웹 개발

웹 프로그래밍 기초

1.언어

저급 언어

* 기계 중심의 언어
* 2진수로 이뤄진 값으로 작성하는 프로그래밍 언어(기계어)

\* 컴파일러

: 기호로 작성된 프로그램을 숫자로 바꿔주는 도구

고급 언어

* 사람 중심의 언어
* 작성된 코드는 컴파일러를 통해 저급 언어로 번역

2. HTTP(Hypertext Transfer Protocol)

- 무상태(Stateless)

ㄴ 요청과 응답을 하면 연결을 끊어버리는 상태

ㄴ 연결 상태 유지 X

장점

1. 특정 다수를 대상으로 하는 서비에는 적합
2. 클라이언트와 서버 간의 최대 연결 수 보다 훨씬 많은 요청과 응답을 처리

단점

1. 클라이언트과의 이전상황을 모름

ㄴ 정보 유지를 위해 쿠키가 등장

○ 인터넷

- TCP / IP 기반의 네트워크가 전세계적으로 확대되어 하나로 연결된 네트워크들의 네트워크

○ URL (Uniform Resource Locator)

- 인터넷 상의 자원의 위치

- 특정 웹 서버의 특정 파일에 접근하기 위한 경로 혹은 주소

IP 주소 or 도메인 이름 문서 이름

EX) <http://www.sunnyvale.co.kr/docs/index.html>

접근 프로토콜 문서의 경로

○ HTTP 동작

텍스트이(가) 표시된 사진

자동 생성된 설명

● 요청 헤더 명령어

GET

* 정보를 요청 (SELECT)
* 요청 바디는 사용 X

POST

* 정보를 밀어 넣기 위해 사용 (INSERT)

PUT

* 정보를 업데이트 (UPDATE)

DELETE

* 정보를 삭제 (DELETE)

HEAD

* (HTTP) 헤더 정보만 요청
* 해당 자원이 존재하는지, 서버 이상 확인

OPTIONS

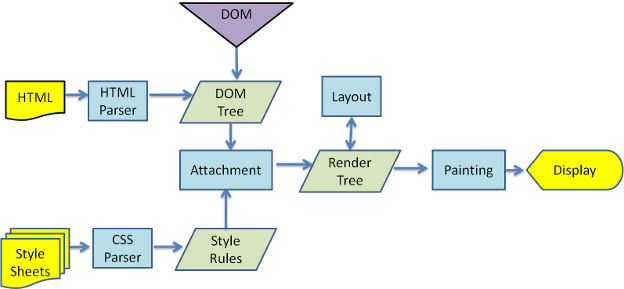
* 웹서버가 지원하는 메서드의 종류를 요청

TRACE

* 클라이언트의 요청을 그대로 반환
* 브라우저 구성 요소



○브라우저 랜더링



○ 웹서버

- 웹 서버 소프트웨어가 동작하는 컴퓨터

- 클라이언트가 요청하는 HTML 문서나 각종 리소스를 전달하는 역할

- 정적 컨텐츠 처리

\* 웹 크롤러

ㄴ 다른 웹사이트의 정보를 읽어갈 때 사용하는 소프트웨어

○ WAS (Web Application Server)

- 미들웨어

- 클라이언트 요청 중 보통 웹 애플리케이션이 동작하도록 지원

- 동적 컨텐츠 처리

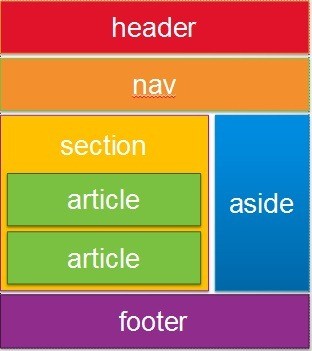
- 정적인 컨텐츠도 처리하는데 있어서 웹서버와 동일

HTML

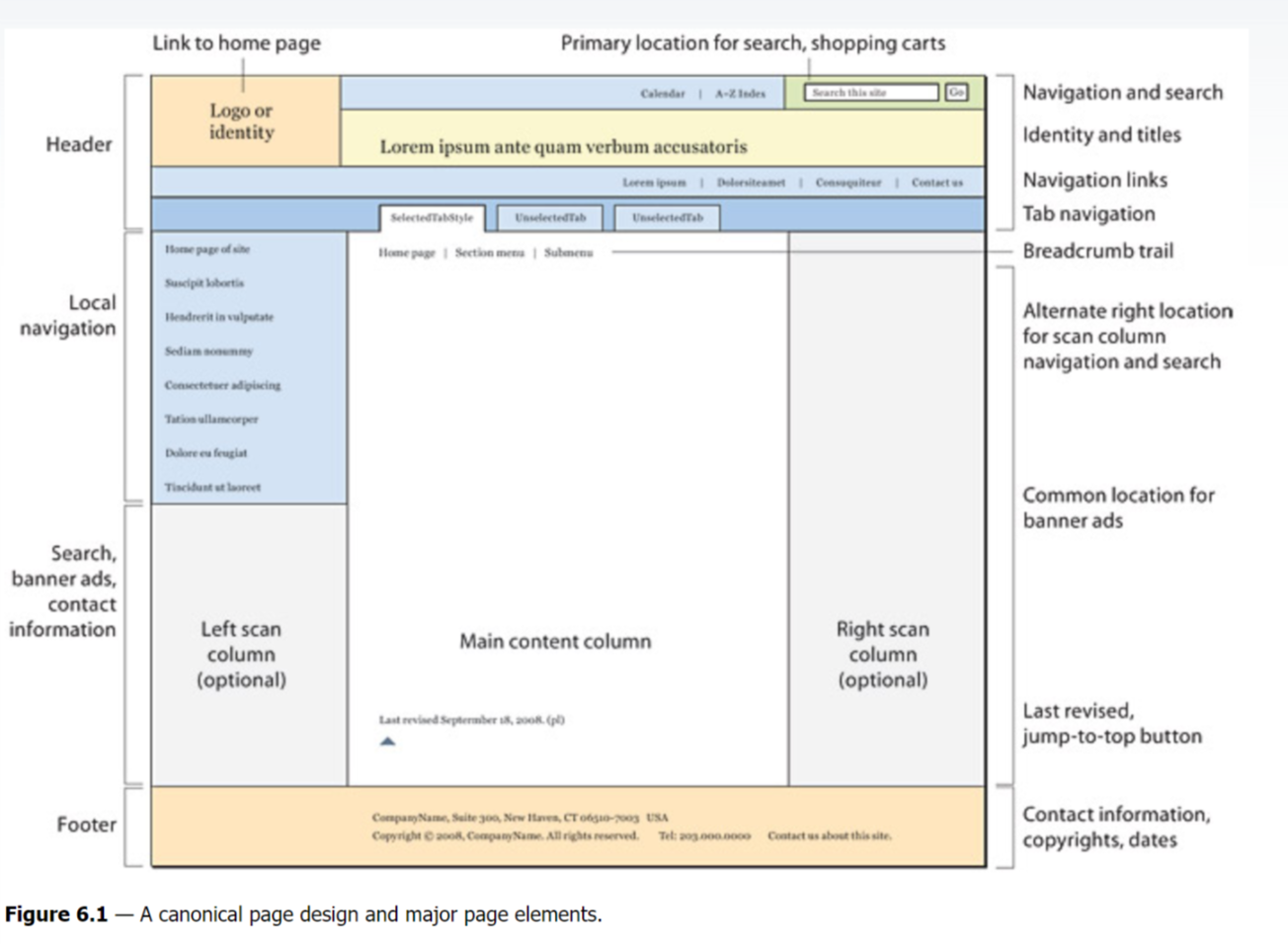
○ 태그의 종류

● 링크 ● 이미지 ● 목록 ● 제목

○ 레이아웃 태그



○ 레이아웃 구조 설계



CSS

○ style 적용

1. inline

- 태그 안에서 선언

2. internal

- head안에 선언

3. external

- 외부파일(.css)로 지정

○ 연산자와 우선순위 (casecading)

1. Inline > internal > external

2. 나중에 선언된 것, 구체적인 것은 먼저

3. id > class > element

○ Selector

- HTML의 요소들을 tag, id, class, html 태그 속성 등을 통해 쉽게 찾아주는 방법

○ font-size 단위

- 기본은 16px

- em 단위는 상속받은 값을 기준

ㄴ ex) 상위 노드 32px , 하위 노드 2em = 64px

○ Layout

● display (block, inline, inline-block)

1. block

- 엘리먼트가 아래로

2. inline

- 왼쪽으로 오른쪽

- 높이와 넓이 반영 X

3. inline-block

● position (static, absolute, relative, fixed)

1. static: 기본값, 순서대로

2. absolute

- 기준점에 따라 top/left/right/bottom로 배치

- 기준점 상위 엘리먼트(position)로 단계적으로 찾아감

3. relative: 자신이 위치해야 할 곳을 기준

4. fixed: 전체화면을 기준

● float(left, right)

