

**DOM(document object model)**

=> HTML,XML 문서의 프로그래밍 interface이다. DOM은 문서의 구조화된 표현을 제공한다.

즉, 문서 구조 및 스타일, 내용을 담당한다. 이들은 웹 페이지를 script 또는 프로그래밍 언어들에 의해 사용될 수 있게 연결시켜주는 역할 담당한다.

-document 객체: 웹페이지를 담당하는 객체

**BOM(browser object model)**

=> javaScript가 브라우저와 소통하기 위한 모델이다. 공식 표준은 아니지만 대부분의 브라우저가 자바스크립트 상호작용에 있어 여러 함수들을 사용한다. BOM은 웹 브라우저 창을 관리할 목적으로 제공되는 객체 모음을 대상으로 하는 모델로서, 자바스크립트 등에서 이를 사용한다.

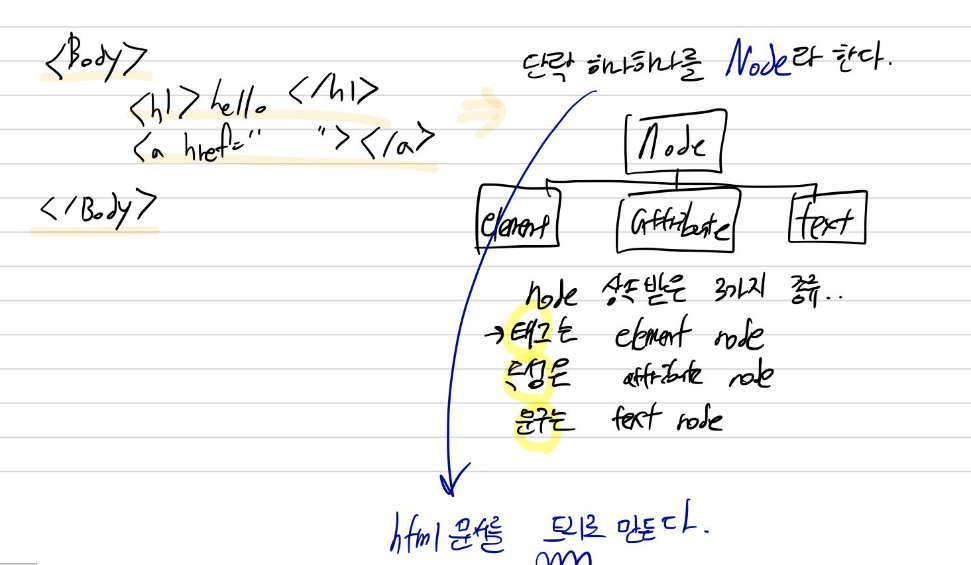
-navigator 객체: 브라우저, 운영체제, 사용자 정보 등에 대한 정보 있다.

-screen: 화면에 대한 정보(너비, 높이,픽셀..)

-location: 주소에 대한 정보

-history: 앞으로 가기, 뒤로가기...

**html 문서를 트리로 만들 수 있다**



1.

**웹 브라우저:** 웹에서의 클라이언트, 사용자의 작업창이라고 할 수 있다.

(웹 서버가 우리한테 응답하는 것은 html문서이다. 해당 문서를 해석해서 그림처럼 보여주는 프로그램이 ‘웹 브라우저’)

**웹 어플리케이션**: 동적으로 응답되는 html 페이지를 제공한다. 동적 웹 페이지와 유사한 개념

**웹 서버:** 웹 브라우저 요청을 받아들이고 요청에 대한 결과를 웹 브라우저에게 응답하는 곳.

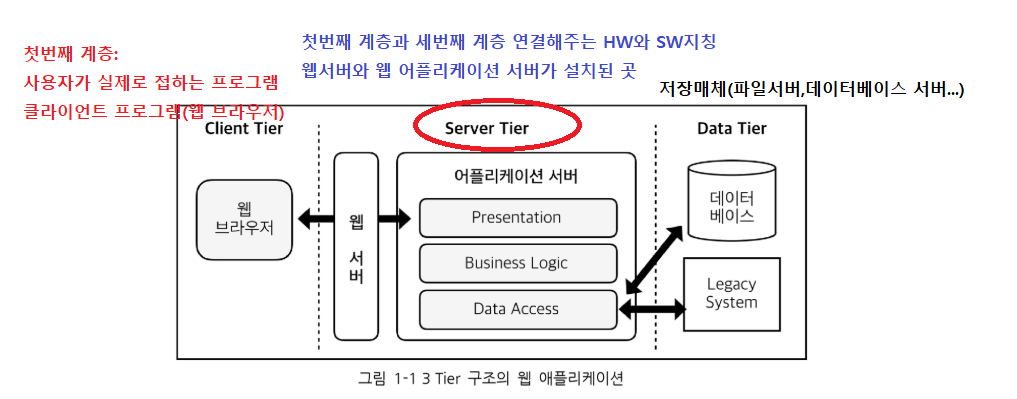
또한, 웹 어플리케이션 서버에 이들의 처리를 요청하는 작업을 수행한다.

