8. 표현언어와 JSTL

1. 표현언어로 단순화하기
2. 표현언어로 요청 파라미터 처리하기
3. 표현언어로 내장 객체 접근하기
4. JSTL 개요
5. JSTL 라이브러리
6. JSTL core 태그
7. JSTL fmt

1. 표현언어로 단순화하기

- **표현언어(EL:Expression Language)?**
  - JSP 출력에 대한 부분을 쉽게하기 위해 개발한 태그
  - 파일 JSP페이지에 사용되는 선언문(<%! %>), 스크립트릿(<% %>), 표현식(<%= %>)과 같은 자바 코드를 대신 사용
  - 예:

표현식	표현언어
<%=expr%>	${expr}
<%="hello%>	${"hello"}

1. 표현언어로 단순화하기

- **표현언어에서 사용 가능한 데이터 타입**

정수형 : ${10}
실수형 : ${5.6}
문자열형: ${"성윤정"}
논리형: ${true}
null : ${null}

EL 식 작성법

- 표현 언어는 항상 ${ 로 시작해서 }로 끝남 : **${expr}**
- 연산식 사용 **: ${num + 1}**
- 브라켓 연산자(bracket ([]) operator)를 사용: **${article[“num”] + 1}**
- 동적으로 값을 받도록 JSTL이나 커스텀 태그의 JSP 액션의 속성에 값을 지정할 때도 사용: **<c:out value="${article.num + 1}"/>**

표현언어의 연산자

종류	연산자
산술	+, -, *, /(or div), % (or mod)
관계형	==(or eq), !=(or ne), <(or lt), >(or gt), <=(or le), >=(or ge)
조건	a ? b : c
논리	&& (or and), || (or or), ! (or not)
null 검사	empt

연산자 표함 EL 식

${5+2} , ${3==3}, ${3 eq 3}, ${empty input}

표현언어 연산자 사용하기

```
<body>
    ₩${5+2} : ${5+2} <br>
    ₩${5/2} : ${5/2} <br>
    ₩${5 div 2} : ${5 div 2} <br>
    ₩${5 mod 2} : ${5 mod 2}<br>
    ₩${5 > 2} : ${5 > 2}<br>
    ₩${2 gt 10} : ${2 gt 10}<br>
    ₩${(5 > 2) ? 5 : 2} : ${(5 > 2) ? 5 : 2}<br>
    ₩${(5 > 2) || (2 < 10)} : ${(5 > 2) || (2 < 10)}<br>
    <%
    String input=null;
    %>
    ₩${empty input} : ${empty input}<br>
</body>
```

표현언어 조건 식 사용

```
<% if(input==null) { %>
        텅빈 객체(null) 입니다
<% } %>
```

```
<c:if test=$(empty input}>
        텅빈 객체(null)입니다
</c:if>
```

2. 표현언어로 요청 파라미터 처리하기

- **jsp 파라미터 처리**
  - 폼의 입력값을 얻어오기 위해 jsp 내장 객체 request의 getParameter() 메소드 사용
- **표현언어 요청 파라미터 처리**
  - param 사용

내장객체	설명
param	JSP의 내장객체인 request의 getParameter()와 동일한 역할인 파라미터 값을 알려 줌
paramValues	동일한 이름으로 전달되는 파라미터 값들을 배열 형태로 얻어오는데 사용 request의 getParameterValues와 동일

2. 표현언어로 요청 파라미터 처리하기

- 사용예

```
<form action="research.jsp">
좋아하는 계절
<input type="checkbox" name="season" value="spring">
<input type="checkbox" name="season" value="summer">
<input type="checkbox" name="season" value="fall">
<input type="checkbox" name="season" value="winter">
</form>
```

```
<%
    String seasonArr[]=request.getParameterValues("season");
    out.println("당신이 좋아하는 계절: ");
    for(String season:seasons){
        out.println(season);
    }
%>
```

일반적인 jsp 방식

```
<c:forEach items="${paramValues.season}" var="season"
        ${season}
</c:forEach>
```

표현언어 방식

3. 표현언어로 내장 객체 접근하기

내장객체 접근 형태

category	내장객체	설명
범위	pageScope	page 기본 객체에 저장된 속성의 <속성, 값>매핑을 저장한 Map 객체
	requestScope	request 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체
	sessionScope	session 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체
	applicationScope	application 기본 객체에 저장된 속성의 <속성, 값>매핑을 저장한 Map 객체