◎ Servlet

- URL를 연결해주는 프로그램

- WAS에 설치되어 동작하는 어플리케이션

- 객체 생성은 1번만 함

○ 메서드

● doGet() 메서드

- Http GET 메서드가 호출될 때 서블릿에서 doGet 메서드를 호출

● service(request, response)

- 템플릿 메소드 패턴으로 구현

- 클라이언트의 요청이 GET일 경우에는 자신이 가지고 있는 doGet 호출

- 클라이언트의 요청이 POST일 경우에는 자신이 가지고 있는 doPost 호출

○ Servlet 생명주기

① WAS에서 서블릿 요청을 받으면 해당 서블릿이 메모리에 있는지 확인

② if 메모리에 X

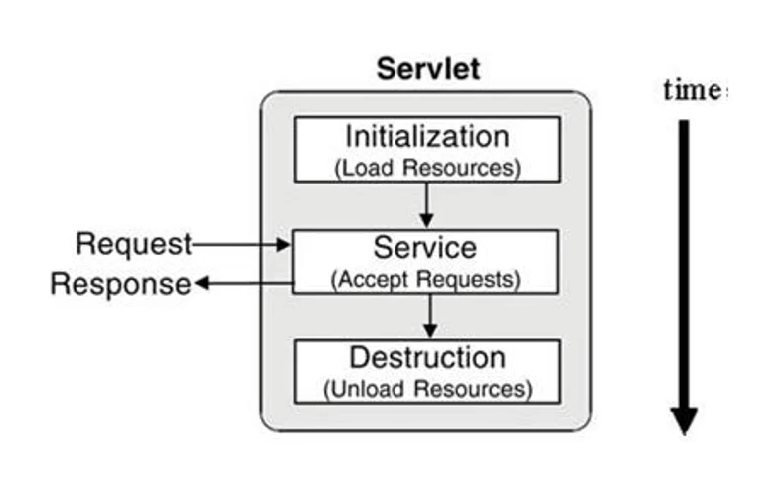
ㄴ 해당 서블릿 클래스를 메모리에 올리고 init() 실행

Else

ㄴ service() 메소드 실행

③ WAS가 종료되거나, 웹 어플리케이션이 새롭게 갱신되는 경우

Destroy() 메소드 실행



○ 요청과 응답

1. WAS는 웹브라우저로부터 Servlet 요청을 받으면

- 요청할 때 가지고 있는 정보를 HttpServletRequest 객체를

생성하여 저장

- 웹 브라우저에게 응답을 보낼 때 사용하기 위하여

HttpServletResponse 객체를 생성

- 생성된 HttpServletRequest, HttpServletResponse 객체를

서블릿에게 전달

● HttpServletRequest

- Http 프로토콜의 request 정보를 서블릿에게 전달하기 위한

목적으로 사용

- 헤더정보, 파라미터, 쿠키, URI, URL 등의 정보를 읽는

메소드를 가지고있음

- Body의 Stream을 읽어 들이는 메소드를 가지고있음

● HttpServletResponse

- WAS는 어떤 클라이언트가 요청을 보냈는지 암

ㄴ 해당 클라이언트에게 응답을 보내기위해 위 객체를

생성하여 서블릿에게 전달

- 서블릿은 해당 객체를 이용하여 contentType, 응답코드,

응답 메시지 등을 전송

자바 스크립트

○ 변수

- var, let, const로 선언

- 어떤 것을 사용하는 가에 따라 scope라는 변수 유효범위가 달라짐

//논리 연산자

var name = "crong";

var result = name || "codesquad";

console.log(result);

//삼항 연산자

var data = 11;

var result = (data>10)? "ok" : "fail";

console.log(result);

//함수

function printName(name){     // 함수 선언문

    return "my name : " + name;

  }

console.log(printName()); //name = undefined

console.log(printName("Jisu"));

console.log(printName("Jisu", "Kiho")); // Jisu만 출력

//함수 표현식

function printName(name){

    var inner = function(){

// result 와 위치가 바뀌면 result는 inner()를 인식 X but 함수 선언문으로 바뀌면 가능, 함수 표현식

      return "inner value";

    }

    var result = inner();

    console.log("name is  " + result);

}

printName();

/\*

호이스팅(hoisting)

function printName(){

    var result = inner();

    var inner = function(){}

}

위 코드에서 inner가 실행되는 순간 is not defined 오류가 아닌 function이 아니라고 나오는 이유는

function printName(){

    var inner; -> 호이스팅

    var result = inner(); // undefined

    inner = function(){}

} 로 함수 안에 필요한 변수들을 모두 끌어올려서 선언하기 때문 따라서 inner를 값이 할당되기 전에 실행되었으므로 undefined이 할당됨

\*/

//function의 반환값이 없다면 undefined를 반환 (void는 없음)

/\*

arguments

자바 스크립트의 함수는 선언한 파라미터보다 더 많은 인자를 보낼 수 있다.

이때 넘어온 인자를 arguments로 배열의 형태로 하나씩 접근이 가능

but argumnets는 배열 타입이 X

\*/

function a(){

    console.log(arguments);

  }

  a(1,2,3);

  //출력값 : [object Arguments] { 0: 1, 1: 2, 2: 3 }

  //arrow function (함수 표현식)

  function getName(name){

      reuturn "Kim " + name;

  }

  var getName = (name) => "Kim " + name; //arrow function

○ window 객체

//window 객체

function run(){

    console.log("run start");

      setTimeout(function() {

        var msg = "hello codesquad";

        console.log(msg);

        console.log("run ing");

      }, 1000); // ms 단위

      console.log("run end");

  } /\* start end ing 순으로 출력

  ->비동기 콜백( 스택이 다 비워지고 비동기는 마지막에 진행) = 지연시켜서 함수를 실행시키는 함수

  \*/

run();

○ DOM ( Document Object Model)

- 브라우저에서는 HTML코드를 DOM이라는 객체 형태의

모델로 저장

ㄴ 저장된 정보를 DOM Tree라고 함

○ Event

- 브라우저에서 발생

● 등록 방법

- addEventListener()

//이벤트 등록

var el = document.querySelector(".outside");

el.addEventListener("click", function(e){

    console.log("clicked",e);

    var target = e.target;

    console.log(target.className,target.nodeName);

});

○ Ajax (XMMLHTTP Request 통신)

- 브라우저 새로고침 없이 서버로부터 데이터를 받음

- 누르지 않은 탭의 컨텐츠를 동적으로 필요한 시점에 컨텐츠를

받아와서 표현

● JSON

- 표준적인 데이터 포맷을 결정하기 위해 JSON 포맷을 사용

{**"name2"**: 50, **"name3"**: "값3", **"name1"**: **true**}

● Ajax 실행 코드

//Ajax 실행 코드

   function reqListener(){

    console.log(this.reponseText);

}

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", "./json.txt");

oReq.send();

JSP(Java Server Page)

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%-- 페이지에 대한 선언 부분 --%>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
<%  
 int total = 0;  
 for (int i = 1; i <= 10; i++) {  
 total += i;  
 }  
%>  
<%-- <% 위 page에서 선언한 언어를 사용하는 부분 %> 스크립트릿 --%>  
1부터 10까지의 합 : <%=total %> <%-- <%= 응답결과로 주는 부분 %> 표현식 --%>  
  
</body>  
</html>

○ 실행 순서

① 브라우저가 웹서버에 JSP에 대한 요청 정보를 전달

② 브라우저가 요청한 JSP가 최초로 요청된 경우

1. JSP로 작성된 코드가 서블릿으로 코드 변환(java 파일 생성)
2. 서블릿 코드를 컴파일, 실행가능한 bytecode로 변환

(class파일 생성)

1. 서블릿 클래스를 로딩하고 인스턴스 샐행

③ 서블릿이 실행되어 요청을 처리하고 응답 정보를 생성

○ 선언식

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
hello  
<%  
 System.*out*.println("jspService()");  
%> <%--실행할 때마다 호출--%>  
  
<%!  
 public void jspInit(){  
 System.*out*.println("jspInit()!!");  
 }  
%>  
<%--  
 <%! 선언식 메서드를 선언하거나 필드를 선언할 때 사용 %>  
--%>  
  
<%!  
 public void jspDestroy(){  
 System.*out*.println("jspDestroy()");  
 }  
%>  
</body>  
</html>

○ 문법

●스크립트 요소

1) 선언문 (Declaration)

<%! %> 멤버 변수(전역) 선언 및 메소드 선언에 사용

2) 스크립트릿 (Scriptlet)

<% %> 프로그래밍 코드 기술에 사용

- 선언된 변수는 지역변수

3) 표현식 (Expresstion)