**웹 어플리케이션 서버(WAS):** 웹 브라우저가 요청한 작업에 필요한 프로그래밍 로직의 처리 및 데이터베이스와의 연동을 처리하는 부분. 처리결과를 웹 브라우저로 응답하기 위해 처리결과를 웹 서버에 보낸다. (우리가 백엔드 프로그래밍 하는 부분)

(<=> 정적 웹 서버: 완성된 html 가지고 있고 요청이 오면 해당 파일을 브라우저에 띄운다.)

**\*Apache Tomcat은 웹서버와 웹 어플리케이션 서버 역할 모두 수행하는 솔루션이다.**

2. **3 Tier구조**의 웹 애플리케이션



MVC(model, controller, view)모델은 디자인 패턴, 3 Tier Architecture은 아키텍쳐 스타일!

3-tier에 mvc패턴 사용할 수 있다.

3. 서블릿(Server Side Applet)

-서버 프로토콜 종류에 관계없이 응용프로그램을 개발 할 수 있다.

-서블릿은 독립된 스레드 기반으로 서비스되낟. 다중 스레드 서비스가 기본적으로 제공된다.

-**서블릿 컨테이너는 클라이언트에서 전송되는 Request와 Response에 대한 처리를 담당한다.**

(서블릿 컨테이너로는 오픈소스인 Tomcat이 대표적)

-서블릿 클래스는 객체를 생성하거나 메소드를 호출하는 주체가 **서블릿 컨테이너**이다.

-business와 presentation 로직이 혼합된 형태 => 이를 보완하기 위해 JSP사용

3-1) 서버에서 response에 대한 상태 코드

-4xx: 클라이언트 오류

-5xx: 서버 오류

-200: 성공적인 수행

3-2) **서블릿 등록과 URL 매핑**

1. HttpServlet 클래스 상속 받아서 원하는 처리를 doGet,doPost를 재정의해서 구현한다.

=>web.xml에서 mapping한다

|  |
| --- |
| **extends** HttpServlet  protected void doGet(HttpServletRequest req, HttpServletResponse resp){...}  HttpServletRequest req => 클라이언트가 보낸 요청에 대한 정보들이 들어있다.  예를들어 웹 클라이언트로부터 전달된 매개변수를 파싱  HttpServletResponse resp => 서블릿이 응답할 내용이나 정책에대해서 정해줄 수 있다. 응답과 관련된 작업을 할 수 있도록 기능 제공 |
|  |

|  |
| --- |
|  |

1. @WetServlet 에노테이션 추가한다.( @WebServlet("/hello.do"))

=>서블릿 3.0 부터 지원되는 어노테이션은 web.xml파일에 자동으로 서블릿으로 등록됨

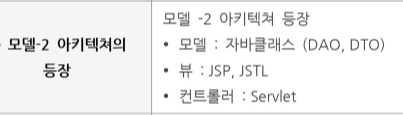
**\*\*web.xml과 어노테이션 동시에 설정할 경우 web.xml의 설정이 우선시된다.**

3-3) Servlet과 JSP 변천사

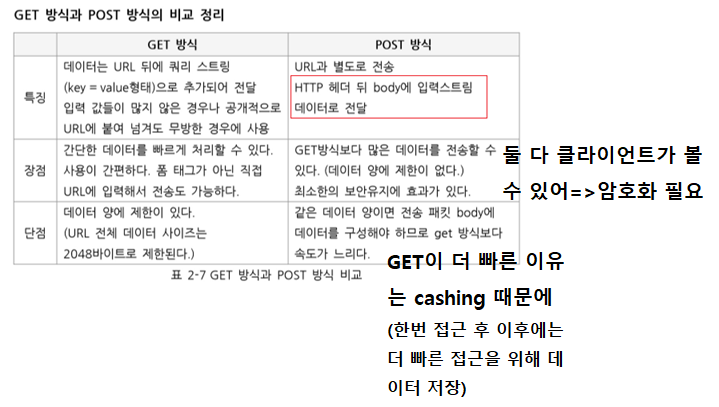
서블릿과 JSP: 웹 어플리케이션에서 동적으로 html 응답을 만드는 주체!(요청에 대한 응답 담당자)

서블릿: java로 응답 <=> jsp: html스러우면서도 java 코드 사용 가능

**=>business는 servlet으로! presentation은 jsp로!**



4.



