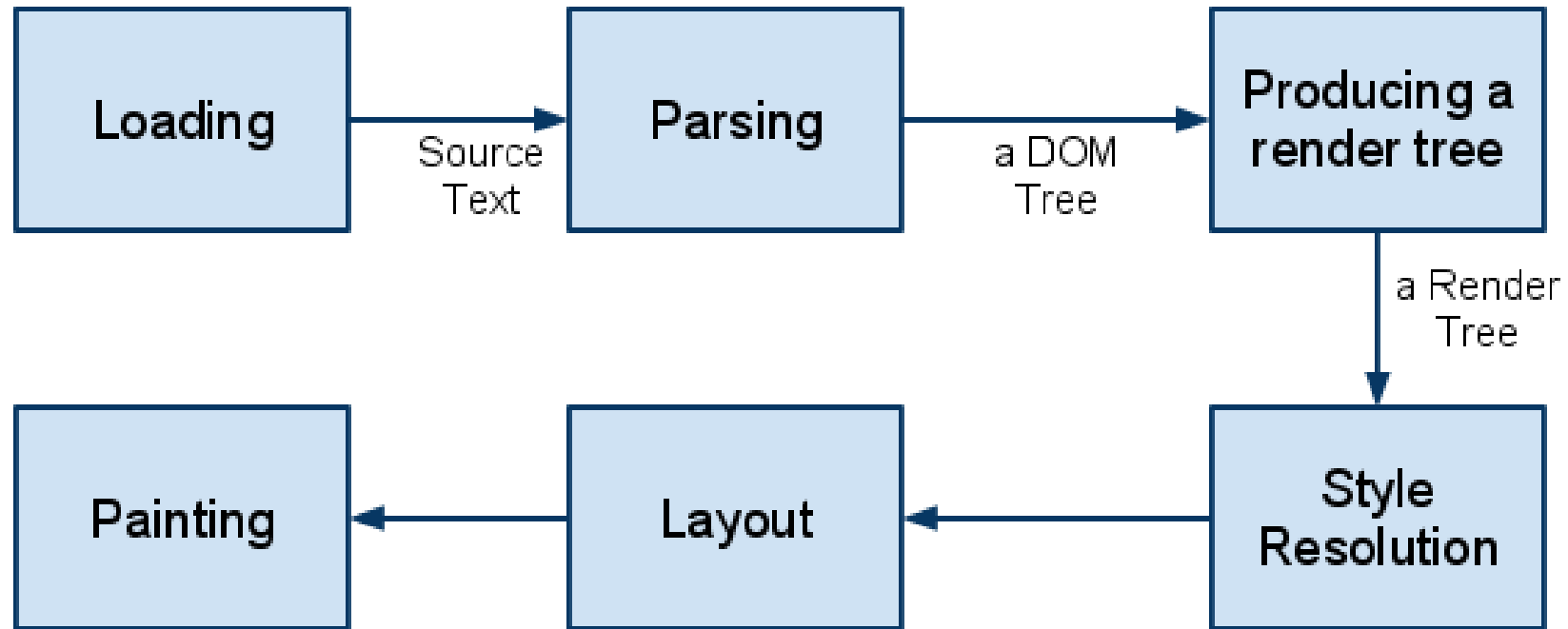
HTTP Status Code(응답코드) 종류



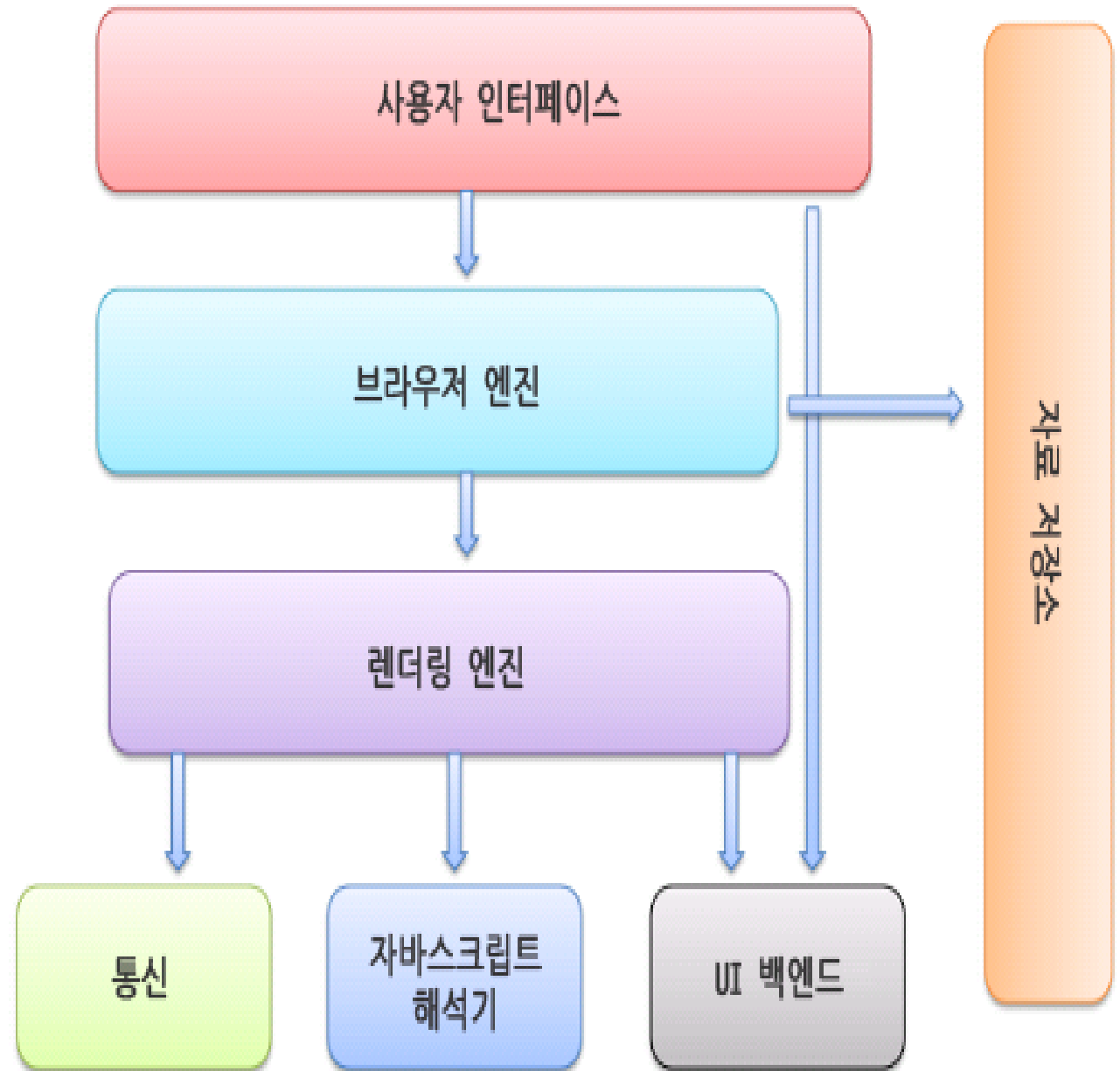
- 200번(요청성공)
- 201번(원격지서버에 파일 생성)
- 302번(페이지이동)
- 304번(로컬 캐쉬정보이용)
- 401번(인증실패)
- 403번(접근금지)
- 404번(페이지없음)
- 500번(서버에러)

브라우저의 동작구조

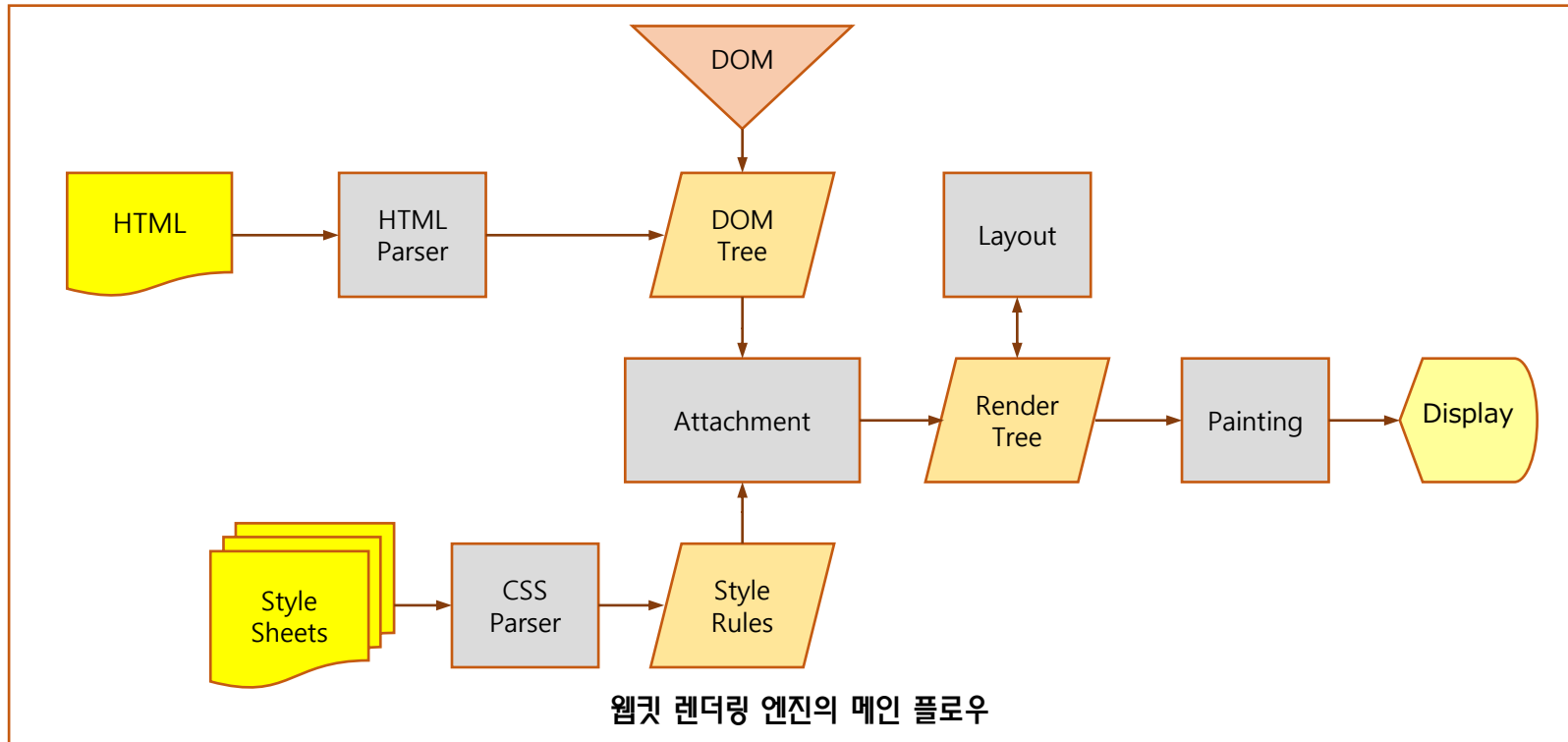
브라우저의 렌더링엔징



IE : Trident
크롬 : Webkit
파이어폭스 : Gecko
사파리 : Webkit
오페라 : Presto

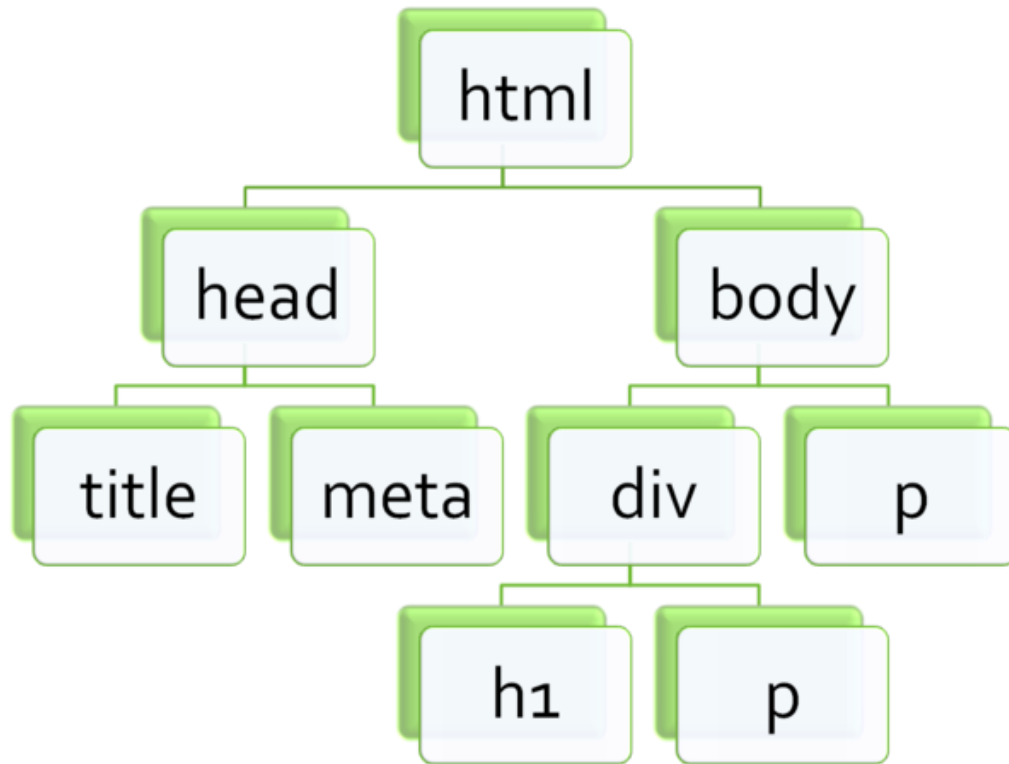


렌더링 엔진의 메인 플로우



DOM(브라우저 렌더링에 의해 트리형태로 구성된 파싱문서)

객체: 속성, 메소드
객체.속성
객체.메소드



자바스크립트
(DOM 트리 접근자)



front end 종류

EcmaScript version 5, 6

European Computer Manufacturers Association (ECMA)

웹표준 프론트엔드 언어

https://www.w3schools.com/js/js_es6.asp

jquery



- 2006년에 도입된 jQuery는 초기 프론트엔드 프레임워크
- 현재도 광범위하게 사용하는 인기있는 프레임워크
- jQuery는 간단하고 사용하기 쉬울 뿐만 아니라 광범위한 JavaScript 코드를 작성할 필요성도 최소화
- 수년 동안 존재한 덕분에 솔루션을 위한 상당한 jQuery 커뮤니티가 있음
- CSS와 DOM을 조작하고 웹 사이트의 기능과 상호 작용을 최적화하기 위해 사용
- 모든 브라우저를 지원

<https://www.w3schools.com/jquery/>

[cheatsheet https://oscarotero.com/jquery/](https://oscarotero.com/jquery/)

jquery chaining



jquery 장점 단점

장점:

DOM은 요소를 추가하거나 제거하기 위해 유연
HTTP 요청 전송이 단순화됨
동적 콘텐츠 촉진

단점:

상대적으로 느린 작업 능력
유지보수가 어렵다(대형프로젝트에 적합하지 않음)

사용 시기:

jQuery는 데스크톱 기반 javascript 애플리케이션을 개발하는 데 사용
코드를 간결하고 매우 단순하게 유지
이벤트 처리 및 애니메이션 수행에 사용.

