익스프레션 언어

강사 : 강병준

EL(Expression Language)의개요

❖ EL(Expression Language)

- 표현언어를의미
- jsp스크립트를대신하여속성값들을좀더편리하게출력하기위해제공되는언어
- 예를들어<%=hello%>라는코드를EL로표현할때는${hello}로표현된다.

❖ 표현식

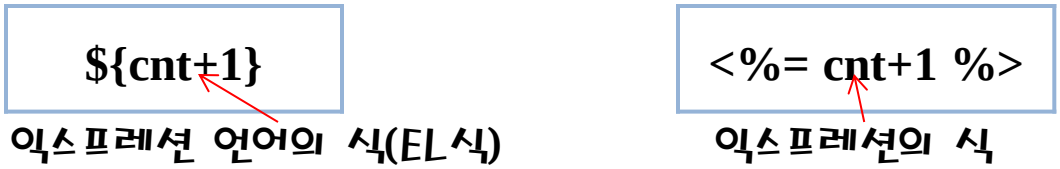
test 변수를 표현할 때 → ${test}

hello 객체의test 변수를표현하면다음과같다

${hello.test} 나 ${hello['test']}

익스프레션 언어란?

- 익스프레션 언어(expression language)란 식(expression)을 중심으로 코드를 기술하는 언어이다.
- 연산자와 피연산자의 조합을 ${와 }로 둘러싸서 표현한다.



- 위에 사용된 cnt 데이터 이름의 의미는 서로 다르다. 익스프레션에서 사용된 cnt는 자바 프로그래밍 언어의 변수 이름이며, EL 식에서 사용된 cnt는 애트리뷰트의 이름으로 해석된다.
- 애트리뷰트란 setAttribute, getAttribute, removeAttribute 메서드를 통해 저장되고, 관리되는 데이터를 의미한다.

Hundred.jsp

```
<%
        int sum = 0;
        for (int cnt = 1; cnt <= 100; cnt++)
              sum += cnt;
        request.setAttribute( “RESULT ”, new Integer(sum));
        RequestDispatcher dispatcher =  request.getRequestDispatcher( “HundredResult.jsp ”);
        dispatcher.forward(request, response);
%>
```

덧셈의 결과를 애트리뷰트로 저장합니다

HundredResult.jsp 호출

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<HTML>
      <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
      <BODY>
            1부터 100까지 더한 결과는? <%= request.getAttribute( “RESULT ”) %>
      </BODY>
</HTML>
```

애트리뷰트 값을 가져다가 출력합니다.

애트리뷰트 형태로 전달되는 데이터

```
<%
      int sum = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
            sum += cnt;
      request.setAttribute( “RESULT ”, new Integer(sum));
      RequestDispatcher dispatcher =
request.getRequestDispatcher( “HundredResult.jsp ”);
      dispatcher.forward(request, response);
%>
```

덧셈의 결과를 애트리뷰트로 저장합니다

호출

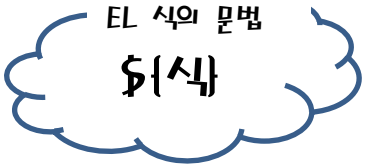
```
<%@page contentType= “text/html; charset=euc-kr ”%>
<HTML>
      <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
      <BODY>
            1부터 100까지 더한 결과는? ${RESULT}
      </BODY>
</HTML>
```

애트리뷰트 값을 가져다가 출력합니다.

애트리뷰트 값을 출력하는 EL 식

익스프레션 언어의 기초 문법

- 익스프레션 언어의 유일한 목적은 식을 계산해서 그 결과를 출력하는 것이므로 다음과 같은 하나의 문법으로 표현할 수 있으며 이 문법을 EL 식이라고 부른다.



- '식' 위치에는 데이터 이름 하나로만 구성된 식이 들어갈 수도 있고, 연산자를 포함하는 식이 들어 갈 수도 있으며, 자바의 정적 메서드를 호출하는 식이 들어갈 수도 있다.

${RESULT}

데이터 이름 하나로만 구성된 EL 식

${RESULT + 101}

연산자를 포함하는 EL 식

${m:sqrt(100)}

자바의 정적 메서드를 호출하는 EL 식

익스프레션 언어의 기초 문법

❖ 데이터 이름 하나로만 구성된 EL 식

- 데이터 이름 하나로만 구성된 EL 식은 가장 간단한 형태의 EL 식이다.
- EL 식 안에 기술되는 데이터 이름은 애트리뷰트 이름으로 해석된다.

```
${RESULT}
```

애트리뷰트 이름

JSP/서블릿 기술에서 사용되는 네 종류의 애트리뷰트

에트리뷰트의 종류	호출할 때 사용하는 내장 변수	메서드의 소속
page 애트리뷰트	pageContext 내장 변수	javax.servlet.jsp.JspContext 클래스
request 애트리뷰트	request 내장 변수	javax.servlet.ServletRequest 인터페이스
session 애트리뷰트	session 내장 변수	javax.servlet.http.HttpSession 인터페이스
application 애트리뷰트	application 내장 변수	javax.servlet.ServletContext 인터페이스

❖ 데이터 이름 하나로만 구성된 EL 식

1부터 100까지의 합을 구하는 JSP 페이지

1부터 100까지의 합을 구하는 JSP 페이지

```
<%
      int sum = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
           sum += cnt;
      request.setAttribute( “RESULT ”, new Integer(sum));
      RequestDispatcher dispatcher =
request.getRequestDispatcher( “HundredResult.jsp ”);
      dispatcher.forward(request, response);
%>
```

request 데이터 영역에 애트리뷰트를 저장합니다

1부터 100까지의 합을 출력하는 JSP 페이지

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<HTML>
      <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
      <BODY>
           1부터 100까지 더한 결과는? ${RESULT}
      </BODY>
</HTML>
```