GET방식 예시



POST의 경우 => URL에 데이터 붙지 않고 **본문의 body에 포함하여 전송한다.**

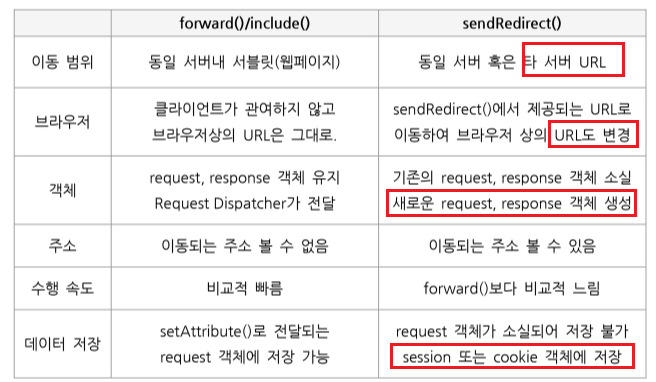
\* <a>태그는 get방식만 지원

<form>태그는 post/get 모두 지원

5. ServletContext 객체는 웹 어플리케이션 당 하나씩 생성된다. Servlet이 Servlet Container와 소통하기 위한 메소드들의 집합을 정의한 객체이다. ServletConfig는 해당 서블릿에서 사용 가능 => **웹 어플리케이션 내의 공통적 내용 공유는 ServletContext로!**

6. **서블릿의 페이지 이동**

|  |  |
| --- | --- |
| Forward | -동일 서버 내부의 서블릿 간 페이지 이동  -최초의 **request 객체가 그대로 유지**되어 전달된다.  -브라우저 상의 url 변화 없다.    -request의 setAttribute로 데이터 저장하여 forward된 서블릿에 데이터 전달  -**이때 setAttribute와 getAttribute의 데이터 타입은 object이다** |
| Redirect | -다른 서버까지 자유롭게 이동 가능하다.  -새 URL로 바뀐다.  -request,response객체 새로 생성된다.    -총 2번의 request, respose 발생한다. 이전의 객체들이 사라지므로 request에 원하는 데이터를 저장하여 전달할 수 없다. **Session 혹은 Cookie 객체에 저장해야 한다.** |
| Include | -현재 페이지에 다른 페이지 포함한다.  -include된 서블릿을 먼저 수행하고 돌아와서 원래 서블릿의 나머지 부분을 수행!  **(forward하면 이후의 코드 실행 X / Include는 돌아와서 이후 코드 실행 O)**  -include된 서블릿의 response가 본래 서블릿의 response에 포함된다. |



7. 쿠키와 세션

-클라이언트 단위로 상태정보를 유지해야 하는 경우 사용한다.

|  |  |
| --- | --- |
| 쿠키 | -클라이언트(local)에 **text 타입**으로 저장된다.  => 각자의 로컬 영역이 허락되는 만큼 저장 가능하다.  => 데이터 크기 제한 있다.  -setMaxAge(초단위)로 시간 정해주지 않는다면, default는 **브라우저 닫는 순간까지!**  -처리방식  1.클라이언트 요청에 따라 웹 서버가 클라이언트에게 쿠키 부여  2.쿠키가 사용자의 디스크나 웹 브라우저 메모리에 저장됨  3.사용자가 쿠키 부여 받은 웹 사이트 다시 요청할 때 쿠키가 전송되며, 웹 서버는 해당 쿠키를 통해 클라이언트를 구별한다. |
| 세션 | -서버에 object 타입으로 저장된다.  => 서버가 수용 할 수 있는 만큼 데이터 저장 가능하다.  => 브라우저 수만큼 세션이 생성된다. 브라우저의 구분위해 jsessionID 사용  -한 사용자와 웹 서버간에 존재하는 하나의 가상 연결이라 할 수 있다.  -처리방식  1.클라이언트(웹브라우저)가 특정 웹사이트(도메인)에 요청  2.서버는 접속한 클라이언트에게 세션ID 생성해주고 ID를 저장한다  3.서버는 세션ID와 함께 클라이언트에게 응답  4.클라이언트는 세션 ID를 쿠키에 저장, 이때 **쿠키 이름은 jsessionId**  5.클라이언트가 도메인에 재요청시 저장된 세션ID를 서버에 넘긴다  6.서버는 세션ID로 클라이언트 구별한다. |

