



## **JSP 13강 – 표현 언어**

---

**양 명 속**

**[[now4ever7@gmail.com](mailto:now4ever7@gmail.com)]**



# 목차

---

- 표현언어의 개요
- 표현언어의 연산자와 내장 객체



# 표현 언어(Expression Language)의 개요

## ■ 표현언어(EL)

- JSTL 1.0 스펙에서 소개되었던 것으로, **jsp 페이지에서 사용되는 자바 코드를 대신해서 액션 태그 엘리먼트** (예 :<jsp:setProperty>)의 **속성에 값을 지정하는 역할**을 함
- 기존방식 예) <tags:kk> 액션 태그 엘리먼트의 속성은 attribute, 속성값은 <%=pageContext.getAttribute("age")%> 이라면

```
<tags:kk attribute="<%=pageContext.getAttribute("age")%>">
```

- 자바빈 컴포넌트에서 자바빈 객체의 메서드에 접근하는 경우
- 예) person 이 자바빈 객체, getAddress()는 자바빈 객체가 가지고 있는 메서드

```
<%=person.getAddress()%>
```

- 기존 방식 : 액션 태그의 엘리먼트 속성의 값 또는 자바빈의 메서드 접근 시에 표현식<%=> 사용



# 표현 언어의 개요

- 표현 언어는 표현식 `<%= %>`을 대신하는 효과를 가짐
- 표현 언어로 표시하면
  - 액션 태그 엘리먼트의 속성값 처리

```
<tags:kk attribute="${age}">
```

```
<tags:kk attribute="<%=pageContext.getAttribute("age")%>">
```

- 자바빈의 메서드에 접근

```
${person.address}
```

```
<%=person.getAddress()%>
```



# 표현 언어의 개요

- 과거에는 표현언어는 JSTL에 상당히 종속적인 면을 가지고 있었음
- JSP 2.0 스펙에 포함되면서, JSP 컨테이너 자신이 표현언어의 표현식을 해석할 수 있게 되었고, 표준과 커스텀 액션 태그, HTML 같은 템플릿 텍스트와 같이 자바 코드를 사용해야 했던 모든 곳에서 표현언어를 사용할 수 있게 됨
- null 값을 가지는 변수에 대해 좀 더 관대하고, 데이터 형변환을 조금 더 자동적으로 해줌
  - => 값이 없을 경우(null)나 형변환 같은 것에 신경 쓸 필요 없음

java EE7 : jsp2.3, servlet 3.1, EL 3.0, JSTL 1.2

java EE8 : jsp2.3, servlet 4.0, EL 3.0, JSTL 1.2



# 표현 언어의 개요

---

- 표현 언어의 표현식의 기능
  - 변수와 연산자를 포함함
  - jsp의 영역(page, request, session, application)에 저장된 어떤 속성 및 자바빈이라도 표현언어의 변수로서 사용될 수 있음
  - 내장 객체를 지원함
- 표현 언어의 작성 방법
  - 표현언어 표현식에는 숫자, 문자열, boolean 값, null 같은 상수값(리터럴)들을 포함할 수 있음
  - 표현언어는 \$ 와 표현식, { } 를 사용해서 표현

# 표현 언어의 개요

- [1] 표현언어는 항상 `${` 로 시작해서 `}` 로 끝남

`${num}`

`${num}`은 `<%=num%>`와 결과가 같다

- 표현식은 jsp 스트립트 내부에서는 쓸 수 없음
  - 즉, `<%%>`, `<%!%>`, `<%= %>` 안에는 사용할 수 없음
  - 그 외의 곳에서는 사용 가능
- [2] 표현언어는 표현식 안에 연산식도 작성 가능
  - num 변수값에 1을 더하는 표현언어의 표현식

`${num + 1}`

- [3] 자바빈의 num 프로퍼티 값에 1을 더하는 표현언어 표현식
  - article은 자바빈 객체, num 은 프로퍼티

`${article.num + 1}`



# 표현 언어의 개요

- 프로퍼티 접근 연산자 dot(.)은 표현언어에서, 자바빈이나 컬렉션 객체에서 다음에 오는 이름과 같은 프로퍼티를 찾게 함
- 표현언어 표현식에는 브라켓 연산자 [] 를 사용해도 됨
  - 브라켓 연산자 배열의 형태로 객체의 프로퍼티나 변수에 접근할 수 있음
  - 자바빈의 프로퍼티 num에 1을 더하는 경우 `${article.num + 1}` 와 동일