request 데이터 영역에 있는 애트리뷰트 값을 가져다가 출력합니다

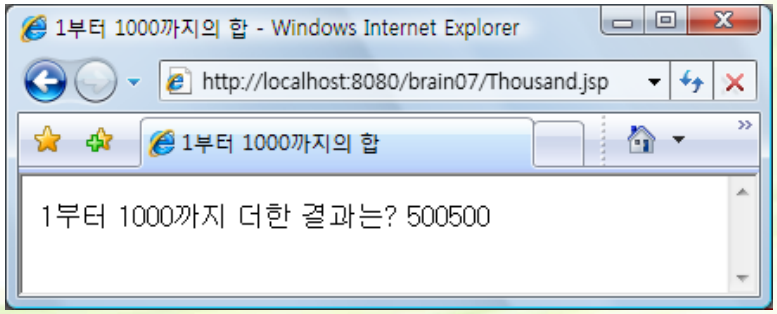
❖ 데이터 이름 하나로만 구성된 EL 식

Thousand.jsp

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<%
      int sum = 0;
      for (int cnt = 1; cnt <= 1000; cnt++)
            sum += cnt;
      pageContext.setAttribute( “RESULT ”, new Integer(sum));
%>
<HTML>
       <HEAD><TITLE>1부터 1000까지의 합</TITLE></HEAD>
       <BODY>
            1부터 1000까지 더한 결과는? ${RESULT}
       </BODY>
</HTML>
```

page 데이터 영역에 애트리뷰트를 저장합니다

page 데이터 영역에 있는 애트리뷰트 값을 가져다가 출력합니다.



실행 결과

기본 문법

❖ attributeCreate.jsp

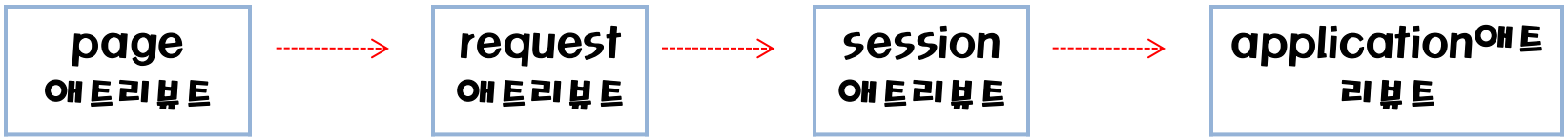
```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<%
    request.setAttribute("name", "김유신");
    RequestDispatcher dispatcher = request.getRequestDispatcher("ELResult.jsp");
    dispatcher.forward(request, response);
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

❖ ELResult.jsp

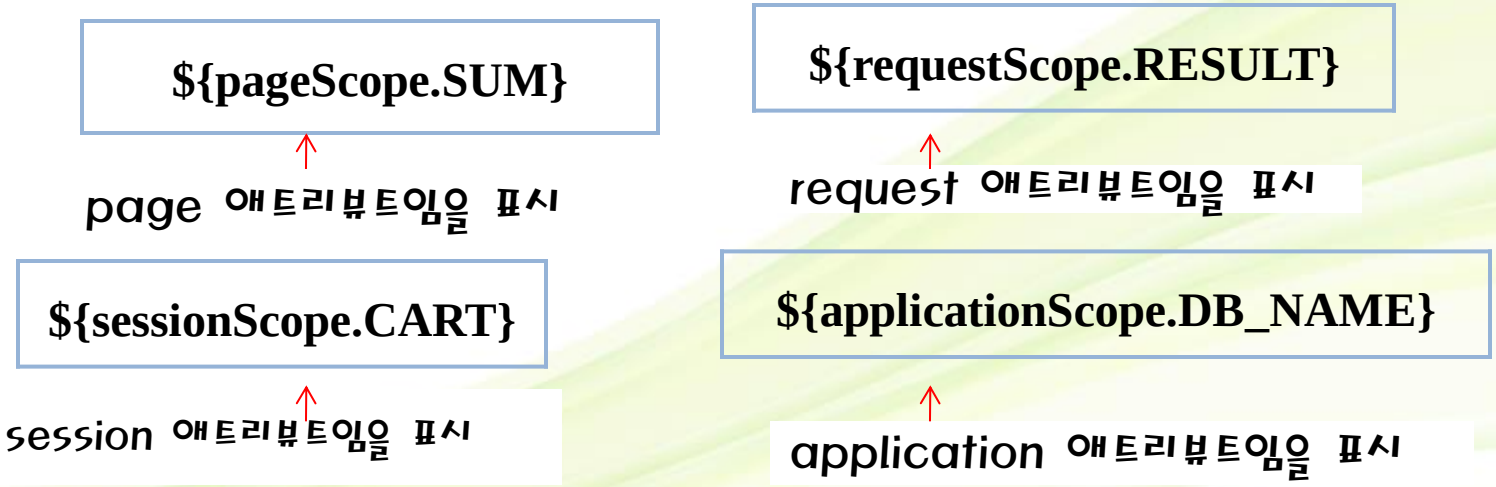
```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>EL의 결과</title>
</head>
<body>
name : ${name}
</body>
</html>
```

❖ 데이터 이름 하나로만 구성된 EL 식

- EL 식에 있는 데이터 이름을 해석하는 순서는 사용 범위가 좁은 애트리뷰트부터 점점 더 사용 범위가 넓은 애트리뷰트 순으로 진행 된다.



- 순서에 상관없이 특정한 종류의 애트리뷰트를 짚어서 출력하고 싶을 때는 다음과 같이 표시하면 된다.



❖ 익스프레션 언어의 내장 객체

익스프레션 언어의 내장 객체

내장 객체 이름	표현하는 데이터	객체의 타입
pageScope	page 애트리뷰트의 집합	Map
requestScope	request 애트리뷰트의 집합	Map
sessionScope	session 애트리뷰트의 집합	Map
applicationScope	application 애트리뷰트의 집합	Map
param	웹 브라우저로부터 입력된 데이터의 집합	Map
paramValues	웹 브라우저로부터 입력된 데이터의 집합 (똑같은 이름의 데이터가 여럿일 때 사용)	Map
header	HTTP 요청 메시지에 있는 HTTP 헤더의 집합	Map
headerValues	HTTP 요청 메시지에 있는 HTTP 헤더의 집합 (똑같은 이름의 HTTP 헤더가 여럿일 때 사용)	Map
cookie	웹 브라우저로부터 전송된 쿠키의 집합	Map
initParam	웹 애플리케이션의 초기화 파라미터의 집합	Map
pageContext	JSP 페이지의 환경 정보의 집합	PageContext

❖ 익스프레션 언어의 내장 객체

- param은 웹 브라우저에서 <FORM> 엘리먼트를 통해 입력된 데이터를 가져올 때 사용하는 내장 객체이다.
- param 객체의 사용 방법은 두 가지
  - param 뒤에 마침표를 찍고 해당 데이터 이름을 쓰는 방법
  - param 뒤에 대괄호를 치고, 그 안에 작은따옴표나 큰 따옴표로 묶은 데이터 이름을 쓰는 방법

```
${param.NUM}
```