>>쿠키 예제 코드



>>세션 예제 코드



\*\* 세션 초기화 부분 알기

**\*\* 시간 정해주기**

8. Filter

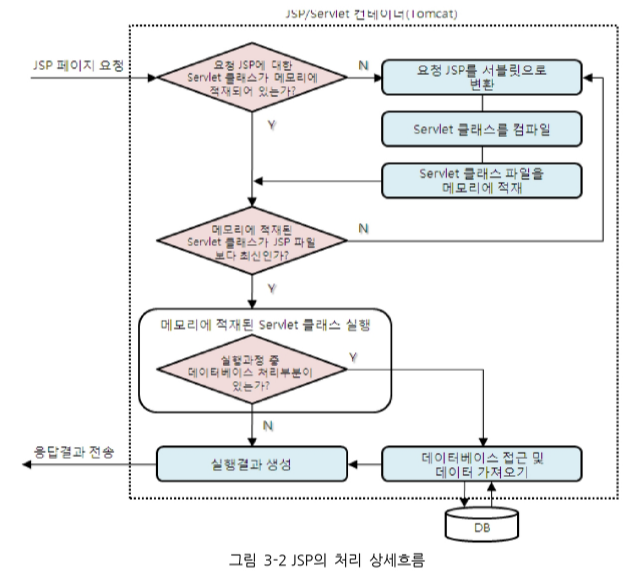
-사용자 인증 등과 같이 모든 서블릿이 공통적으로 필요로 하는 기능들을 서블릿이 호출되기 전에 전처리하고 싶거나, 서블릿이 호출되고 난후에 후처리하고 싶다면 필터로 구현한다.

-예) 사용자 인증, 이미지 변환 및 압축, 암호화 etc



9. **JSP(java server pages) 웹페이지를 동적으로 처리**할 수 있는 기술

-서블릿 프로그램 기능을 HTML 파일내에 스크립트 형식으로 구현한다.

