

# EL과 JSTL

동의과학대학교 컴퓨터정보과  
김진숙

# 이번 장에서 공부할 것

- EL과 JSTL을 이용하여 HTML 부분에서 자바 코드를 없애는 방법
- 학습내용
  - EL
  - 포워드와 리다이렉트
  - 자바 빈과 EL
  - 게시판에 EL 적용
  - JSTL

# EL(Expression Language)

- 자바 코드가 들어가는 표현식(출력)을 편리하게 사용하기 위해 JSP 2.0 스펙에 추가된 개념으로 JSTL 1.0 에서 소개됨
- EL은 JSP의 표현식(expression)을 대신하여, 좀 더 알아보기 편한 표현으로 바꾸어 쓸 수 있도록 만들어진 것
- EL과 JSTL로 MVC 모델의 **View**에서 자바 코드 최소화 사용
- 사용 설정
  - 페이지 지시자의 **isELIgnored** 속성이 false로 설정되어야 함
  - isELIgnored의 기본값이 false

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8" isELIgnored="false"%>
```

# EL(Expression Language)

- 특징

- 브라우저에 출력하기 위해 만들어진 언어
- 'null'은 출력하지 않음
- 문자열은 " "(쌍따옴표) 또는 ' '(홀따옴표) 모두 사용 가능
- 데이터가 **미리 저장소에 확보(\*\*\*)**되어 있어야 출력 가능
- 연산자 사용 가능
- 제어문(조건문, 반복문)은 없음 → JSTL과 함께 사용
- 스크립트릿에서 만들어진 변수를 직접 출력할 수 없음
  - scope객체(pageContext, request, session, application)를 사용하면 가능

	형식	예
세션에 값 저장	session.setAttribute("세션_속성_이름", "값");	session.setAttribute("userId", "lee");
세션 속성 값 읽어서 출력	<%=session.getAttribute("세션_속성_이름")%>	<%=session.getAttribute("userId")%>
<b>EL을 사용한 읽기</b>	<b>\${출력할_값}</b>	<b>\${userId}</b>

# EL(Expression Language)

- 표현 언어의 형식

- 표기법 : **`${표현식}`**
- 기본적으로 변수명, 또는 '객체명.멤버변수명'구조

`${member.no}`

↑      ↑  
객체명    속성

`${member["no"]}`

↑      ↑  
객체명    속성(작은 따옴표도 사용 : 'no')

변수명에 특수문자가 있을 경우, 두 번째 표기법만 사용 가능

- 표현언어에는 자바 객체, 배열, 숫자, 문자열, boolean, null 같은 상수 값도 있음
- 기본적인 연산 표현 가능