입력 데이터의 이름

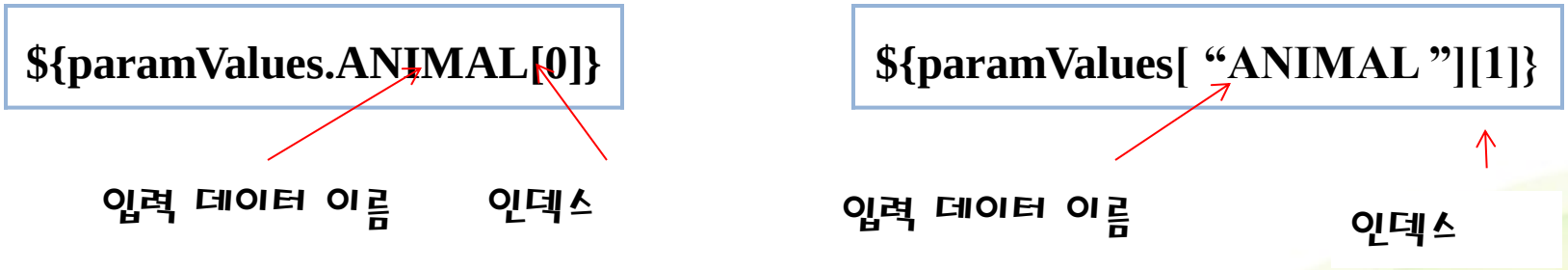
```
${param[ "COLOR "]}
```

입력 데이터의 이름

- <FORM> 엘리먼트를 통해 똑같은 이름의 데이터가 여러 개 입력되는 경우도 있는데, 그럴 때는 paramValuse 내장 객체를 사용하면 된다.

❖ 익스프레션 언어의 내장 객체

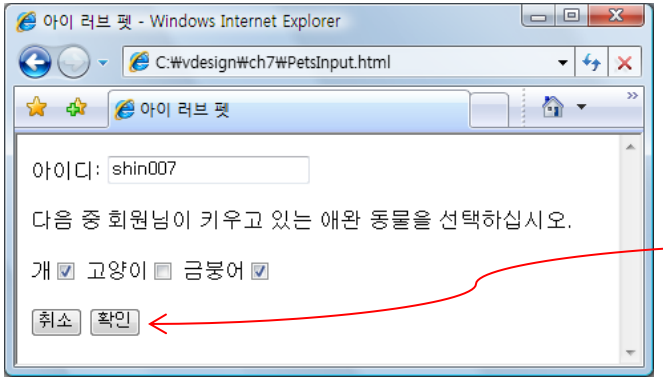
- paramValuse 내장 객체를 이용해서 데이터를 가져오는 방법은 두 가지이다.
- 하나는 객체 이름 뒤에 마침표를 찍고, 그 다음에 데이터 이름을 쓰고, 그 다음에 데이터 값의 인덱스를 대괄호로 묶어서 표시하는 것이고, 다른 하나는 객체 이름 뒤에 두 개의 대괄호를 치고 그 안에 각각 따옴표로 묶은 데이터 이름과 인덱스를 쓰는 것이다.



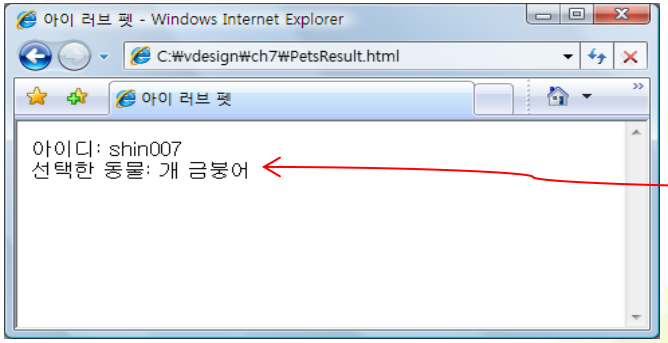
- EL 식의 인덱스가 0부터 시작하므로, 첫 번째 데이터 값을 가져오기 위해서는 인덱스를 0이라고 써야 하고, 두 번째 데이터 값을 가져오기 위해서는 1이라고 써야 한다.

❖ 익스프레션 언어의 내장 객체

- **param과 paramValues 내장 객체를 사용하는 웹 애플리케이션을 작성해보자**



① 데이터를 선택하고 '확인' 버튼을 누르면



② 선택한 데이터가 나타납니다.

애완동물 웹 애플리케이션의 화면 설계

❖ 익스프레션 언어의 내장 객체

- 앞 두 화면을 HTML문서와 JSP 페이지로 구현하고, URL을 각각 다음과 같이 정한다.

http://localhost:8181/ch05_web/PetsInput.html ← 위쪽 화면의 URL

http://localhost:8181/ch05_web/PetsResult.jsp ← 아래쪽 화면의 URL

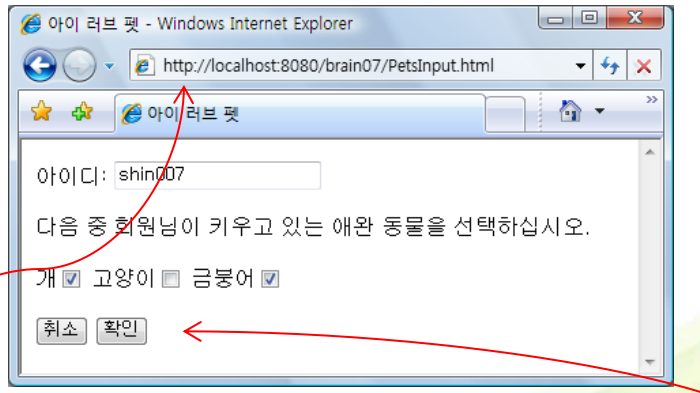
PetsInput.html

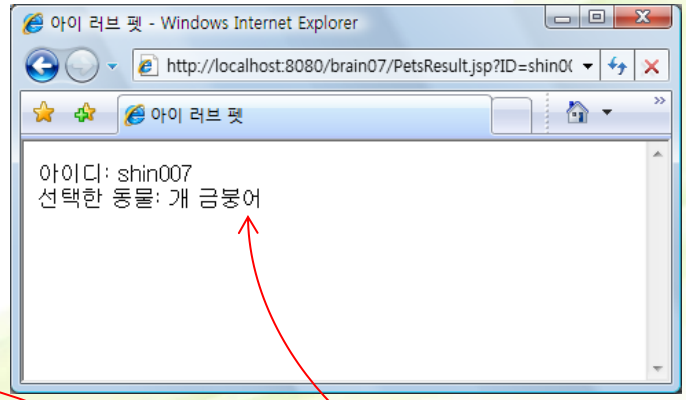
```
<HTML>   <HEAD>
      <META http-equiv= “Content-Type ”  content= “text/html;charset=euc-kr ”>
      <TITLE>아이 러브 펫</TITLE>    </HEAD>
  <BODY>
       <FORM ACTION=PetsResult.jsp>
        아이디: <INPUT TYPE=TEXT NAME=ID><BR><BR>
        다음 중 회원님이 키우고 있는 애완동물을 선택하십시오.<BR><BR>
             개<INPUT TYPE=CHECKBOX NAME=ANIMAL VALUE= “개 ” >
             고양이<INPUT TYPE=CHECKBOX NAME=ANIMAL VALUE= “고양이 ” >
             금붕어<INPUT TYPE=CHECKBOX NAME=ANIMAL VALUE=
“금붕어 ” ><BR><BR>
       <INPUT TYPE=RESET VALUE= “취소 ” >
       <INPUT TYPE=SUBMIT VALUE= “확인 ” >
    </FORM>
 </BODY>
</HTML>
```

