Chapter

10

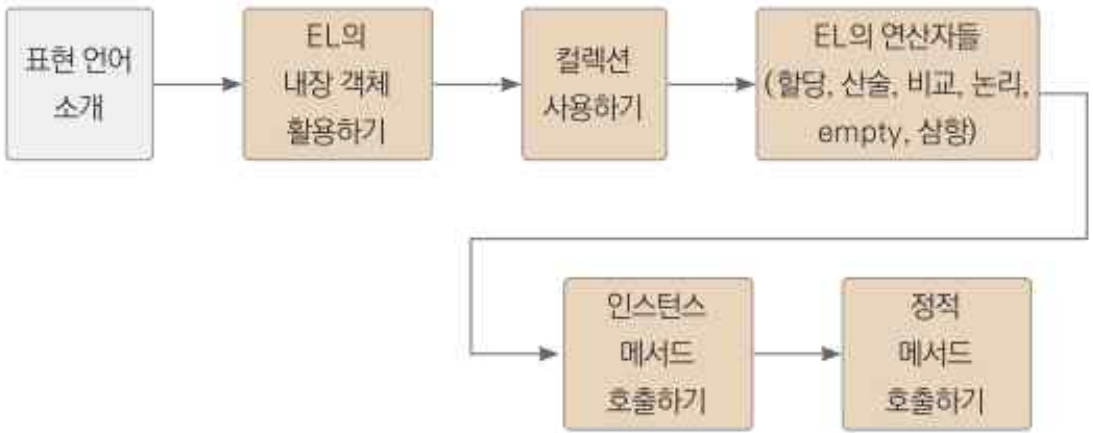
표현 언어(EL)
Expression Language

학습 목표 및 순서

▣ 학습 목표

. 변수의 값을 표현식보다 편하게 출력할 수 있게 해주고, 4가지 영역에 저장된 속성도 더 쉽게 읽을 수 있는 표현 언어의 문법을 학습.

▣ 학습 순서



학습 목표 및 순서

▣ 활용 사례

. 표현 언어는 모델2 방식으로 웹 애플리케이션을 개발할 때 주로 사용

. 4가지 영역(page, request, session, application)에 저장된 속성에 접근할 때 사용

. JSTL(11장)과 서블릿(13장)을 학습한 후 표현 언어를 확실하게 활용하게 될것임

10.1 표현 언어란?

▣ 표현 언어란..??

- 표현 언어(Expression Language, EL)는 변수의 값을 출력할 때 사용하는 스크립트 언어
- 표현식(<%= %>)과 다른점은 4가지 영역에 저장된 값을 출력하는 것
- 사용법이 매우 간결하고, 예외와 형변환에 관대하다는 특징이 있음
- 기능
  - JSP 내장 객체의 영역에 담긴 속성을 사용
  - 산술 연산, 비교 연산, 논리 연산이 가능
  - 자바 클래스에 정의된 메서드를 호출할 수 있음
  - 표현 언어만의 객체를 통해 JSP와 동일한 기능 수행

▣ 기본사용법

```
${ 속성 }
```

- 영역에 저장된 속성을 출력
- 스크립틀릿에서 저장한 변수는 사용할 수 없음

10.1 표현 언어란?

▣ 기본사용법

형식	${ 속성 }

```
<h2>${ reuqestScope.saveVar }</h2> ⇐ request영역에 저장된 속성 출력

<c:set var="elVar" value="${ elVar }" /> ⇐ JSTL과 함께 사용

<jsp:include page="${ pathVar }" /> ⇐ 액션태그와 함께 사용
```

```
<%!
void myMethod(${ errorVar }) { } ⇐ 선언부에서 사용했으므로 에러 발생
%>

<%@ include file="${ errorVar }" %> ⇐ 스크립틀릿에서 사용했으므로 에러 발생

<%= ${ errorVar } %> ⇐표현식에서 사용했으므로 에러 발생
```

10.1 표현 언어란?