`${article['num'] + 1}` 또는 `${article["num"] + 1}`

- 브라켓 안의 값은 프로퍼티의 이름을 나타내는 문자열이거나 프로퍼티 이름을 값으로 가진 변수가 올 수 있음





# 표현 언어의 개요

---

- 표현언어는 동적으로 값을 받도록 JSTL이나 커스텀 태그의 jsp 액션의 속성에 값을 지정할 때도 사용 가능
  - 화면에 article 객체의 num 프로퍼티 값에 1을 더한 값을 출력 (JSTL 문법)

```
<c:out value="${article.num + 1}"/>
```

# 표현언어의 연산자와 내장 객체

## ■ 표현언어의 연산자

연산자	설 명
.	빈의 프로퍼티나 Map의 엔트리 접근
[]	배열이나 List 엘리먼트 접근
()	괄호, 표현식의 연산 순서를 바꿔서 연산하게 할 때
a?b:c	조건 테스트-조건(a) ? true 일때 리턴값(b) : false일때 리턴값(c)
+	더하기
-	빼기
*	곱하기
/ 또는 div	나누기
% 또는 mod	나머지
== 또는 =	같다

```
<c:set var="result" value='${fn:split(birth, "-")}' />
${result[0]} 년 ${result[1]} 월 ${result[2]} 일
```

```
<c:set var="bean" value="${alist[curPos]}"></c:set>
```

# 표현언어의 연산자와 내장 객체

연산자	설 명
!= 또는 ne	다르다
< 또는 lt	보다 작다
> 또는 gt	보다 크다
<= 또는 le	작거나 같다
>= 또는 ge	크거나 같다
&& 또는 and	논리 AND
또는 or	논리 OR
! 또는 not	단항 not (true를 false로, false를 true로)
empty	빈 변수 값 체크. null, 빈 문자열, 빈 배열, 엔트리가 없는 Map 이나 컬렉션인지 등을 테스트

<c:if test="\${empty sessionScope.userId }">  
컨텍스트 패스 : \${pageContext.request.contextPath}

# 표현언어에서 제공되는 내장객체

내장 객체	설 명
pageScope	모든 page 영역 객체들에 대한 컬렉션
requestScope	모든 request 영역 객체들에 대한 컬렉션
sessionScope	모든 session 영역 객체들에 대한 컬렉션
applicationScope	모든 application 영역 객체들에 대한 컬렉션
param	모든 request 파라미터들을 문자열로 가진 컬렉션
paramValues	모든 request 파라미터들을 파라미터당 문자열 배열로 가진 컬렉션
header	HTTP 요청 헤더를 문자열로 가진 컬렉션
headerValues	HTTP 요청 헤더들을 헤더당 문자열 배열로 가진 컬렉션
cookie	모든 쿠키의 컬렉션
initParam	모든 어플리케이션의 초기화 파라미터의 이름 컬렉션
pageContext	현재 페이지를 위한 javax.servlet.jsp.PageContext

If(session.getAttribute("userId")==null || session.getAttribute("userId").isEmpty())  
컨텍스트 패스 : request.getContextPath()

# 예제-연산자와 내장객체 사용

← → 🏠 💰 http://localhost:9090/projectStudy2/EL/eLEx1.jsp

## 간단한 표현언어(EL)예제

연산자를 사용한 예와 내장객체의 사용한 예:

표현식	값
<code>\${2 + 5}</code>	7
<code>\${4/5}</code>	0.8
<code>\${5 div 6}</code>	0.8333333333333334
<code>\${5 mod 7}</code>	5
<code>\${2 &lt; 3}</code>	true
<code>\${2 gt 3}</code>	false
<code>\${3.1 le 3.2}</code>	true
<code>\${(5 &gt; 3) ? 5 : 3}</code>	5
<code>\${header ["host"]}</code>	localhost:9090
<code>\${header ["user-agent"]}</code>	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; Win64; x64; Trident/6.0; .NET4.0E; .NET4.0C; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET CLR 2.0.50727; MASMJS)