❖ 익스프레션 언어의 내장 객체

PetsResult.jsp

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<HTML>
    <HEAD><TITLE>아이 러브 펫</TITLE></HEAD>
    <BODY>
        아이디: ${param.ID} <BR>
        선택한 동물: ${paramValues.ANIMAL[0]}
                    ${paramValues.ANIMAL[1]}
                    ${paramValues.ANIMAL[2]}
    </BODY>
</HTML>
```





① (예제 7-3)의 URL을 입력하세요.

② 데이터를 선택하고 '확인' 버튼을 누르면

③ 결과 화면이 나타날 것입니다.

❖ 익스프레션 언어의 내장 객체

- header 내장 객체는 HTTP 요청 메시지에 포함된 HTTP 헤더 값을 가져올 때 사용한다.
- header 내장 객체를 이용해서 HTTP 헤더 값을 가져오는 방법은 두 가지이다. 하나는 이 객체의 이름 뒤에 마침표를 찍고 그 다음에 해당 헤더 이름을 쓰는 것이며, 또 하나는 객체의 이름 뒤에 대괄호를 치고 그 안에 작은 따옴표나 큰 따옴표로 묶은 헤더 이름을 쓰는 것이다.

```
${header.Host}
```

HTTP 헤더 이름

```
${header[ "User-Agent "]}
```

HTTP 헤더 이름

- 첫 번째 사용 방법의 제약 사항 – HTTP 헤더 이름이 자바의 식별자 명명 규칙을 따르지 않을 때는 사용할 수 없다.

```
${header.User-Agent}
```

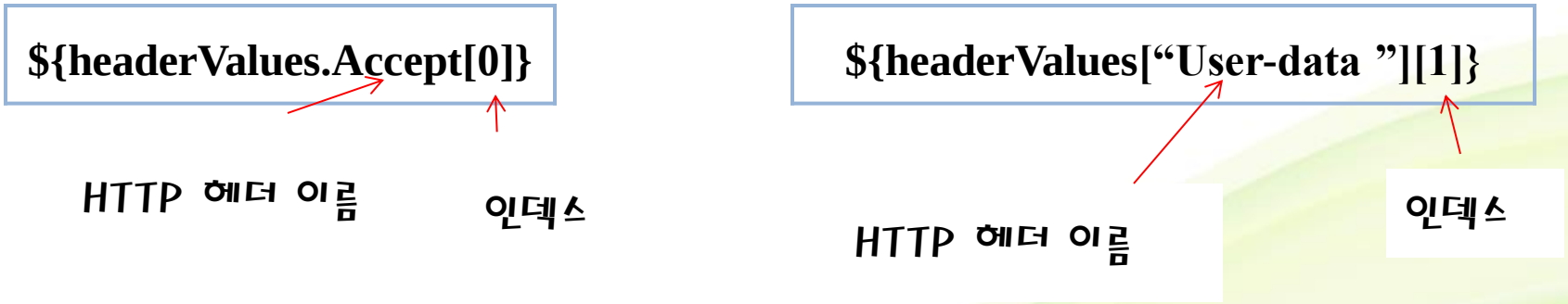
잘못된 EL 식

```
${header[ "User-Agent "]}
```

올바른 EL 식

❖ 익스프레션 언어의 내장 객체

- HTTP 요청 메시지 안에 똑같은 이름의 HTTP 헤더가 둘 이상 있을 때는 header 내장 객체 대신 headerValues 내장 객체를 사용해야 한다.
- headerValues 내장 객체의 이름 뒤에 마침표나 대괄호를 이용해서 헤더 이름을 표시하고, 그 다음에 대괄호로 묶은 인덱스를 표시해야 한다.



❖ 익스프레션 언어의 내장 객체

- cookie 내장 객체는 웹 브라우저가 웹 서버로 보낸 쿠키를 가져올 때 사용하며, 마침표와 대괄호를 이용하는 사용 방법 두 가지가 있다.

```
${cookie.CART}
```
쿠키의 이름

```
${cookie[ "USER_NAME "]}
```
쿠키의 이름

- 위 EL 식이 가져오는 것은 쿠키의 값이 아니라 쿠키 객체이므로, JSP 페이지 안에 이런 EL 식을 써 놓으면 사용자에게 아무 의미 없는 쿠키 객체의 참조 값만 출력될 것이다.

- 쿠키를 가져오는 식 뒤에 마침표를 찍고 value라고 쓰거나, 대괄호를 치고 그 안에 'value' 또는 "value" 라고 쓰면 쿠키 값이 출력된다.

```
${cookie.CART.value}
```
쿠키의 값을 가져오라는 표시

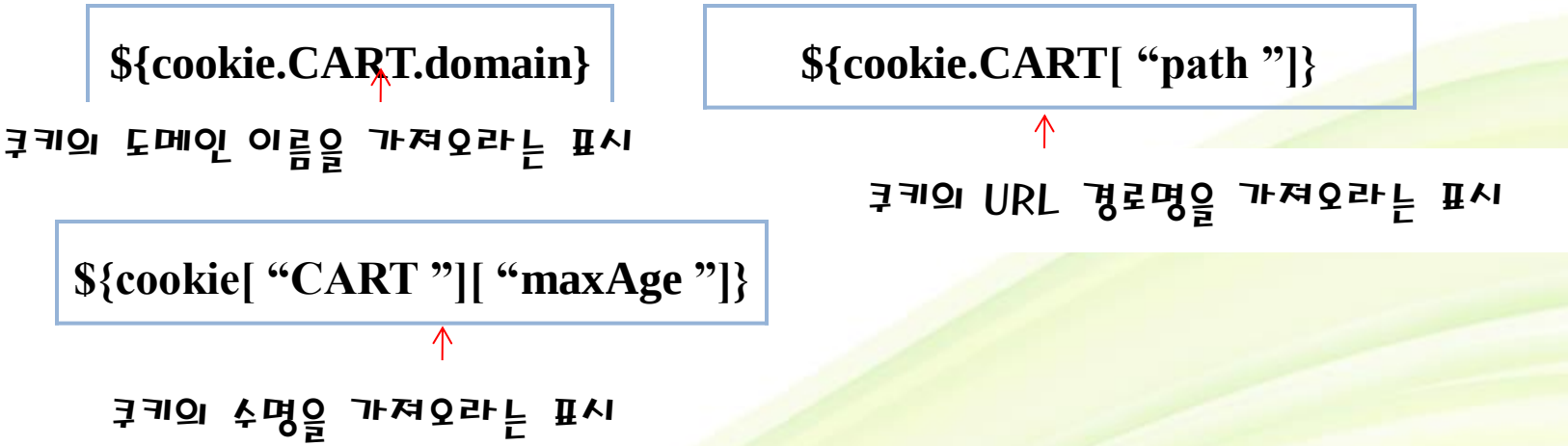
```
${cookie.CART[ "value "]}
```
쿠키의 값을 가져오라는 표시