```
request.setAttribute("cnt", 12)
```

```
request.getAttribute("cnt")
```



```
${cnt} 또는 ${requestScope.cnt}
```

java code

EL

JSP 표현식  
사용

```
<H2>
```

```
<jsp:useBean id="test" class="TestBean" />
```

```
<%=test.getName() %> 혹은 <jsp:getProperty name="test" property="name" />
```

```
</H2>
```



```
<H2>
```

```
${test.name}
```

```
</H2>
```

EL 사용

# EL(Expression Language)

- 표현 언어는 **객체가 생성**되어 전달된다는 것을 **전제**함
- 사용 시점에 객체 선언할 필요 없음

`${객체.속성}`

`${member.id}` 또는 `${member["id"]}` → member 객체의 getId() 메서드 호출

`${row[0]}` → row라는 이름의 컬렉션 객체의 첫 번째 값

- 속성명은 private 멤버 변수가 아니라, **getter/setter**를 보고 결정된다.
- 메소드 이름에서 get/set을 떼고, 첫 글자를 소문자로 바꾼 것이 속성명

# EL에서 사용할 수 있는 기본 내장 객체

구분	객체	설명	코드
스코프	pageScope	JSP의 page와 같은 기능을 하고 page 영역에 바인딩된 객체를 참조	<code>\${pageScope.객체명.변수명}</code>
	<b>requestScope</b>	JSP의 request와 같은 기능을 하고 request 영역에 바인딩된 객체를 참조	<code>\${requestScope.객체명.변수명}</code>
	<b>sessionScope</b>	JSP의 session와 같은 기능을 하고 session 영역에 바인딩된 객체를 참조	<code>\${sessionScope.객체명.변수명}</code>
	<b>applicationScope</b>	JSP의 application와 같은 기능을 하고 application 영역에 바인딩된 객체를 참조	<code>\${applicationScope.객체명.변수명}</code>
요청 매개변수	<b>param</b>	request.getParameter()를 호출한 것과 같으며 한 개의 요청 매개변수 처리	<code>\${param.매개변수명}</code>
	<b>paramValues</b>	request.getParameterValues()를 호출한 것과 같으며 여러 개의 요청 매개변수 처리	<code>\${paramValues.매개변수명}</code>
헤더값	header	request.getHeader()를 호출한 것과 같으며 단일값 반환	<code>\${header.헤더명}</code>
	headerValues	request.getHeader()를 호출한 것과 같으며 배열로 반환	<code>\${headerValues.헤더명}</code>
쿠키값	<b>cookie</b>	쿠키 이름의 값을 반환	<code>\${cookie.쿠키명}</code>
초기 매개변수	initParam	컨텍스트 초기화 매개변수 저장 map객체	<code>\${initParam.매개변수명}</code>
JSP 내용	pageContext	pageContext(현 JSP 페이지 지칭) 객체를 참조할 때 사용	<code>\${pageContext.request.requestURI}</code>



메소드	설명
String <b>getProtocol()</b>	웹 서버로 요청 시, 사용 중인 프로토콜을 리턴