`request.getHeader("host")`  
`request.getHeader("user-agent")`

127.0.0.1:9090

# 예제-연산자와 내장객체 사용

```
<%@ page contentType="text/html;charset=utf-8"%>
```

```
<HTML><HEAD><TITLE>간단한 표현언어(EL)예제</TITLE></HEAD>
```

```
<BODY><H3>간단한 표현언어(EL)예제</H3><P>
```

연산자를 사용한 예와 내장객체의 사용한 예:

```
<TABLE BORDER="1">
```

```
<THEAD>
```

```
<TD><B>표현식</B></TD> <TD><B>값</B></TD>
```

```
</THEAD>
```

```
<TR>
```

```
<TD>W${2 + 5}</TD> <TD>${2 + 5}</TD> </TR>
```

```
<TR>
```

```
<TD>W${4/5}</TD> <TD>${4/5}</TD> </TR>
```

```
<TR>
```

```
<TD>W${5 div 6}</TD> <TD>${5 div 6}</TD> </TR>
```

```
<TR>
```

```
<TD>W${5 mod 7}</TD> <TD>${5 mod 7}</TD> </TR>
```

```
<TR>
```

```
<TD>W${2 < 3}</TD> <TD>${2 < 3}</TD> </TR>
```

```
<TR>
```

```
<TD>W${2 gt 3}</TD> <TD>${2 gt 3}</TD> </TR>
```

\$라는 문자를 그냥 화면에 표시하기 위해 앞에 `<math>`를 붙여서 사용



# 예제-연산자와 내장객체 사용

<TR>

<TD>W\${3.1 le 3.2}</TD> <TD>\${3.1 le 3.2}</TD> </TR>

<TR>

<TD>W\${(5 > 3) ? 5 : 3}</TD>

<TD>\${(5 > 3) ? 5 : 3}</TD>

</TR>

<TR>

<TD>W\${header["host"]}</TD>

호스트의 ip와 port 번호를 화면에 표시

<TD>\${header["host"]}</TD>

</TR>

<TR>

<TD>W\${header["user-agent"]}</TD>

웹 브라우저의 종류를 화면에 표시

<TD>\${header["user-agent"]}</TD>

</TR>

</TABLE>

</BODY>

</HTML>



# 예제

---

```
<%
    //표현언어-EL : 기존의 jsp 표현식을 대체하는 효과
    //el 표현식에는 연산자 사용 가능, 내장객체 지원함
    //null에 관대하고 형변환도 자동으로 해줌
%>
<h3>el 표현식 연습</h3>
W${2+5} : ${2+5} <br>
W${10/3} : ${10/3}<br>
W${10%3} : ${10%3}<br>
W${header.host} : ${header.host} <br>
W${header["user-agent"]} : ${header["user-agent"]} <br>

<h4>기존 jsp 방식</h4>
<%
    String sHost = request.getHeader("host");
    String sAgent = request.getHeader("user-agent");
%>
<hr>
header host 는 <%=sHost %> <br>
header user-agent 는 <%=sAgent %>
```