<%= %> 화면에 출력할 내용 기술에 사용

- 웹 브라우저에 출력할 부분

4) 주석

- HTML, Java, JSP 주석을 사용가능

1) HTML

<!-- -->

- 웹 페이지 상 나타나지 않지만 소스 보기할 때 보임

2) JSP 3) Java

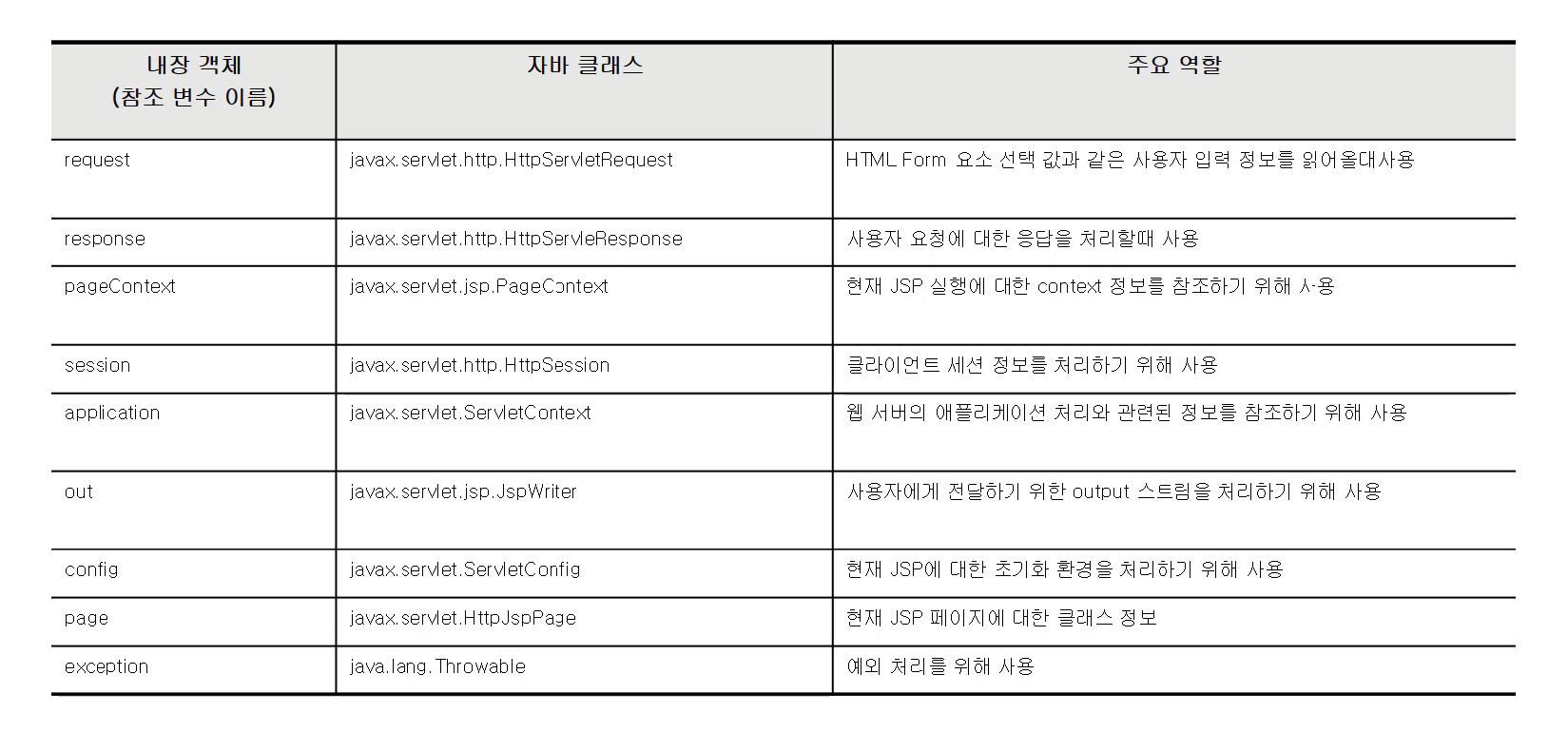
<%-- --%> // , /\* \*/

<body>  
id : <%= getId()%> <%-- 표현식 --%>  
<%!  
 String id = "u001"; // 멤버 변수 선언  
 public String getId(){ //메소드 선언  
 return id;  
 }  
 // 선언문  
%>  
  
<%  
 //스크립트릿  
 for (int i = 1; i <= 5; i++) {  
  
%>  
<h<%=i%>>아름다운 한글</h<%=i%>>  
<%  
 }  
 //끊어 쓰기 가능  
%>  
</body>

○ JSP 내장 객체

- JSP를 실행하면 서블릿 소스가 생성되고 실행

Ex) response, request, application, session, out 등등



<body>  
<%  
 StringBuffer url = request.getRequestURL(); //내장객체  
  
 System.*out*.println("url: " + url.toString());  
 System.*out*.println("<br>");  
%>  
</body>

리다이렉트(redirect)