String <b>getServerName()</b>	웹 서버로 요청 시, 서버의 도메인 이름을 리턴
String <b>getMethod()</b>	웹 서버로 요청 시, 요청에 사용된 요청 방식(GET, POST, PUT등)을 리턴
String <b>getQueryString()</b>	웹 서버로 요청 시, 요청에 사용된 QueryString을 리턴
String <b>getRequestURI()</b>	웹 서버로 요청 시, 요청에 사용된 URL로부터 URI값을 리턴
String <b>getRemoteHost()</b>	웹 서버로 정보를 요청한 웹 브라우저의 호스트 이름을 리턴
String <b>getRemoteAddr()</b>	웹 서버로 정보를 요청한 웹 브라우저의 IP주소를 리턴
int <b>getServerPort()</b>	웹 서버로 요청 시, 서버의 Port번호를 리턴
String <b>getContextPath()</b>	해당 JSP페이지가 속한 웹 어플리케이션의 컨텍스트 경로를 리턴
String <b>getHeader(name)</b>	웹 서버로 요청 시, HTTP요청 헤더(header) 헤더이름 name에 해당하는 속성값을 리턴
Enumeration <b>getHeaderNames()</b>	웹 서버로 요청 시, HTTP요청 헤더(header)에 있는 모든 헤더이름을 리턴

# EL (Expression Language)

- 내장 객체의 생명주기

- **pageContext**

- 하나의 JSP 프로그램이 실행될 때 생성되었다가, 그 프로그램 실행이 끝날 때 삭제됨. 여기에 값을 저장하면 해당 JSP 파일 내에서만 그 값이 남아있음

- **request**

- JSP 프로그램 실행 요청을 받았을 때 생성되었다가, 그 요청을 모두 처리했을 때 삭제됨. JSP의 포워드 태그를 사용하지 않은 경우는 pageContext와 생명주기가 비슷하지만 포워드 사용시 달라짐

- **session**

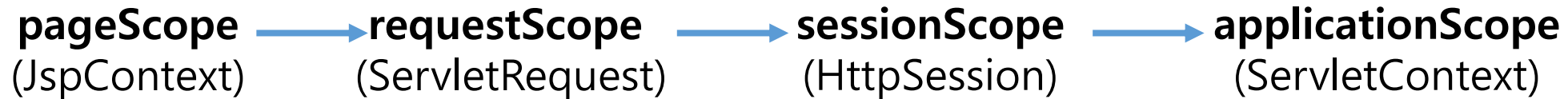
- 한 사용자가 웹 사이트에 접속하여 세션이 수립되었을 때 생성되고, 세션이 종료되었을 때 삭제됨. 세션은 일정 시간 동안 사용자가 아무런 요청을 하지 않으면 종료됨

- **application**

- 웹 애플리케이션이 서비스를 시작할 때 생성되고, 서비스 종료될 때 삭제됨. 웹 애플리케이션 서비스를 시작하는 것은 서블릿 컨테이너인 톰캣이 시작될 때이므로, 쉽게 풀어서 얘기하자면 톰캣이 시작될 때 생성되고, 톰캣이 종료될 때 삭제된다고 생각해도 무방.

# Scope

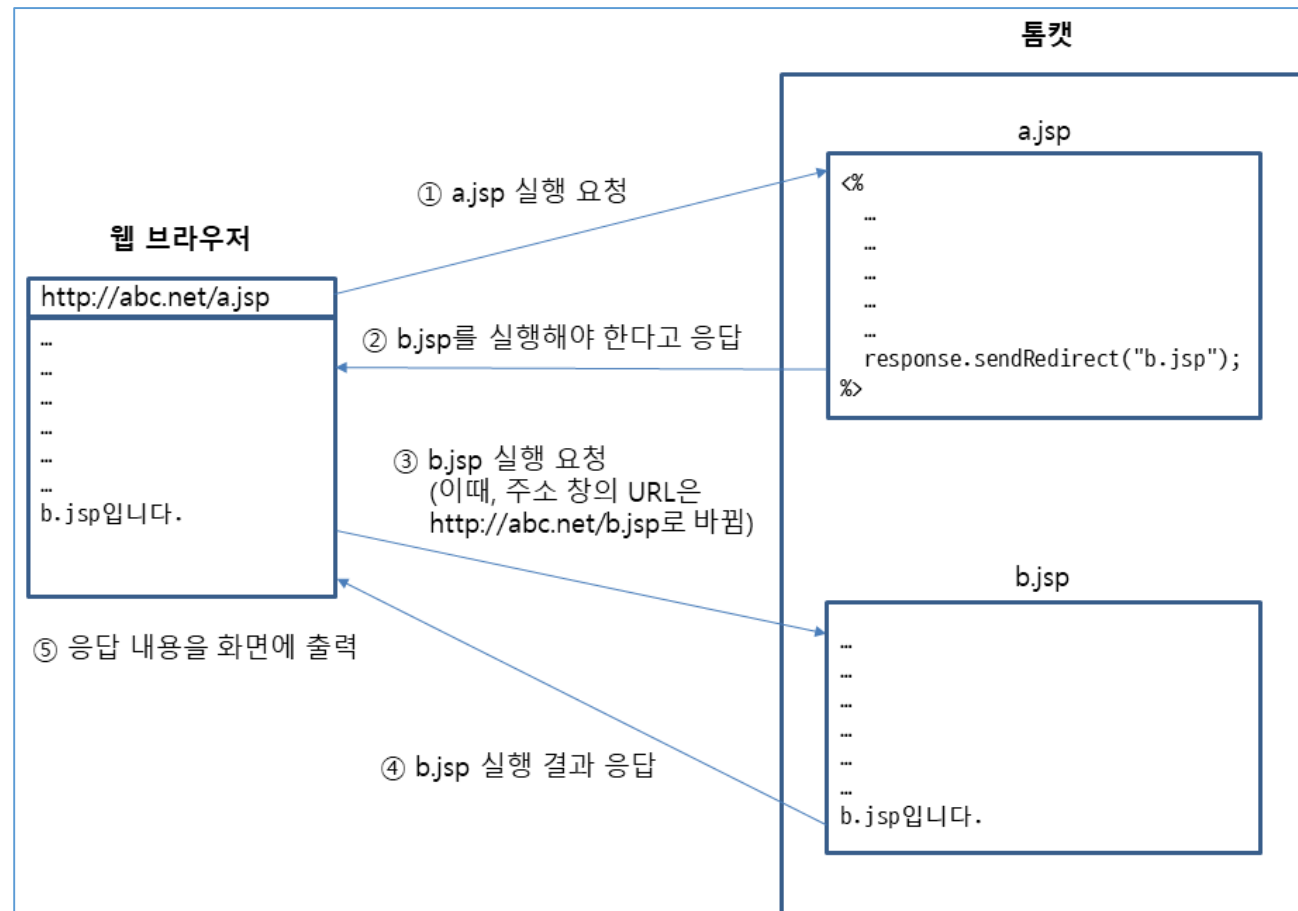
- scope 보관소의 이름을 명시하지 않은 경우에는 작은 scope에서 큰 scope로 값을 찾음
- 찾지 못한 경우 NULL 반환



# 포워드와 리다이렉트

교재 : p.271 - 274

- 리다이렉트(redirect)
  - `response.sendRedirect("이동할_페이지");`

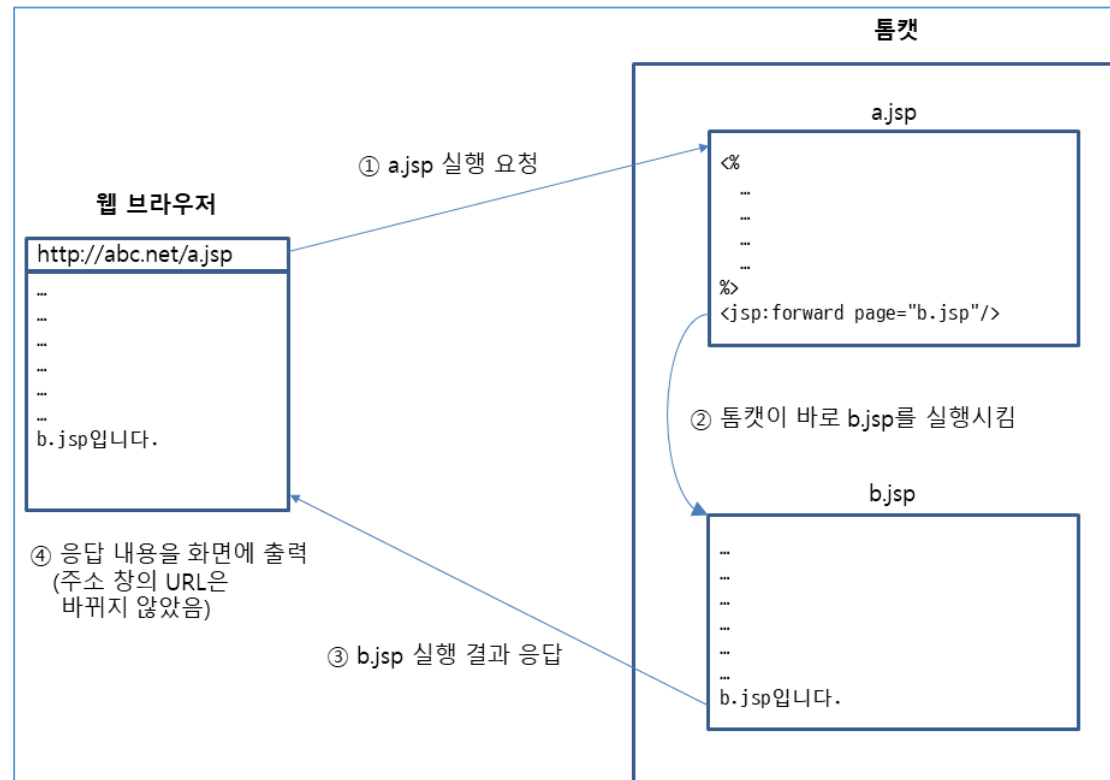


# 포워드와 리다이렉트

교재 : p.271 - 274

- 포워드(forward)

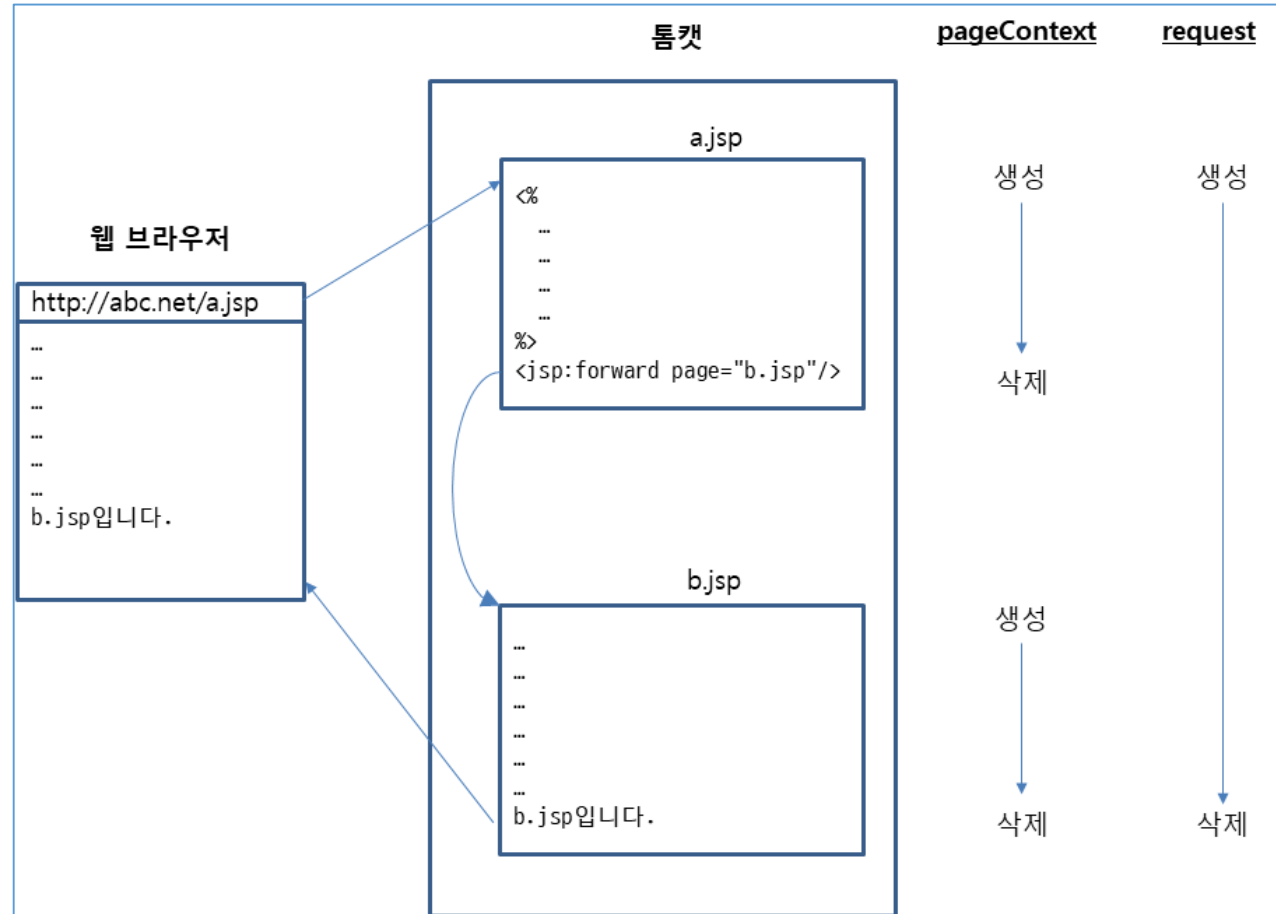
- 액션 태그 : `<jsp:forward page="이동할_페이지_URL"/>`
- 자바 코드 : `request.getRequestDispatcher("이동할_URL").forward(request, response);`



# 포워드와 리다이렉트

교재 : p.271 - 274

- pageContext와 request 객체의 생명주기



# 실습1 : 기본 실습

- elTest.jsp 파일에서 실습해봅시다. 결과는 어떨까요?