사용하지 않을 경우:

대규모 애플리케이션을 개발할 때는 javascript 코드를 많이 추가하여 애플리케이션을 무겁게 하기 때문에 jQuery를 사용할 수 없다.
자바스크립트의 진보된 촉진, 낮은 코딩 및 구성요소의 재사용성을 가지고 현대의 프레임워크와 경쟁할 수 없다.

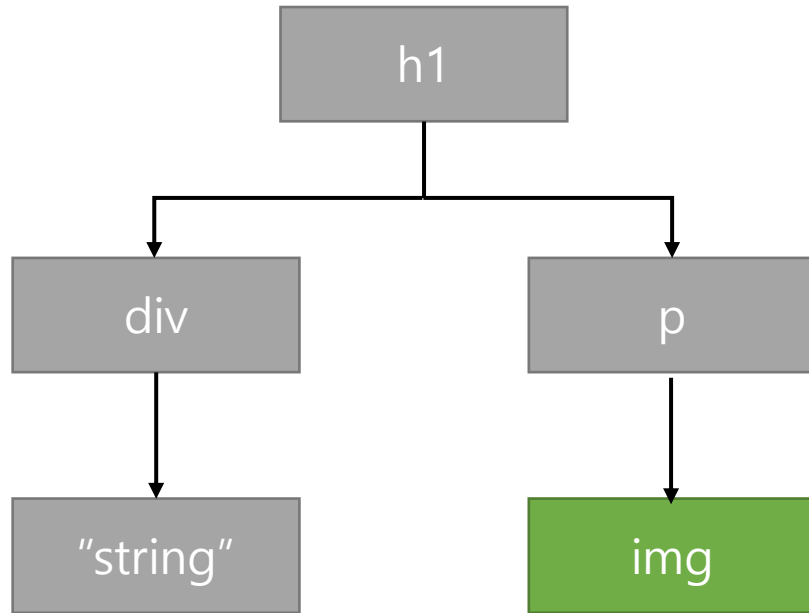
vue.js



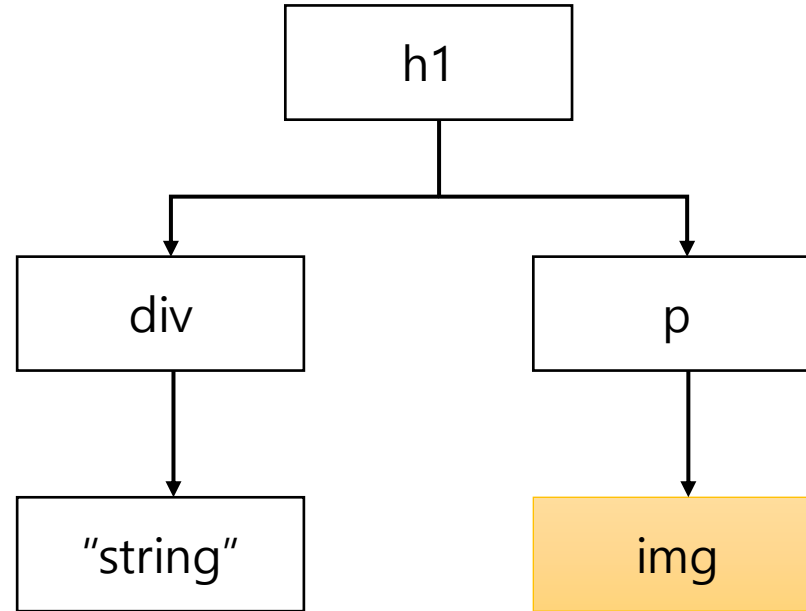
- 오늘날 가장 인기 있는 프론트 엔드 프레임워크 중 하나
- Vue는 간단한 프레임워크이다.
- 앵글러 개발자들이 겪는 복잡성을 없애는 데 좋다.
- 크기는 더 작으며 시각적 DOM과 구성요소 기반이라는 두 가지 주요 이점을 제공합니다. 또한
- 2방향 바인딩
- 웹 애플리케이션과 모바일 애플리케이션 구축부터 진보적인 웹 애플리케이션까지 간편하고 역동적인 프로세스를 모두 손쉽게 처리할 수 있다.
- 앱 성능을 최적화하고 복잡성을 타개하기 위해 구축
- 알리바바, 로이터, 샤오미 등은 이 프레임워크 사용중

https://www.w3schools.com/whatis/whatis_vue.asp

Virtual DOM



DOM



vue.js 장단점

장점:

- 광범위하고 상세한 설명서
- 간단한 구문 - 자바스크립트 배경을 가진 프로그래머들은 Vuejs로 쉽게 시작할 수 있다.
- 애플리케이션 구조 설계의 유연성

단점:

- 구성 요소의 안정성 결여
- community 가 아직까지 적다
- 플러그인 및 구성 요소를 포함한 언어 장벽(대부분의 플러그인은 중국어로 작성됨)

사용:

유연한 설계 구조에는 vue를 사용하는 것이 좋습니다.

모든 것을 처음부터 설계할 수 있게 해주고 대형 프로젝트 개발에도 좋음

사용하지 않을 경우:

복잡성에 대한 답을 얻기 위한 지원 커뮤니티가 부족하다

또한 안정적인 구성 요소가 필요한 애플리케이션은 프레임워크가 구성 요소의 안정성에 문제를 보여왔기 때문에 Vuejs로 구축하기에 적합하지 않다.

angular

- Angular는 2010년 Google에 의해 처음 개발된 TypeScript 기반의 JavaScript 프레임워크
- AngularJS는 자바스크립트로 만든 client 측 MVC/MVVM 프레임워크로 모던 단일 페이지 웹 애플리케이션 개발의 정수
- 자바스크립트로 작성할 코드량을 줄여준다.
- Dom을 선택하고 조작하는 자바스크립트 코드를 작성하지 않아도 된다.
- 양방향 데이터 바인
- 모델의 데이터와 뷰 데이터가 양방향 데이터 바인딩이 되어, 모델이 바뀌면 뷰 데이터도 같이 변경
- HTML, CSS, 로직 등의 개발 영역을 명확하게 분리해줍니다.
- 기존 자바스크립트에서는 Dom 조작과 이벤트 처리를 위해 HTML을 잘 알고 있어야 했으나, AngularJS는 뷰 코드와 로직 코드가 명확히 분리

참고:<https://www.w3schools.com/angular/>

참고: <https://ithub.tistory.com/68>

장점:

- 이원 데이터 바인딩과 같은 대부분의 주요 기능이 기본적으로 제공되므로 코드 양을 줄여줌
- 구성 요소를 외부 요소를 정의하여 종속성에서 분리합니다.
- 구성 요소를 재사용할 수 있으며 종속성 주입을 사용하여 관리하기 쉽습니다.
- 학습 및 지원을 위한 광범위한 커뮤니티

단점 :

- 앵글롤러는 완전한 동적 솔루션이기 때문에 작업을 수행할 수 있는 여러 가지 방법이 있으므로 학습이 쉽지 않다.
- 동적 앱은 구조와 크기가 복잡하기 때문에 성능이 좋지 않다.

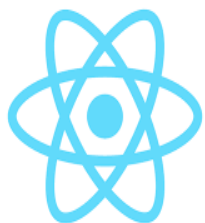
angular 사용:

양방향 데이터 바인딩을 사용하기 때문에 콘텐츠를 순식간에 동적으로 업데이트하여 브라우저 기반 애플리케이션의 성능을 각도로 향상시킵니다. 엔터프라이즈 기반 애플리케이션과 동적 웹 애플리케이션의 경우 angular 사용하는 것이 가장 좋습니다.

사용 안 할 경우

angular 는 프론트엔드 프레임워크로서의 완전한 솔루션이다. 제한된 범위의 애플리케이션을 빌드하려면 앵글러가 제공하는 리소스를 사용할 수 없다. 또한 더 작은 크기의 팀이 있는 경우 더 적은 복잡성과 간단한 구문으로 더 작은 프레임워크를 선택.

react



- 가장 간단한 프레임워크 중 하나인 React는 페이스북에서 앱의 기능이 지속적으로 추가되어 코드 유지관리성 문제를 해결하기 위해 개발
- 오픈 소스 프레임워크인 React는 뛰어난 기능을 제공하는 가상 문서 객체 모델(DOM)을 사용한다.
- 높은 트래픽이 예상되며 이를 처리하기 위한 안정적인 플랫폼이 필요한 사람에게 이상적인 프레임워크.
- 기술 컨설턴트로서, 단일 [페이지](#) 웹 [애플리케이션](#) 및 PWA [구축](#)과 관련된 프로젝트에 대해 React를 추천.

장점:

- 구성요소를 재사용할 수 있으므로 응용프로그램의 다른 부분에서 구성요소를 쉽게 협업하고 재사용할 수 있다.
- 가상 DOM 사용으로 일관되고 원활한 성능 제공
- React 후크에서 구성 요소를 쓰는 가장 좋은 대안으로서, 클래스 없이 구성 요소를 작성할 수 있으며 React를 보다 쉽게 학습할 수 있습니다.
- React Dev 도구는 고급이며 매우 유용.

단점:

- 프레임워크의 다중적이고 지속적인 업데이트로 인해 적절한 설명서를 만드는 것이 어렵고 초보자가 학습시 어렵다.
- JSX의 복잡성을 이해하기 어렵다.

React 사용 :

- React는 특히 단일 페이지 응용프로그램을 개발하려는 경우 사용자 인터페이스를 구축하는 데 사용됩니다.
- 구성요소를 재사용할 수 있으므로 시간을 단축하고 대화형 인터페이스를 개발하고자 할 때 가장 강력한 프론트엔드 프레임워크

React를 사용하지 않을 경우:

- Javascript에 대한 실제 경험이 없는 경우 React는 권장되는 옵션이 아닙니다.
- 또한 경험이 부족한 개발자들에게는 JSX 학습이 어렵다.

ember.js



- 2011년에 개발된 Eberjs는 구성요소 기반이며 앵글러와 유사한 이원 데이터 바인딩을 제공
- 현대 기술의 증가하는 수요를 원활하게 처리하도록 설계되됨.
- Eberjs를 사용하여 복잡한 모바일 및 웹 애플리케이션을 개발할 수 있으며, 효율적인 아키텍처로 문제를 해결할 수 있다.
- 학습이 어렵다.
- 개발자 커뮤니티는 작다.

장점:

- 잘 조직된
- 가장 빠른 프레임워크
- 이원 데이터 바인딩
- 적절한 문서

단점:

- 작은 커뮤니티, 덜 인기 있는
- 복잡한 구문 및 느린 업데이트
- 하드러닝곡선
- 소규모 애플리케이션을 위한 무거운 프레임워크

사용 시기:

풍부한 사용자 인터페이스를 갖춘 Linkedin과 같은 현대적인 애플리케이션을 구축하고자 한다면, Eberjs는 Eberjs가 제공하는 뛰어난 라우팅 덕분에 다양한 애플리케이션 상태를 보는 것과 같은 모든 기술적 프론트엔드 축진을 갖춘 프레임워크입니다.

이 프레임워크는 필요에 따라 페이지를 렌더링할 수 있는 준비된 구성, 유용한 바인딩 및 사용자 지정 속성을 제공하기 때문에 대규모 프로젝트를 위한 완벽한 프론트엔드 솔루션입니다.

사용하지 않을 경우:

엠버즈는 복잡한 문제를 해결하기 위해서는 경험과 비즈니스 논리가 필요하기 때문에 소규모 개발 팀에서는 올바른 선택이 아닙니다. 초기 비용은 엠버지가 더 높을 수 있다. 또한 간단한 ajax 기능을 작성하고 간단한 사용자 인터페이스를 구현하는 경우 프레임워크가 올바른 선택이 아닐 수 있습니다.

backbone.js



BACKBONE.JS

- 가장 쉬운 프레임워크 중 하나인 backbone.js를 사용하면 단일 페이지 애플리케이션을 신속하게 개발할 수 있다.
- MVC 아키텍처를 기반으로 하는 프레임워크이다.
- 컨트롤러와 비슷하게 MVC 아키텍처의 뷰는 요소 로직을 구현할 수 있다.

장점 :

- 가장 빠른 자바스크립트 프레임워크 중 하나
- 배우기 쉽다
- 경량 프레임워크

단점:

- 앱 구조를 만들기 위한 기본 도구 제공(프레임이 준비된 구조를 제공하지 않음)
- 뷰를 모델 및 모델로 전달하기 위해 보일러 플레이트 코드를 작성해야 합니다.

사용 시기:

백본js는 Trello와 같은 동적 용도에 사용됩니다. 이를 통해 개발자는 클라이언트 측 모델을 구축하고, 업데이트를 더 빠르게 하며, 코드를 재사용할 수 있습니다. 따라서 업데이트를 동적으로 처리하고 클라이언트를 유지하며 서버와 지속적으로 동기화하는 데 효율적입니다.

사용하지 않을 경우:

backbone.js는 다른 MVC 클라이언트 측 프레임워크에 비해 웹 애플리케이션을 구축하기 위한 최소한의 요구 사항을 제공합니다. 그러나 플러그인 및 확장을 지원하여 기능을 확장할 수 있습니다. 따라서 개발 팀이 하나의 프레임워크에서 완전한 솔루션을 목표로 하는 상황에서 backbone.js를 목표로 해서는 안 된다

frontend UI

jquery ui: <https://jqueryui.com/>

jquery mobile: <https://jquerymobile.com/>

bootstrap: <https://getbootstrap.com/>
<https://www.w3schools.com/bootstrap4/>

semantic ui:<https://semantic-ui.com/>

JSP(Java Server Page)

목차

- page 디렉티브(Directive) - <%@ page%>
- include 디렉티브(Directive) - <%@ include%>
- taglib 디렉티브 - <%@ taglib%>

page 디렉티브(Directive)

- `<%@ page%>`
- JSP 페이지에 대한 정보는 page 디렉티브(Directive)의 속성들을 사용해서 정의
- 생성되는 문서의 타입, 스크립팅언어, import할 클래스, 세션 및 버퍼의 사용여부, 버퍼의 크기 등 JSP페이지에서 필요한 설정 정보를 지정.

page 디렉티브(Directive)

- page 디렉티브(Directive)의 속성

속성명	속성의 기본값	사용법	속성 설명
info		info="설명... "	페이지를 설명해 주는 문자열을 지정하는 속성
language	"java"	language="java"	JSP 페이지의 스크립트 요소에서 사용할 언어를 지정하는 속성
contentType	"text/html; charset=ISO-8859-1"	contentType="text/html; charset=utf-8"	JSP 페이지가 생성할 문서의 타입을 지정하는 속성
extends		extends="system.MasterClass"	자신이 상속 받을 클래스를 지정할 때 사용하는 속성
import		import="java.util.Vector" import="java.util.*"	다른 패키지에 있는 클래스를 가져다 쓸 때 사용하는 속성

page 디렉티브(Directive)

- page 디렉티브(Directive)의 속성

속성명	속성의 기본값	사용법	속성 설명
session	"true"	session="true"	HttpSession을 사용할지 여부를 지정하는 속성
buffer	"8kb"	buffer="10kb" buffer="none"	JSP 페이지의 출력버퍼의 크기를 지정하는 속성
autoFlush	"true"	autoFlush="false"	출력버퍼가 다 찰 경우에 저장되어 있는 내용의 처리를 설정 하는 속성
isThreadSafe	"true"	isThreadSafe="true"	현재 페이지에 다중쓰레드를 허용할지 여부를 설정하는 속성

page 디렉티브(Directive)