▣ 객체 표현방식

. EL에서 객체를 표현할때는 .(점) 이나 [](대괄호)를 사용

```
${ param.name } ⇐ 점으로 표현
${ param["name"] } ⇐ 더블쿼테이션 사용
${ param['name'] } ⇐ 싱글쿼테이션 사용
```

. 속성명에 특수기호나 한글이 포함된 경우에는 대괄호만 사용 가능

```
${ header["user-agent"] } ⇐ 속성명에 특수기호가 포함되어 대괄호 사용
${ header.user-agent } ⇐ 에러발생
${ King['한글'] } ⇐ 속성명이 한글이면 대괄호 사용
${ King.한글 } ⇐ 에러발생
```

. 대괄호 사용시에는 “(큰따옴표)와 ‘(작은따옴표) 모두 사용 가능

10.2 EL의 내장 객체

▣ 4가지 영역에 속성값 저장하고 읽어오기

내장객체명	설 명
pageScope	pageContext 내장객체와 같이 page 영역에 저장된 속성값을 읽음
requestScope	request 내장객체와 같이 request영역에 저장된 속성값을 읽음
sessionScope	session 내장객체와 같이 session 영역에 저장된 속성값을 읽음
applicationScope	applicatioin 내장객체와 같이 application 영역에 저장된 속성값을 읽음

```
<%
pageContext.setAttribute("scopeValue", "페이지 영역");
request.setAttribute("scopeValue", "리퀘스트 영역");
session.setAttribute("scopeValue", "세션 영역");
application.setAttribute("scopeValue","애플리케이션 영역");
%> ⇐ 4가지 영역에 동일한 속성명으로 값 저장
```

ImplicitObjMain.jsp

10.2 EL의 내장 객체

▣ 4가지 영역에 속성값 저장하고 읽어오기(계속)

```
    <h3>각 영역에 저장된 속성 읽기</h3>
    <ul>
        <li>페이지 영역 : ${ pageScope.scopeValue }</li>
        <li>리퀘스트 영역 : ${ requestScope.scopeValue }</li>
        <li>세션 영역 : ${ sessionScope.scopeValue }</li>
        <li>애플리케이션 영역 : ${ applicationScope.scopeValue }</li>
    </ul>

    <h3>영역 지정 없이 속성 읽기</h3>
    <ul>
        <li>${ scopeValue }</li>  ❸
    </ul>

<%--    <jsp:forward page="ImplicitForwardResult.jsp" /> --%>  ❹
</body>
```

❷ EL의 내장객체를 통해 출력
❸ 영역 지정이 없으면 가장 좁은 page 영역의 속성 출력

❹ 주석을 해제한 후 포워드

10.2 EL의 내장 객체

▣ 4가지 영역에 속성값 저장하고 읽어오기(계속)

```
<h3>각 영역에 저장된 속성 읽기</h3>
<ul>
    <li>페이지 영역 : ${ pageScope.scopeValue }</li>
    <li>리퀘스트 영역 : ${ requestScope.scopeValue }</li>
    <li>세션 영역 : ${ sessionScope.scopeValue }</li>
    <li>애플리케이션 영역 : ${ applicationScope.scopeValue }</li>
</ul>
<h3>영역 지정 없이 속성 읽기</h3>
<ul>
    <li>${ scopeValue }</li>  ❷
</ul>
```

❶ 페이지 영역은 출력 안됨
❷ 포워드 하면 가장 좁은 영역이 request 영역이 됨

ImplicitForwardResult 페이지

각 영역에 저장된 속성 읽기

- 페이지 영역 : ❶
- 리퀘스트 영역 : 리퀘스트 영역
- 세션 영역 : 세션 영역
- 애플리케이션 영역 : 애플리케이션 영역

영역 지정 없이 속성 읽기

- ❷ 리퀘스트 영역

[Note] EL을 통해 영역의 속성을 읽을때는 영역명을 명시하지 않아도 됨

ImplicitForwardResult.jsp

10.2 EL의 내장 객체

▣ 폼값 처리하기

내장객체명	설 명
param	request.getParameter("매개변수명")과 동일하게 요청 매개변수의 값을 받아옴
paramValues	request.getParameterValues("매개변수명")과 동일하게 요청 매개변수의 값을 문자열 배열로 받아옴

```
<h2>폼값 전송하기</h2>
<form name="frm" method="post" action="FormResult.jsp">
    이름 : <input type="text" name="name" /><br />
    성별 : <input type="radio" name="gender" value="Man" />남자
          <input type="radio" name="gender" value="Woman" />여자<br />
    학력 :
        <select name="grade">
            <option value="ele">초딩</option>
            <option value="mid">중딩</option>
```

FormSubmit.jsp

10.2 EL의 내장 객체

▣ 폼값 처리하기(계속)

```
<ul>
    <li>이름 : ${ param.name }</li>
    <li>성별 : ${ param.gender }</li>
    <li>학력 : ${ param.grade }</li>
    <li>관심사항 : ${ paramValues.inter[0] }
        ${ paramValues.inter[1] }
        ${ paramValues.inter[2] }
        ${ paramValues.inter[3] }</li>
</ul>
```