```
${cookie[ "CART "][ "value "]}
```
쿠키의 값을 가져오라는 표시

```
${cookie[ "CART "].value}
```
쿠키의 값을 가져오라는 표시

❖ 익스프레션 언어의 내장 객체

- 쿠키 객체 안에는 쿠키 값 외에도 쿠키가 속하는 도메인 이름, URL 경로명, 쿠키의 수명 같은 중요한 정보들이 들어 있다.
- 그런 정보를 출력하기 위해서는 앞 페이지와 같은 형식의 EL 식에서 value라는 이름을 빼고 대신 domain, path, maxAge라는 이름을 써 넣으면 된다.



❖ 익스프레션 언어의 내장 객체

CookieDataWriter.jsp 쿠키 데이터를 저장하는 JSP 페이지

```
<%@page contentType= “text/html; charset=euc-kr ” %>
<%
    Cookie cookie = new Cookie( “NAME ”, “John ”);
    response.addCookie(cookie);
%>
<HTML>
    <HEAD><TITLE>쿠키 데이터 저장 프로그램</TITLE></HEAD>
    <BODY>
        쿠키 값이 설정되었습니다.
    </BODY></HTML>
```

쿠키 데이터를 웹 브라우저 쪽에 저장합니다

쿠키 데이터 값을 출력하는 JSP 페이지 CookieDataReader.jsp

```
<%@page contentType= “text/html; charset=euc-kr ” %>
<HTML>
    <HEAD><TITLE>쿠키 데이터 출력 프로그램</TITLE></HEAD>
    <BODY>
        NAME 쿠키 데이터의 값은? ${cookie.NAME.value}
    </BODY>
</HTML>
```

쿠키 데이터의 값을 가져와서 출력합니다

기본 문법

- 익스프레션 언어의 내장 객체
  - initParam은 웹 애플리케이션의 초기화 파라미터 값을 가져다가 출력할 때 사용하는 내장 객체입니다.
  - initParam 객체의 이름 뒤에 마침표나 대괄호를 이용해서 해당 초기화 파라미터의 이름을 표시합니다.

❖ web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
        <welcome-file-list>
                <welcome-file>index.html</welcome-file>
                <welcome-file>index.htm</welcome-file>
                <welcome-file>index.jsp</welcome-file>
        </welcome-file-list>
        <context-param>
        <param-name>DB_NAME</param-name>
        <param-value>Oracle</param-value>
        </context-param>

</web-app>
```

❖ initParam.jsp

```
<%@page contentType="text/html; charset=euc-kr" %>
<HTML>
    <HEAD><TITLE>애플리케이션 초기화 파라미터 예제
</TITLE></HEAD>
    <BODY>
        DB_NAME 초기화 파라미터의 값은? ${initParam.DB_NAME}
    </BODY>
</HTML>
```

익스프레션 언어의 내장 객체

- pageContext 내장 객체는 JSP 페이지의 주변 환경에 대한 정보를 제공한다. 이 내장 객체의 사용 방법은 다소 독특하다.
- pageContext 내장 객체의 타입은 PageContext라고 되어 있는데 이것은 java.servlet.jsp 패키지에 속하는 클래스 이름이다. 이 객체를 이용하면 PageContext 클래스에 속하는 get으로 시작하는 이름의 메서드를 호출 할 수 있다.
- pageContext 클래스에는 8개의 get-메서드가 있으며, EL 식을 이용해서 이 메서드들을 호출할 때는 메서드 이름 제일 앞에 있는 get이라는 단어를 떼고, 그 다음에 있는 첫 문자를 소문자로 고친 이름을 사용하면 된다.

```
${pageContext.request}
```

getRequest 메서드를 가리키는 단어

```
${pageContext[ "request "]}
```

getRequest 메서드를 가리키는 단어

  - 이 EL 식은 getRequest 메서드의 리턴값을 출력하는데, 그 값은 JSP 페이지의 request 내장 변수의 값과 동일한 객체이므로 사용자에게 아무 의미도 없는 참조값만 출력된다.

❖ 익스프레션 언어의 내장 객체

- 다음과 같은 EL 식을 이용하면 JSP 페이지의 URL 경로명을 출력할 수 있다

```
${pageContext.request.requestURI}
```

getRequestURI 메서드의 리턴값을 가져오는 단어

```
${pageContext[ "request "][ "requestURI "]}
```

getRequestURI 메서드의 리턴값을 가져오는 단어

```
${pageContext.request[ "requestURI "]}
```

getRequestURI 메서드의 리턴값을 가져오는 단어

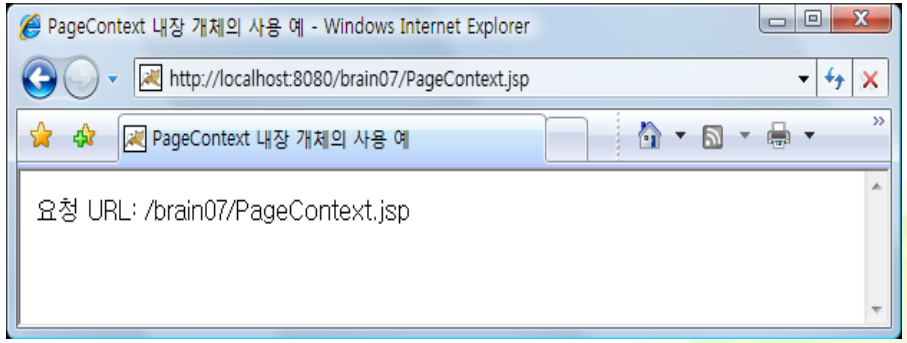
```
${pageContext[ "request "].requestURI}
```

getRequestURI 메서드의 리턴값을 가져오는 단어

❖ 익스프레션 언어의 내장 객체

PageContext.jsp

```
<%@page contentType= “text/html; charset=euc-kr ” %>
<HTML>
    <HEAD><TITLE>PageContext 내장 개체의 사용 예</TITLE></HEAD>
    <BODY>
        요청 URL: ${pageContext.request.requestURI} <BR>
    </BODY>
</HTML>
```



실행 결과

EL의연산자

❖ EL 연산자의종류와쓰임

▪ EL 연산자

연산자	설명
.	빈 또는 맵에 접근하기 위한 연산자이다.
[]	배열 또는 리스트에 접근하기 위한 연산자이다.
()	연산할 때 우선 순위를 주려고 할 때 사용한다.
x ? a : b	x의 조건이 만족하면 a를 리턴하고, 만족하지 않으면 b를 리턴한다.
Empty	값이 NULL일 경우 true를 반환한다.