- page 디렉티브(Directive)의 속성

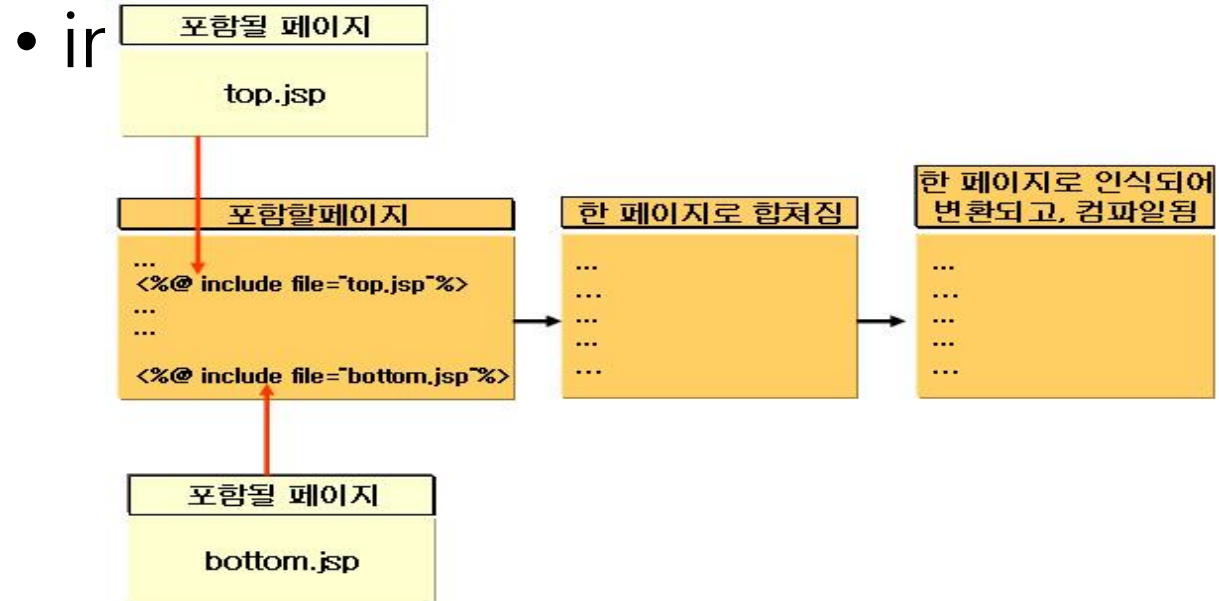
속성명	속성의 기본값	사용법	속성 설명
errorPage		errorPage="error/fail.jsp"	에러발생시 에러를 처리할 페이지를 지정하는 속성
isErrorPage	"false"	isErrorPage="false"	해당페이지를 에러페이지로 지정하는 속성
pageEncoding	"ISO-8859-1"	pageEncoding="utf-8"	해당페이지의 문자 인코딩을 지정하는 속성
isELIgnored	jsp버전 및 설정에 따라 다르다.	isELIgnored="true"	표현 언어(EL)에 대한 지원여부를 설정하는 속성

include 디렉티브(Directive)

- <%@ include %>
- JSP페이지에서는 여러 JSP 페이지에서 공통적으로 사용되는 내용이 있을 때, 이러한 내용을 별도의 파일로 저장해 두었다가 필요한 JSP 페이지 내에 삽입할 수 있는 기능을 제공
- 공통적으로 포함될 내용을 가진 파일을 해당 JSP 페이지 내에 삽입하는 기능을 제공하는 것이 include 디렉티브

include 디렉티브(Directive)

- include 디렉티브는 <%@ include로 시작되며, 포함시킬 파일명을 file속성의 값으로 기술.
 - <%@ include file="포함될 파일의 url"%>



include 디렉티브(Directive)

- include 디렉티브의 처리 과정
 - include 디렉티브를 사용한 JSP 페이지가 컴파일 되는 과정에서 include 되는 JSP페이지의 소스 내용을 그대로 포함해서 컴파일
 - 복사 & 붙여넣기 방식으로 두 개의 파일이 하나의 파일로 합쳐진 후 하나의 파일로서 변환되고 컴파일.
- include 디렉티브는 주로 조각 코드를 삽입할 때 사용.
 - 조각코드를 가지는 페이지의 내용은 어떤 값을 가지는 변수를 정의하고 있는 경우에 주로 사용.

taglib 디렉티브 (Directive)

- <%@ taglib%>
- taglib 디렉티브는 표현 언어(EL :Expression Language), JSTL(JSP Standard Tag Library), 커스텀 태그(Custom Tag)를 JSP페이지 내에 사용할 때 사용됨.

taglib 디렉티브 (Directive)

- 사용방법

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

--중략--

```
<c:set var="aInt" value="123"%>
```

- prefix속성 : 별명과 같은 역할, prefix속성의 값을 사용하면 uri속성의 값을 사용하는 것과 같음.
- uri속성 : 사용자가 정의한 어떤 태그의 설정 정보를 가짐.

스크립트 요소의 이해

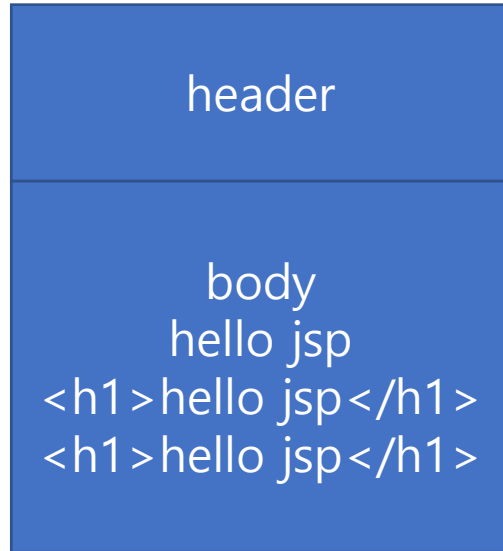
- JSP페이지에서는 선언문(Declaration), 스크립트릿(Scriptlet), 표현식(Expression) 이라는 3가지의 스크립트 요소를 제공.

선언문(Declaration) - `<%! %>` : 전역변수 선언 및 메소드 선언에 사용

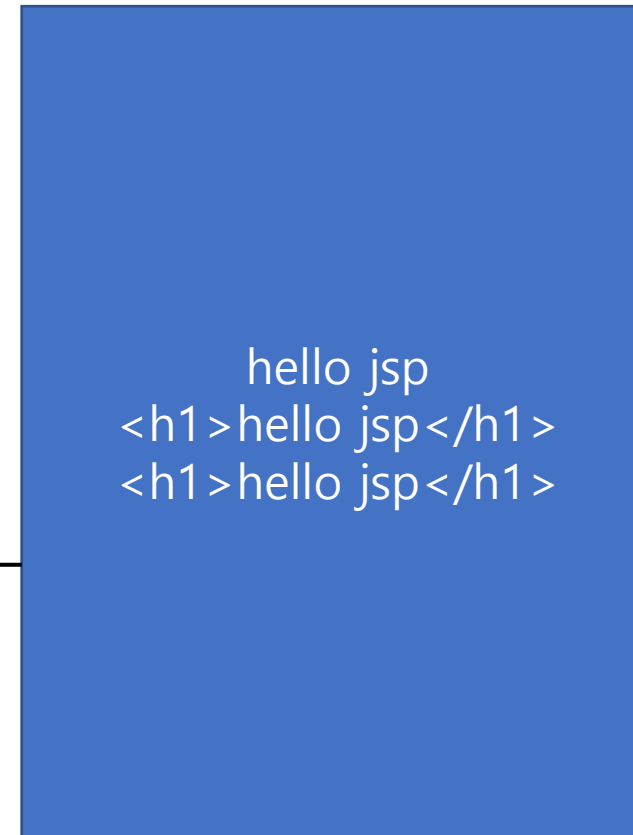
스크립트릿(Scriptlet) - `<% %>` : 프로그래밍 코드 기술에 사용

표현식(Expression) - `<%= %>` : 화면에 출력할 내용 기술에 사용

http response



html make buffer



write



선언문(Declaration)

- 선언문:<%! %>
- 선언문은 JSP 페이지 내에서 필요한 멤버변수나 메소드가 필요할 때 선언해 사용하는 요소
- 선언문의 문법
 - <%! 문장 %>

선언문(Declaration)

- 선언문에서 변수 및 메소드 선언

<%!

String id = "Kingdora"; //멤버변수 선언

public String getId() { //메소드 선언

return id;

}

%>

스크립트릿(Scriptlet)

- 스크립트릿 :<% %>
- 가장 일반적으로 많이 쓰이는 스크립트 요소
- 주로 프로그래밍의 로직을 기술할 때 사용.
- 스크립트릿에서 선언된 변수는 지역변수
- 스크립트릿의 문법
 - <% 문장%>

표현식(Expression)

- 표현식 : `<%= %>`
- JSP 페이지에서 웹 브라우저에 출력할 부분을 표현. 즉, 화면에 출력하기 위한 것.
- 스크립트릿 내에서 출력할 부분은 내장객체인 out객체의 `print()` 또는 `println()` 메소드를 사용해서 출력.
- 표현식의 문법
 - `<%=문장%>`

주석(Comment)

- JSP페이지에서 사용할 수 있는 주석
 - HTML주석, 자바주석, JSP주석
- HTML 주석
 - HTML 주석은 <!--로 시작해서 -->로 끝나는 형태
 - HTML 주석은 HTML주석을 사용한 페이지를 웹에서 서비스할 때 화면에 주석이 내용이 표시되지는 않으나 , [소스보기]수행하면 HTML주석의 내용이 화면에 표시.
 - HTML주석의 예시
<!-- html 주석입니다. -->

주석(Comment)

- JSP주석
 - JSP 페이지에서만 사용되며 <%--로 시작해서 --%>로 끝나는 형태
 - JSP 주석은 해당 페이지를, 웹 브라우저를 통해 출력 결과로서 표시하거나, 웹 브라우저 상에서 소스 보기를 해도 표시 되지 않음. 또한 JSP 주석 내에 실행코드를 넣어도 그 코드는 실행되지 않음.
 - JSP주석의 예시
 - %-- JSP 주석입니다. --%>

주석(Comment)

- 자바주석

- 자바 주석은 `//`, `/**`을 사용해서 작성.
- `//`은 한 줄짜리 주석을 작성할 때 사용되고, `/**`은 여러 줄의 주석을 작성할 때 사용
- 스크립트릿이나 선언문에서 사용되는 주석으로, 자바와 주석 처리 방법이 같음

- 자바주석의 예시

`//주석`

`/*주석`

여러 줄에 걸친 주석이다.

`*/`

JSP페이지의 모듈화

- include 액션태그(<jsp:include>)의 사용법

- <jsp:include page="포함될 페이지" flush="true"/>

- page속성: 현재 페이지에 결과가 포함될 페이지명

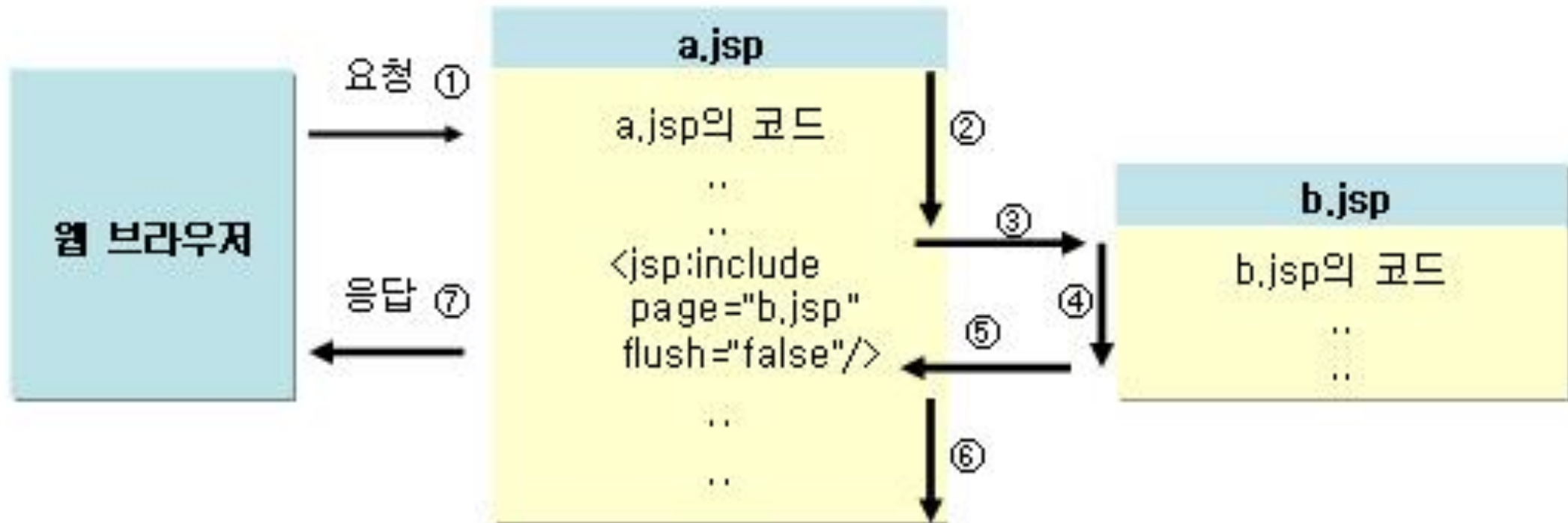
- flush 속성: 포함될 페이지로 제어가 이동될 때, 현재 포함하는 페이지가 지금까지 출력 버퍼에 저장한 결과를 처리하는 방법을 결정

- include 액션 태그의 권장 형태

- <jsp:include page="포함될 페이지" flush="false"/>

JSP페이지의 모듈화

- include 액션태그의 처리 과정



JSP페이지의 모듈화

- ① 웹 브라우저가 a.jsp페이지를 웹 서버에 요청.
- ② 서버는 요청받은 a.jsp페이지를 처리하는데, a.jsp페이지 내에서 출력 내용은 출력버퍼에 저장하는 등의 작업을 처리.
- ③ 이때 `<jsp:include page="b.jsp" flush="false"/>`문장을 만나면 하던 작업을 멈추고 프로그램제어를 b.jsp페이지로 이동.
- ④ b.jsp페이지를 처리한다. b.jsp페이지 내에 출력내용은 출력버퍼에 저장하는 등의 작업을 처리.
- ⑤ b.jsp페이지를 처리가 끝나면, 다시 a.jsp페이지로 프로그램의 제어가 이동하는데, 이동위치는 `<jsp:include page="b.jsp" flush="false"/>`문장 다음 행이 됨.
- ⑥ a.jsp페이지의 나머지 부분을 처리한다. 출력할 내용이 있으면 출력버퍼에 저장.
- ⑦ 출력버퍼의 내용을 웹 브라우저로 응답.