3. 표현언어로 내장 객체 접근하기

- **request 내장 객체 정보 전달 메소드들**

메소드	설명
setAttribute(name, value)	주어진 이름(name)에 값(value)을 설정
getAttribute(name)	주어진 이름(name)에 설정된 값(value)을 얻어 옴
getAttributeNames()	현재 객체에 관련된 모든 속성 이름을 얻어낸다
removeAttribuete(name)	주어진 이름(name)에 설정된 값(value)을 제거

예제 : p326 서블릿 클래스에서 두 수에 대한 합을 구해 jsp에 출력

3. 표현언어로 내장 객체 접근하기

기타 내장객체 속성

속성	jsp 내장객체	서블릿 내장 객체	표현언어 내장객체
page 속성	pageContext	javax.servlet.jsp.JspContext 클래스	pageScope
request 속성	request	javax.servlet.ServletRequest 인터페이스	requestScope
session 속성	session	javax.servlet.http.HttpServlet 인터페이스	sessionScope
application 속성	application	javax.servlet.ServletContext 인터페이스	applicationScope

3. 표현언어로 내장 객체 접근하기

기타 내장객체 속성

- 사용예

범위	자바코드	표현언어
pageContext	pageContext.getAttribute("name");	${pageScope.name}
reuqest	request.getAttribute("name")	${request.name}
session	session.getAttribute("name")	${session.name}
application	application.getAttribute("name")	${application.name}

- 객체를 지정하지 않으면 다음 순서로 검색 : {name}

pageScope->requestScope->sessionScope->applicationScope

예제 : p332 표현언어 내장객체 명시적으로 사용하기

1. JSTL 개요

- **JSTL(JSP Standard Tag Library)**
  - 표준 커스텀 태그(Custom Tag)
  - 커스텀 태그 : 사용자 정의 태그
  - JSP 페이지의 로직을 담당하는 부분인 제어문 및 데이터베이스 처리 등을 표준 커스텀 태그로 제공
  - 코드를 깔끔하게 하고 가독성을 좋게 함

4. JSTL 개요

제어문 사용 예

```
<%if(request.getParameter("color").equals("1")){%>
        <span style="color:red;">빨강<span>
<%} else if(request.getParameter("color").equals("2")){%>
        <span style="color:green;">초록<span>
<%} else if(request.getParameter("color").equals("3")){%>
        <span style="color:blue;">초록<span>
<%} %>
```

```
<c:if test="${param.color==1}"
        <span style="color:red;">빨강</span>
</c:if>
c:if test="${param.color==2}"
        <span style="color:green;">초록</span>
</c:if>
c:if test="${param.color==3}"
        <span style="color:blue;">빨강</span>
</c:if>
```

4. JSTL 개요

- **JSTL과 커스텀 태그도 XML기반**
  - 시작 태그와 종료 태그를 쌍으로 작성
    - 시작 태그와 종료 태그가 존재하는 경우

```
<c:if test="empty ${c.getEmail}"> <!--시작 태그-->
  <input type="text" name="email"><!--태그 내용-->
</c:if><!--종료 태그-->
```

    - 종료 태그가 없는 단독 태그의 경우
      - 태그를 닫는 기호(>)전에 /를 입력해 태그의 종료를 알림

```
<c:set var="name" scope="page"/><!--단독 태그-->
```

5. JSTL 라이브러리

- **JSTL에서 제공해주는 기능**
  - 간단한 로직 구현(자바 변수 선언, if문, for문에 해당) :core
  - 다른 페이지 호출(<c:redirect>, <c:import>) : core
  - 날짜, 시간, 숫자 포맷 :format
  - JSP 페이지 하나를 가지고 여러 가지 언어의 웹페이지 생성 : core
  - 데이터베이스로 입력, 수정, 삭제, 조회 :sql
  - XML 문서의 처리 : xml
  - 문자열을 처리하는 함수 : functions

5. JSTL 라이브러리

- **JSTL을 사용하기 위한 라이브러리 추가하기**
  - JSTL 라이브러리 다운로드
    - JSTL 라이브러리 다운로드 사이트
      - *https://jstl.java.net*
    - 라이브러리 다운로드
      - *javax.servlet.jsp.jstl-api-1.2.1.jar*와 *javax.servlet.jsp.jstl-1.2.1.jar* 다운로드
  - 라이브러리 배치
    - [프로젝트]-[WEB-INF]-[lib]에 라이브러리 복사