- http 프로토콜로 정해진 규칙

- 서버는 클라이언트로부터 요청을 받은 후 , 클라이언트에게

특정 URL로 이동하라고 요청

- 서블릿이나 JSP에서 redirect하기위해

HttpservletResponse.sendRedirect()를 사용

○ 쓰는 이유

- 신뢰할 수 있는 HTTP 트랜잭션의 수행

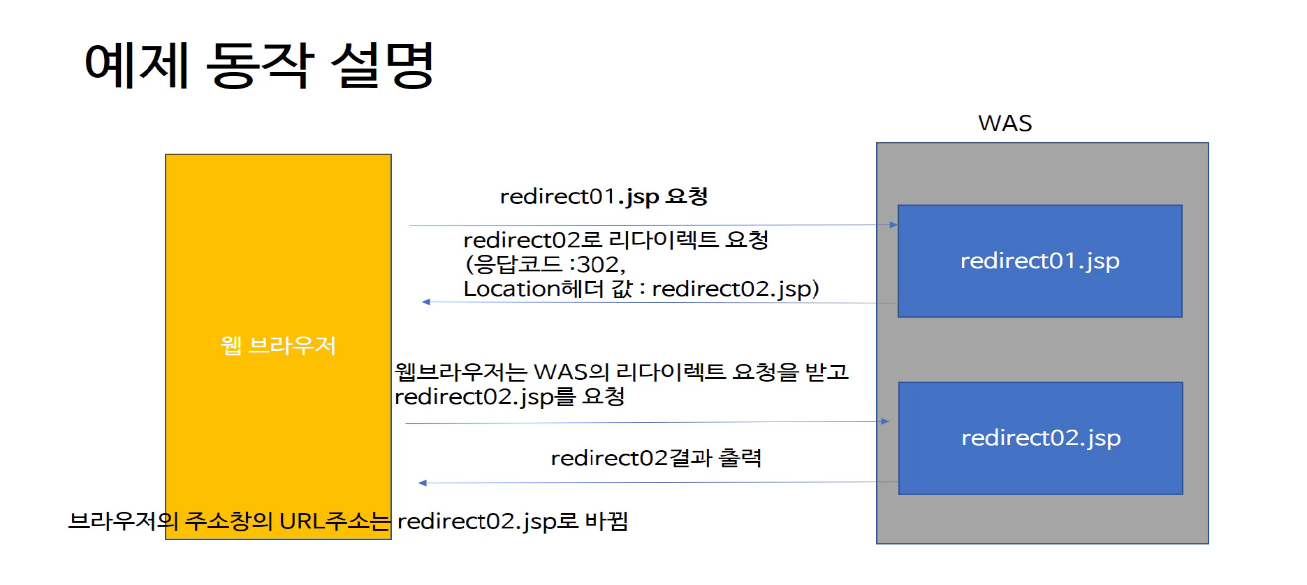
- 지연 최소화

- 네트워크 대역폭 절약

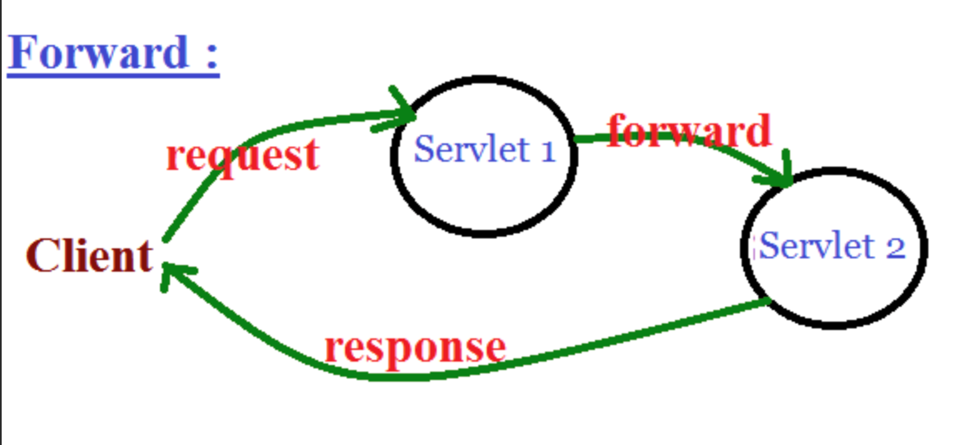
ㄴ 한곳에서 실패하여도 다른 곳을 이용, 신뢰성이 개선

ㄴ 클라이언트가 보다 가까운 리소스에 접근, 응답시간 줄어듬

ㄴ 목적지 서버가 분산, 네트워크 혼잡도 줄어듬



Forward



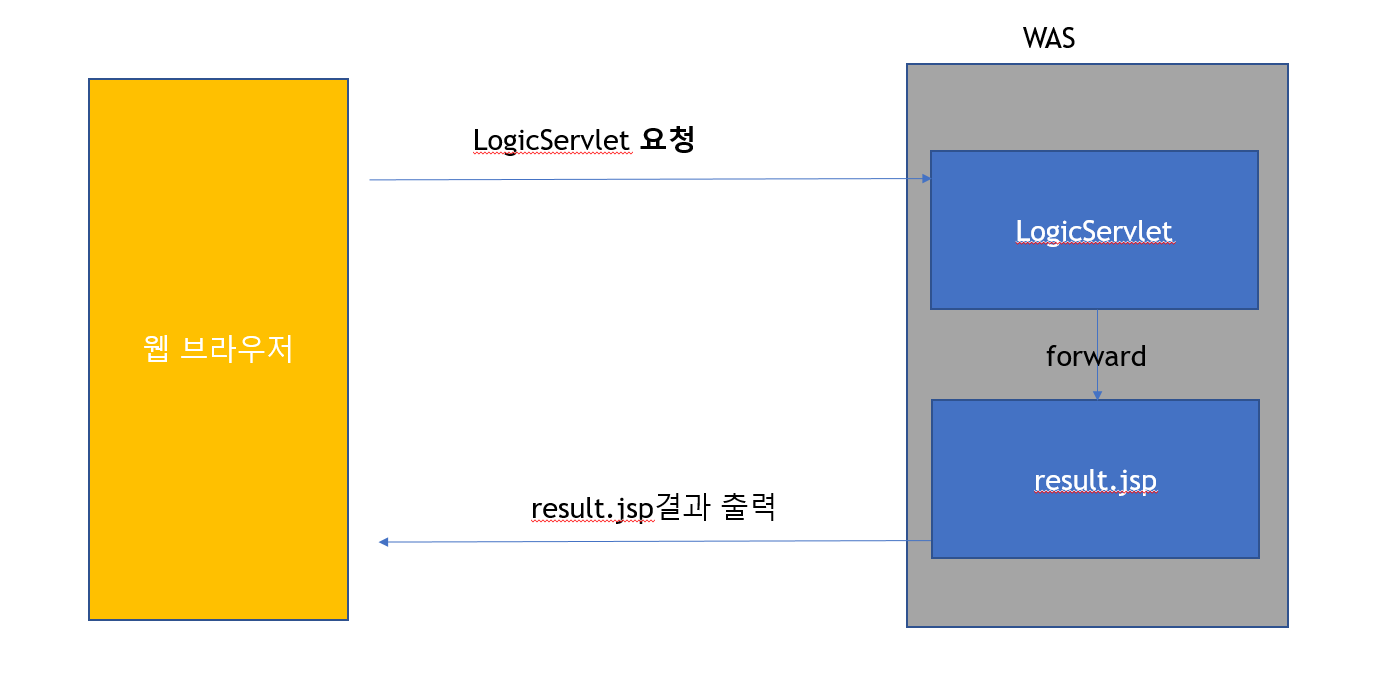
- request, response 객체는 요청과 응답이 끝날 때까지 존재

- redirect와 달리 URL이 바뀌지 않음

@Override  
protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 int diceValue = (int) (Math.*random*() \* 6) + 1;  
 req.setAttribute("dice", diceValue); // 값 맡김  
  
 RequestDispatcher requestDispatcher = req.getRequestDispatcher("/next"); //다음에 넘어갈 URL;  
 requestDispatcher.forward(req,resp);   
}

위는 서블릿1, 밑 서블릿 2

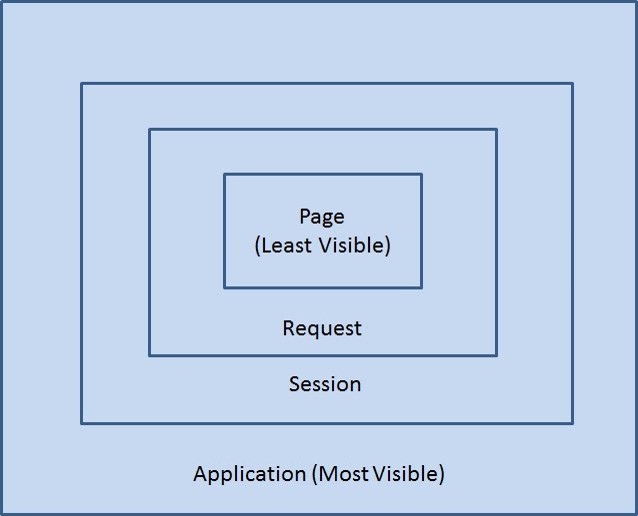
@Override  
protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 resp.setContentType("text/html;charset=utf-8");  
 PrintWriter out = resp.getWriter();  
  
 out.println("<html>");  
 out.println("<head><title>form</title></head>");  
 out.println("<body>");  
  
 int dice = (Integer) req.getAttribute("dice"); //값 받기  
  
 out.println("dice: " + dice);  
 for (int i = 0; i < dice; i++) {  
 out.println("<br>hello");  
 }  
  
 out.println("</body>");  
 out.println("</html>");  
}



Scope

○ 종류

- Application > Session > Request > Page



○ Page scope

- PageContext 추상 클래스 사용

◈ forward가 될 경우 해당 Page scope에 지정된 변수 사용 X

(지역 변수 개념)

○ Request scope

◈ http 요청을 WAS가 받아서 웹 브라우저에게 응답할 때까지

변수값을 유지하고자 할 때 사용

- HttpServletResponse 객체를 사용

- 값 저장 request.setAttribute()

- 값 받기 request.getAttribute()