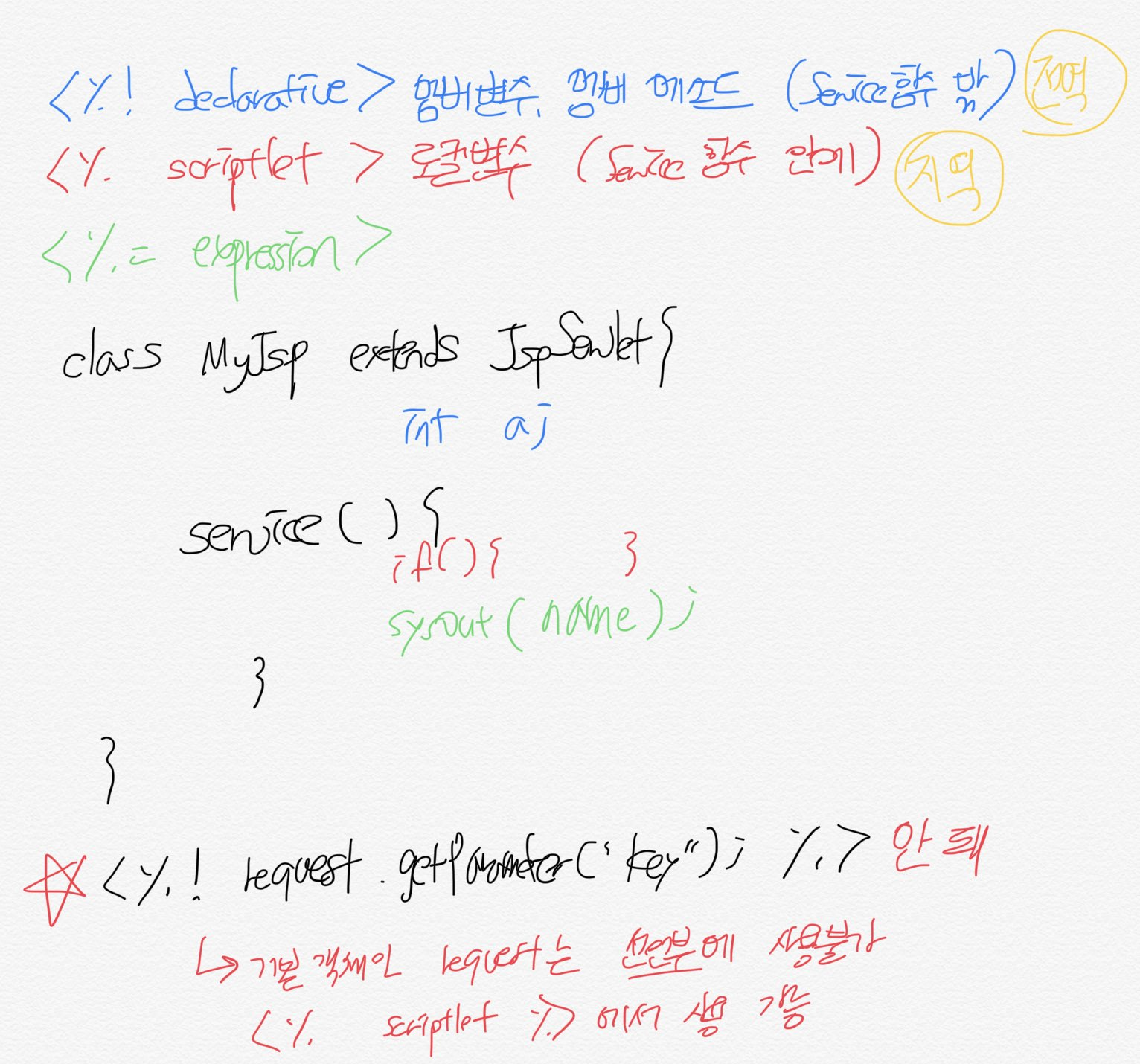


-맨 처음 요청 및 JSP페이지가 변화됐을 때에만 컴파일 과정 걸친다.

-한 번이라도 요청했던 JSP파일은 재요청시 메모리에 적재된 클래스가 응답하므로 평균 응답 시간이 매우 짧다.

10. JSP의 Scripting Element

|  |  |
| --- | --- |
| Scriptlet | <% 자바 코드 ; %> (=>지역)  -JSP페이지 내에서 java 코드 구현 위해 사용  -서블릿으로 변환시: 서블릿의 **service 메소드**로 구현  -함수 선언 안 된다!(함수 안에 함수 선언 안 되는걸 생각해봐~)  -로컬변수로 사용된다 |
| Declaration(선언) | <%! 속성선언 및 method 작성 %> (=>전역)  -서블릿으로 변환시: 서블릿의 멤버변수나 멤버 메소드로 선언된다.  -멤버변수 선언이나 메소드 선언에 사용된다. |
| Expression(표현식) | <%= 출력값,변수,함수결과 %> (=>자바코드 아님[결과값만])  -**코드에 세미콜론(;) 없는 것에 주의**  -출력할 값, 변수, 함수 결과 출력을 위해 사용 |



p.118~119 확인하기

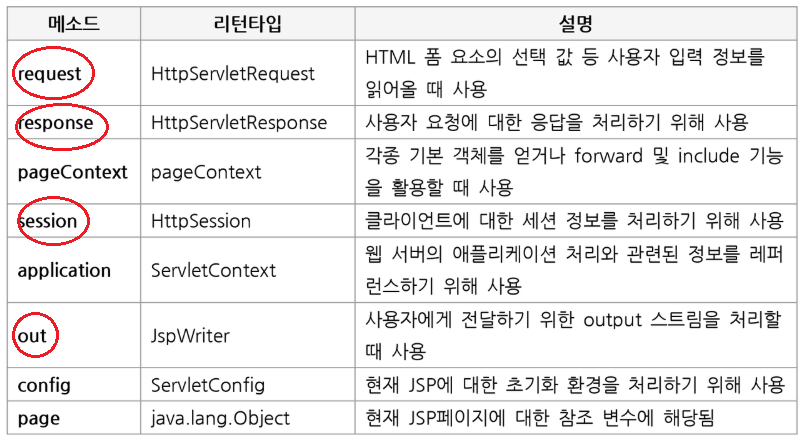
11. 지시자(Directive) 태그

|  |  |  |
| --- | --- | --- |
| page Directive  =>속성 알기! | -컨테이너에게 현재 JSP페이지를 **어떻게 처리**할 것인가에 대한 정보를 제공     |  | | --- | | contentType: 브라우저로 내보내는 내용의 형식 지정 및 문자 집합 지정  pageEncoding: 현재 JSP 페이지 문자집합 지정  import: 현재 JSP페이지에서 사용할 Java 패키지나 클래스 지정  errorPage: 에러가 발생할 때에 대신 처리될 JSP 페이지 지정  isErrorPage: 에러 핸들링 하는 페이지인지 지정하는 요소(true로 지정해야지 exception 내장객체 사용가능=> error 정보 알 수 있다.) | |
| include Directive | -다른 JSP페이지를 파일에 삽입하여 사용 (이후 액션태그와 비교하기)    - |
| taglib Directive | -custom tag이용할 때 사용. ( => JSTL 사용할 때 taglib 필요) |