```
<h2>EL 예제</h2>
<ul>
  <li>문자열1 : ${"오늘 요일은?"}</li>
  <li>문자열 2 : ${'화요일'}</li>
  <li>연산식1 : ${25-11}</li>
  <li>연산식2 : ${5>3}</li>
  <li>내장객체1 : ${header.host}</li>
  <li>내장객체2 : ${param.id}</li>
  <li>내장객체3 : ${param["id"]}</li>
</ul>
```

← → ↻ ⓘ localhost:8080/web04/elTest1.jsp?id=홍길동

## EL 예제

- 문자열1 : 오늘 요일은?
- 문자열 2 : 화요일
- 연산식1 : 14
- 연산식2 : true
- 내장객체1 : localhost:8080
- 내장객체2 : 홍길동
- 내장객체3 : 홍길동

메소드	설명
String <b>getProtocol()</b>	웹 서버로 요청 시, 사용 중인 프로토콜을 리턴
String <b>getServerName()</b>	웹 서버로 요청 시, 서버의 도메인 이름을 리턴
String <b>getMethod()</b>	웹 서버로 요청 시, 요청에 사용된 요청 방식(GET, POST, PUT 등)을 리턴
String <b>getQueryString()</b>	웹 서버로 요청 시, 요청에 사용된 QueryString을 리턴
String <b>getRequestURI()</b>	웹 서버로 요청 시, 요청에 사용된 URL로부터 URI값을 리턴
String <b>getRemoteHost()</b>	웹 서버로 정보를 요청한 웹 브라우저의 호스트 이름을 리턴
String <b>getRemoteAddr()</b>	웹 서버로 정보를 요청한 웹 브라우저의 IP주소를 리턴
int <b>getServerPort()</b>	웹 서버로 요청 시, 서버의 Port번호를 리턴
String <b>getContextPath()</b>	해당 JSP페이지가 속한 웹 어플리케이션의 컨텍스트 경로를 리턴
String <b>getHeader(name)</b>	웹 서버로 요청 시, HTTP요청 헤더(header) 헤더이름 name에 해당하는 속성값을 리턴
Enumeration <b>getHeaderNames()</b>	웹 서버로 요청 시, HTTP요청 헤더(header)에 있는 모든 헤더이름을 리턴



# 실습2 : 배열, 쿠키 값 출력

```
<!-- 배열에서 값 꺼내기 -->
<%// 값 저장하기
String list[] = {"moon", "sun", "jupiter", "mars", "venus", "mercury"};
pageContext.setAttribute("list", list);
%>
```

1. 위의 list 배열에서 "mars" 출력하기
2. 모두 출력하기