○ Session scope

◈ 웹 브라우저 별로 변수를 관리할 때 사용

◈ 웹 브라우저 탭 간 세션 정보가 공유

- HttpSession 인터페이스를 구현한 객체를 사용

- JSP에서는 session 내장 변수 사용

- 서블릿에서는 HttpServletRequest.getSession()를 통해

session 객체를 얻음

- session.setAttribute() or getAttribute()로 값 저장 및 로드

- 예로 장바구니처럼 사용자 별로 유지 되어야 할 정보

○ Application scope

- 웹 어플리케이션이 시작되고 종료될 때까지 사용

- ServletContext 인터페이스를 구현한 객체를 사용

- jsp에서는 application

- servlet에서는 getServletContext

- 웹 어플리케이션 하나당 하나의 application 객체가 사용

◈ 모든 클라이언트가 공통으로 사용해야할 값들이 있을 때 사용

EL(Expression Language) 표현 언어

- 값을 표현하는데 사용되는 스크립트 언어

- JSP 보완

${표현식}

○ 객체 접근 규칙

${<표현>,<표현2>}

- 둘 중 1개가 null이면 null 반환, 정수가 아니면 오류 발생

○ 표현 언어 비활성화 – 2.3이전에 버전은 기본값이 false

<%@page isELIgnored = “true” %>

JSTL

- JSP 페이지에서 조건문 처리, 반복문 처리 등을

HTML Tag 형태로 작성할 수 있게 도와줌

- html과 자바 코드가 섞여 있어서 유지보수가 어려웠음

ㄴ JSTL 등장한 계기

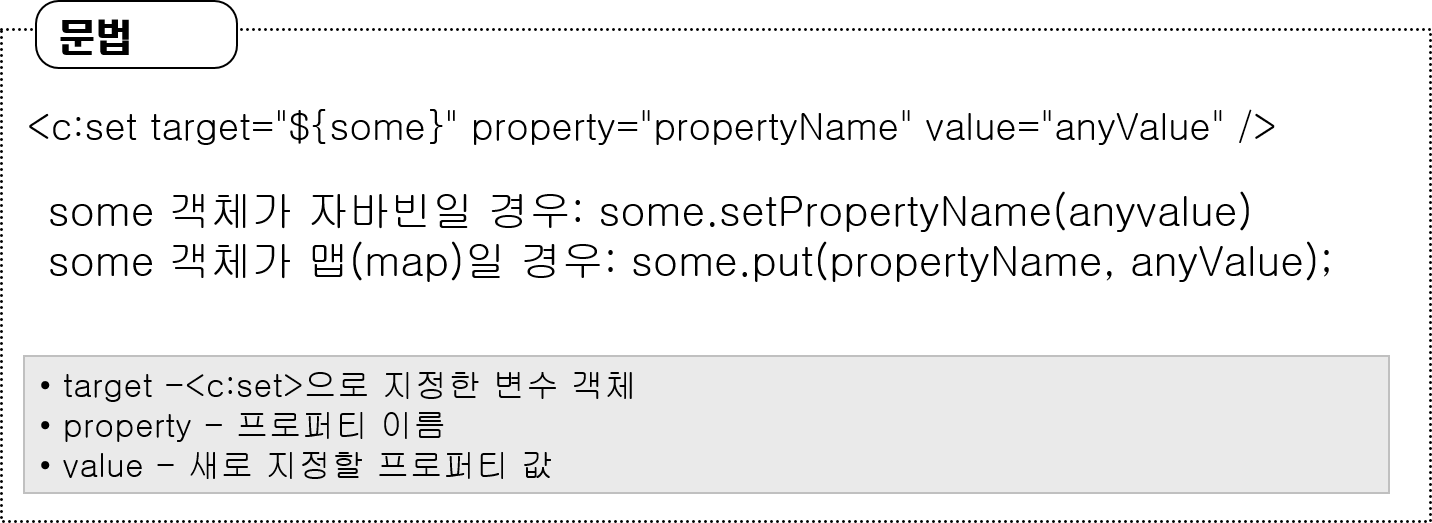


○ 코어 태그

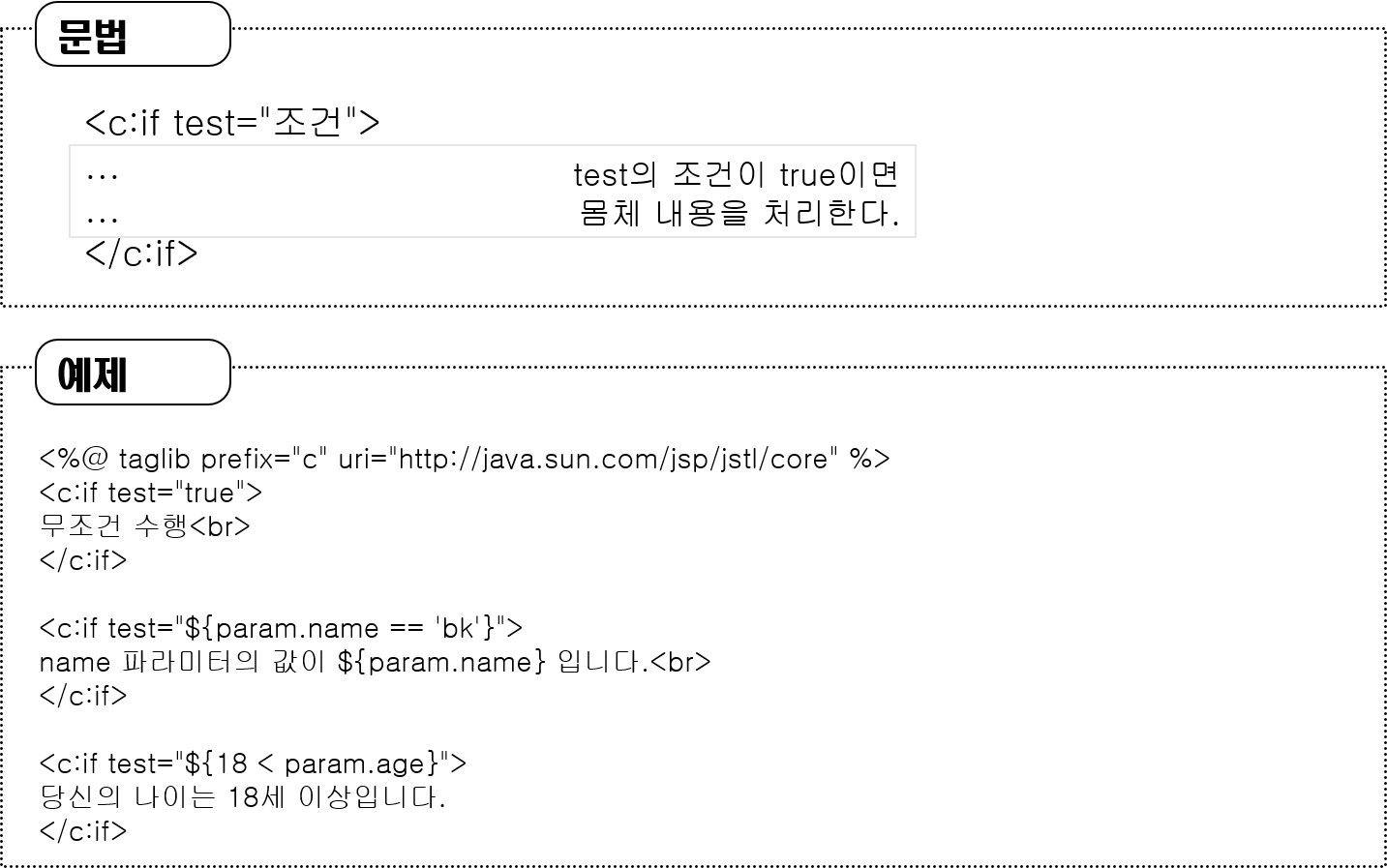


<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<**c:set** var="value1" scope="request" value="kang"></**c:set**>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
성 : ${value1 }<br>  
<**c:remove** var="value1" scope="request"/>  
성 : ${value1 }<br>  
  
</body>  
</html>

○ 변수 지원 – 프로퍼티(get,set), 맵 처리



○ 흐름 제어 (if문)



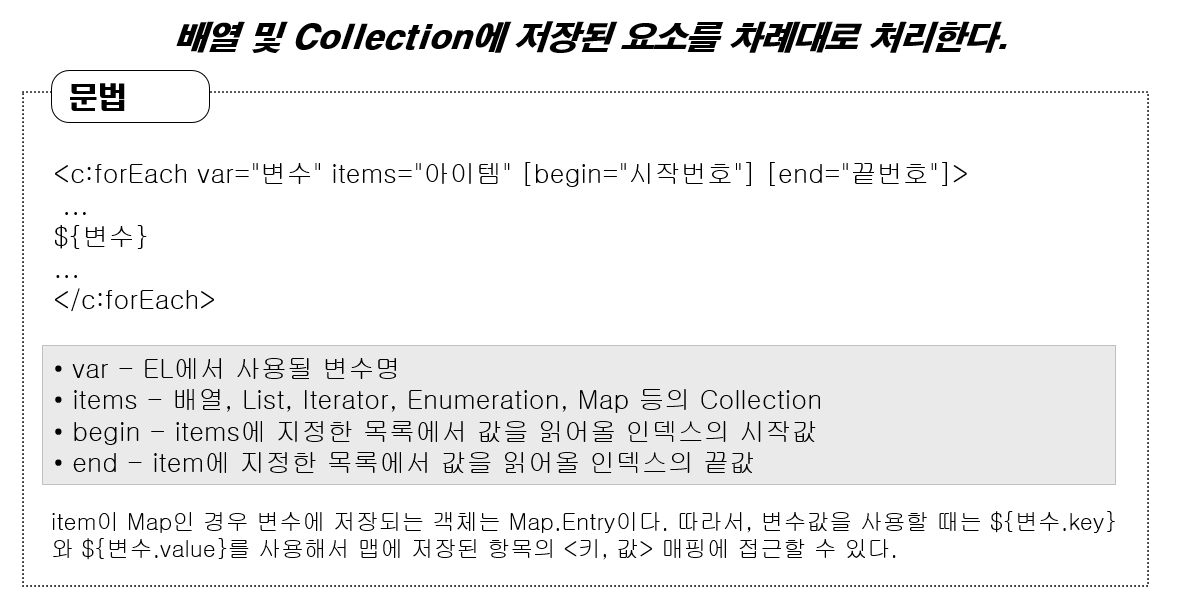
<**c:set** var="n" scope="request" value="10"/>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
<**c:if** test="${n == 0}">  
 n과 0은 같습니다.  
</**c:if**>  
  