12. JSP 기본 객체

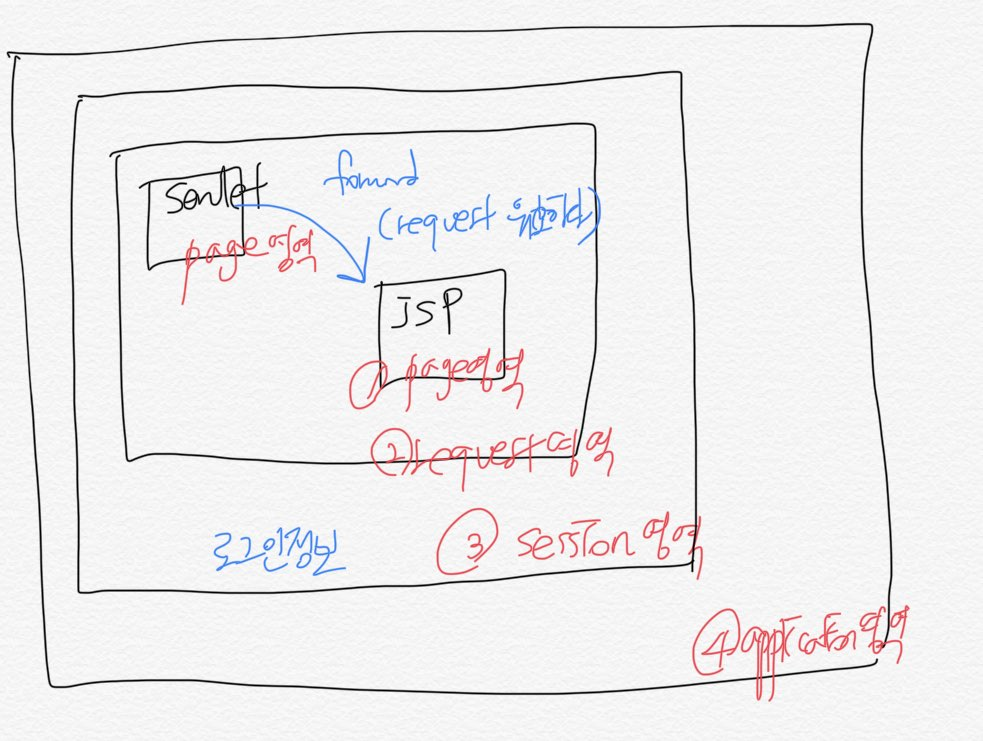
-선언부에는 기본 객체 사용 못 한다

<%! request.getParameter(“name”)%> 불가!

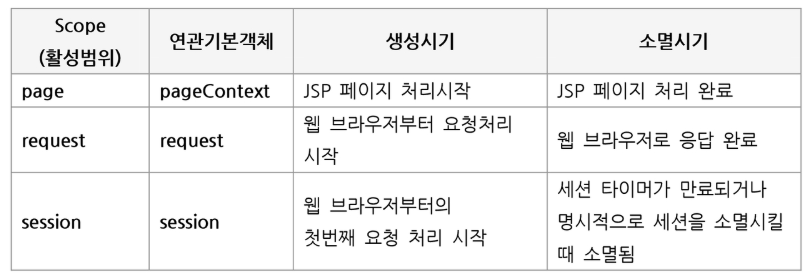


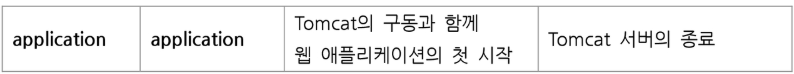
|  |  |
| --- | --- |
| request 기본 객체의  주요 메소드 | getParameter(String name)  getCookies() => 리턴 타입 Cookie[]  getSession()  **setCharacterEncoding()** |
| response 기본 객체의  주요 메소드 | **setContentType(String type)** => contentType을 설정  **sendRedirct(url)** => 클라이언트 요청을 다른 페이지로 보낸다 |

13. 활성 범위(Scope)와 기본 객체



Life Cycle 비교: application>session>request>page





-application scope에서 javax.servlet.**ServletContext** 인터페이스 구현한 객체를 생성하고 사용한다. 컨테이너(tomcat)구동과 함께 생성되고 서버의 종료에 함께 소멸된다.

14. JSP 액션 태그

**<jsp:include page=”페이지”></jsp:include>**

|  |  |
| --- | --- |
| Include Directive | Include Action Tag |
| -JSP 파일을 JAVA파일로 변환하기 전에 include 지시문 위치에 파일 내용 그대로 삽입시키고 서블릿 코드로 변환된다.  -포함하는 jsp와 포함되는 jsp 합쳐져서 새로운 페이지 생성되어 컴파일 된다,  -정적인 특징 | -include되는 파일과 include하는 파일이 각각 컴파일된 후 두 파일의 실행 결과를 한곳에 합쳐서 출력된다,  -즉, 별도로 동작하기 때문에 변수명 겹쳐도 된다.  -요청시점에 동적으로 처리한다.  -request 기본객체의 attribute나 <jsp:param>이용한 파라미터 전달된다. |

**<jsp:useBean> 액션태그 ... 우선 패스**

15.