❶ 하나의 값이 전송되는 경우에는 EL의 내장객체 param을 사용

❷ 둘 이상의 값이면 paramValues 를 통해 배열로 값을 받음

EL로 폼값 받기

- 이름 : 성낙현
- 성별 : Man
- 학력 : uni
- 관심사항 : eco spo

FormResult.jsp

. EL은 null을 출력해도 예외가 발생하지 않음

. 따라서 checkbox를 하나도 체크하지 않더라도 null체크 없이 간단히 코딩할 수 있다.

10.2 EL의 내장 객체

▣ 객체 전달하기

- 폼을 통해 전송할 수 있는 값은 문자열
- 객체인 경우에는 영역의 공유범위를 통해 전달해야 함

ObjectParams.jsp

```
<%
request.setAttribute("personObj", new Person("홍길동", 33));  ⇐ request영역에 Person객체 저장
request.setAttribute("stringObj", "나는 문자열");
request.setAttribute("integerObj", new Integer(99));
%>
<jsp:forward page="ObjectResult.jsp"> ⇐ 전달할 페이지로 포워드
    <jsp:param value="10" name="firstNum" /> ⇐ 파라미터도 같이 전달
    <jsp:param value="20" name="secondNum" />
</jsp:forward>
```

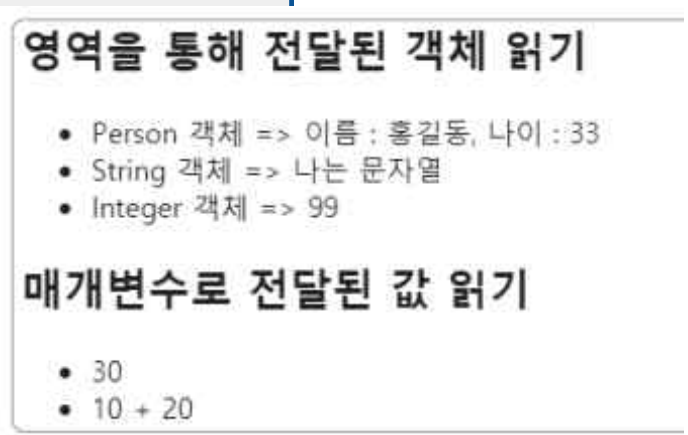
10.2 EL의 내장 객체

▣ 객체 전달하기

- request영역은 포워드 된 페이지에서는 공유되므로 값을 읽어올 수 있음
- 파라미터로 전달된 값도 읽어올 수 있음

```
<h2>영역을 통해 전달된 객체 읽기</h2>
<ul>
    <li>Person 객체 => 이름 : ${ personObj.name }, 나이 : ${ personObj.age }</li>
    <li>String 객체 => ${ requestScope.stringObj }</li>
    <li>Integer 객체 => ${ integerObj }</li>
</ul>
<h2>매개변수로 전달된 값 읽기</h2>
<ul>
    <li>${ param.firstNum + param['secondNum'] }</li>
    <li>${ param.firstNum } + ${param["secondNum"] }</li>
</ul>
```

ObjectResult.jsp



10.2 EL의 내장 객체

▣ 쿠키, HTTP 헤더, 컨텍스트 초기화 매개변수 출력

- cookie : 쿠키를 읽을 때 사용
- header : request.getHeader()와 동일하게 헤더값을 읽을 때 사용
- headerValues : request.getHeaders()와 동일하게 헤더값을 배열 형태로 읽을 때 사용
- initParam : web.xml에 설정한 컨텍스트 초기화 매개변수를 읽을 때 사용
- pageContext : JSP의 pageContext 내장 객체와 동일한 역할

```
<%
CookieManager.makeCookie(response, "ELCookie", "EL좋아요", 10);  ❶
%>
    <h3>쿠키값 읽기</h3>
    <li>ELCookie값 : ${ cookie.ELCookie.value }</li>  ❷
```