<**c:if** test="${n == 10}">  
 n과 10은 같습니다.  
</**c:if**>

○ 흐름제어 choose



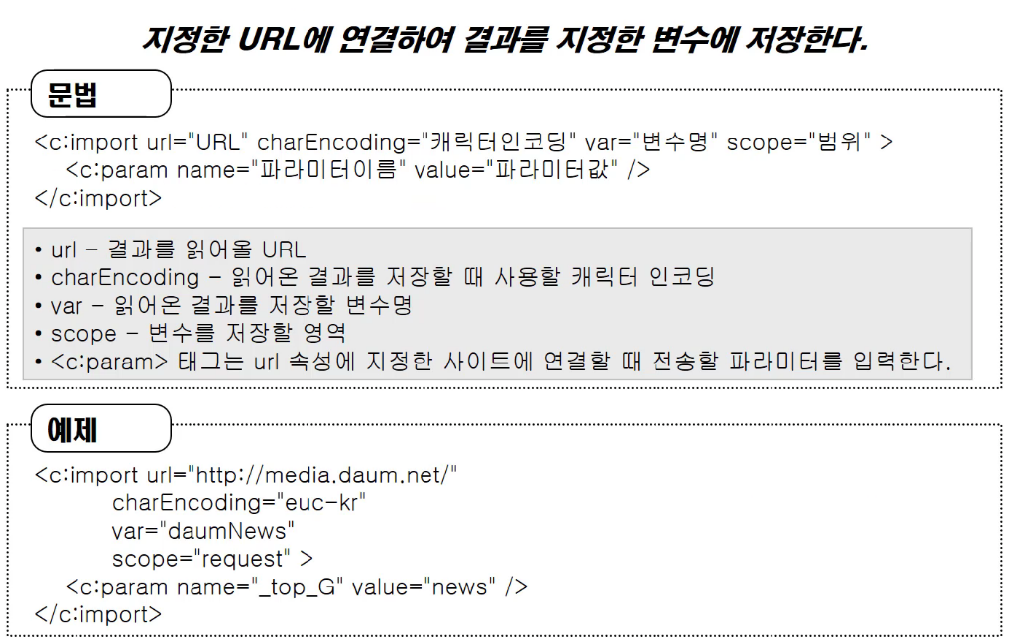
<**c:choose**>  
 <**c:when** test="${score>=90}">  
 A학점입니다.  
 </**c:when**>  
 <**c:when** test="${score>=80}">  
 B학점입니다.  
 </**c:when**>  
 <**c:when** test="${score>=70}">  
 C학점입니다.  
 </**c:when**>  
 <**c:when** test="${score>=60}">  
 D학점입니다.  
 </**c:when**>  
 <**c:otherwise**>  
 F학점입니다.  
 </**c:otherwise**>  
</**c:choose**>

○ 흐름제어 – forEach



<**c:forEach** items="${list}" var="item" begin="1">  
 ${item} <br>  
</**c:forEach**>

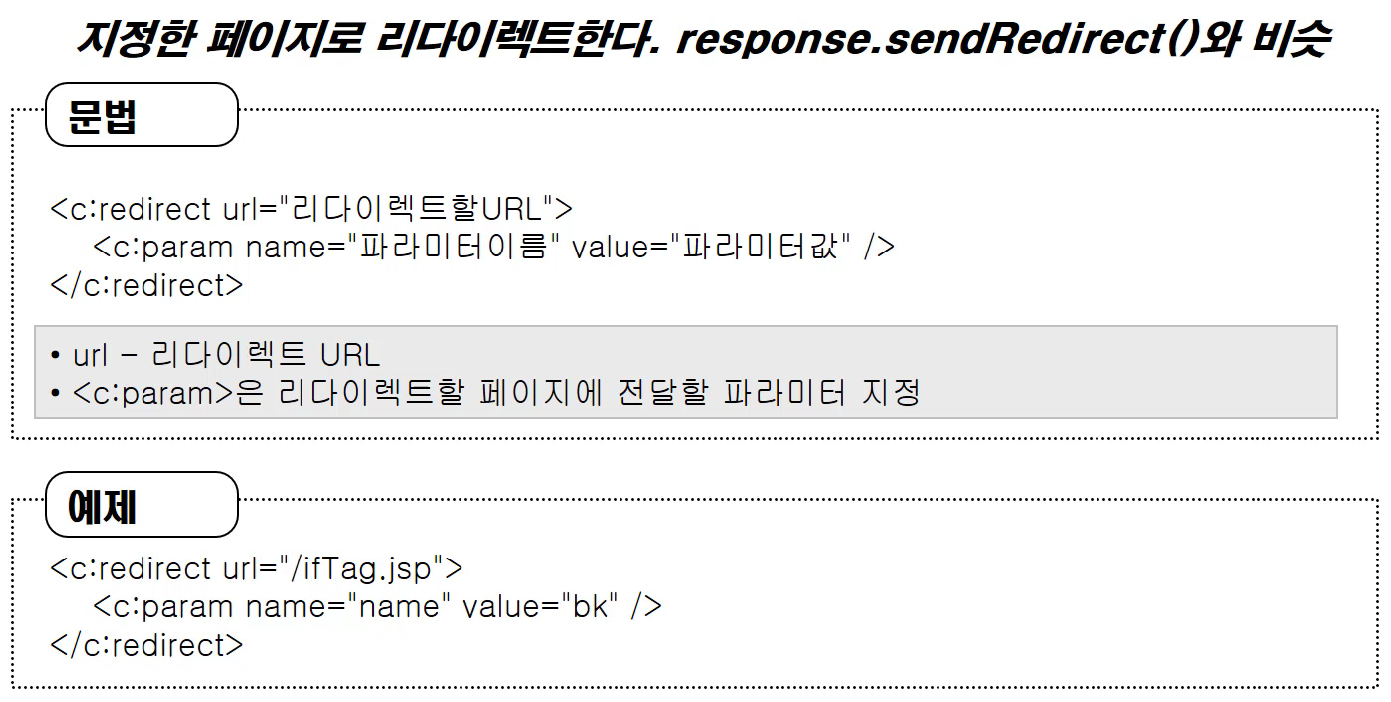
○ 흐름제어 – import



<**c:import** url="jstlValue.jsp" var="urlValue" scope="request"></**c:import**>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
${urlValue}

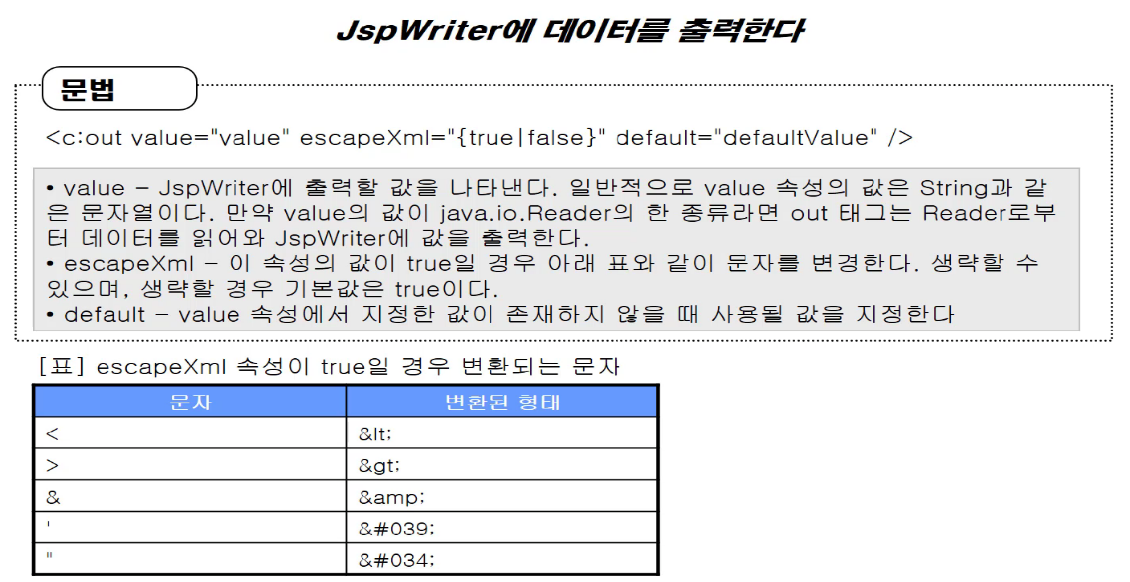
○ 흐름제어 – redirect

- response.sendRedirect()와 비슷



<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<**c:redirect** url="jstl05.jsp"></**c:redirect**>

○ 기타 태그 – out



<**c:set** var="t" value="<script type='text/javascript'>alert(1);</script>"></**c:set**>  
  
<**c:out** value="${t}" escapeXml="false"/>  
<%-- escapeXml="true" 일때는 문자열로 인식하기때문에  
<script type='text/javascript'>alert(1);</script>가 출력됨  
--%>

SQL(Structured Query Language)

- 데이터를 보다 쉽게 검색하고 추가, 삭제, 수정 같은 조작을

할 수 있도록 고안된 컴퓨터 언어

- 관계형 데이터베이스에서 데이터를 조작하고 쿼리하는

표준언어

○ DML (Data Manipulation L)

- 데이터를 조작하기위해 사용

- INSERT, UPDATE, DELETE, SELECT등

○ DDL (Data Definition L)

- 데이터베이스의 스키마를 정의, 조작하기 위해 사용

- CREATE, DROP, ALTER 등

○ DCL (Data Control L)