# Request Header

---

## ▼ Request Headers [view parsed](#)

GET /mymvc/pd/pdDetail.do?no=4 HTTP/1.1

Host: localhost:9090

Connection: keep-alive

Cache-Control: max-age=0

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.80 Safari/537.36

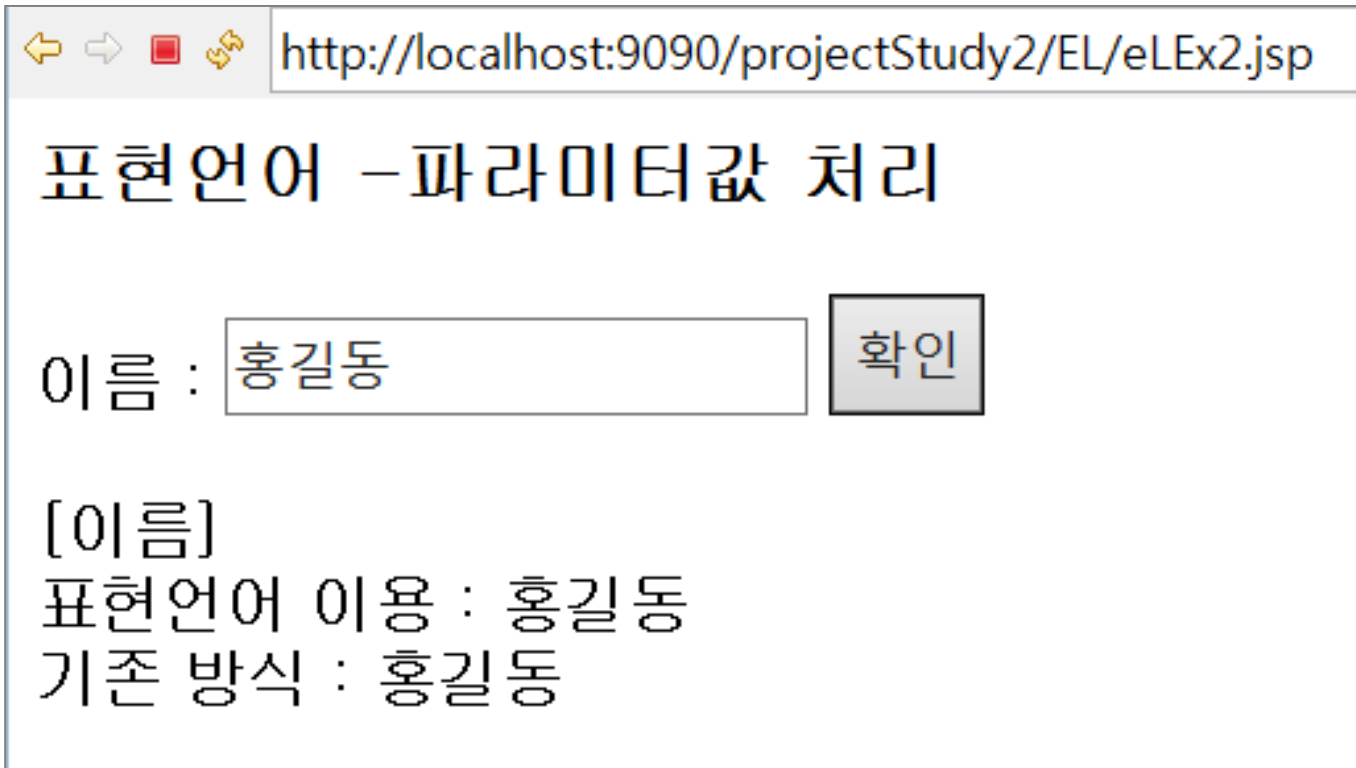
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3

Referer: http://localhost:9090/mymvc/pd/pdEditOk.do

Accept-Encoding: gzip, deflate, br

## 예제-param 내장 객체의 사용

- 폼에 이름을 입력하고 [확인]단추를 클릭하면 현재 페이지의 input 태그에서 value 속성의 값에 폼에서 입력한 이름이 표시됨



← → □ ⌂ http://localhost:9090/projectStudy2/EL/eLEx2.jsp

### 표현언어 - 파라미터값 처리

이름 :

[이름]  
표현언어 이용 : 홍길동  
기존 방식 : 홍길동



# 예제-param 내장 객체의 사용

```
<%@ page contentType="text/html; charset=utf-8"%>
<% request.setCharacterEncoding("utf-8");%>
<HTML><HEAD><TITLE>표현언어의 사용예제2</TITLE></HEAD>
<BODY>
<H3>표현언어 - 파라미터값 처리</H3><P>

<FORM action="eLEx2.jsp" method="post">
    이름 : <input type="text" name="name" value="${param['name']}">
        <input type="submit" value="확인">
</FORM>

<P>
[이름] <br>
표현언어 이용 : ${param.name} <br>
기존 방식 : <%=request.getParameter("name") %>

</BODY>
</HTML>
```