산술연산자

산술 연산자	설명
+	더하기 연산자
-	빼기 연산자
*	곱하기 연산자
/ 또는 div	나누기 연산자
% 또는 mod	나머지 연산자

❖ 산술 연산자, 비교 연산자, 논리 연산자, 조건 연산자

- 자바 연산자와 동일한 기능을 하는 연사자들에 대해 알아보자.

Operators.jsp

```
<%@page contentType= “text/html; charset=euc-kr ” %>
<HTML>
    <HEAD><TITLE>익스프레션 언어 연산자 연습</TITLE></HEAD>
    <BODY>
        X = ${param.NUM1}, Y = ${param.NUM2} <BR><BR>
        X + Y = ${param.NUM1 + param.NUM2} <BR>
        X - Y = ${param.NUM1 - param.NUM2} <BR>
        X * Y = ${param.NUM1 * param.NUM2} <BR>
        X / Y = ${param.NUM1 / param.NUM2} <BR>
        X % Y = ${param.NUM1 % param.NUM2} <BR><BR>
        X가 더 큽니까? ${param.NUM1 - param.NUM2 > 0} <BR>
        Y가 더 큽니까? ${param.NUM1 - param.NUM2 < 0} <BR><BR>
        X와 Y가 모두 양수입니까? ${(param.NUM1 > 0) && (param.NUM2 > 0)}
<BR><BR>
        X와 Y가 같습니까? ${param.NUM1 == param.NUM2? “예 ” : “아니오 ” }
<BR><BR>
    </BODY>
</HTML>
```

❖ 산술 연산자, 비교 연산자, 논리 연산자, 조건 연산자

- 익스프레션 언어의 연산자 중에는 같은 모습으로 다른 기능을 하는 것도 있다

${gender == “female”}

두 값이 같으면 true 다르면 false

${ “CAR” < “CAT”}

유니코드에 따른 사전식 비교.
CAR가 먼저이므로 true

${ “CAT” > “DOG”}

유니코드에 따른 사전식 비교.
CAT이 먼저이므로 false

StringOperators.jsp

```
<%@page contentType= “text/html; charset=euc-kr ” %>
<HTML>
    <HEAD><TITLE>문자열 비교</TITLE></HEAD>
    <BODY>
        입력 문자열 : ${param.STR1}, ${param.STR2} <BR><BR>
        두 문자열이 같습니까? ${param.STR1 == param.STR2} <BR>
        어느 문자열이 먼저입니까? ${param.STR1 < param.STR2 ? param.STR1 : param.STR2}
    </BODY>
</HTML>
```

연산자

- 엠프티 연산자
  - empty라는 단어 형태의 엠프티 연산자는 데이터의 존재 여부를 확인하는 단항 연사자이다. 피연산자인 데이터 이름은 empty라는 연산자 이름 뒤에 써야 합니다.

```
${empty NAME}
```

데이터 이름

emptyOperator.html

```
<!DOCTYPE html><html><head>
<meta charset="EUC-KR"><title>Insert title here</title></head>
<body>
    <form action="emptyOperator.jsp">
        <input type="text" name="ID"><p>
         <input type="submit" value="확인">
    </form>
</body>
</html>
```

❖ emptyOperator.jsp

```
<%@page contentType="text/html; charset=euc-kr" %>
<HTML>
    <HEAD><TITLE>엠프티 연산자</TITLE></HEAD>
    <BODY>
        안녕하세요, ${empty param.ID ? "guest" : param.ID}님
    </BODY>
</HTML>
```

❖ 연산자의 우선순위를 바꾸는 괄호 연산자

- 여러 개의 연산자가 포함된 수학식에는 왼쪽에서부터 오른쪽으로 계산되는 것이 순서이지만, 가감승제 연산자가 뒤섞여 있을 때는 곱셈, 나눗셈이 덧셈, 뺄셈보다 먼저 계산된다. 익스프레션 언어의 연산자에도 마찬가지로 우선순위가 있다.
- EL 식 안에 여러 개의 연산자가 있으면 왼쪽부터 오른쪽으로 순서대로 처리되지만, 우선순위가 다른 연산자가 섞여 있으면 높은 우선순위의 연사자가 먼저 처리된다.

```
${2 + 3 * 4}
```

곱셈이 먼저 수행됩니다

- 우선순위를 바꾸기 위해서는 수학식과 마찬가지로 괄호를 사용하면 된다.

```
${(2 + 3) * 4}
```

덧셈이 먼저 수행됩니다.

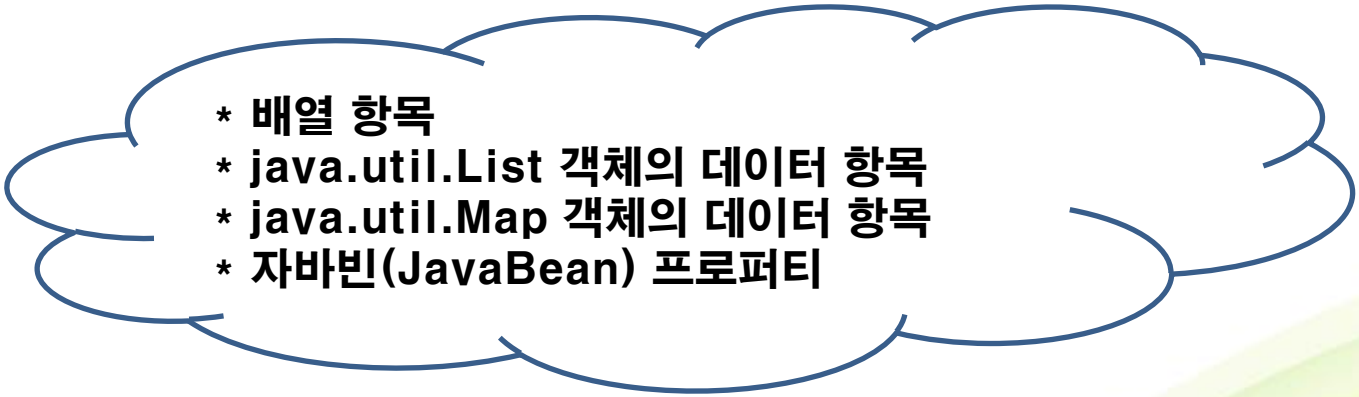
❖ 연산자의 우선순위를 바꾸는 괄호 연산자

- 익스프레션 언어의 연산자 우선 순위

우선순위	연 산 자
↑높음	[ ]
	( )
	-(부호) ! not empty
	* / % div mod
	+ -
	< > <= >= lt gt le ge
	== != eq ne
	&& or
	|| or
↓낮음	? :

❖ 대괄호 연산자와 마침표 연산자

- 대괄호와 마침표 연산자의 용도: 자바에서는 배열 항목과 객체 멤버를 가리키기 위해 사용되지만, 익스프레션 언어에서는 다음과 같은 데이터 항목을 가리키기 위해 사용된다.