5. JSTL 라이브러리

JSTL 라이브러리 개요 및 사용 방법

- JSTL은 많은 태그를 제공하며, 이들을 사용하기 위해 각각 네임스페이스로 라이브러리를 제공
  - Core(코어), XML, I18N(국제화, 포매팅), SQL(데이터베이스), Functions(함수)
- 라이브러리들은 URI로 제공되며 이들을 사용할 때는 접두어(prefix)를 사용
- JSTL 라이브러리에서 제공하는 태그를 사용
  - taglib 디렉티브에 사용할 라이브러리의 prefix 속성과 uri 속성에 해당하는 값을 기술
    - 예 : *<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>*

6. JSTL Core

- **JSTL Core 태그 라이브러리**
  - 라이브러리
    - Core(코어)
  - URI
    - http://java.sun.com/jsp/jstl/core
  - Prefix
    - c
  - 제공 기능
    - 변수의 선언 및 삭제 등의 변수와 관련된 작업
    - if, for문 등과 같은 제어문
    - URL처리 및 그밖에 예외처리 및 화면 출력

6. JSTL Core

JSTL core 태그

태그	설명
<c:set>	변수에 값을 설정한다.
<c:remove>	변수에 설정된 값을 제거
<c:if>	조건에 따라 처리를 달리할 때 사용
<c:choose>	여러 조건에 따라 처리할 때 사용
<c:forEach>	반복처리를 위해서 사용
<c:forTokens>	구분자로 분리된 각각의 토큰을 처리할 때 사용
<c:import>	외부의 자원을 url을 지정하여 가져다 사용
<c:redirect>	지정한 경로로 이동
<c:url>	url을 재작성
<c:out>	데이터를 출력할 때 사용
<c:catch>	예외 처리를 할 때 사용

6. JSTL Core

- **<c:set>**
  - 해당범위(scope) 내에 속성을 생성하고 속성 값을 지장
  - jsp의 setAttribute() 메소드와 동일

```
pageContext.setAttribute("msg", "Hello");
<c:set var="msg" value="Hello" scope="page"/>
```

  - <c:set> 태그 속성 목록

속성	설명
var	변수 이름을 String 으로 지정
value	변수에 저장될 값을 지정
scope	변수가 효력을 발휘할 영역, 생략 가능, 기본값은 page

6. JSTL Core

<c:set>

형식

```
<c:set var="변수 이름" [scope="{page|request|session|application}"]>
        저장할 값
</c:set>
```

```
<c:set var="변수 이름" value="저장할 값"
        [scope="{page|request|session|application}"]>
```

<jsp:setProperty> 기능 사용

```
<jsp:useBean id="member" class="el.MemberBean" >
<jsp:setProperty name="member" property="name" value="전수빈" />
</jsp:useBean>
```

```
<c:set var="member" value="<%=new el.MemberBean()%>">
<c:set target="${member}" property="name" value="전수빈"/>
```

6. JSTL Core

- **<c:set>**
  - 연산자 사용

```
<c:set var="add" value="${10+5}">
<c:set var="flag" value="${10>5}">
```

  - 예제: p353

- **<c:remove>**
  - removeAttribute()

```
<c:remove var="변수 이름" [scope="{page|request|session|application}"]
<c:remove var="age">
```

6. JSTL Core

- **<c:if>**
  - 참과 거짓 값만 사용할 수 있는 if만 제공
  - if-else, if- else if 제공 않음
  - 형식

```
<c:if test="조건식">
  조건이 참일 경우 실행할 문장
</c:if>
```

```
<c:if test="${param.color==1}">
  <span style="color:red">빨강</span>
</c:if>
```

```
<%
  String str=request.getParameter("color");
  int color=Integer.parseInt(str);
  if(color==1){
%>
    <span style="color:red">빨강</span>
<% } %>
```

예제 : p356

6. JSTL Core

- **<c:choose>**
  - <c:if> 참과 거짓 값만 선택 가능
  - 여러 가지 값 중에 하나를 선택 가능하게 함
  - 형식

```
<c:choose>
    <c:when test="조건1">몸체1</c:when><!- - 조건1에 만쪽할 때 - ->
    <c:when test="조건2">몸체2</c:when><!- - 조건2에 만쪽할 때 - ->
    <c:otherwise>몸체3</c:otherwise><!- - 조건을 만족하지 않을 때 - ->
</c:choose>
```