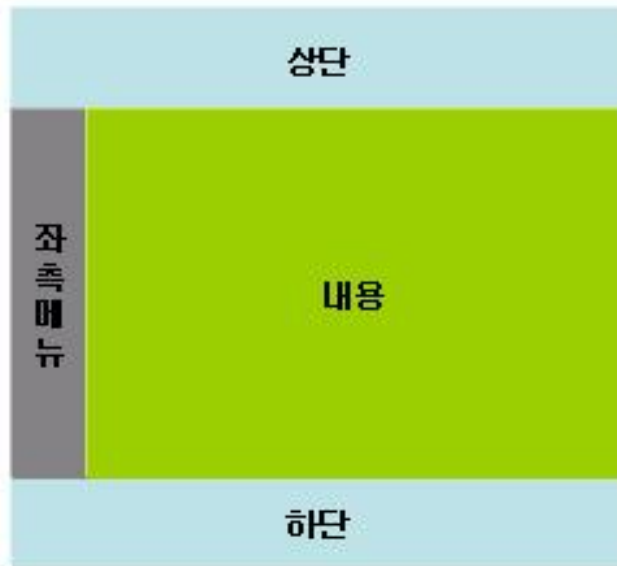
JSP페이지의 모듈화

- include 액션 태그에서 포함되는 페이지에 값 전달하기
 - 포함되는 JSP 페이지에 값 전달은 요청 파라미터를 추가적으로 지정해서 사용.
 - include액션 태그의 바디(body) 안에 param 액션 태그(<jsp:param>)를 사용.
 - name속성: JSP 페이지에 전달할 파라미터의 이름
 - value 속성: 전달할 파라미터의 값

```
<jsp:include page="b.jsp" flush="false">  
  <jsp:param name="p1" value="<%=var%>"/>  
</jsp:include>
```

JSP페이지의 모듈화

- JSP 페이지의 중복 영역 처리
 - 중복되는 페이지의 호출은 include액션 태그



상단 : 로고 포함한 메뉴

좌측 메뉴 : 하위 메뉴 포함

중앙 : 내용

하단 : 회사소개, 찾아오는 길, 보안 정책 등의 내용을 포함

주로 중앙의 내용부분의 내용만 계속 바뀌게 되는 같은 구조를 계속 유지

JSP페이지의 모듈화

- 각각 상단, 좌측, 하단은 같은 페이지를 유지하고 중앙의 내용만 바뀌는 이것은 <jsp:include>액션 태그를 사용

```
<TABLE>
<TR>
<TD colspan="2"><jsp:include page="top.jsp" flush="false"/></TD>
</TR>
<TR>
<TD><jsp:include page="left.jsp" flush="false"/></TD>
<TD><jsp:include page="<%=content%>" flush="false"/></TD>
</TR>
<TR>
<TD colspan="2"><jsp:include page="bottom.jsp" flush="false"/></TD>
</TR>
</TABLE>
```

JSP페이지의 흐름제어

- forward 액션태그(<jsp:forward>) 는 다른 페이지로 프로그램의 제어를 이동할 때 사용.
- JSP 페이지 내에 forward 액션태그를 만나게 되면, 그전까지 출력버퍼에 저장되어 있던 내용을 제거하고, forward 액션태그가 지정하는 페이지로 이동.
- 사용자가 입력한 값에 따라 여러 페이지로 이동해야 할 경우에 사용하면 좋음.
 - forward액션태그를 잘 이해하면 모델2(Model2)에서 컨트롤러에 대한 이해가 쉬움.

JSP페이지의 흐름제어

- forward 액션태그의 사용법

- <jsp:forward page="이동할 페이지명"/>

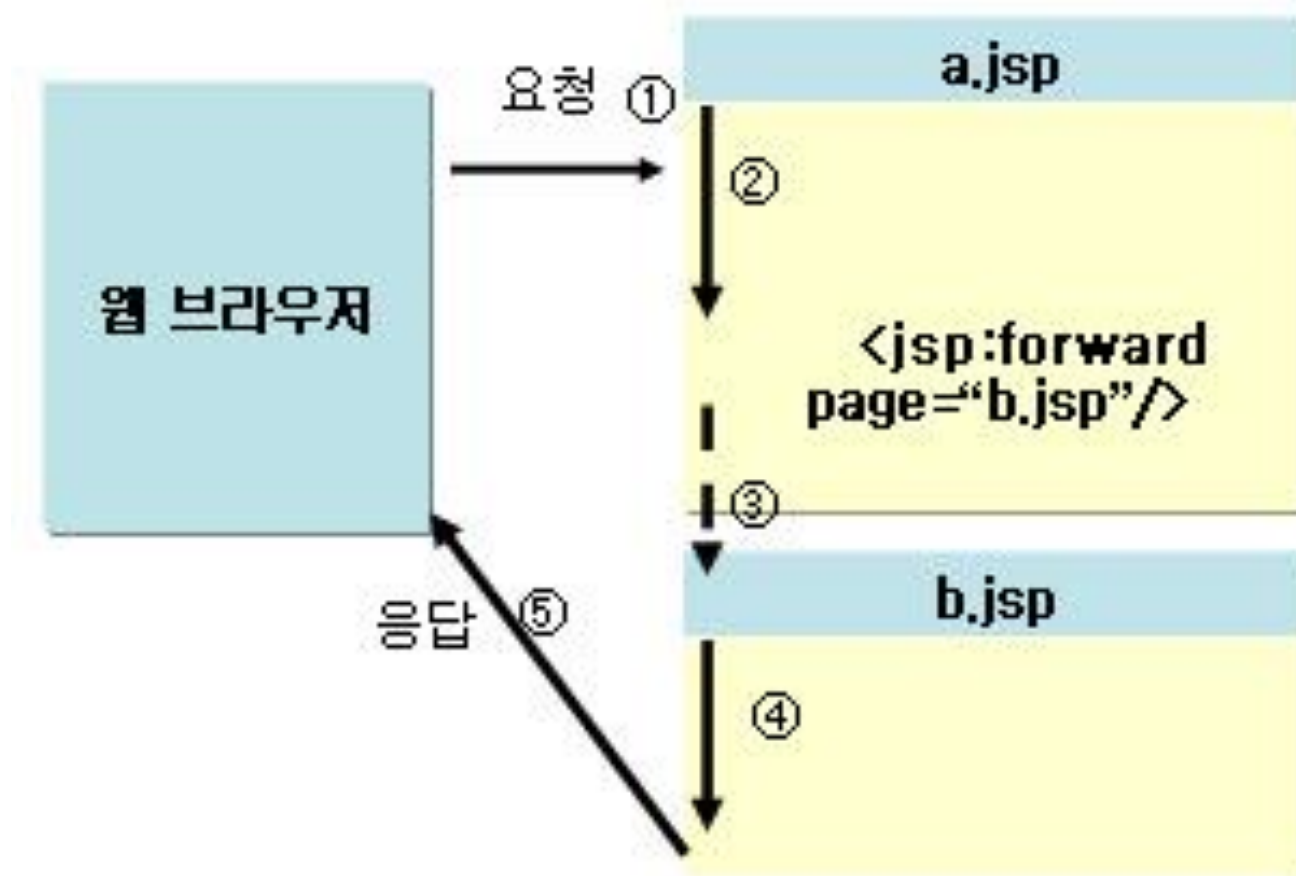
- <jsp:forward page="이동할 페이지명"></jsp:forward>

- <jsp:forward page='<%=expression + ".jsp"%>'/>

- page속성: 이동할 페이지 명을 기술

JSP페이지의 흐름제어

- forward액션태그의 처리과정



JSP페이지의 흐름제어

- ① 웹 브라우저에서 웹 서버로 a.jsp페이지를 요청.
- ② 요청된 a.jsp페이지를 수행.
- ③ a.jsp페이지를 수행하다가 <jsp:forward>액션 태그를 만나면 이제까지 저장되어있는 출력버퍼의 내용을 제거하고 프로그램제어를 page속성에서 지정한 b.jsp로 이동.
- ④ b.jsp페이지를 수행.
- ⑤ b.jsp페이지를 수행한 결과를 웹 브라우저에게 응답.

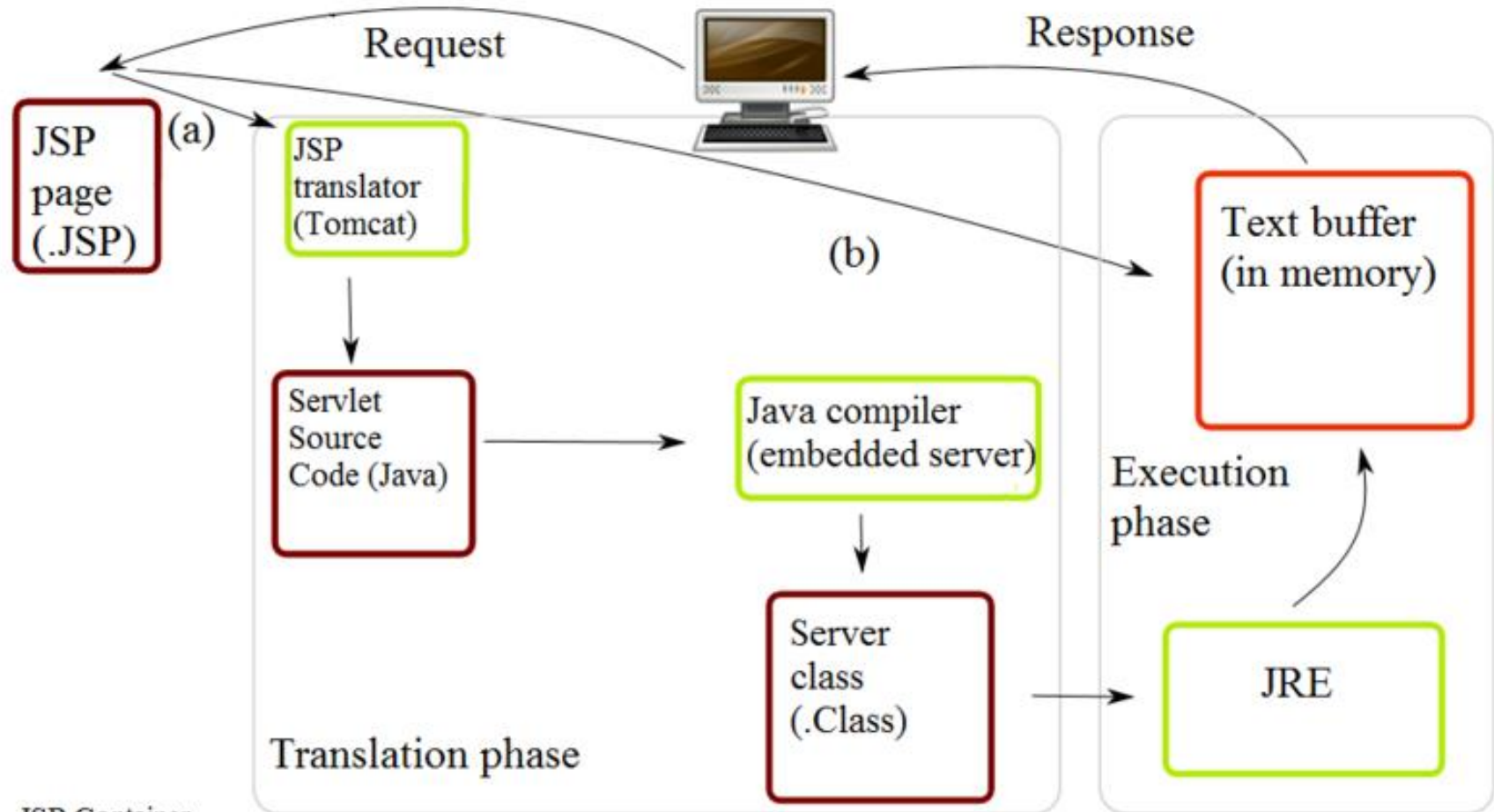
JSP페이지의 흐름제어

- forward 액션 태그에서 포워딩되는 페이지에 값 전달하기

```
<jsp:forward page="이동할 페이지명">  
  <jsp:param name="paramName1" value="var1"/>  
  <jsp:param name="paramName2" value="var2"/>  
</jsp:forward>
```

서블릿

자바 서블릿(Java Servlet)은 [자바](#)를 사용하여 웹페이지를 동적으로 생성하는 서버측 프로그램 혹은 그 사양을 말하며, 흔히 "서블릿"이라 불린다. 자바 서블릿은 웹 서버의 성능을 향상하기 위해 사용되는 자바 클래스의 일종이다. 서블릿은 JSP와 비슷한 점이 있지만, [JSP](#)가 HTML 문서 안에 Java 코드를 포함하고 있는 반면, 서블릿은 자바 코드 안에 HTML을 포함하고 있다는 차이점이 있다.



JSP Container

(a) Translation occurs at this point, if JSP has been changed or is new.

(b) If not, translation is skipped.

a tag 속성

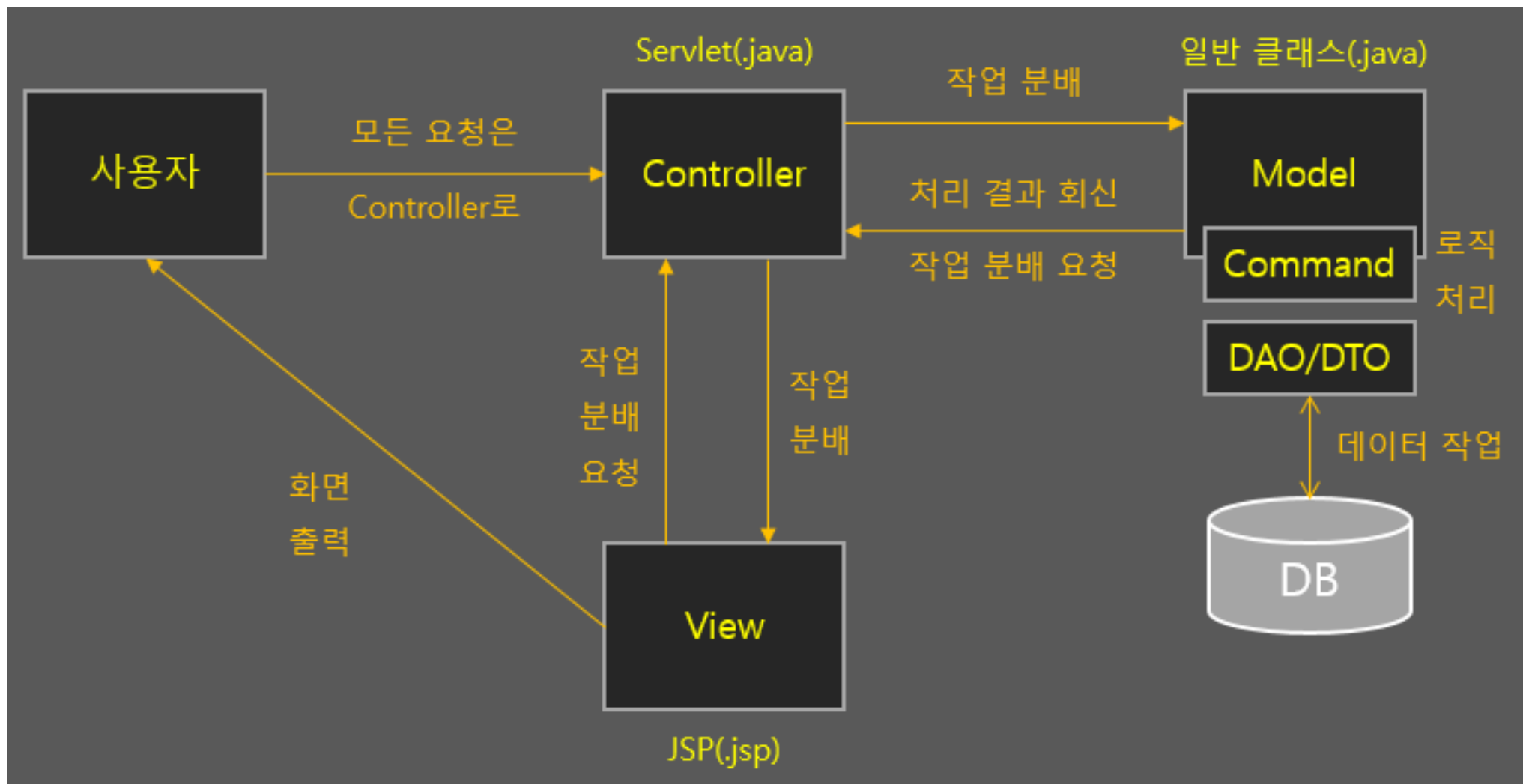
_self : 링크를 클릭한 해당 창에서 연다.

