JSP 프로그래밍

(재)대덕인재개발원

15 표현 언어(Expression Language)

- 1. 표현 언어란?
- 2. 표현 언어의 기본 객체
- 3. 표현 언어의 기본
- 4. 표현 언어에서 클래스 함수 호출하기
- 5. 표현 언어의 사용법
- 6. 표현 언어 비활성화 방법

15.1 표현 언어

- 표현 언어는 값을 표현하는 데 사용되는 새로운 스크립트 언어로서 JSP의 기본 문법을 보완하는 역할을 한다.
- 스크립트 요소 중의 하나인 표현식 보다 간결하고 표현식 대신 사용할 수 있다.(<%= expr %> ➔ ${ expr })
- 표현 언어의 기능
  - JSP의 네 가지 기본 객체가 제공하는 영역의 속성 사용
  - 집합 객체에 대한 접근 방법 제공
  - 수치 연산, 관계 연산, 논리 연산자 제공
  - 자바 클래스 메서드 호출 기능 제공
  - 표현 언어만의 기본 객체 제공

15.1.1 표현 언어의 기본 문법

- 표현 언어는 $와 표현식 그리고 괄호('{' 와 '}')를 사용하여 값을 표현한다.

```
${expr}  or  #{expr}
```

```
<jsp:include page="/module/${skin.id}/header.jsp"  flush="true" />
<b>${sessionScope.member.id}</b> 님 환영합니다.
```

```
<%
  Member m = new Member();
  m.setName("이름 1");
%>
<c:set var="m" value="<%= m %>" />
<c:set var="name" value="${ m.name }" />  <%-- 이 시점에 곧 바로 값 계산 --%>
<% m.setName("이름 2"); %>
${name} <%-- name의 값은 "이름1" --%>
```

15.2 표현언어의 기본 객체

- EL이 제공하는 11개의 기본 객체

기본 객체	설 명
pageContext	JSP의 page 기본 객체와 동일하다.
pageScope	pageConext 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체
requestScope	request기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체
sessionScope	session기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체
applicationScope	application기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체
param	요청 패러미터의 <패러미터이름, 값> 매핑을 저장한 Map 객체. 값의 타입은 String으로서 request.getParameter(이름)의 결과와 동일하다.
paramValues	요청 패러미터의 <패러미터이름, 값배열> 매핑을 저장한 Map 객체. 값의 타입은 String[]으로서 request.getParameterValues(이름)의 결과와 동일하다.
header	요청 정보의 <헤더이름, 값> 매핑을 저장한 Map 객체. request.getHeader(이름)의 결과와 동일하다.
headerValues	요청 정보의 <헤더이름, 값배열> 매핑을 저장한 Map 객체. request.getHeaders(이름)의 결과와 동일하다.
cookie	<쿠키 이름, Cookie> 매핑을 저장한 Map 객체. request.getCookies()로 구한 Cookie배열로 부터 매핑을 생성한다.
initParam	초기화 패러미터의 <이름, 값> 매핑을 저장한 Map 객체. application.getInitParameter(이름)의 결과와 동일하다.

15.2.1 표현 언어의 기본 객체

- EL에서는 기본 객체의 값에 접근할 때 자바빈 객체의 프로퍼티를 사용한다.

```
<%@ page contentType="text/html; charset=utf-8"  %>
<%
  request.setAttribute("name", "이산");
%>

<html>
<head><title> EL Object</title></head>
<body>

요청 URI : ${pageConext.request.requestURI}<br>
request의 name 속성 : ${requestScope.name}<br>
code 패러미터 : ${param.code}
ID Cookie의 값 : ${cookie.ID.value} <!– Cookie 객체의 getValue() 메서드의 리턴 값-->
</body>
</html>
```

15.3 표현 언어의 기본

- EL도 일종의 스크립트 언어로서 자료 타입, 수치 연산자, 논리 연산자, 비교 연산자 등을 제공한다.

15.3.1 EL의 데이터 타입

- EL은 불리언 타입, 정수 타입, 실수 타입, 문자열 타입 그리고 널 타입의 5가지 타입을 제공한다.

기본 객체	표현방식
Boolean 타입	true 와 false
정수 타입	0 ~ 9로 이루어진 정수 값 음수의 경우 '-'가 붙음
실수 타입	0 ~ 9로 이루어져 있으며, 소수점('.') 을 사용할 수 있고, 3.24e3과 같이 지수형으로 표현 가능하다.
문자열 타입	따옴표(' 또는 ") 로 둘러싼 문자열. 만약 작은 따옴표(')를 사용해서 표현할 경우 값에 포함된 작은 따옴표는 \' 와 같이 \ 기호와 함께 사용해야 한다. \ 기호 자체는 \\ 로 표시한다.
널 타입	null

15.3.2 객체에 접근하기

- EL 언어는 객체에 저장된 값에 접근할 때 점(.)이나 대괄호([])를 사용한다.
- ${cookie.ID.value}
- cookie.name 과 cookie[name]은 같은 결과를 낸다.