❶ 4장에서 생성한 쿠키메니져 클래스를 통해 10초간 유효한 쿠키 생성
❷ EL을 통해 쿠키값 읽어 출력

OtherImplicitObj.jsp

10.2 EL의 내장 객체

▣ 쿠키, **HTTP** 헤더, 컨텍스트 초기화 매개변수 출력(계속)

```
<h3>HTTP 헤더 읽기</h3>
<ul>
    <li>host : ${ header.host }</li>
    <li>user-agent : ${ header['user-agent'] }</li>
    <li>cookie : ${ header.cookie }</li>
</ul>

<h3>컨텍스트 초기화 매개변수 읽기</h3>
<li>OracleDriver : ${ initParam.OracleDriver }</li>

<h3>컨텍스트 루트 경로 읽기</h3>
<li>${ pageContext.request.contextPath }</li>
```

❸ HTTP 헤더 정보 및 그외 값들 읽어와 출력

쿠키값 읽기

- ELCookie값 : EL좋아요 — 값이 채워짐

HTTP 헤더 읽기

- host : localhost:8081
- user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36
- cookie : JSESSIONID=F0266E17E2DA30E9CC315D657E5CFA65; ELCookie=ELi¢□i□□i□□

컨텍스트 초기화 매개변수 읽기

- OracleDriver : oracle.jdbc.OracleDriver

컨텍스트 루트 경로 읽기

- /MustHaveJSP

10.3 컬렉션 사용하기

▣ 컬렉션 사용하기

. EL을 사용하면 자바 코드보다 훨씬 더 간단하게 컬렉션을 사용할 수 있음

```
<h2>List 컬렉션</h2>
<%
List<Object> aList = new ArrayList<Object>();  ❶
aList.add("청해진");  ❷
aList.add(new Person("장보고", 28));  ❸
pageContext.setAttribute("Ocean", aList);  ❹
%>
<ul>
    <li>0번째 요소 : ${ Ocean[0] }</li>
    <li>1번째 요소 : ${ Ocean[1].name }, ${ Ocean[1].age }</li>  ❺
    <li>2번째 요소 : ${ Ocean[2] }<!--출력되지 않음--></li>
</ul>
```

❶ List 컬렉션을 생성
❷~❹ String과 Person을 추가한 후 page영역에 속성 저장
❺ List는 배열의 속성을 가지므로 인덱스로 접근하여 출력

List 컬렉션

- 0번째 요소 : 청해진
- 1번째 요소 : 장보고, 28
- 2번째 요소 : ❶

CollectionUse.jsp

10.3 컬렉션 사용하기

▣ 컬렉션 사용하기(계속)

```
<h2>Map 컬렉션</h2>
<%
Map<String, String> map = new HashMap<String, String>();
map.put("한글", "훈민정음");
map.put("Eng", "English");
pageContext.setAttribute("King", map);
%>
<ul>
    <li>영문 Key : ${ King["Eng"] }, ${ King['Eng'] }, ${ King.Eng }</li>
    <li>한글 Key : ${ King["한글"] }, ${ King['한글'] }, \${King.한글 }<!--에러-->
</li>
</ul>
```

❻ Map 컬렉션을 생성한 후 2개의 데이터를 저장. page영역에 속성 저장

❼ Map은 키를 통해 읽어야 하므로 3가지 형태로 접근가능.

Note 단, Key가 한글이면 .(점)으로는 접근할 수 없어 에러발생 코드에서는 EL주석을 통해 처리.

Map 컬렉션

- 영문 Key : English, English, English
- 한글 Key : 훈민정음, 훈민정음, ${King.한글}

10.4 EL의 연산자들

▣ 할당 연산자

. 초기의 EL은 할당이 불가능했지만, EL3.0부터 할당이 가능해짐.
. 단, 할당과 동시에 출력되므로 세미콜론과 작은따옴표를 함께 사용.

```
${ numberVar = 10 }        ← 할당과 동시에 출력
${ numberVar = 10;'' }     ← 할당만 되고 출력은 되지 않음
```

▣ 할당 연산자

. +, -, * : 덧셈, 뺄셈, 곱셈
. / 또는 div : 나눗셈
. % 또는 mod : 나머지

▣ 논리 연산자

. && 또는 and : 논리 And
. || 또는 or : 논리 Or
. ! 또는 not : 논리 No

10.4 EL의 연산자들

▣ 비교 연산자

- > 또는 gt : Greater Than(~보다 크다.)
- >= 또는 ge : Greater than or Equal(~보다 크거나 같다.)
- < 또는 lt : Less Than(~보다 작다.)
- <= 또는 le : Less than or Equal(~보다 작거나 같다.)
- == 또는 eq : EQual(같다.)
- != 또는 ne : Not Equal(같지 않다. 즉 다르다.)