_blank : 링크를 새창으로 연다.

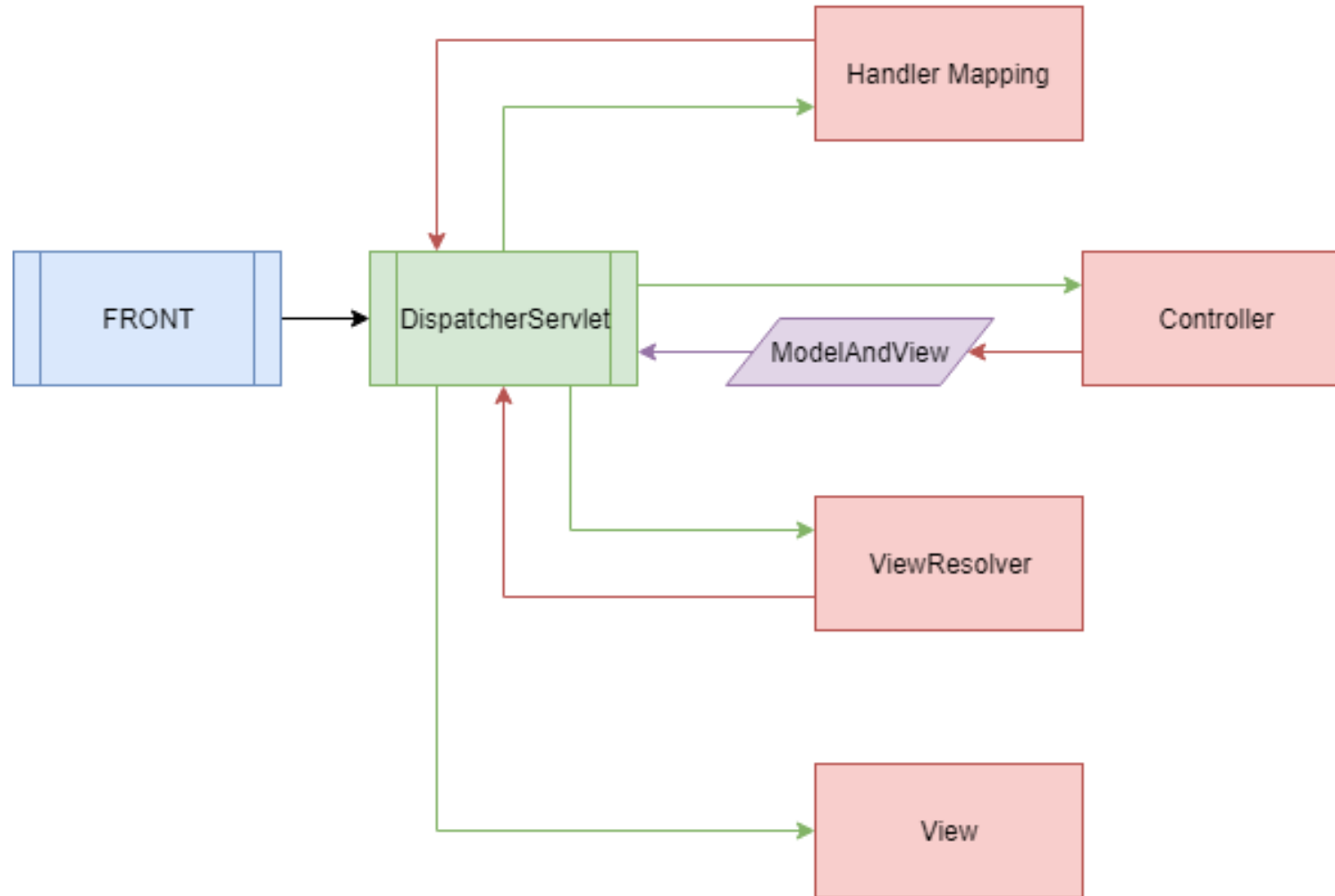
_parent : 부모 창에서 연다. (부모 창이 없으면 _self 속성으로 처리)

_top : 전체 브라우저 창에서 가장 상위의 창에서 연다. (부모 창이 없으면 _self 속성으로 처리)

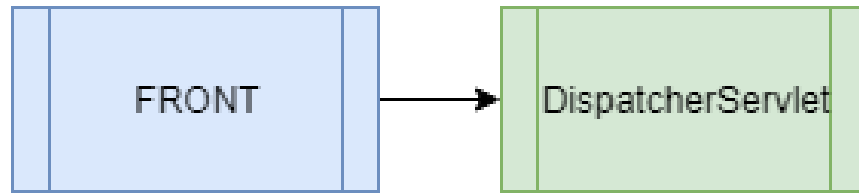
MVC 구조



Spring Mvc 구조

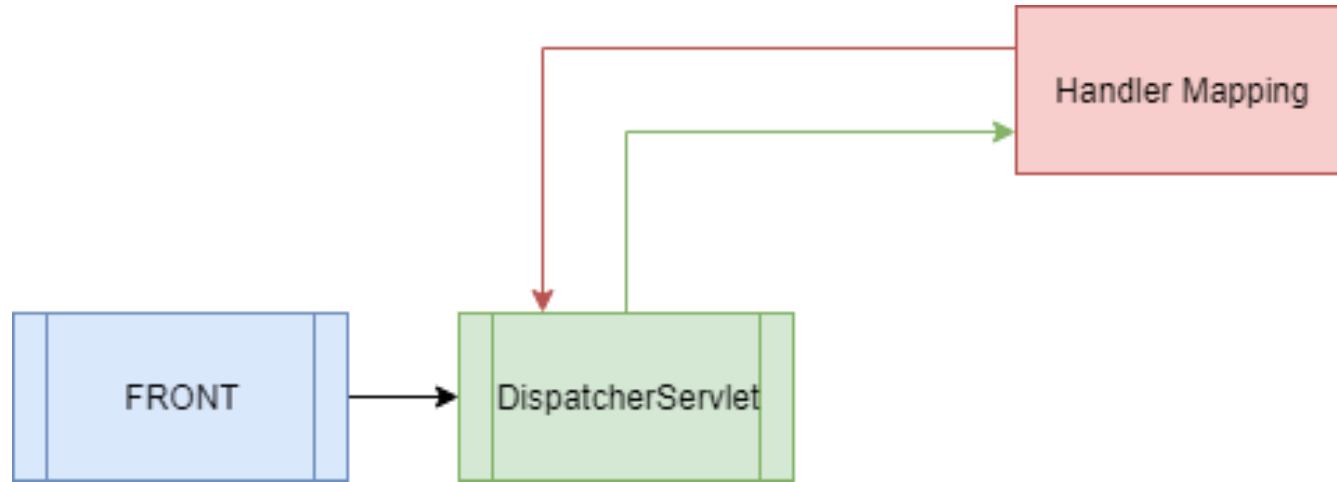


1. DispatcherServlet 역할(MVC의 핵심 요소)



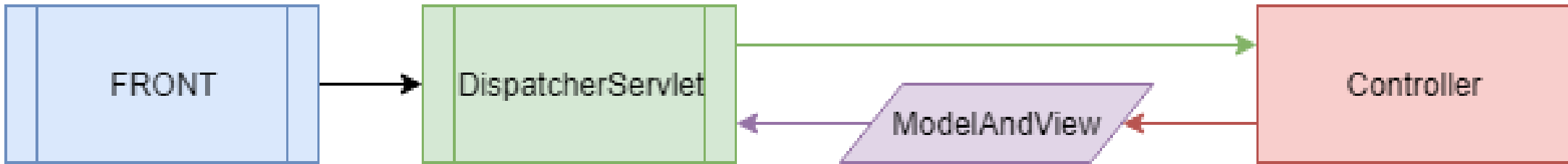
1. Front 가 URL로 호출
2. DispatcherServlet이 클라이언트 요청을 최초로 받음.
(중앙집중식 프론트 컨트롤러)

2. Handler Mapping 역할



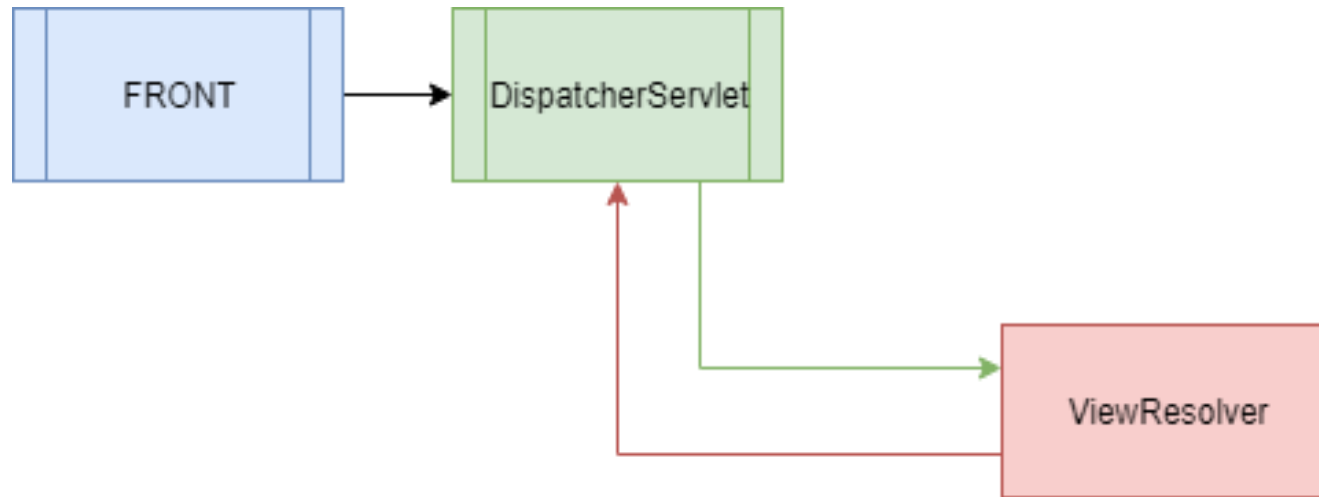
Handler Mapping은 DispatcherServlet에게 해당 request가 어떤 컨트롤러를 사용하는지 응답

3. Controller



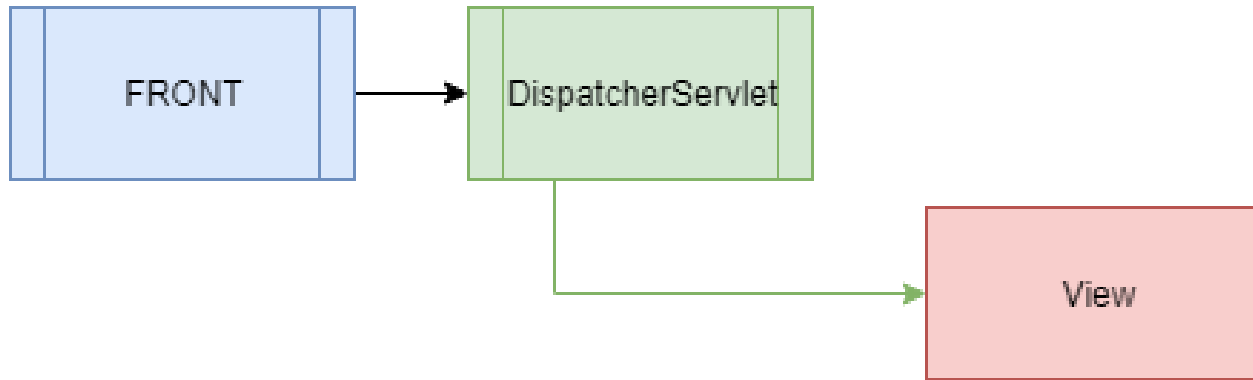
DispatcherServlet은 로직을 처리할 컨트롤러를 응답 받았으니 해당 Controller로 처리 요청을 넘김.
해당 컨트롤러는 로직을 수행하고 서비스를 요청하며 Model을 만듭니다.
그리고 ModelAndView 객체에 담아 다시 DispatcherServlet으로 넘겨줍니다.

4. ViewResolver



DispatcherServlet는 ViewResolver로 해당 ModelAndView에 맞는 view를 찾아달라고 요청.
ViewResolver는 해당 view를 응답

5. View



ViewResolver으로 부터 받은 view 정보로 DispatcherServlet 필요한 view를 찾아
Model 값을 넘겨 view를 출력

Spring MVC의 처리 순서

- 1.클라이언트가 서버에 요청을 하면, front controller인 DispatcherServlet 클래스가 요청을 받는다.
- 2.DispatcherServlet는 프로젝트 파일 내의 servlet-context.xml 파일의 @Controller 인자를 통해 등록된 요청 위임 컨트롤러를 찾아 매핑(mapping)된 컨트롤러가 존재하면 @RequestMapping을 통해 요청을 처리할 메소드로 이동한다.
- 3.컨트롤러는 해당 요청을 처리할 Service(서비스)를 받아 비즈니스로직을 서비스에게 위임한다.
- 4.Service(서비스)는 요청에 필요한 작업을 수행하고, 요청에 대해 DB에 접근해야한다면 DAO에 요청하여 처리를 위임한다.
- 5.DAO는 DB정보를 DTO를 통해 받아 서비스에게 전달한다.
- 6.서비스는 전달받은 데이터를 컨트롤러에게 전달한다.
- 7.컨트롤러는 Model(모델) 객체에게 요청에 맞는 View(뷰) 정보를 담아 DispatcherServlet에게 전송한다.
- 8.DispatcherServlet는 ViewResolver에게 전달받은 View정보를 전달한다.
- 9.ViewResolver는 응답할 View에 대한 JSP를 찾아 DispatcherServlet에게 전달한다.
- 10.DispatcherServlet는 응답할 뷰의 Render를 지시하고 뷰는 로직을 처리한다.
- 11.DispatcherServlet는 클라이언트에게 Rendering된 뷰를 응답하며 요청을 마친다.

DAO(Data Access Object)는 DB를 사용해 데이터를 조회하거나 조작하는 기능을 전담하도록 만든 오브젝트

DTO(Data Transfer Object)는 VO(Value Object)로 바꿔 말할 수 있는데
계층간 데이터 교환을 위한 오브젝트

나이:

삭제

삭제되었습니다.

```
delete from student where age=20;
```

숫자1:

숫자2:

계산

합:

Spring 기반으로
작성하시요.