15.3.3 객체의 탐색

- EL 언어에서 PAGE, REQUEST, SESSION, APPLICATION 영역에 저장된 속성에 접근할 때에는 pageScope, requestScope, sessionScope, applicationScope 기본 객체를 사용한다고 했다.
- ${pageScope.NAME} : PAGE 영역에 저장되어 있는 NAME 속성의 값을 참조.
- 영역을 나타내는 EL 기본 객체를 사용하지 않고 이름만 지정할 경우 EL은 네 개의 영역을 차례대로 검색해서 속성이 존재하는지 확인한다.

15.3.4 연산자

- EL의 연산자 종류

연산	표기
산술연산자	+, -, *, / (div), % (mod)
논리연산자	and (&&), or (||), not
관계 연산자	== (eq), != (ne), < (lt), > (gt), <= (le), >= (ge)
조건부	A ? B : C (A가 참이면 B, 거짓이면 C 를 수행한다.)

15.3.4 수치 연산자

- +, -, *, / 또는 div, % 또는 mod
- 숫자가 아닌 객체와 수치 연산자를 사용할 경우 객체를 숫자 값으로 변환한 후 연산자를 수행한다.
  - ${"10" + 1}
- 객체가 null이면 0으로 처리한다.
  - ${null + 1}
- 나눗셈의 경우 피연사자가 모두 정수라 하더라도 피연산자를 Double 타입으로 변환한 뒤 연산을 수행한다.
  - ${3 / 2}

15.3.5 비교 연산자

- == 또는 eq
- != 또는 ne
- < 또는 lt
- > 또는 gt
- <= 또는 le
- >= 또는 ge
- 문자열을 비교할 경우 String.compareTo() 메서드 사용
  - ${someValue == '2004'}

15.3.6 논리 연산자

- && 또는 and
- || 또는 or
- ! 또는 not

15.3.7 empty 연산자

- empty 연산자는 검사할 객체가 텅 빈 객체인지를 검사하기 위해 사용한다. = empty <값>
- <값>이 null이면 true를 리턴.
- <값>이 빈 문자열(“”)이면 true를 리턴.
- <값>이 길이가 0이면 true를 리턴.
- <값>이 빈 Map이면 true를 리턴.
- <값>이 빈 Collection이면 true를 리턴.
- 이 외의 경우에는 false를 리턴한다.

15.3.8 비교 선택 연산자

- 형식 : <수식> ? <값1> : <값2>
- <수식>의 결과 값이 true이면 <값1>을 리턴하고, false이면 <값2>를 리턴한다.
- 연산자 우선순위
  1. [] .
  2. ()
  3. - (단항) not ! empty
  4. * / div % mod
  5. + - (이항)
  6. < > <= >= lt gt le ge
  7. == != eq ne
  8. && and
  9. || or
  10. ? :

15.3.9 특수 문자 처리하기

- 표현식 기본 문법 : \${expr} 또는 \#{expr}
- “${expr} ” 과 “#{expr}”을 출력한다.

15.4 표현 언어에서 클래스 함수 호출하기

- 표현식에서의 자바 클래스 사용

```
<%
        SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
        Date date = new Date();
%>
……..
오늘은 <%= formatter.format(date) %>
```

- EL에서는 위 코드처럼 직접적으로 자바 코드를 사용할 수 없다.
- EL에서 클래스의 메서드를 사용하기 위해서는 클래스의 메서드는 static으로 정의해야 하며, static이 아닌 메서드는 사용할 수 없다.

15.4.1 표현 언어에서 클래스 함수 호출하기

- static 메서드를 가진 예제 클래스

```
package com.san.util;

import java.text.SimpleDateFormat;
import java.util.Date;

public class DateUtil{
        private static SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");

        public static String format(){
                return formatter.format(date);
        }
}
```

15.4.2 함수를 정의한 TLD 파일 작성

- 클래스 파일을 작성 후 TLD(Tag Library Descriptor) 파일을 작성한다.
- TLD 파일은 태그 라이브러리에 대한 설정 정보를 담고 있다.

 # webContent/WEB-INF/tlds/el-functions.tld

```
<?xml version="1.0" encoding="utf-8" ?>
<taglib xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_1.xsd"
        version="2.1">

        <description>EL에서 함수실행</description>
        <tlib-version>1.0</tlib-version>
        <short-name>Elfunction</short-name>

        <function>
                <description>Date 객체 포맷팅</description>
                <name>dateFormat</name>
                <function-class>com.san.util.DateUtil</function-class>
                <function-signature>java.lang.String format(java.util.Date)</function-signature>
        </function>
```

15.4.3 web.xml 파일에 TLD 내용 추가하기

- TLD 파일을 작성한 다음에는 web.xml 파일에 TLD 파일에 대한 내용을 추가해 주어야 한다.

 # webContent/WEB-INF/web.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                http://java.sun.com/xml/ns/j2ee/web-app_2_5.xsd"
        version="2.5">

        <jsp-config>
                <taglib>
                        <taglib-uri>
                                /WEB-INF/tlds/el-functions.tld
                        </taglib-uri>
                        <taglib-location>
                                /WEB-INF/tlds/el-functions.tld
                        </taglib-location>
                </tablib>
        </jsp-config>
</web-app>
```

15.4.4 EL에서 함수 사용하기

- EL에서 함수를 사용하기 위해서는 다음과 같은 형태를 갖는다.