- 배열과 java.util.List 객체의 데이터 항목을 가리킬 때는 반드시 대괄호 연산자를 이용해야 한다.
- java.util.Map 객체의 데이터 항목과 자바빈 프로퍼티를 가리킬 때는 대괄호 연산자와 마침표 연산자 중 하나를 사용할 수 있다.

❖ 대괄호 연산자와 마침표 연산자

- 대괄호 연산자를 이용해서 배열 항목을 가져다가 출력하는 방법 – 자바에서와 마찬가지로 배열 이름 다음에 대괄호로 묶은 인덱스를 표시하면 된다.

- EL 식의 인덱스는 자바에서와 마찬가지로0부터 시작하고, 0은 첫 번째 데이터 항목을 가리킨다.

❖ 대괄호 연산자와 마침표 연산자

Winners.jsp

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<%
        String winners[] = new String[3];
        winners[0] = “이수현” ;
        winners[1] = “정세훈” ;
        winners[2] = “김진희” ;
        request.setAttribute( “WINNERS ”, winners);
        RequestDispatcher dispatcher =
request.getRequestDispatcher( “WinnersView.jsp ”);
        dispatcher.forward(request, response);
%>
```

배열을 애트리뷰트 형태로 저장합니다

호 출

WinnersView.jsp

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<HTML>
    <HEAD><TITLE>우승자 명단</TITLE></HEAD>
    <BODY>
        <H3>우승자 명단</H3>
        1등. ${WINNERS[0]} <BR>
        2등. ${WINNERS[1]} <BR>
        3등. ${WINNERS[2]} <BR>
    </BODY>
</HTML>
```

배열 항목을 가져다가 출력하는 EL 식입니다

❖ 대괄호 연산자와 마침표 연산자

Fruits.jsp

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<%@page import= “java.util.* ” %>
 <%
        ArrayList<String> items = new ArrayList<Strin
        items.add( “딸기” );
        items.add( “오렌지” );
        items.add( “복숭아” );
        request.setAttribute( “FRUITS ”, items);
        RequestDispatcher dispatcher = request.getRequestDispatcher( “FruitsView.jsp ”);
        dispatcher.forward(request, response);
%>
```

List 객체를 애트리뷰트 형태로 저장합니다

호출

```
<%@page contentType= “text/html; charset=euc-kr ”%>
<HTML>
     <HEAD><TITLE>인기 상품 목록</TITLE></HEAD>
     <BODY>
          <H3>이달에 가장 많이 팔린 과일입니다.</H3>
          1위. ${FRUITS[0]} <BR>
          2위. ${FRUITS[1]} <BR>
          3위. ${FRUITS[2]} <BR>
     </BODY>
</HTML>
```

List 객체의 항목을 가져다가 출력하는 EL 식입니다

❖ 대괄호 연산자와 마침표 연산자

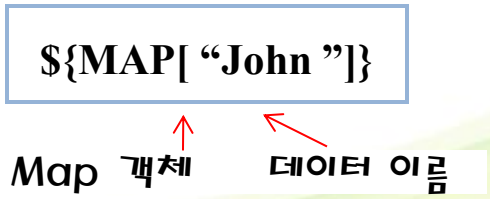
- java.util.Map는 java.util.List와 마찬가지로 자바의 표준 라이브러리에 있는 인터페이스 이름이며, 이 인터페이스에는 여러 데이터 항목을 <이름, 값> 쌍으로 저장해서 관리할 수 있는 메서드들이 있습니다.

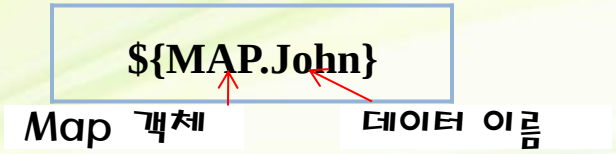
```
HashMap<String, Integer> map = new HashMap<String, Integer>();
map.put( “Edgar ”, new Integer(95));
map.put( “Thomas ”, new Integer(100));
map.put( “John ”, new Integer(75));
```

이름이 String 타입이고, 값이 Integer 타입인 데이터를 저장할 수 있는 HashMap 객체를 만듭니다

HashMap 객체에 3개의 데이터를 저장합니다

- Map 객체에 저장한 데이터 값을 가져오기 위해서는 대괄호 연산자나 마침표 연산자를 이용해서 데이터의 이름을 지정해야 합니다.





❖ mapCreate.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>
<%@page import="java.util.*" %>
<%
HashMap<String, String> map = new HashMap<String, String>();
map.put("Park", "목동");
map.put("Jasica", "크라이스 처치");
map.put("Susan", "시드니");
request.setAttribute("ADDRESS", map);
RequestDispatcher dispatcher =
request.getRequestDispatcher("mapView.jsp?NAME=Park");
dispatcher.forward(request, response);
%>
```

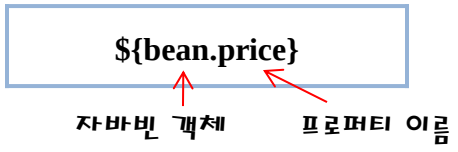
❖ mapView.jsp

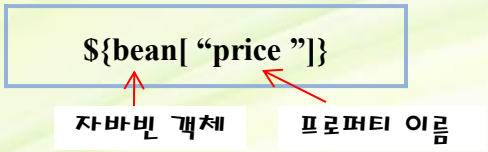
```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
   ${param.NAME}의 주소는? ${ADDRESS[param.NAME]} <p>
   ${ADDRESS.Jasica} <p>
   ${ADDRESS.Susan} <p>
</body>
</html>
```

표현언어에서 객체 메소드 호출

- 자바빈(JavaBean)이란 JavaBean 규격서에서 따라 작성된 자바 클래스를 의미합니다.
- JavaBean 규격서에 따르면 JavaBean 클래스에는 파라미터 없는 생성자가 반드시 있어야 하고, 이 클래스 외부에서 필드(클래스의 멤버 변수)에 접근할 필요가 있을 때는 직접 접근할 수 없고, 반드시 메서드를 통해서 접근해야 합니다.