- 데이터를 제어하는 언어

- 권한을 관리하고, 데이터의 보안, 무결성들을 정의

- GRANT, REVBOKE 등

○ Database

● 계정 접속

mysql -u root -p

● DB 생성

create database dbName;

● 사용자 생성

create user 'connectuser'@'localhost' identified by 'connect123!@#';

create user 'connectuser'@'%' identified by 'connect123!@#';

grant all privileges on connectdb.\* to 'connectuser'@'localhost';

grant all privileges on connectdb.\* to 'connectuser'@'%';

flush privileges; /DBMS에게 적용하라는 명령

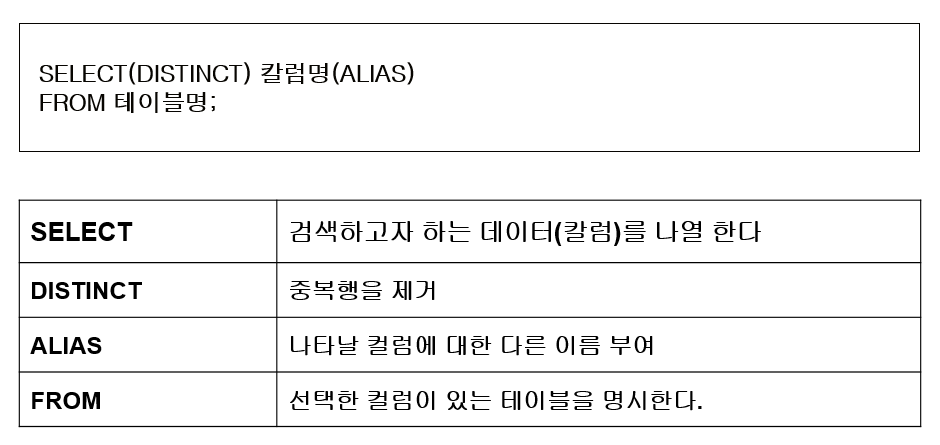
● 데이터베이스 전환

use dbName;

● 테이블 구조

desc tableName;

○ Select문

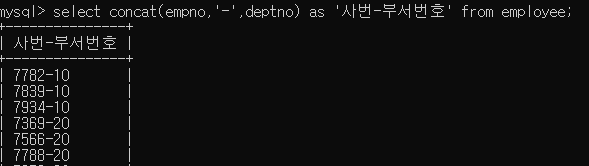


1) ALIAS 부여

select empno as 사번 from employee;

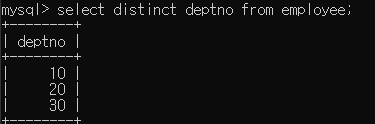
2) 컬럼 합성(Concatenation)

- concat



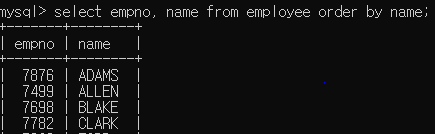
3) 중복 행 제거

- DISTINCT



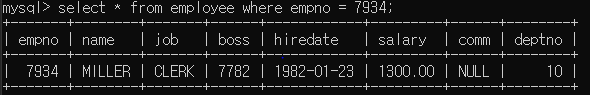
4)정렬

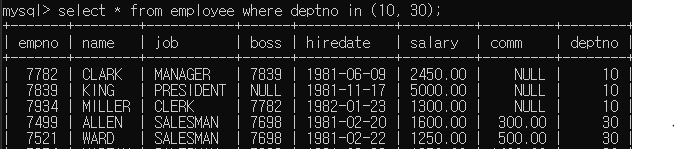
- order by columnName or Num ASC(오름, default) , DESC(내림)



5) 특정 행 검색

- where , in 키워드



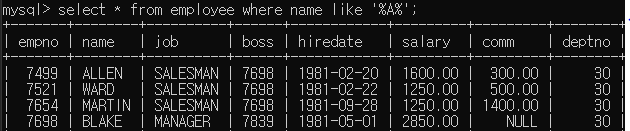


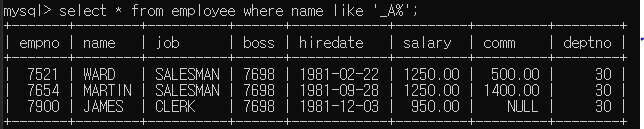
- like

ㄴ 와일드 카드를 사용하여 특정문자를 포함한 값에 대한 처리

ㄴ %는 0에서부터 여러 개의 문자열을 나타냄

ㄴ \_는 단 하나의 문자





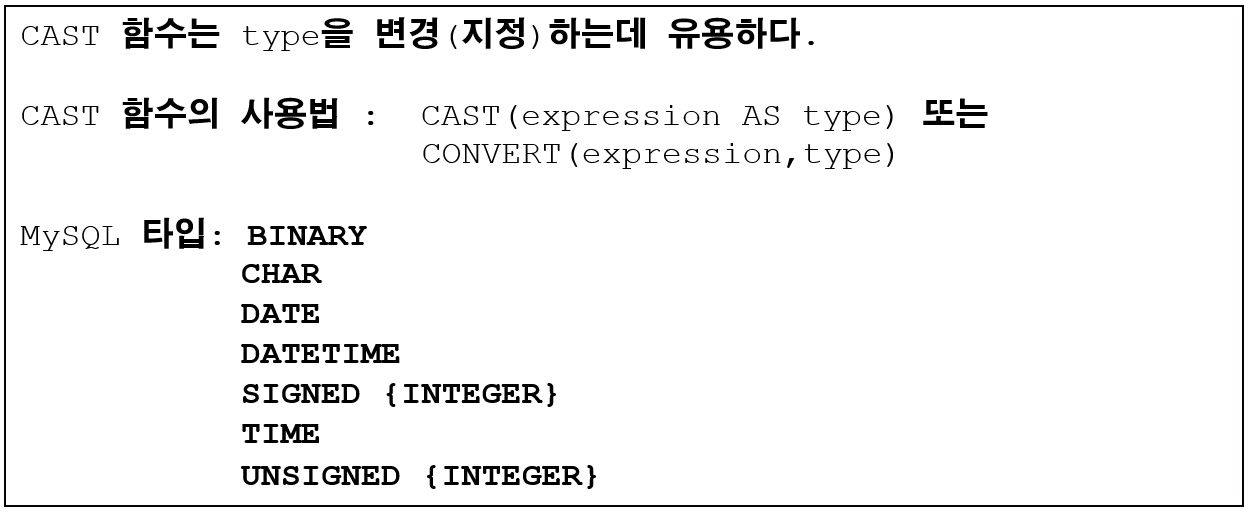
6) 함수

- ucase, upper 대문자 - lcase, lower 소문자

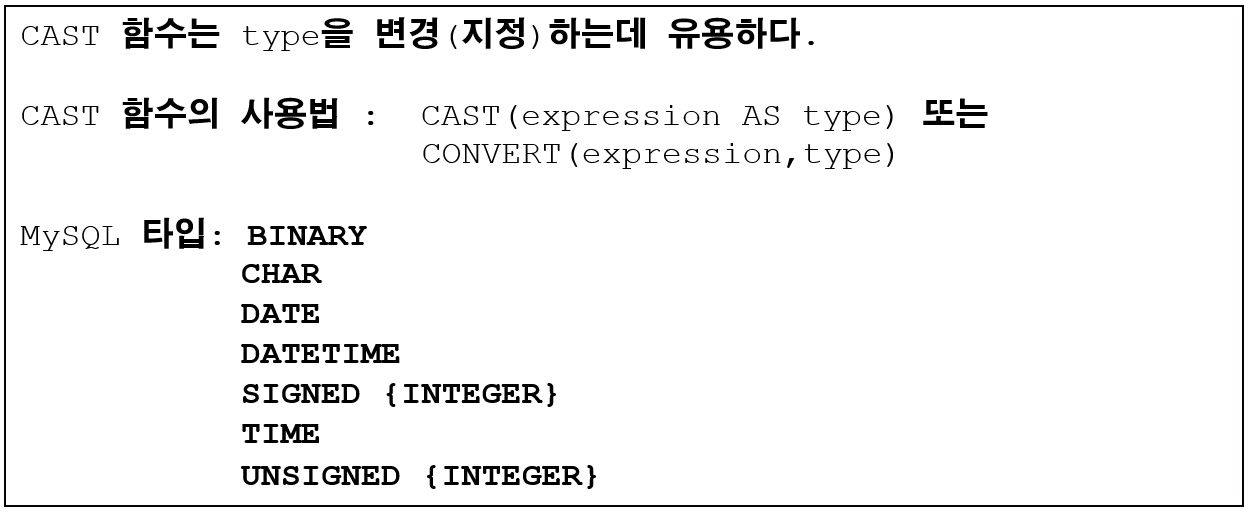
- substring 문자 끊기 - lpad, rpad 공백 존재 시 문자 채움

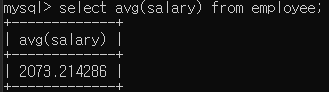
- trim 공백 제거

7) CAST 형변환



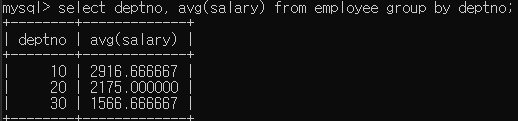
8) 그룹함수





9) group by

- 어떠한 값에 대한 그룹 함수 결과값 출력



○ INSERT 문

INSERT INTO 테이블명(필드1, 필드2, 필드3 ……)

VALUES (필드1의 값, 필드 2의 값 …….)