의존성 주입(Dependency Injection)

- 의존성 주입은 Spring에서 아주 중요한 개념이자 장점이다.
- Bean 객체를 생성할 때 Bean 객체가 관리할 값이나 객체를 주입하는 것을 의미한다.
- Bean 객체 생성 후 Bean 객체가 가질 기본 값을 자바 코드로 설정하는 것이 아닌 Bean을 정의하는 xml 코드에서 정의하는 개념이다.



Spring Framework

Bean 생성하기

Spring Bean 객체 생성

- Spring에서는 사용할 Bean 객체를 bean configuration file에 정의를 하고 필요할 때 객체를 가져와 사용하는 방법을 이용한다.
- bean 태그 : 사용할 Bean을 정의하는 태그

bean 태그의 기본 속성

- class : 객체를 생성하기 위해 사용할 클래스를 지정한다.
- id : Bean 객체를 가져오기 위해 사용하는 이름을 지정한다.
- lazy-init : 싱글톤인 경우 xml을 로딩할 때 객체 생성 여부를 설정한다.
true : xml 로딩 시 객체를 생성하지 않고 객체를 가져올 때 생성한다.
- scope : 객체의 범위를 설정한다.
singleton : 객체를 하나만 생성해서 사용한다.
prototype : 객체를 가져올 때 마다 객체를 생성한다.

scope

Scope	Description
singleton	하나의 Bean 정의에 대해서 Spring IoC Container 내에 단 하나의 객체만 존재한다.
prototype	하나의 Bean 정의에 대해서 다수의 객체가 존재할 수 있다.
request	하나의 Bean 정의에 대해서 하나의 HTTP request의 생명주기 안에 단 하나의 객체만 존재한다; 즉, 각각의 HTTP request는 자신만의 객체를 가진다. Web-aware Spring ApplicationContext 안에서만 유효하다.
session	하나의 Bean 정의에 대해서 하나의 HTTP Session의 생명주기 안에 단 하나의 객체만 존재한다. Web-aware Spring ApplicationContext 안에서만 유효하다.
global session	하나의 Bean 정의에 대해서 하나의 global HTTP Session의 생명주기 안에 단 하나의 객체만 존재한다. 일반적으로 portlet context 안에서 유효하다. Web-aware Spring ApplicationContext 안에서만 유효하다.

학습정리

- Spring에서는 프로그램에서 사용할 객체를 bean configuration 파일에 정의하여 사용한다.

BeanPostProcessor

- Bean 객체를 정의할 때 init-method 속성을 설정하면 객체가 생성될 때 자동으로 호출될 메서드를 지정할 수 있다.
- 이 때 BeanPostProcessor 인터페이스를 구현한 클래스를 정의하면 Bean 객체를 생성할 때 호출될 init 메서드 호출을 가로채 다른 메서드를 호출 수 있도록 할 수 있다.

BeanPostProcessor

- `postProcessBeforeInitialization` : `init-method`에 지정된 메서드가 호출되기 전에 호출된다.
- `postProcessAfterInitialization` : `init-method`에 지정된 메서드가 호출된 후에 호출된다.
- `init-method` 가 지정되어 있지 않더라도 자동으로 호출된다.

학습정리

- Spring에서는 객체가 생성될 때 init-method로 지정된 메서드가 호출되기 전, 후에 다른 메서드를 호출할 수 있도록 지원하고 있다.

의존성 주입(Dependency Injection)

- 의존성 주입은 Spring에서 아주 중요한 개념이자 장점이다.
- Bean 객체를 생성할 때 Bean 객체가 관리할 값이나 객체를 주입하는 것을 의미한다.
- Bean 객체 생성 후 Bean 객체가 가질 기본 값을 자바 코드로 설정하는 것이 아닌 Bean을 정의하는 xml 코드에서 정의하는 개념이다.

생성자를 통한 주입

- Bean을 정의할 때 constructor-arg 태그를 이용해 주입하게 되면 생성자를 통해 주입할 수 있다.

```
<bean id="t7" class="kr.co.softcampus.beans.TestBean2">  
<constructor-arg value="100" type="int" index="2"/>  
<constructor-arg value="200" type="int" index="1"/>  
<constructor-arg value="300" type="int" index="0"/>  
</bean>
```

```
<bean id="data" class="kr.co.softcampus.beans.DataClass" scope="prototype"/>
```

```
<bean id="t8" class="kr.co.softcampus.beans.TestBean3">  
<constructor-arg ref="data"/>  
<constructor-arg ref="data"/>  
</bean>
```

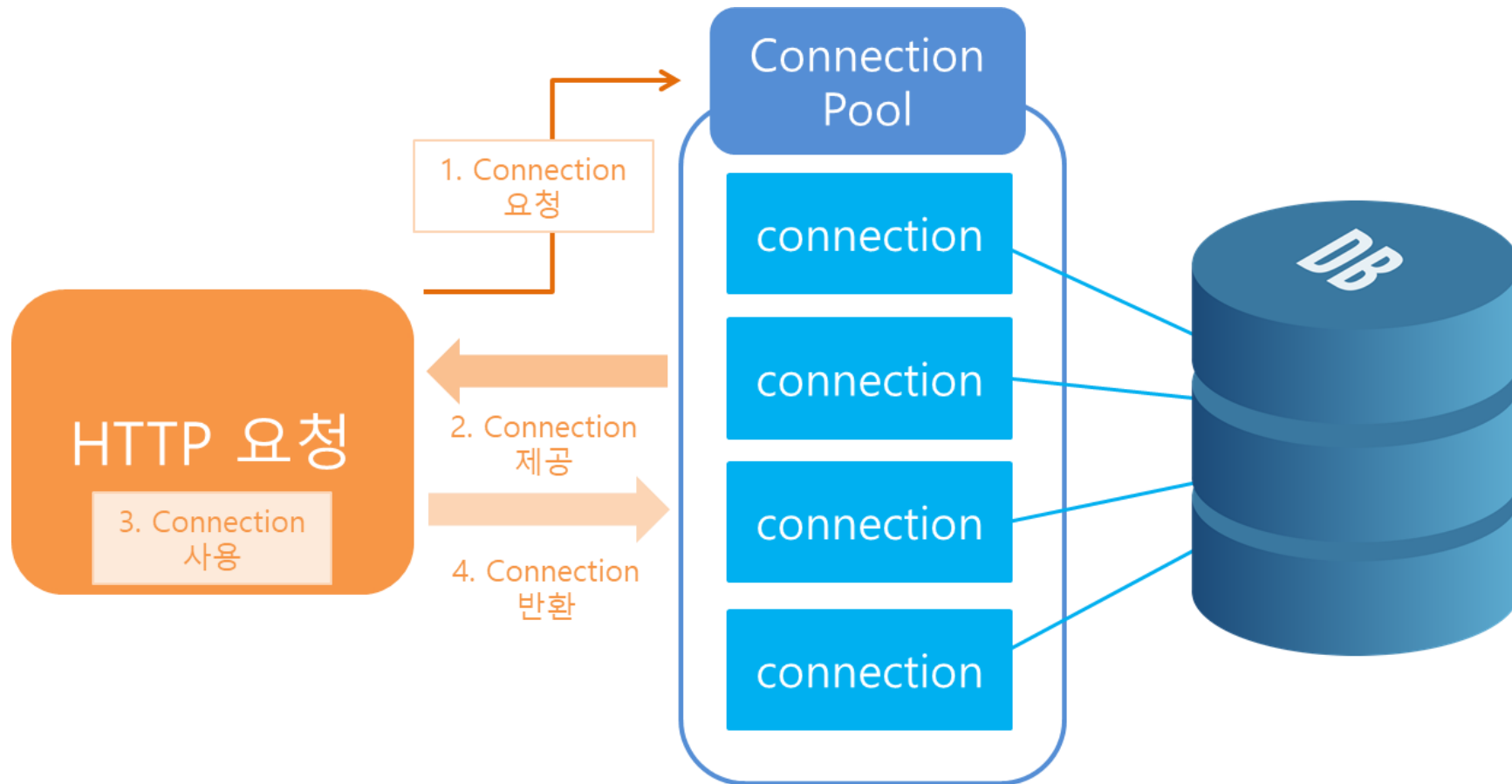
생성자를 통한 주입

- value : 기본 자료형 값과 문자열 값을 설정한다.
- ref : 객체를 설정한다.
- type : 저장할 값의 타입을 설정한다.
- index : 지정된 값을 주입할 생성자의 매개변수 인덱스 번호

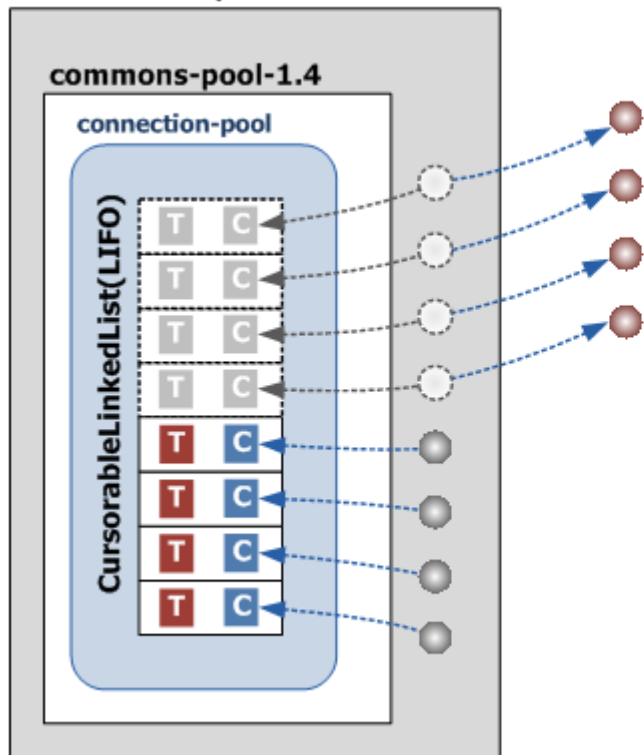
학습정리

- Bean 객체를 생성할 때 객체의 기본 값을 생성자를 통해 주입할 수 있다.

웹 컨테이너(WAS)가 실행되면서 connection 객체를 미리 pool에 생성
HTTP 요청에 따라 pool에서 connection객체를 가져다 쓰고 반환한다.
이와 같은 방식으로 물리적인 데이터베이스 connection(연결) 부하를 줄이고
연결 관리 한다.



commons-dbcp-1.2.2



maxActive

Default = 8, non-positive = no limit

maxIdle

Default = 8, negative = no limit

minIdle

default = 0

initialSize

default = 0

maxWait

Default = indefinitely

 **activeConnection**
numActive = 4

 **idleConnection**
numIdle = 4

속성 이름 설명

initialSize BasicDataSource 클래스 생성 후 최초로 getConnection()

메서드를 호출할 때 커넥션 풀에 채워 넣을 커넥션 개수

maxActive 동시에 사용할 수 있는 최대 커넥션 개수(기본값: 8)

maxIdle 커넥션 풀에 반납할 때 최대로 유지될 수 있는 커넥션 개수(기본값: 8)

minIdle 최소한으로 유지할 커넥션 개수(기본값: 0)

웹 컨테이너(WAS)가 실행되면서 connection 객체를 미리 pool에 생성 HTTP 요청에 따라 pool에서 connection객체를 가져다 쓰고 반환한다. 이와 같은 방식으로 물리적인 데이터베이스 connection(연결) 부하를 줄이고 연결 관리 한다.

pool에 미리 connection이 생성되어 있기 때문에 connection을 생성하는 데 드는 요청 마다 연결 시간이 소비되지 않는다.

커넥션을 계속해서 재사용하기 때문에 생성되는 커넥션 수를 제한적으로 설정 함

동시 접속 할 경우 pool에서 미리 생성 된 connection을 제공하고 없을 경우는 사용자는 connection이 반환될 때까지 번호순대로 대기상태로 기다린다.

여기서 WAS에서 커넥션 풀을 크게 설정하면 메모리 소모가 큰 대신 많은 사용자가 대기시간이 줄어들고, 반대로 커넥션 풀을 적게 설정하면 그 만큼 대기 시간이 길어진다.

```
<bean id="dataSource-mysql"  
    class="org.apache.commons.dbcp.BasicDataSource"  
    destroy-method="close">  
    <property name="driverClassName"  
value="${Globals.DriverClassName}"/>  
    <property name="url" value="${Globals.Url}" />  
    <property name="username" value="${Globals.UserName}"/>  
    <property name="password" value="${Globals.Password}"/>  
    <property name="maxActive" value="${Globals.maxActive}"/>  
    <property name="maxIdle" value="${Globals.maxIdle}"/>  
    <property name="maxWait" value="${Globals.maxWait}"/>  
</bean>
```

보기

이름	나이	생일

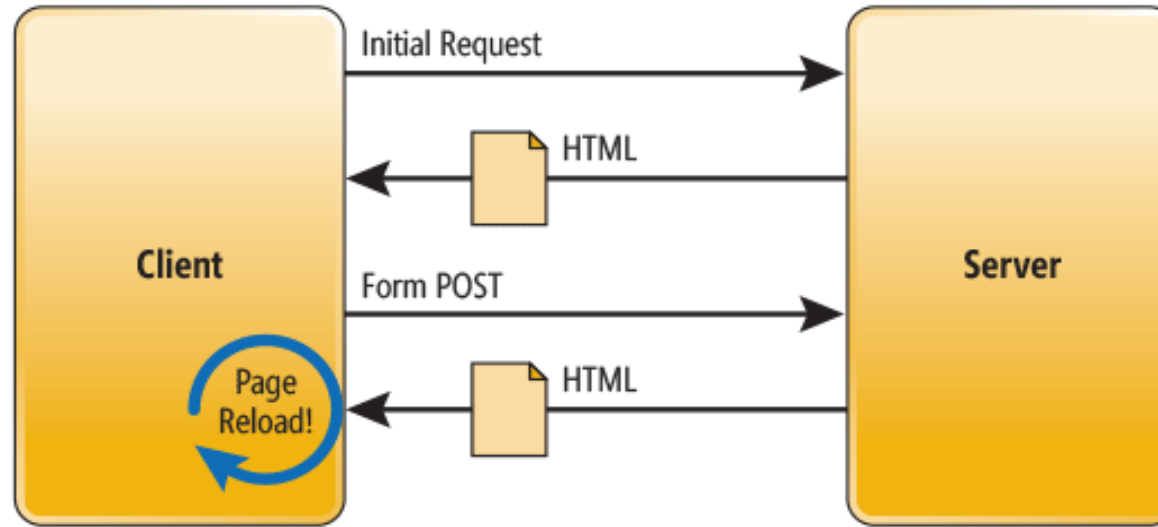
ajax

AJAX (Asynchronous Javascript And XML):

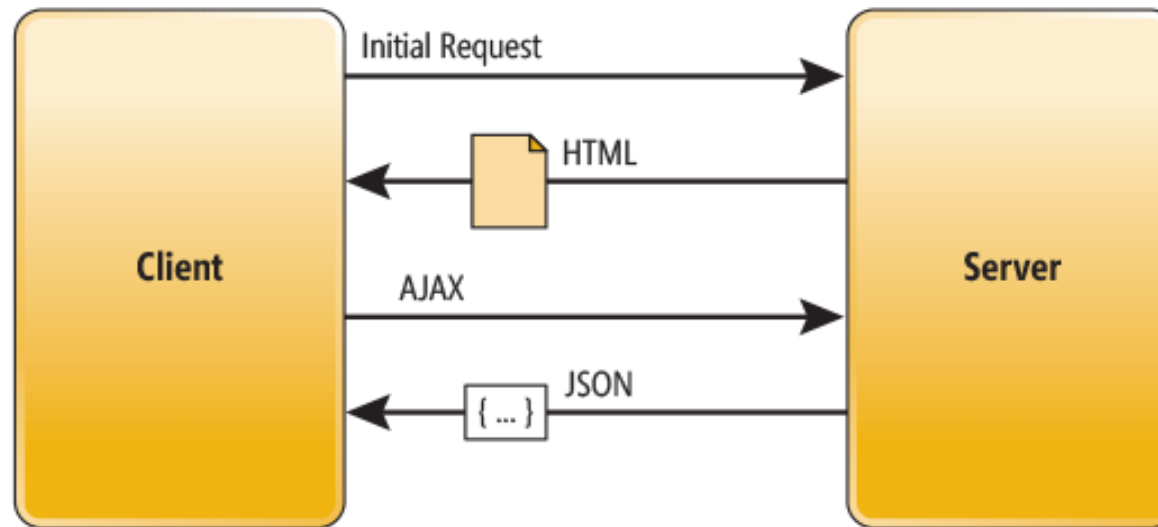
자바스크립트를 통해서 서버에 데이터를 요청

JavaScript의 라이브러리 중 하나이며 Asynchronous Javascript And Xml(비동기식 자바스크립트와 xml)의 약자이다. 브라우저가 가지고있는 XMLHttpRequest 객체를 이용해서 전체 페이지를 새로 고치지 않고도 페이지의 일부만을 위한 데이터를 로드하는 기법이며 **JavaScript**를 사용한 비동기 통신, 클라이언트와 서버간에 **XML** 데이터를 주고받는 기술이다.