```
<%@ taglib prefix="pre" uri="….." %>
……

${pre:functionName(arg1, arg2, ….)}
……
```

- taglib 디렉티브는 web.xml 파일에서 설정한 태그 라이브러리를 JSP 페이지에서 사용한다는 것을 명시한다.
- prefix 속성은 태그 라이브러리를 구분할 때 사용할 접두어를 나타낸다.
- EL에서 태그 라이브러리에 정의된 함수를 사용하려면 ${태그라이브러리접두어:함수이름(인자, 인자,…)}의 코드를 사용하면 된다.

15.4.4 EL에서 함수 사용하기

- EL에서 함수를 사용 예

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page session="false" %>
<%@ taglib prefix="elfunc" uri="/WEB-INF/tlds/el-functions.tld" %>

<%
        java.util.Date today = new java.util.Date();
        request.setAttribute("today", today);
%>
<html><head><title>EL 함수호출</title></head>
<body>

오늘은 <b>${ elfunc:dateFormat(today) }</b>입니다.

</body>
</html>
```

- EL에서 태그 라이브러리로 지정한 함수를 호출할 때는 사용할 클래스의 메서드 이름이 아닌 TLD 파일의 <name> 태그에서 지정한 이름을 사용한다.

15.5 표현 언어의 사용법

15.5.1 <jsp:forward>나 <jsp:include>에 속성으로 전달한 값 활용

- 페이지에서 request 기본 객체의 속성으로 전달 받은 값을 사용할 경우

```
<%-- 포함되는 JSP에서 스크립트릿과 표현식 사용 --%>
<%
        MemberInfo memberInfo = (MemberInfo)request.getAttribute("memberInfo");
%>
이름 : <%= memberInfo.getName() %>
……

<%--  포함되는 JSP에서 EL 사용--%>
이름 :  ${memberInfo.name}
```

15.5.2 액션 태그나 커스텀 태그의 속성값으로 사용하기

- EL을 이용한 액션 태그에 속성값 지정하기

```
<jsp:include page='<%= "/layout" + layout.getModuleName() + ".jsp"%>' flush="true" />
```

```
<jsp:include page="/layout/${layout.moduleName}.jsp" flush="true" />

<jsp:include page="/${folderName}/${layout.moduleName}.jsp" flush="true" />
```

15.5.3 함수 호출을 사용한 값의 포맷팅

- EL을 이용한 자바 메서드 호출

```
<%= StringUtil.substring(0, 20, false, /* HTML 코드 없앰 */)%>
${ elfunc:substring(0. 20, false)}
```

15.6 표현 언어 비활성화 방법

- JSP 규약은 ${expr}나 #{expr}과 같은 EL을 비활성화 시키는 세 가지 방법을 제공한다.
  - web.xml 파일에 비활성화 옵션 지정하기
  - JSP 페이지에 비활성화 옵션 지정하기
  - web.xml 파일을 서블릿 2.3 또는 2.4 버전에 맞게 작성하기

15.6.1 web.xml 파일에 EL 비활성호 옵션 추가하기

- web.xml 파일에 EL 비활성화 옵션 추가

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app …>
        <jsp-config>
                <jsp-property-group>
                        <url-pattern>/oldversion/*</url-pattern>
                        <el-ignored>true</el-ignored>
                </jsp-property-group>

                <!-- #{expr} 형식의 EL만 문자열 처리 -->
                <jsp-property-group>
                        <url-pattern>/oldversion2_4/*</url-pattern>
                        <deferred-syntax-allowed-as-literal>
                           true
                        </deferred-syntax-allowed-as-literal>
                </jsp-property-group>

        </jsp-config>
</web-app>
```

15.6.2 JSP 페이지에서 EL 비활성화 시키기

- web.xml 파일에 EL을 비활성화 했는지의 여부와 상관없이 JSP 페이지의 page 디렉티브를 이용해서 EL을 활성화시키거나 비활성화 시킬 수 있다.

```
<%-- EL을 비활성화 시키는 경우--%>
<%@ page isELIgnored="true"%>

<%-- #{expr} 형식의 EL을 비활성화 시키는 경우--%>
<%@ page deferredSyntaxAllowedAsLiteral="true"%>
```

15.6.3 web.xml 파일의 버전에 따른 EL 처리

- web.xml 파일이 따르는 서블릿 버전에 따라 EL 지원 여부가 결정된다.
  - 서블릿 2.3 버전(JSP 1.2)의 web.xml : EL을 지원하지 않는다.
  - 서블릿 2.4 버전(JSP 2.0) 의 web.xml : #{expr} 을 지원하지 않는다.
  - 서블릿 2.5 버전의 web.xml : ${expr} 및 #{expr} 을 지원한다.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
…..
</web-app>
```

```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/j2ee
                http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
…..
</web-app>
```