<c:choose>

- 예

예제 : p359

```
//jsp 코드

if(request.getParameter("userType").equals("admin")){₩
  out.println(request.getParameter("id") + "(관리자)");
}else{
  out.println(request.getParameter("id") + "(회원)");
}
```

```
//<c:if> 코드

<c:if test="${param.userTyep=='admin'}">
  ${param.id} + (관리자)
</c:if>
<c:if test="${param.userTyep=='member'}">
  ${param.id} + (회원)
</c:if>
```

```
//<c:choose> 코드
<c:choose>
<c:when test="${param.userTyep=='admin'}">
  ${param.id} + (관리자)
</c:when>
<c:otherwise>
   ${param.id} + (회원)
</c:otherwise>
</c:choose>
```

6. JSTL Core

<c:forEach>

- 배열(array), 컬렉션(collections) 또는 맵(Map) 등과 같이 집합체에 저장된 값을 순차적으로 처리할 때 사용
- 형식

```
<c:forEach [var="변수명"] items="collections">
  몸체
<c:forEach>
```

```
<c:forEach var="movie items="${movielist}">
  ${movie}<br>
<c:forEach>
```

예제 : p361

6. JSTL Core

<forEach>

- varStatus 속성

```
<c:forEach var="movie items="${movielist}" varStatus="status">
  ${status.index}
  S{status.count}
  ${movie}<br>
<c:forEach>
```

속성	설명
index	items에 지정한 집합체의 현재 반복 중인 index를 알려 줌 0부터 순서가 부여됨
count	루핑을 돌 때 현재 몇 번째 반복 중인지 알려 줌 1부터 순서 부여

예제: p362

6. JSTL Core

<forEach>

varStatus 속성

```
<c:forEach var="movie" items="${movieList}" varStatus="status">
  <c:choose>
    <c:when test="${status.first }">
      <li style="font-weight: bold; color: red;">${movie}</li>
    </c:when>
    <c:otherwise>
      <li>${movie}</li>
    </c:otherwise>
  </c:choose>
</c:forEach>
```

속성	설명
first	현재 루프가 처음인지 여부를 알려 줌. 첫 번째일 경우 true, 아니면 false를 리턴
last	현재 루프 마지막인지 여부를 알려 줌. 마지막일 경우 true, 아미면 false 리턴

<forEach>

- begin, end 속성

속성	설명
begin	반복 시작 항목의 인덱스
end	반복의 마지막 항목의 인덱스
step	인덱스 증가 치

```
<table>
<c:forEach var="cnt" begin="1" end="10" step=2 varStatus="status">
  <tr>
  <td>${status.index}</td>
  <td>${status.count}</td>
  <td>${cnt}</td>
 </tr>
</c:forEach>
</table>
```

예제 :p367

6. JSTL Core

<c:forTokens>

- 문자열을 구분자로 분리하여 하나씩 추출

형식

```
<c:forTokens var="토큰을 저장할 변수" items="토큰으로나눌 문자열"
        delims="구분자">
  몸체
</c:forTokens>
```

```
<c:forTokens var="city" items="서울,인천,대구,부산" delims=",">
  ${city}<br>
</c:forTokens>
```

예제 p 371

6. JSTL Core

- **<c:import>**
  - <jsp:include> 태그와 같이 다른 페이지 내용을 포함시키기 위해 사용
  - 형식

```
<c:import url="URL" [var="변수명"][scope="영역"]
                                [charEncoding="charEncoding"]>
</c:import>
```

```
<c:import url="http://localhost:8181/web-study/el.jsp" var="data">
</c:import>
```

예제: p372

6. JSTL Core

- **<c:url>**
  - 여러 번 반복 사용되는 주소를 변수에 저장하기 위해 사용
  - 형식

```
<c:url value="URL" [var="변수명"] [scope="영역"]>
</c:url>
```

```
<c:url value="images/pic.jgp" var="data">
</c:url>
```

예제 p:373

6. JSTL Core

- **<c:redirect>**
  - response.sendRedirect() 메소드와 동일, 지정한 페이지로 이동
  - 형식

```
<c:redirect url="URL" [context="경로명"]></c:redirect>
```

```
<c:redirect url="jstlUrl.jsp" ></c:redirect>
```

예제: p374

6. JSTL Core

- **<c:out>**
  - value 속성에 지정한 값을 출력
  - 형식

```
<c:out value="value" [default="기본값"]>
```

```
<c:out value="${age}" default="10">
```

```
<c:out value="value" >
 기본값"
</c:out>
```

```
<c:out value="${age}">
 10
</c:out>
```