Traditional Page Lifecycle

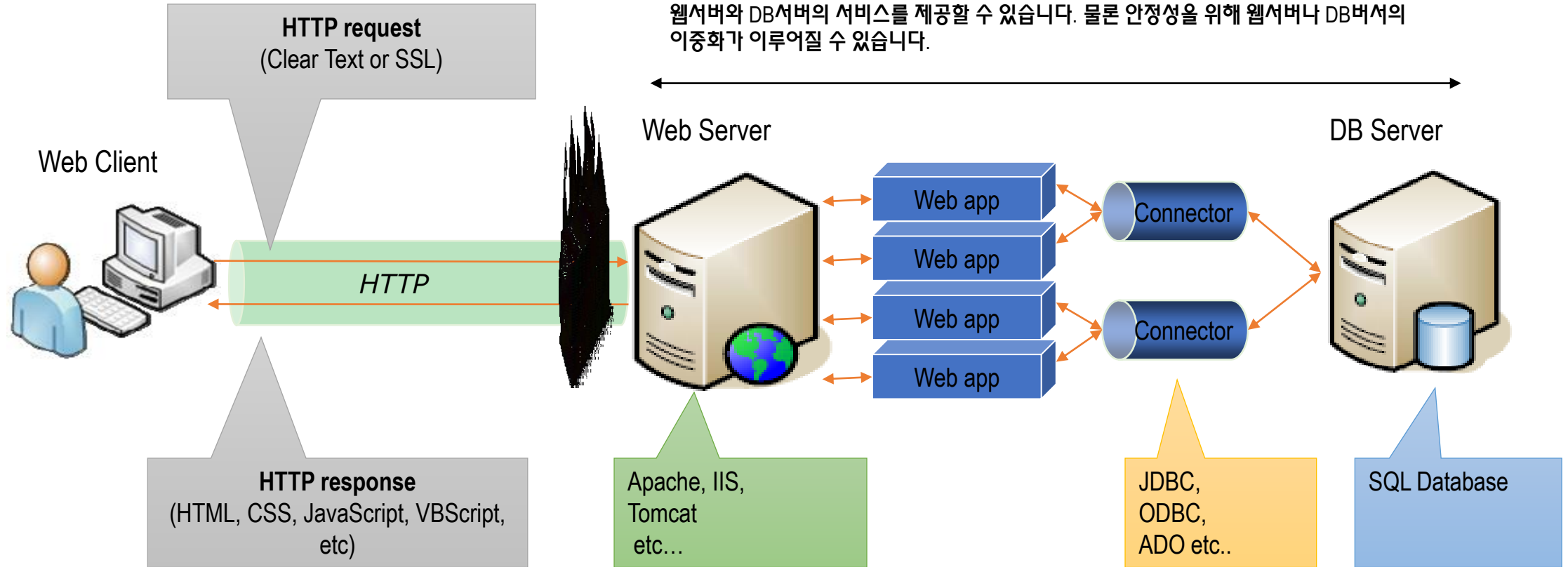


SPA Lifecycle



웹어플리케이션 아키텍처

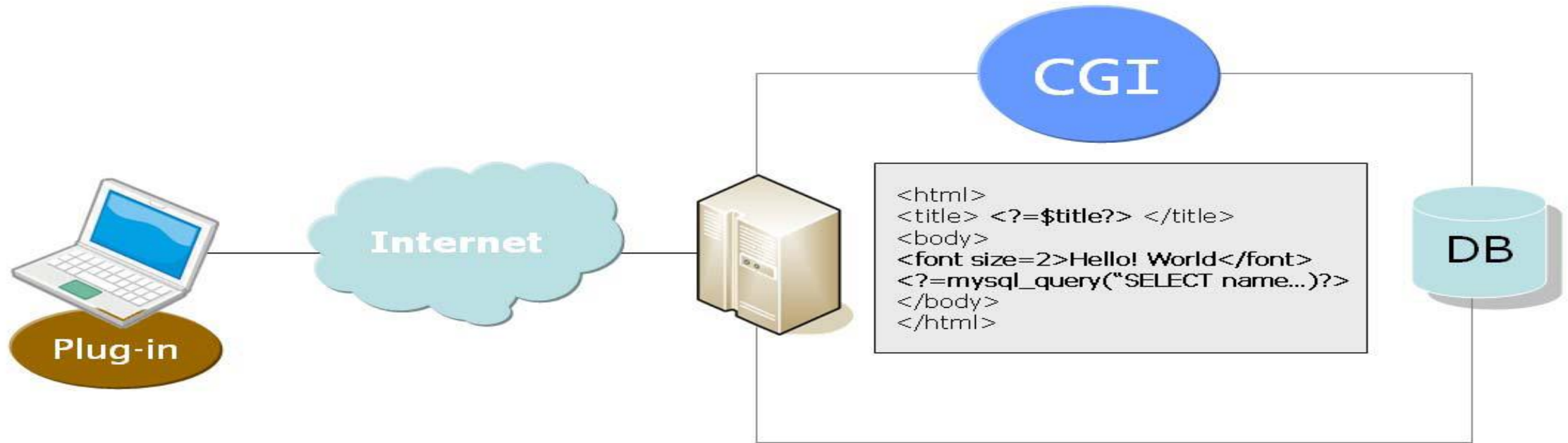
대부분이 2Tier로 분리되어 있을지만 3Tier가 존재할 수도 있습니다. 물론 소규모의 경우 1Tier에서 웹서버와 DB서버의 서비스를 제공할 수 있습니다. 물론 안정성을 위해 웹서버나 DB서버의 이중화가 이루어질 수 있습니다.



웹 문서 시대(1990년대)

1. Inside Browser

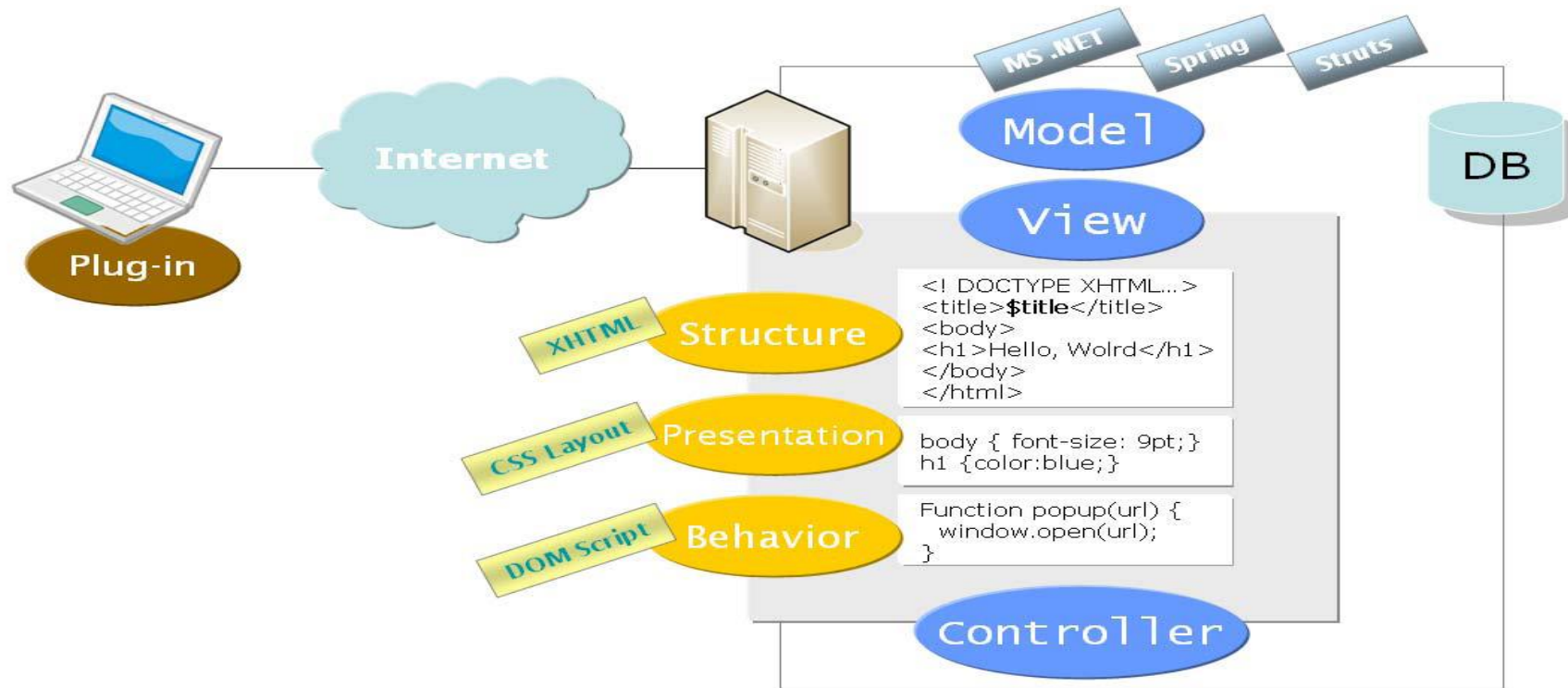
- 웹 서버와 웹 브라우저간 정적 HTML문서를 주로 보내거나 CGI(Common Gateway Interface)를 이용하여 개발하는 경우, 마크업과 프로그램 코드가 섞여있는 개발 방식을 사용했다.
- 이 때는 개발 직군간의 업무 분담이 전혀 이루어지지 않는 상태였습니다.



웹 표준 시대(2000년대 초반)

1. Inside Browser

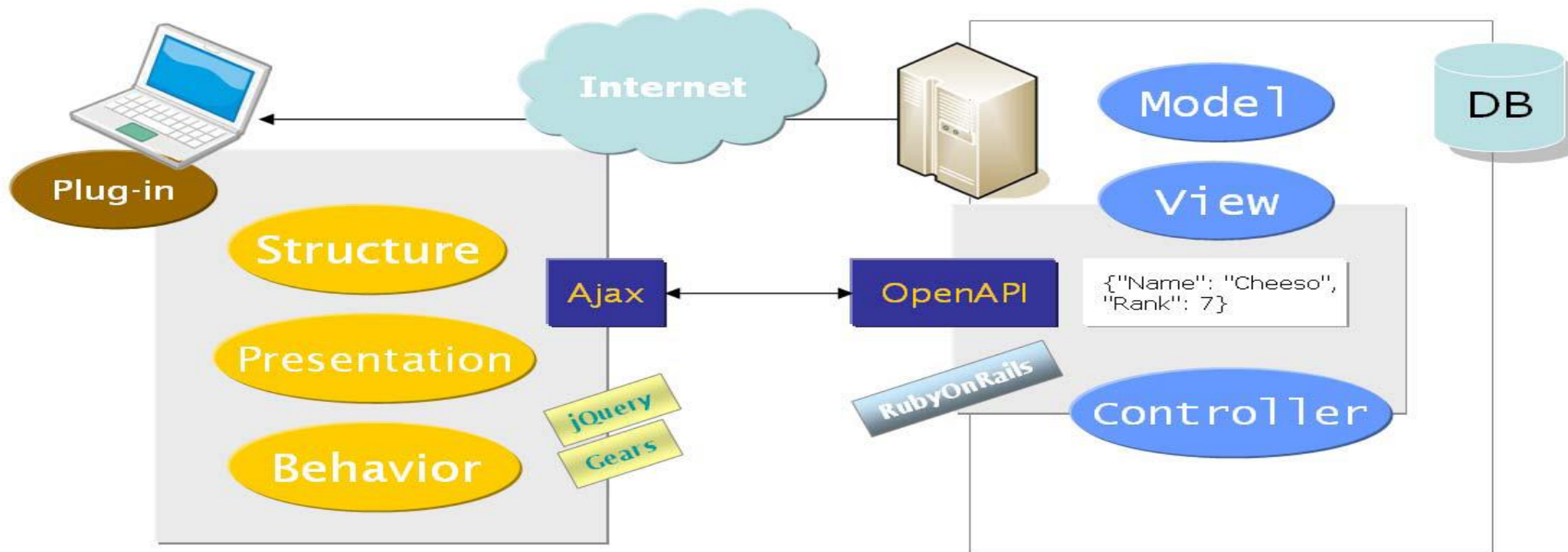
- 2000년대 초반으로 오면서 백엔드 개발에서는 이른바 **MVC 모델**이라는 기법을 통해 **데이터 모델과 템플릿 그리고 비즈니스 로직이 분리된 코드**를 통해 개발 생산성을 높이게 됩니다.
- **프론트엔드**에서도 웹 표준 개발 방법론을 통해 **구조(HTML), 표현(CSS), 동작(DOM Script)**를 분리하고 **CSS 레이아웃과 W3C 기반 DOM**을 통한 웹 개발 방식을 많이 이용하게 됩니다.