[Note] Java에서 사용하던 연산자는 동일하게 사용가능. 여기에 추가로 EL에서 사용할 수 있는 문자형태의 연산자가 추가로 제공되어 편의성을 제공함.

예제 10-9] webapp/10EL/Operator1.jsp
예제 10-10] webapp/10EL/Operator2.jsp

[Note] 해당 예제는 산술연산, 비교연산 등의 매우 일반적인 내용이므로 교제를 참고하여 작성하세요.

10.5 인스턴스 메서드 호출

▣ 호출할 메서드 준비

. EL에서는 자바 코드를 직접 사용할 수 없으나, 메서드는 호출 기능을 제공

```
public class MyELClass {
    // 주민번호를 입력받아 성별을 반환합니다. ❶
    public String getGender(String jumin) {
        String returnStr = "";
        int beginIdx = jumin.indexOf("-") + 1;
        String genderStr = jumin.substring(beginIdx, beginIdx + 1);
        int genderInt = Integer.parseInt(genderStr);
        if (genderInt == 1 || genderInt == 3)  ❷
            returnStr = "남자";
        else if (genderInt == 2 || genderInt == 4)  ❸
            returnStr = "여자";
        else
            returnStr = "주민번호 오류입니다.";
        return returnStr;
    }
```

❶ 주민번호를 매개변수로 받아 성별을 판단하여 반환하는 메서드 정의

Java Resources/el/MyELClass.java

10.5 인스턴스 메서드 호출

▣ 메서드 호출하기

. page 영역에 먼저 저장한 후 메서드 호출. 공유가 가능한 범위라면 다른 영역도 가능.

```
<%
MyELClass myClass = new MyELClass();  // 객체 생성 ❶
pageContext.setAttribute("myClass", myClass);  // page 영역에 저장 ❷
%>
<html>
<head><title>표현 언어(EL) - 메서드 호출</title></head>
<body>
    <h3>영역에 저장 후 메서드 호출하기</h3>
    001225-3000000 => ${ myClass.getGender("001225-3000000") } <br />
    001225-2000000 => ${ myClass.getGender("001225-2000000") }
</body>
```

❶ 객체 생성후 page 영역에 저장
❸ EL을 통해 메서드 호출

영역에 저장 후 메서드 호출하기

001225-3000000 => 남자
001225-2000000 => 여자

ELCallMethod.jsp

10.6 정적 메서드 호출

▣ 클래스명을 통한 정적 메서드 호출

```
// 입력받은 정수까지의 구구단을 HTML 테이블로 출력해줍니다. ❺
public static String showGugudan(int limitDan) {
    StringBuffer sb = new StringBuffer();
    try {
        sb.append("<table border='1'>");
        for (int i = 2; i <= limitDan; i++) {
            sb.append("<tr>");
            for (int j = 1; j <= 9; j++) {
                sb.append("<td>" + i + "*" + j + "=" + (i * j) + "</td>");
            }
            sb.append("</tr>");
        }
        sb.append("</table>");
    } catch (Exception e) {
        e.printStackTrace();
    }
    return sb.toString();
```

❺ 매개변수로 전달된 숫자까지의 구구단을 테이블 형태로 반환

예제 10-11] Java Resources/el/MyELClass.java(메서드 추가)

10.6 정적 메서드 호출

▣ 클래스명을 통한 정적 메서드 호출(계속)

```
    <h3>클래스명을 통해 정적 메서드 호출하기</h3>
    ${ MyELClass.showGugudan(7) } ❶
</body>
```

클래스명을 통해 정적 메서드 호출하기

2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63

예제 10-13] webapp/10EL/ELCallMethod.jsp(코드 추가)

[Note] TLD를 이용해서 메서드를 호출하는 방식은 JSP 2.0(톰캣 5.5)에서 사용하던 방식으로 현재의 거의 사용되지 않으므로 참고만 하고 넘어가도 좋음

학습 마무리

▣ **핵심요약**

- EL은 내장 객체를 통해 4가지 영역에 저장된 속성값을 읽을 수 있음
- 전송된 폼값이나 객체를 EL을 통해 읽을 수 있음
- 컬렉션을 보다 쉽게 사용할 수 있음
- 자바에서 제공하는 연산자와 함께 문자 형태의 연산자를 추가로 사용할 수 있음
- JSP 코드를 직접 사용할 수는 없지만, 메서드를 호출할 수 있는 기능을 제공