속성	설명
value	출력할 값을 지정
default	지정한 값이 없을 경우 사용할 값을 지정

6. JSTL Core

- **<c:catch>**
  - 예외처리를 위해 사용, try-catch 과 동일
  - 형식

예제 : p 377

```
<c:catch var="변수명>
  예외가 발생할 수 있는 코드
</c:catch>
```

```
<c:catch var="errmsg">
 예외 발생전
  <%=1/0%>
  예외 발생 후
</c:catch>
```

7. JSTL fmt

- **JSTL I181N 태그 라이브러리**
  - 라이브러리
    - I18N(국제화, 포매팅)
  - URI
    - http://java.sun.com/jsp/jstl/fmt
  - Prefix
    - fmt
  - 제공 기능
    - 로케일, 메시지, 숫자 문자 형식 등

7. JSTL fmt

- **JSTL I181N 태그 라이브러리**

기능	태그	설명
숫자 날짜 형식	<fmt:formatNumber>	숫자 형식을 표현할 때 사용
	<fmt:formatDate>	날짜 형식을 표현할 때 사용
	<fmt:parseNumber>	문자열을 숫자로 파싱
	<fmt:parseDate>	문자열을 날짜로 파싱
	<fmt:setTimeZone>	특정 scope의 타임 존을 설정할 때 사용
	<fmt:timeZone>	타임 존(Time Zone)을 적용할 때 사용
로케일 지정	<fmt:setLocale>	국제화 태그를 사용할 로케일 지덩
	<fmt:requestEncoding>	request.setCharacterEncoding() 역할
메시지	<fmt:bundle>	태그 몸체에서 사용할 리소스 번들 지정
	<fmt:setBundle>	특정 리소스 번들을 사용할 수 있도록 로딩
	<fmt:message>	메시지 출력

7. JSTL fmt

숫자 날짜 형식 지정 관련 태그

- <fmt:formatNumber>
  - 수자 데이터 표현하기 위해 사용
  - 형식

```
<fmt:formatNumber value="수치 데이터"
                  [type="{number|currency|percent}]
                  [pattern="패턴"]
                  [currencySymbol="화폐단위"]
                  [groupingUsed="{true|false}"]
                  [var="변수명"]
                  [scope="{page|request|session|application}">
```

숫자 날짜 형식 지정 관련 태그

- <fmt:formatNumber>

```
<fmt:formatNumber value="1234567.89"/>
 1,234,567.89    <- 출력결과
```

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

```
<fmt:formatNumber value="0,5" type="percent"/>
 50%<- 출력결과
```

```
<fmt:formatNumber value="10000" type="currency"/>
￦10,000    <- 출력결과
```

```
<fmt:formatNumber value="10000" type="currency" currencySymbol="$"/>
$10,000    <- 출력결과
```

숫자 날짜 형식 지정 관련 태그

- <fmt:formatNumber>

```
<fmt:formatNumber value="1234567.891234" pattern="#,#00.0#""/>
 1,234,567.89    <- 출력결과
```

```
<fmt:formatNumber value="1234567.8" pattern="#,#00.0#"/>
 1,234,567.8    <- 출력결과
```

```
<fmt:formatNumber value="1234567.89" pattern="000"/>
 1234567.890   <- 출력결과
```

숫자 날짜 형식 지정 관련 태그

- <fmt:formatDate>
  - 날짜의 형식을 지정
  - 형식

```
<fmt:formatDate value="수치 데이터"
        [type="{time|date|both}]
        [dateStyle="{default|short|medium|full}"]
        [timeStyle="{default|short|medium|full}"]
        [pattern="customPattern"]
        [timeZone="tomeZone"]
        [groupingUsed="{true|false}"]
        [var="변수명"]
        [scope="{page|request|session|application}">
```

숫자 날짜 형식 지정 관련 태그

- <fmt:formatDate>

예제 p:385

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

<fmt:formatDate value="${now}"/>
time<fmt:forDate value="${now}" type="time:/>
time<fmt:forDate valie="${now}" type="both"
```

숫자 날짜 형식 지정 관련 태그

- <fmt:setTimeZone><fmt:timeZone>
  - 특정지역의 타임존을 설정하는 태그
  - 형식
  - 기능

```
<fmt:setTimeZone value="timeZonr"
            [var="변수명"]
            [scope="{page|request|session|application}">
```

```
<fmt:TimeZone value="timeZonr">
  몸체
</fmt:TimeZone>
```

예제 p:387

- **<fmt:setLocal>**
  - 로케일 지정을 위한 태그

예제: p389

```
<fmt:setLocal value="local">
```

- **<fmt:requestEncoding>**
  - post 방식으로 전송된 데이터의 한글 깨짐 방지
  - 예제: p391