Ajax 시대 (2000년대 후반)

1. Inside Browser

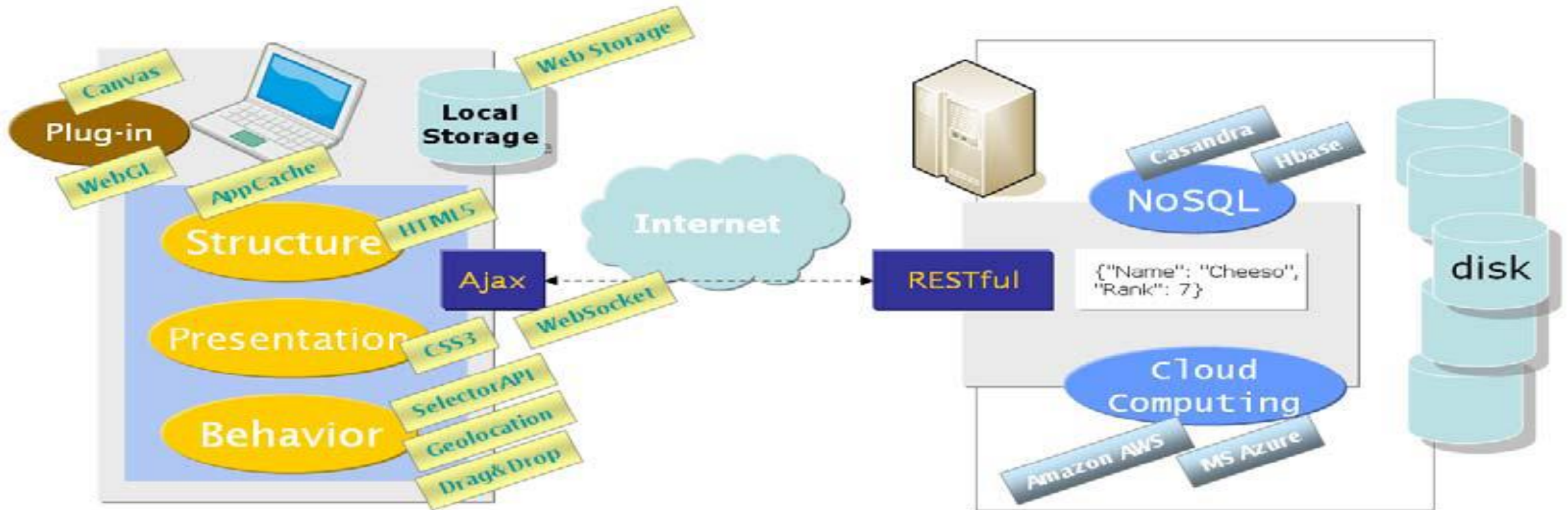
- 2004년 지메일과 구글맵스가 소개되면서 프론트 엔드 부분의 웹 기술의 혁신이 일어나게 됩니다.
- 즉, 프론트엔드 웹 콘텐츠가 고객의 PC에 일단 로딩이 된 후, 웹 서버에 Ajax 호출을 통해 데이터를 받아 온 후 기존 DOM을 갱신하는 개발 방법입니다.
- 이 방법을 통하면 백엔드 개발자가 json과 같은 데이터 기반 응답만 하게 되므로 더 간단한 웹 개발이 이루어진다. 이에 반해 프론트엔드 개발자는 다양하고 풍부한 사용자 경험을 제공하는 웹 애플리케이션 개발이 가능해졌습니다.



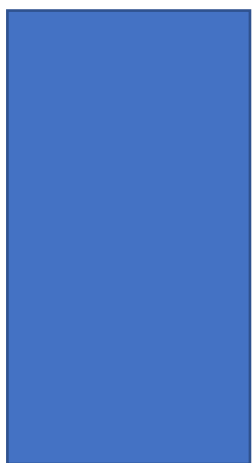
HTML5시대 (2010년대 초반)

1. Inside Browser

- HTML5가 가져올 가장 큰 변화는 **서버와 독립적인 웹 애플리케이션의 개발이 가능**하다는 것입니다.
- 특히, **모바일 환경에서 오프라인 기능과 로컬 데이터베이스의 지원**은 웹 서버와 독립할 수 있는 **여건**을 만들어 줍니다.
- 특히, HTML5의 Canvas, 드래그앤드롭, 지오로케이션, 파일API 등을 통한 사용자 경험을 확대해 줄 수 있습니다.







Ajax를 이용하여 다음을 작성
하십시오
연습문제
10분 시간드립니다.

태국

필리핀

홍콩

여기는 태국
태국 이미지

AOP

- Aspect Oriented Programming : 관점 지향 프로그래밍
- 하나의 프로그램을 관점(혹은 관심사)라는 논리적인 단위로 분리하여 관리하는 개념
- 로깅, 감사, 선언적 트랜잭션, 보안, 캐싱 등 다양한 곳에서 사용되고 있다.
- 여기에서는 메서드 호출을 관심사로 설정하여 AOP에 관한 실습을 진행한다.
- 관심사를 통해 Spring Framework가 어떤 메서드가 호출되는지 관심있게 지켜보다가 특정 메서드가 호출되면 자동으로 메서드 전과 후에 다른 메서드가 호출 될 수 있도록 한다.

핵심적인 관점 (Core Concerns) : 업무 로직을 포함하는 기능 (우리가 적용하고자 하는 핵심 비즈니스 로직)

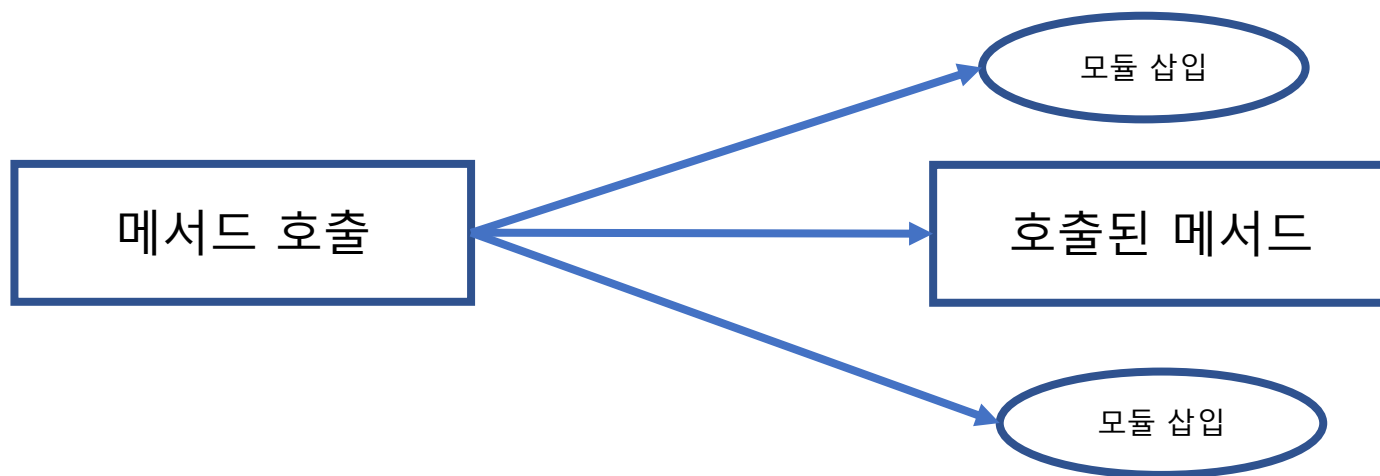
부가적인 관점 (Cross-cutting Concerns(인프라로직)) : 핵심 기능을 도와주는 부가적인 기능 (핵심 로직을 실행하기 위해서 행해지는 데이터베이스 연결, 로깅, 파일 입출력 등)



트랜잭션, 권한 체크, 시간 확인
등등

- 어플 전영역에 나타남
- 중복코드로 유지보수 저하
- 비즈니스 로직과 혼용되어 코드가독성 저하
- SRP: 2개의 이상의 기능X
- Open-closed

Spring AOP



Spring AOP 용어

- Joint Point : 모듈이 삽입되어 동작하게 되는 특정 위치(메서드 호출 등)
- Point Cut : 다양한 Joint Point 중에 어떤 것을 사용할지 선택
- Advice : Joint Point에 삽입되어 동작할 수 있는 코드
- Weaving : Advice를 핵심 로직 코드에 적용하는 것
- Aspect : Point Cut + Advice

Spring AOP Advice 종류

- before : 메서드 호출 전에 동작하는 Advice
- after-returning : 예외 없이 호출된 메서드의 동작이 완료되면 동작하는 Advice
- after-throwing : 호출된 메서드 동작 중 예외가 발생했을 때 동작하는 Advice
- after : 예외 발생 여부에 관계없이 호출된 메서드의 동작이 완료되면 동작하는 Advice
- around : 메서드 호출 전과 후에 동작하는 Advice

Spring AOP 구현

- XML을 이용한 구현방법
- @AspectJ 어노테이션을 이용한 구현방법

라이브러리 추가

- Pom.xml에 다음 라이브러리를 추가한다.

```
<dependency>  
<groupId>org.aspectj</groupId>  
<artifactId>aspectjweaver</artifactId>  
<version>${org.aspectj-version}</version>  
</dependency>
```

학습정리

- Spring은 AOP를 지원함으로써 메서드 호출 전과 후에 자동으로 동작하는 코드를 삽입할 수 있다.

Pointcut	JoinPoints
execution(public **(..))	public 메소드 실행
execution(* set*(..))	이름이 set으로 시작하는 모든 메소드명 실행
execution(* get*(..))	이름이 get으로 시작하는 모든 메소드명 실행
execution(* com.xyz.service.AccountService.*(..))	AccountService 인터페이스의 모든 메소드 실행
execution(* com.xyz.service.*.*(..))	service 패키지의 모든 메소드 실행
execution(* com.xyz.service..*.*(..))	service 패키지와 하위 패키지의 모든 메소드 실행
within(com.xyz.service.*)	service 패키지 내의 모든 결합점 (클래스 포함)
within(com.xyz.service..*)	service 패키지 및 하위 패키지의 모든 결합점 (클래스 포함)
bean(*Repository)	이름이 “Repository”로 끝나는 모든 빈
bean(*)	모든 빈
bean(account*)	이름이 'account'로 시작되는 모든 빈
bean(*dataSource) bean(*DataSource)	이름이 “dataSource” 나 “DataSource” 으로 끝나는 모든 빈



MyBatis

MyBatis 특징

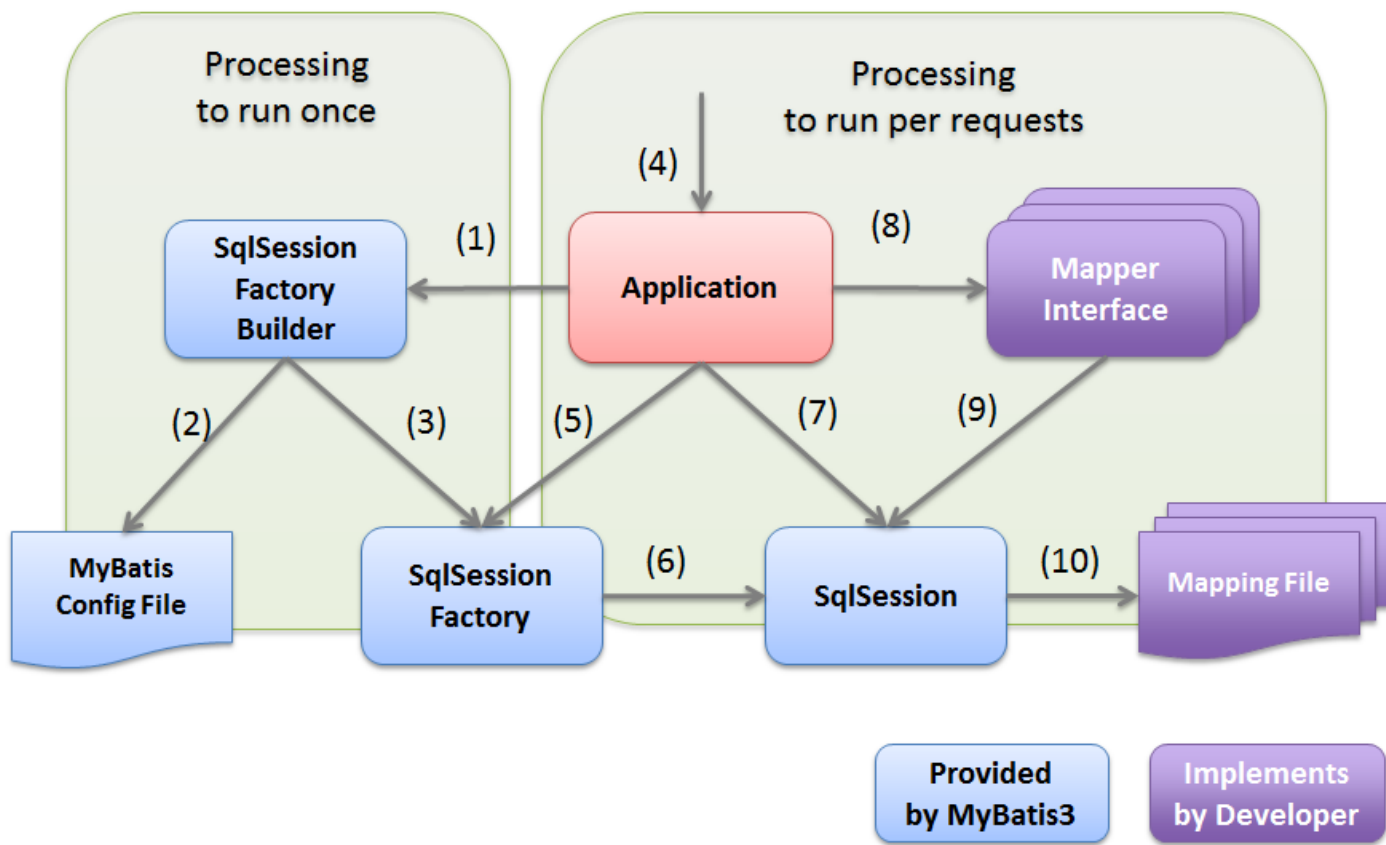
- 간단하다 : 간단한 퍼시스턴스(persistent) 프레임워크
- 생산성 : 62%정도 줄어드는 코드 , 간단한 설정
- 성능 : 구조적 강점(데이터 접근 속도를 높여주는 Join 매핑)
여러 가지 방식의 데이터를 가져오기 전략 (가져오기 미루기 , SQL 줄이기 기법)
- 관심사의 분리 : 설계를 향상 (유지보수성)
 - 리소스를 관리하여 계층화를 지원(커넥션,PreparedStatement,결과셋)
- 작업의 분배 : 팀을 세분화하는 것을 도움
- SQL문이 애플리케이션 소스 코드로부터 완전 분리
- 이식성 : 어떤 프로그래밍 언어로도 구현가능 (자바,C#,.NET,RUBY)



SQL문 구현 문제

- 자바 코드에 SQL문이 직접적으로 구현
- MyBatis환경에서는 SQL을
자바코드에서 분리하여 <mapper> xml 설정

- 자바와 DB 연결 시에 Connection객체 필요
 - 사용자가 웹사이트 접속할 때마다 생성 비효율
 - ConnectionPool 생성하여 (기존코드직접 환경 설정)
 - Mybatis는 이러한 환경 자동 설정
-
- DB 접속~종료까지의 라이프 사이클 생애주기 관리
 - 기존 코드에서는 connect()-close()까지 직접 구현하여 생애주기 관리
 - Mybatis는 SqlSession을 이용해 자동으로 관리



1. 클라이언트는 응용 프로그램에 대한 프로세스를 요청.
2. 응용 프로그램은 SqlSessionFactoryBuilder를 사용하여 작성된 SqlSessionFactory에서 SqlSession을 가져옴
3. SqlSessionFactory는 SqlSession을 생성하여 이를 애플리케이션으로 리턴
4. 응용 프로그램은 SqlSession에서 Mapper Interface의 구현 객체를 가져옴
5. 응용 프로그램은 Mapper Interface 메서드를 호출
6. Mapper Interface의 구현 객체는 SqlSession 메서드를 호출하고 SQL 실행을 요청
7. SqlSession은 매핑 파일에서 실행할 SQL을 가져 와서 SQL을 실행