```
<!-- List 객체에서 값 꺼내기 -->
<%// 값 저장하기
List<String> namelist = new LinkedList<String>();
namelist.add("홍길동");
namelist.add("임꺽정");
namelist.add("일지매");
pageContext.setAttribute("namelist", namelist);
%>
```

3. 위의 List 컬렉션에서 두번째 값 출력하기
4. 모든 값 출력하기

```
<!-- 쿠키에서 값 꺼내기 -->
<% String value = request.getParameter("v");
Cookie cookie = new Cookie("v", value);
response.addCookie(cookie);
%>
```

5. 전달 파라미터 v 값 출력하기
6. 쿠키값 출력하기

7. 호스트명 출력하기
8. 연결상태 출력하기

# EL의 연산자

## • 산술 연산자

연산자	기능	연산자	기능
+	더하기	-	빼기
*	곱하기	/ or div	나누기
% of mod	몫		

## • 비교/조건 연산자

연산자	기능	연산자	기능
== 혹은 eq	같다.	!= 혹은 ne	같지 않다.
< 혹은 lt	좌변이 우변보다 작다.	> 혹은 gt	좌변이 우변보다 크다.
<= 혹은 le	좌변이 우변보다 같거나 작다.	>= 혹은 ge	좌변이 우변보다 같거나 크다.
a?b : c	a가 참이면 b, 거짓이면 c를 반환한다.		

## • 관계 연산자

연산자	기능
&& 혹은 and	AND 연산
혹은 or	OR 연산
! 혹은 not	NOT

## • empty 연산자

- 값이 null, 빈문자 면 true 반환
- \${empty param.id}

## • 3항 연산자

- \${조건?<true경우>:<false경우>}

## • EL 예약어

- 변수이름으로 사용할 수 없음

and	eq	gt	true
instanceof	or	ne	le
false	empty	not	lt
ge	null	div	mod

### • 연산자가 기호와 문자가 있는 이유

- ✓ 엄격한 xml 규칙에서 사용할 수 없는 기호가 있을 수 있으므로(<, >) 두가지 다 제공한다. 일반적으로는 기호를 사용한다.

# 실습4

- 다음과 같이 값이 주어졌을 때 계산기 작성

```
X : ${param.X}  
Y : ${param.Y}
```

1. 두 값이 비어 있는지 확인할 것

n1값이 비었나요? :

n2값이 비었나요? :

2. 아래와 같이 계산하기

X + Y =

X - Y =

X \* Y =

X / Y =

X % Y =

3. 두 값을 비교 할 것

X > Y :

X < Y :

X == Y :

X != Y :

4. X값이 비어 있는 것이 아니라면 화면 출력하고 아니면 "비어 있음" 메시지 출력

# 실습5

- EL로 변경할 수 있는 코드를 변경하시오.

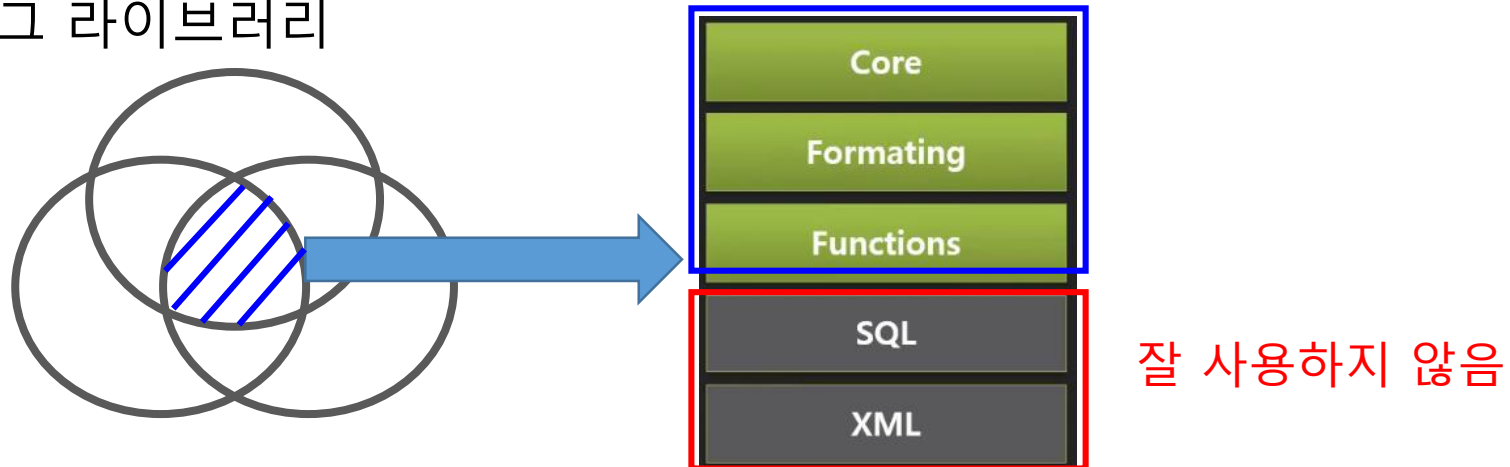
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2 | pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7   <title>Insert title here</title>
8 </head>
9 <body>
10  <%
11    String title= (String)request.getAttribute("title");
12    String author= (String)request.getAttribute("author");
13    String company= (String)request.getAttribute("company");
14  %>
15  <table style="border:1px solid gray">
16    <tr>
17      <th>제목</th>
18      <td><%=title %></td>
19    </tr>
20    <tr>
21      <th>작가</th>
22      <td><%=author %></td>
23    </tr>
24    <tr>
25      <th>출판사</th>
26      <td><%=company %></td>
27    </tr>
28  </table>
29
30 </body>
31 </html>
```