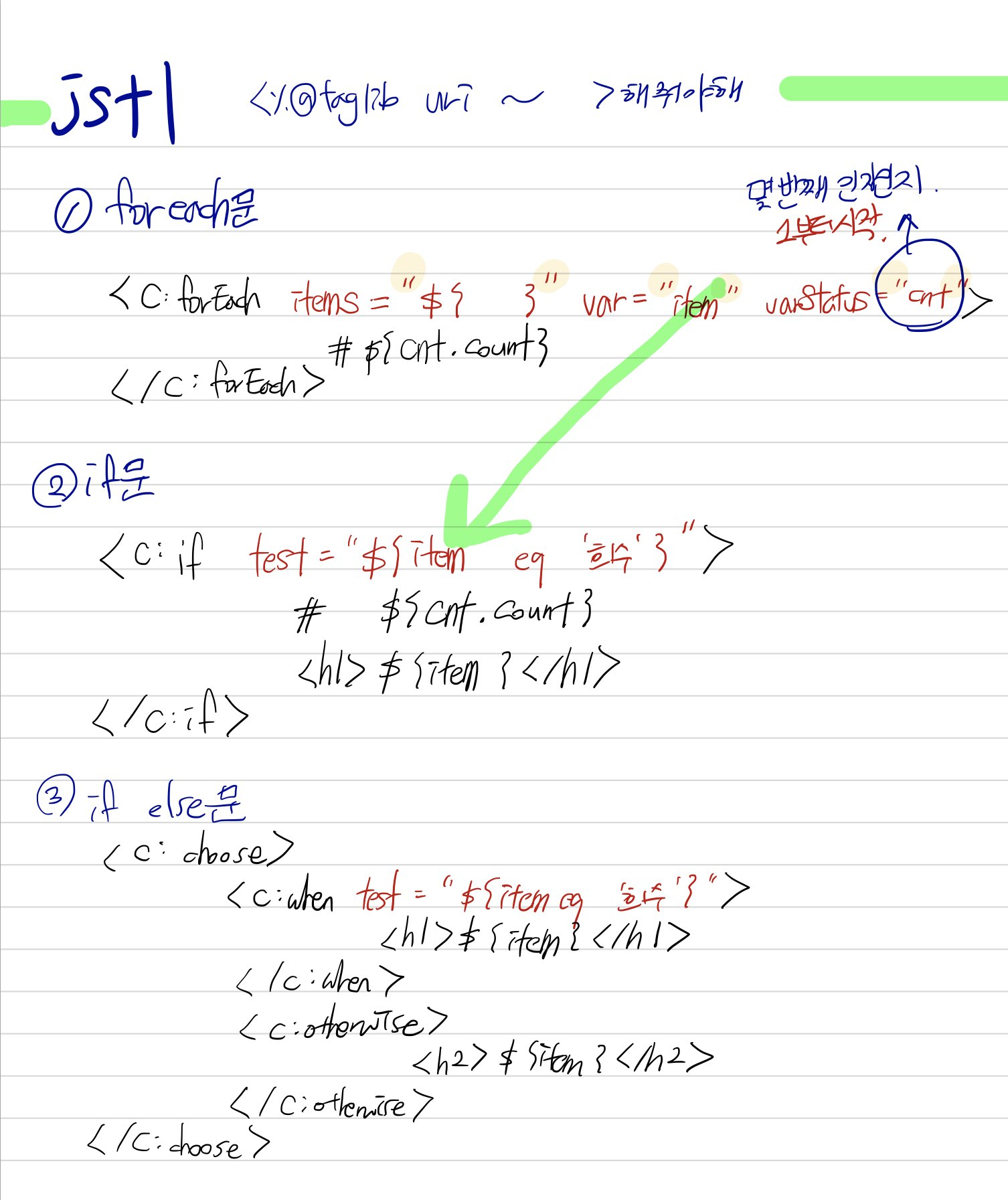
15-1) custom tag => **JSTL** (java standard tag library)

-jstl은 태그를 통해 JSP 코드를 관리하는 라이브러리이다. (if, for ...)

-**JSTL Core 라이브러리:** 주로 presentation 단계에서 사용되며 4가지 scope의 Attribute의 객체에 모두 접근 가능하다.