- 필드명은 생략은 가능 but 모든 필드 값을 반드시 입력

○ UPDATE 문

UPDATE 테이블명

SET 필드1 = 필드1의 값, 필드2 = 필드2의 값…….

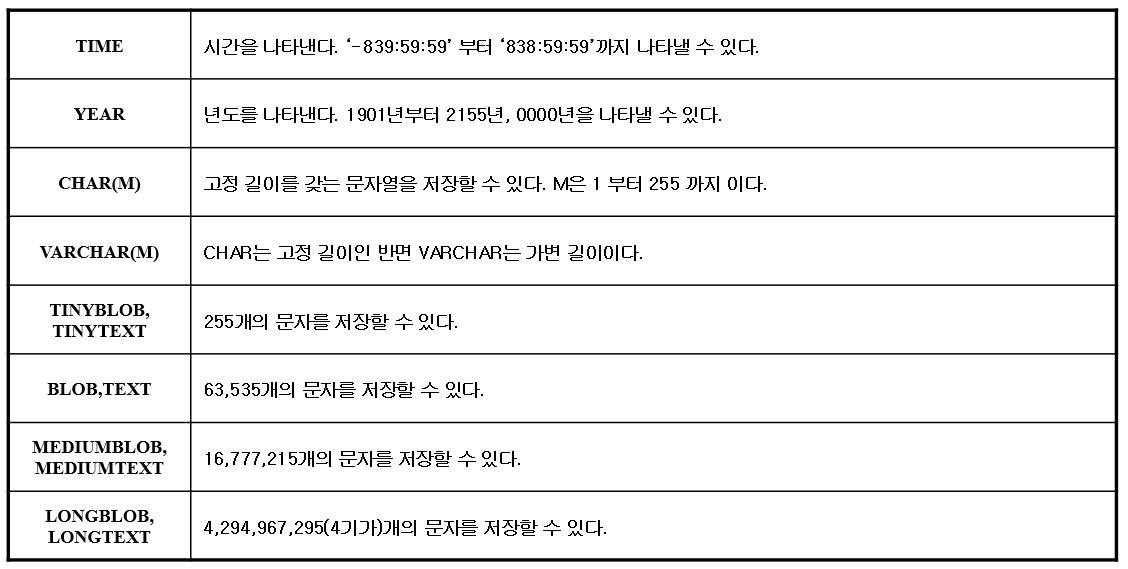
WHERE 조건식

○ DELETE

DELETE FROM 테이블명 WHERE 조건식

○ MySQL 데이터 타입





○ CREATE

create table 테이블명(

필드명1 타입 NOTNULL ,

필드명2 타입 NULL AUTO\_INCREMENT,

...........

PRIMARY KEY(필드명) );

○ alter 테이블 수정

- 필드 추가

alter table 테이블명 add 필드명 타입 옵션;

- 필드 삭제

alter table 테이블명 drop 필드명;

- 필드 수정

alter table 테이블명 change 필드명 새로운필드타입 옵션;

- 테이블 이름 변경

alter table 테이블명 rename 변경이름

Maven

- 반복적으로 진행해왔던 작업들을 지원하기 위해 등장한 도구

- 빌드(Build), 패키징, 문서화, 테스트와 테스트 리포팅, git, 의존성 관리,

svn등 과 같은 형상관리 서버와 연동, 배포 등의 작업을 지원

- pom.xml 파일로 관리

○ 장점

- 편리한 의존성 라이브러리 관리

ㄴ 설정 파일에 적어줌으로써 다운로드 받지 않아도 사용 가능

○ pom.xml

1) project : pom.xml 파일의 최상위 루트 엘리먼트

2) modelVersion : POM model의 버전

3) groupId : 프로젝트를 생성하는 조직의 고유 아이디,

보통 도메인 이름 거꾸로

4) artifactId : 해당 프로젝트에 의하여 생성되는 artifact의 고유 아이디

5) version : 프로젝트의 현재 버전

6) name : 프로젝트의 이름

7) url : 프로젝트 사이트가 있다면 사이트 URL을 등록 가능

○ Scope

1) compile

- 컴파일 할 때 필요.

- 테스트 및 런타임에도 클래스 패스에 포함, 기본값

2) runtime

- 런타임에 필요, JDBC 드라이버

- 컴파일에는 필요 X, 실행 시 에 필요한 경우

3) provided

- 컴파일 시에 필요하지만, 실제 런타임에는 컨테이너 같은 것에서

제공되는 모듈

- servlet, jsp api 등

4) test

- 테스트 코드를 컴파일 할 때 필요

JDBC(Java Database Connectivity)

- 자바를 이용한 데이터베이스 접속과 SQL 문장의 실행, 그리고

실행 결과로 얻어진 데이터의 핸들링을 제공하는 방법과 절차

- 자바 프로그램내에서 SQL문을 실행하기위한 API

○ 사용 단계별 정리

1) import.java.sql.\*;

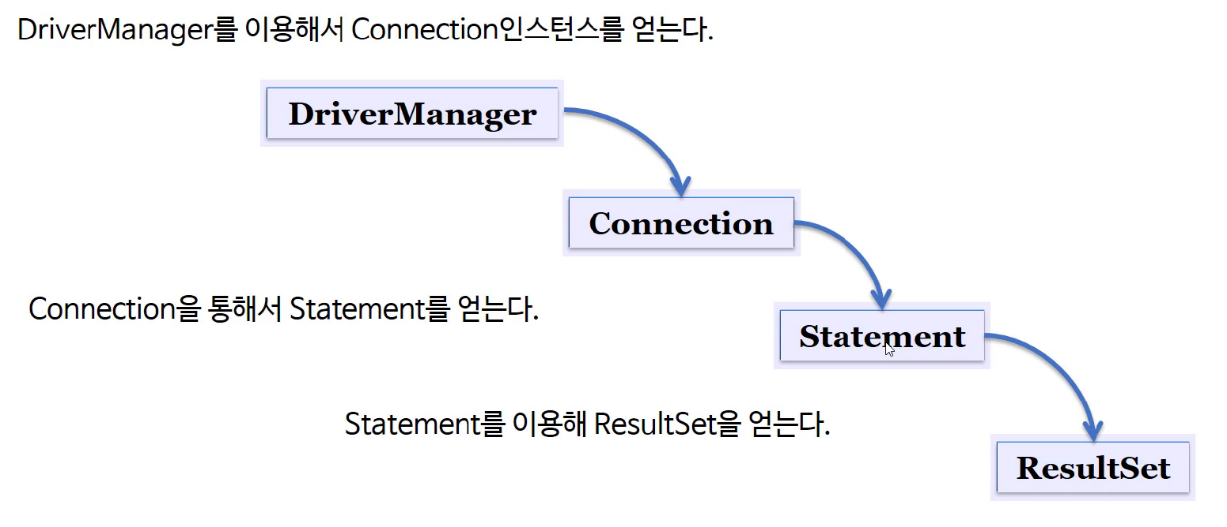
2) 드라이버를 로드

3) Connection 객체를 생성

4) Statement 객체 생성 및 질의 수행 (쿼리문 생성)

5) SQL문에 결과물이 있다면 ResultSet 객체를 생성

6) 모든 객체를 닫음



+

API (Application Programming Interface)

- 응용 프로그램에서 사용할 수 있도록, 운영체제나 프로그래밍 언어가

제공하는 기능을 제어할 수 있게 만든 인터페이스

- 주로 파일 제어, 창 제어, 화상 처리, 문자 제어 등 인터페이스 제공

○ REST API (Representational State Transfer)

- REST형식의 API

- 핵심 컨텐츠 및 기능을 외부 사이트에서 활용할 수 있도록

제공하는 인터페이스

● 규칙

- client-server - stateless

- cache - uniform interface

- layered system - code-on-demand(optional)

uniform interface

- 리소스가 URI로 식별

- 리소스를 생성, 수정, 추가할 때 HTTP메시지로 표현 및 전송

- 메시지는 스스로 설명할 수 있어야함

- 어플리케이션의 상태는 Hyperlink를 이용해 전이

WebAPI

● 규칙

- URI는 정보의 자원을 표현

- 자원에 대한 행위는 HTTP Method로 표현

● HTTP method