# JSTL(JSP Standard Tag Library)

동의과학대학교 컴퓨터정보과  
김진숙

# Custom Tag

- View 영역에서 액션태그나 표현언어를 사용하더라도 조건식, 반복문 등은 자바 코드를 사용하게 되는 것을 보완하기 위해 도입된 기능
- 종류
  - 개발자 커스텀 태그
    - 개발자가 필요에 의해 만든 태그로 스프링, 스트러츠 같은 프레임워크에서 미리 만들어 제공
  - JSTL(JSP Standard Tag Library)
    - JSP에서 빈번하게 사용하는 기능을 태그로 제공하며 JSTL 라이브러리를 설치하여 사용
    - 표준화된 태그 라이브러리



# JSTL

- JSTL 매뉴얼 : <https://docs.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/>
- 커스텀 태그 중 가장 빈번하게 사용되는 태그를 표준화
  - 반복과 조건, 데이터 관리 포맷, XML 조작, 데이터베이스 액세스
- 로직과 화면 출력 분리 용이
  - 자바코드를 View(화면출력) 영역에서 제거(가독성 ↑)
  - MVC Model2에 사용됨(화면 출력)
- 라이브러리를 WEB-INF/lib에 등록하여 사용

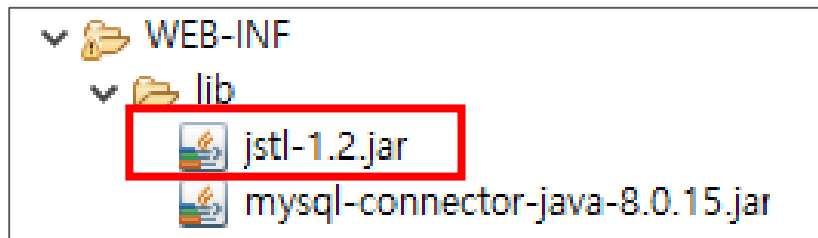
# JSTL

## 1. 다운로드

- <https://mvnrepository.com/artifact/javax.servlet/jstl/1.2>



## 2. WEB-INF/lib 에 복사



다운로드



# JSTL

- 페이지 지시자에 taglib 등록

- Core

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

- Formatting

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

- Functions

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

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

태그 식별을 위한 도메인

uri를 편하게 사용하기 위해 짧은 접두어 사용

```
<c:forEach>
```

.....

```
</c:forEach>
```

# Core

- 자바로 구현한 변수 선언, 조건식, 반복문 등을 태그로 대체
- 사용 전 taglib 지시자에 선언

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

기능	태그	설명
변수지원	<c:set>	변수 지정
	<c:remove>	지정된 변수 제거
흐름 제어	<c:if>	조건문
	<c:choose>	switch 문으로 <c:when>, <c:otherwise>와 함께 사용
	<c:forEach>	반복문
	<c:forTokens>	구분자로 분리된 각 토큰 처리 시 사용
URL 처리	<c:url>	요청 매개변수로부터 URL 생성
기타	<c:catch>	예외처리에 사용

## <c:set>

- scope에 변수를 선언
- <jsp:setProperty> 액션 태그와 유사

```
<c:set var="income" value="${4000*2}" scope="session"/>  
<c:out value="${income }"/>
```

```
<c:set var="income" value="${4000*2}" scope="session"/>  
${income}
```

## <c:remove>

```
<c:remove var="income" scope="session"/>
```

# 실습1

```
<% String id = "gildong";  
    int income = 2000000;  
    out.println(id + "의 수입은 " + income + "입니다.");  
%>
```

- EL과 JSTL로 코드를 변경하시오.

## <c:out>

- 수식을 평가하고 평가된 결과 출력
  - <c:out value="출력값" default="기본값"/>
  - 지정된 값 출력
  - value값이 null이면 default 값 사용

EL이 있어도 <c:out> 사용하는 이유는?  
- 불순한 자바스크립트 코드의 실행을 막고 단순한 문자열로 출력되도록 하는 방법으로 사용  
➔ 보안 때문

```
<c:out value="${param.id}" default="<반가워요>"/>
```

## <c:if>

- test 조건이 true인 경우 body 내용 수행
- <c:if test="조건" var="변수명" scope="범위">
- else문 없음

```
<c:set var="income" value="${4000*2}" scope="session"/>
<c:if test="${income >= 8000}" var="result">
  <p>My income is : ${income}</p>
  <p>Result : ${result}</p>
</c:if>
```

# 실습2

- 아래 코드를 EL과 JSTL로 코드를 변경하시오.
- 삼항연산자를 사용하여 코드하시오.

1.

```
<% String score = request.getParameter("score");  
  
    if(score==null){  
        out.println("매개변수값이 비어있습니다.");  
    }else{  
        out.println(score);  
    }  
%>
```

2.

```
<% String loginid = "gildong";  
    String name = "홍길동";  
  
    if(loginid=="gildong" && name=="홍길동")  
        out.println("아이디는 " + loginid + "이고, " + "이름은 " + name + "입니다.");  
%><br>
```

## <c:choose>

- 여러 조건 처리(switch 문과 유사)
- income이 200만원 이하이면 Income is not good!, 200만원 초과하면 good...

```
<c:set var="income" value="${param.id}"/>
<p>Your income is : ${income }</p>

<c:choose>
  <c:when test="${income<=2000000 }">
    Income is not good!
  </c:when>
  <c:when test="${income>2000000 }">
    Income is good!
  </c:when>
  <c:otherwise>
    Income is undetermined...
  </c:otherwise>
</c:choose>
```