-조건,반복,이동에 관한 태그 지원



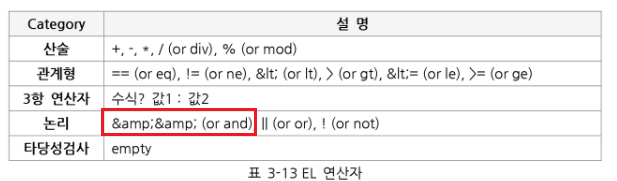
15-2) EL(expression language)

-EL은 숫자는 숫자로, 문자는 문자로 인식한다.

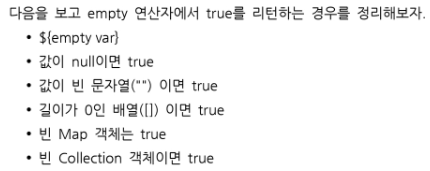
-${requestScope.속성명}으로 접근되지만 보통 ${속성명}으로 접근한다. 이와 같은 경우에는 좁은 범위의 Scope부터 해당 속성을 찾는다.**(순차적인 범위에서 getAttribute(“name”)함수 적용됨)**

|  |
| --- |
|  |

-연산이 가능하다.

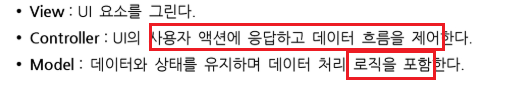


-타당성 검사: 특정 scope에 저장되어 있으면 true, 저장되어 있지 않으면 false



16. MVC pattern model

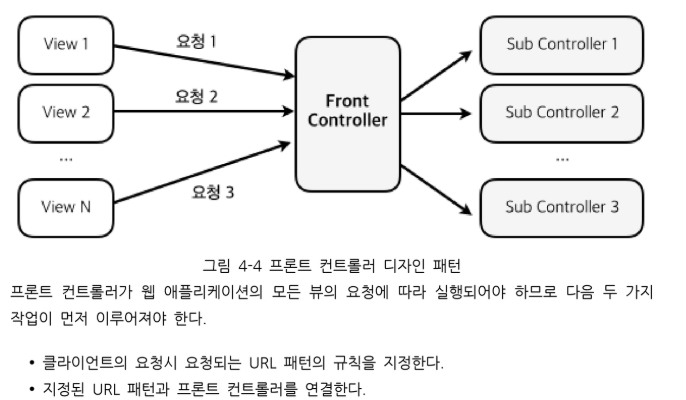
Model-View-Controller



-Model1과 2 모두 MVC 패턴에 기초한다.

|  |  |
| --- | --- |
| Model 1 아키텍쳐 | -client(브라우저)에서 들어오는 request와 처리된 response가 JSP 페이지에 의해 모두 처리된다.  -간단한 어플리케이션에 적합하다.  -JSP페이내에 스크립트릿이 많아질수록(자바코드 많아질 수록) 코드가 복잡해진다.  -**JSP페이지에서 request에 의한 처리(컨트롤러 역할)도 하고, 뷰의 역할도 하게 된다.** (모두 JSP가 수행하므로) => Model 2 아키텍쳐 도입 |
| Model 2 아키텍쳐 | -**client의 요청은 서블릿이 처리 + JSP는 컨텐츠를 화면에 생성하거나 보여주는 역할**  컨트롤러 역할 JSP => model1  컨트롤러 역할 서블릿 => model2 |

17. 프론트 컨트롤러 디자인 패턴



|  |  |
| --- | --- |
| getParameter() | getAttribute() |
| -GET/POST 방식으로 파라미터 값 넘기고, 그 값 사용할 때 사용  -String타입으로 반환된다.  -setParameter는 없다  -웹 브라우저에서 전송받은 request 영역의 값을 읽어온다. | -page,request,session,application과 같은 scope의 임의의 속성을 저장하고 가져올 때 사용  -key와 value값으로 지정  -object 타입으로 반환한다.  -setAttribute()하지 않은 값은 null을 리턴 |

|  |  |  |
| --- | --- | --- |
| -서블릿과 jsp는 java 코드이므로 재컴파일 해야지 새로운 내용이 반영된다.  -jsp도 서블릿이므로 forward로 호출!  -jsp=>서블릿=>.class   |  | | --- | | jsp는 서블릿이다 O  서블릿은 jsp이다 X  jsp는 서블릿의 모든 기능이 가능하다 O  서블릿은 jsp의 모든 기능이 가능하다 X |   -model2는 presentation을 JSP로 business를 servlet으로!   |  | | --- | | <form action="#" method="post">  #은 현재페이지  <input type="text" name="pname">  name="서버로 넘어가는 키값" | |