- 대괄호 연산자와 마침표 연산자
  - EL 식에서 자바빈 프로퍼티의 값을 가져올 때는 프로퍼티 이름을 사용해야 한다. 즉, 자바빈 객체의 이름 뒤에 마침표를 찍고 그 다음에 프로퍼티 이름을 쓰거나, 자바빈 객체의 이름 뒤에 대괄호를 치고 그 안에 따옴표로 묶은 프로퍼티 이름을 써야 합니다.





Thermometer.java

```
package el;
import java.util.HashMap;
import java.util.Map;
public class Thermometer {
	Map<String, Double> th = new HashMap<>();
	public void setTh(String loc, double value) {
		th.put(loc, value);
	}
	public double getTh(String loc) {
		return th.get(loc);
	}
	public double getFarenheit(String loc) {
		return th.get(loc)*1.8 +32;
	}
	public String info() {
		return "온도계";
	}
}
```

Thermometer.jsp

```
<%@ page import="el.Thermometer"%>
<%@ page contentType="text/html; charset=euc-kr"%>
<%
Thermometer thermometer = new Thermometer();
request.setAttribute("t", thermometer);
%>
<html>
<head>
<title>온도 변환 예제</title>
</head>
<body>
${t.setCelsius('서울', 27.3)}
서울 온도: 섭씨${t.getCelsius('서울')}도 / 화씨 ${t.getFahrenheit('서울')}

<br/>
정보: ${t.info}
</body>
</html>
```

람다 사용

1. EL3.0 에서부터는 람다 사용 가능
2. 함수적 스타일의 람다식 작성법

```
(타입 매개변수, ...) -> { 실행문; ... }
```

```
(int a) -> { System.out.println(a); }
```

1) 매개변수 타입은 런타임 시에 대입 값에 따라 자동 인식하므로 생략 가능
2) 하나의 매개변수만 있을 경우에는 괄호( ) 생략 가능
3) 하나의 실행 문장만 있다면 중괄호 { } 생략 가능
4) 매개변수 없다면 괄호 ( ) 생략 불가
5) 리턴 값이 있는 경우 return 문 사용
6) 중괄호 { }에 return 문만 있을 경우, 중괄호 생략 가능
7) 참조형 변수에 대입 가능
8) EL에서 람다를 변수에 할당하면 변수의 해시코드가 출력됩니다.

스트림

1. 스트림은 JDK 1.8에서 추가된 컬렉션의 저장 요소를 하나씩 참조하여 람다식으로 처리할 수 있는 반복자
2. Iterator는 컬렉션의 데이터를 접근할 수 있도록 포인터만 제공하지만 스트림은 데이터를 가지고 작업을 수행 할 수 있도록 해줍니다.
3. List 인터페이스의 stream() 메소드를 호출하면 Stream 객체를 생성해서 리턴해 줍니다.
4. 스트림의 특징
    1) 람다식으로 요소 처리 코드를 제공
    2) 내부 반복자를 사용하므로 병렬처리 쉬움
    3) 스트림은 중간 처리(매핑, 필터링, 정렬)와 최종 처리(반복, 카운팅, 평균, 합계)를 할 수 있다.

스트림

중간 처리 메소드

1. distinct(): 중복 제거
2. filter(): 원하는 조건에 맞는 데이터 추출
3. map(): 데이터 변환
4. sorted(): 데이터 정렬
5. limit(): 데이터 개수 제한

스트림

최종 처리 메소드

1. toList(), toSet(), toMap(): 자료구조로 리턴
2. sum(): 합계
3. count():개수
4. average(): 합계
5. min(), max(): 최대, 최소
6. average(), min(), max()는 Optional 타입을 리턴하기 때문에 get()을 이용해서 값을 가져오면 데이터가 있으면 데이터가 리턴되고 없으면 ELException을 발생시킵니다.
7. orElse(값): Optional 타입의 값이 없으면 매개변수를 리턴

EL3.0에 추가된 기능

1. 세미콜론 연산자

   ${1+1;10+10} 세미콜론 뒤만 결과 출력

2. 할당연산자

   변수 생성할 때 사용

   ${var1=10}

   ${var1=10} ${var1} 선언하고 사용하면 두 번 출력 될 수 있음

   따라서 세미콜론 연산자를 사용하여 선언해야 출력하지 않음

   ${var1=10;""}

EL3.0에 추가된 기능

3. 람다,스트림 사용

```
public class Member {
   private String name;
   private int age;
   public Member(String name, int age) {
          this.name = name;
          this.age = age;
   }
   public String getName() {
        return name;
   }
   public int getAge() {
       return age;
  }
   public String toString() {
       return "["+this.name+","+this.age+"]";
   }
}
```

EL3.0에 추가된 기능

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ page import="ch07.Member, java.util.*" %><html>
<head><title>EL min, max</title></head>
<body>
${ vals = [20, 17, 30, 2, 9, 23]  }<br>
${ vals.stream().min().get() }
<hr>
<%
    List<Member> memberList = Arrays.asList(
        new Member("홍길동", 20), new Member("이순신", 54),
        new Member("유관순", 19), new Member("왕건", 42) );
    request.setAttribute("members", memberList);
%>
${ maxAgeMemOpt = members.stream().max((m1, m2) ->
m1.age.compareTo(m2.age)) ; ''}
${ maxAgeMemOpt.get().name } (${maxAgeMemOpt.get().age}세)
</body>
</html>
```

EL3.0에 추가된 기능

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ page import="ch07.Member,java.util.*" %><html>
<head><title>EL 정렬</title></head><body>
    ${ value =[20, 17, 30, 2, 9, 23]}<br>
    작은순서 : ${ minSort=value.stream().sorted().toList() }<br>
    큰순서 : ${maxSort=value.stream().sorted((x,y)-> x<y?1:-1).toList() }<p>
    <%
        List<Member> list = new ArrayList<>();
        list.add(new Member("이순신",23));
        list.add(new Member("강감찬",18));
        list.add(new Member("을지문덕",45));
        list.add(new Member("양만춘",21));
        request.setAttribute("list", list);
    %>
    작은 순서 : ${minageMan=list.stream()
        .sorted((m1,m2)->m1.age.compareTo(m2.age)).toList() }<br>
    큰순서 :  ${maxageMan=list.stream()
        .sorted((m1,m2)->m2.age.compareTo(m1.age)).toList() }<br>
    작은 순서 : ${minMan=list.stream()
         .sorted((m1,m2)->m1.name.compareTo(m2.name)).toList() }<br>
    큰순서 :  ${maxMan=list.stream()
        .sorted((m1,m2)->m2.name.compareTo(m1.name)).toList() }<br>
</body></html>
```