# 실습3

- param.name으로 아이디를 입력 받아 다음을 처리하시오.
  - 값이 비어 있으면 폼을 작성하여 이름을 다시 입력 받는다.
  - 값이 'admin'과 같으면 "관리자"로 표시한다.
  - 값이 'user'이면 "일반 사용자"로 표시한다.
  - 어느 것도 아니면 원하는 것을 출력한다.

```
<c:choose>
  <c:when test="${empty param.name}">
    <form action="test1.jsp" method="get">
      이름 : <input type="text" name="name">
        <input type="submit" value="확인">
    </form>
  </c:when>
  <c:when test="${param.name == 'admin' }">
    관리자 작업
  </c:when>
  <c:when test="${param.name eq 'user' }">
    일반 사용자. 회원 <c:out value="${param.name}" />
  </c:when>
  <c:otherwise>
    기타 작업
  </c:otherwise>
</c:choose>
```



# 실습3 - 선택

- param으로 점수 데이터를 받아 학점을 처리하는 코드를 EL과 JSTL(choose)로 작성하시오

```
<!-- choose : 조건문 -->
<c:set var="score" value="${param.score }"/>
점수<c:out value="${score }" />는
<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>불합격이거나 점수가 없습니다.</c:otherwise>
</c:choose>
<br><br>
```

# <c:forEach>

- <c:forEach **var**="변수" **items**="객체" **begin**="시작" **end**="끝" **step**="증감" **varStatus**="상태 변수">

항목 속성	설명	비고
var	사용할 변수명	필수 항목
items	Collection 객체 (List, ArrayList, Map 등)	
begin	시작 index, 정의되지 않을 경우 디폴트 0	
end	종료 index, 정의되지 않을 경우 디폴트 items 크기 -1	
step	반복할 때 이동할 index 갯수	
varStatus	반복상태를 알 수 있는 변수	

```
<c:forEach var="i" begin="1" end="3">
    ${i}
</c:forEach>
```

```
< c:foreach items="${RESULT}" var="RESULT" varStatus="status">

    ${status.current}<br/> <!-- 현재 아이템 -->
    ${status.index}<br/>      <!-- 0부터의 순서 -->
    ${status.count}<br/>      <!-- 1부터의 순서 -->
    ${status.first}<br/>      <!-- 현재 루프가 처음인지 반환 -->
    ${status.last}<br/>      <!-- 현재 루프가 마지막인지 반환 -->
    ${status.begin}<br/>      <!-- 시작값 -->
    ${status.end}<br/>        <!-- 끝값 -->
    ${status.step}<br/>      <!-- 증가값 -->

< /c:forEach>
```

# 실습4

```
<!-- 배열에서 값 꺼내기 -->  
<%// 값 저장하기  
    String list[] = {"moon", "sun", "jupiter", "mars", "venus", "mercury"};  
    pageContext.setAttribute("list", list);  
%>
```

1.

```
<% String list[] ={"moon", "sun", "jupiter", "mars","venus","mercury"};  
    pageContext.setAttribute("list", list);  
%>
```

```
<c:forEach var="i" items="${list }">  
    ${i }<br>  
</c:forEach>
```

```
<c:set var="list2" value="${fn:split('moon, sun, jupiter, mars, venus, mercury','(',')}'" />
```

```
<c:set var="list">moon, sun, jupiter, mars, venus, mercury</c:set>
```

```
<c:forEach var="item" items="${list}" varStatus="idx">  
    ${idx.index}, ${item} <br>  
</c:forEach>
```

2.

```
<% List<String> list = new ArrayList<String>();  
    list.add("홍길동");  
    list.add("임꺽정");  
    list.add("이순신");  
    request.setAttribute("list", list);  
%><br>
```

# 실습5

- EL과 JSTL로 구구단 만들기

2 * 2 = 4	3 * 2 = 6	4 * 2 = 8	5 * 2 = 10	6 * 2 = 12	7 * 2 = 14	8 * 2 = 16	9 * 2 = 18
2 * 3 = 6	3 * 3 = 9	4 * 3 = 12	5 * 3 = 15	6 * 3 = 18	7 * 3 = 21	8 * 3 = 24	9 * 3 = 27
2 * 4 = 8	3 * 4 = 12	4 * 4 = 16	5 * 4 = 20	6 * 4 = 24	7 * 4 = 28	8 * 4 = 32	9 * 4 = 36
2 * 5 = 10	3 * 5 = 15	4 * 5 = 20	5 * 5 = 25	6 * 5 = 30	7 * 5 = 35	8 * 5 = 40	9 * 5 = 45
2 * 6 = 12	3 * 6 = 18	4 * 6 = 24	5 * 6 = 30	6 * 6 = 36	7 * 6 = 42	8 * 6 = 48	9 * 6 = 54
2 * 7 = 14	3 * 7 = 21	4 * 7 = 28	5 * 7 = 35	6 * 7 = 42	7 * 7 = 49	8 * 7 = 56	9 * 7 = 63
2 * 8 = 16	3 * 8 = 24	4 * 8 = 32	5 * 8 = 40	6 * 8 = 48	7 * 8 = 56	8 * 8 = 64	9 * 8 = 72
2 * 9 = 18	3 * 9 = 27	4 * 9 = 36	5 * 9 = 45	6 * 9 = 54	7 * 9 = 63	8 * 9 = 72	9 * 9 = 81

```
<c:forEach var="i" begin="2" end="9"><br>
  <c:forEach var="j" begin="2" end="9">
    ${j} * ${i} = ${i*j} &nbsp;
  </c:forEach>
</c:forEach>
```

## <c:forTokens>

- 구분자로 나누어진 객체를 반복문으로 추출
- <c:forTokens **var**="변수" **items**="객체" **delims**="구분자" **begin**="시작" **end**="끝" **step**="증감" **varStatus**="상태변수">

```
<!-- forToken : 파싱하기 -->
<c:forTokens var="token" items="소설/역사/인문/정치/미술/종교/여행/과학/만화/건강" delims="/">
  ${token }
</c:forTokens><br><br>

<c:set var="v" value="mercury, venus, earth, mars, jupiter, saturn, uranus, naptune, pluto"/>
<c:forTokens var="i" items="${v}" delims=",">
  ${i }
</c:forTokens><br><br>
```

실습6 :

<c:forTokens>를 사용하여 위의 배열을 출력하시오.

# 실습7

- Java Bean 객체인 BookBean의 속성에 값을 쓰고 읽는 방법

```
<!-- BookBean 객체 생성 -->
<jsp:useBean id="book" class = "el.web04.BookBean" scope="request"/>

<!-- 액션 태그로 BookBean의 속성 author에 값 설정 -->
<jsp:setProperty name="book" property="author" value="chunhyang"/>

<!-- JSTL로 BookBean의 속성 title에 값 설정 -->
<c:set value="춘향전" target="${book}" property="title"/>
<c:set value="동의과학대출판사" target="${book}" property="publisher"/>

<!-- BookBean에 설정된 값 출력하기 -->
title : <c:out value="${book.title }"/><br>
author: <c:out value="${book.author }"/><br>
publisher: <c:out value="${book.publisher }"/><br>
```

- 지역과 언어 관련 지원 기능/ 숫자, 날짜, 시간을 포맷하는 기능

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

기능	태그	설명
숫자 날짜 형식	formatNumber	숫자를 양식에 맞춰서 출력한다.
	formatDate	날짜 정보를 담고 있는 객체를 포매팅하여 출력할 때 사용한다.
	parseDate	문자열을 날짜로 파싱한다.
	parseNumber	문자열을 수치로 파싱한다.
	setTimeZone	시간대별로 시간을 처리할 수 있는 기능을 제공한다.
	timeZone	시간대별로 시간을 처리할 수 있는 기능을 제공한다.
로케일 지정	setLocale	국제화 태그들이 사용할 로케일을 지정한다.
	<b>requestEncoding</b>	요청 파라미터의 인코딩을 지정한다.
메시지 처리	bundle	태그 몸체에서 사용할 리소스 번들을 지정한다.
	message(param)	메시지를 출력한다.
	setBundle	특정 리소스 번들을 사용할 수 있도록 로딩한다.

```
<fmt:requestEncoding value="UTF-8"></fmt:requestEncoding>
```

ex 1 :

```
<fmt:formatNumber value="1234567.89"/><br>
```

ex 2 :

```
<fmt:formatNumber value="1234567.89" groupingUsed="false"/><br>
```

ex 3 :

```
<fmt:formatNumber value="0.5" type="percent"/><br>
```

ex 4 :

```
<fmt:formatNumber value="1234567.89" pattern="#,#00.0#"/><br>
```

ex 5 :

```
<fmt:formatNumber value="10000" type="currency"/><br>
```

```
<fmt:formatNumber value="10000" type="currency" currencySymbol="$"/><br>
```

ex 1 : 1,234,567.89

ex 2 : 1234567.89

ex 3 : 50%

ex 4 : 1,234,567.89

ex 5 : ₩10,000

\$10,000



```

${now} : Tue Oct 08 11:06:36 KST 2019
2019. 10. 8
date : 2019. 10. 8
time : 오전 11:06:36
both : 2019. 10. 8 오전 11:06:36
default : 2019. 10. 8 오전 11:06:36
short : 19. 10. 8 오전 11:06
medium : 2019. 10. 8 오전 11:06:36
long : 2019년 10월 8일 (화) 오전 11시 06분 36초
full : 2019년 10월 8일 화요일 오전 11시 06분 36초 KST
pattern="yyyy년 MM월 dd일 hh시 mm분 ss초" : 2019년 10월 08일 11시 06분 36초

```

```
<!-- now변수 value 정의 -->
```

```
<c:set var="now" value="<%=new java.util.Date() %>"></c:set>
```

```
<!-- fmt:formatDate - 날짜와 시간 표현하는 방식을 정의 -->
```

```
\${now} : ${now }<br>
```

```
<fmt:formatDate value="\${now}"></fmt:formatDate><br>
```

```
date : <fmt:formatDate value="\${now}" type="date"></fmt:formatDate><br>
```

```
time : <fmt:formatDate value="\${now}" type="time"></fmt:formatDate><br>
```

```
both : <fmt:formatDate value="\${now}" type="both"></fmt:formatDate><br>
```

```
<!-- dateStyle, timeStyle의 날짜, 시간 형식 4가지. (생략 시 default) -->
```

```
default : <fmt:formatDate value="\${now}" type="both" dateStyle="default" timeStyle="default"></fmt:formatDate><br>
```

```
short : <fmt:formatDate value="\${now}" type="both" dateStyle="short" timeStyle="short"></fmt:formatDate><br>
```

```
medium : <fmt:formatDate value="\${now}" type="both" dateStyle="medium" timeStyle="medium"></fmt:formatDate><br>
```

```
long : <fmt:formatDate value="\${now}" type="both" dateStyle="long" timeStyle="long"></fmt:formatDate><br>
```

```
full : <fmt:formatDate value="\${now}" type="both" dateStyle="full" timeStyle="full"></fmt:formatDate><br>
```

```
pattern="yyyy년 MM월 dd일 hh시 mm분 ss초" :
```

```
<fmt:formatDate value="\${now}" pattern="yyyy년 MM월 dd일 hh시 mm분 ss초"></fmt:formatDate><br>
```