2024/10/15 9일차

선생님이 정리하신것

JSTL(JSP Standard Tag Library)

- JSP 표준태그 라이브러리
- 스크립틀릿을 사용해서 작성했던 자바코드를 대체할 수 있다.
  - 변수 선언/삭제, 값 출력, 제어문 처리, 반복문 처리, 숫자나 날짜에 대한 포맷팅, 국제화처리, URL처리
- 사용법
  - JSTL 파일을 다운받아서 WEB-INF/lib에 복사한다.
  - JSP 파일에 사용할 태그라이브러리를 지시어를 사용해서 정의한다.

```
<%@ taglib prefix="별칭" uri="태그라이브러리식별자" %>
<별칭:태그명 value="${EL표현식}" />
```

JSTL 태그라이브러리의 태그 종류

- core 태그
  - 가장 많이 사용되는 태그 라이브러디다.
  - 변수/출력/제어문/반복문/URL 처리를 지원한다.
  - JSP에 아래의 지시어를 정의한다.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

- fmt 태그
  - 숫자나 날짜에 대한 포맷팅을 지원한다.
  - 국제화처리를 지원한다.
  - JSP에 아래의 지시어를 정의한다.

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/j
```

```
stl/fmt" %>
```

- function 태그
  - String 클래스의 주요 메소드 사용을 지원한다.
  - JSP에 아래의 지시어를 정의한다.

```
  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/js
tl/function" %>
```

- xml 태그
  - 거의 사용하지 않는다.
- sql 태그
  - 거의 사용하지 않는다.

Core 태그 라이브러리

- **<c:out>** 태그
  - 값을 출력한다.
  - <%=값 %>, ${표현식} 과 동일한 작업을 수행한다.
  - XSS(Cross Site Scripting) 취약점 공격에 대한 가장 기본적인 방어을 지원한다.
    - XSS는 악의적인 사용자가 공격하려는 사이트에 스크립트를 넣어서 쿠키나 세션아이디와 같은 정보를 취득하는 것을 말한다.
    - 주로 공개된 게시판의 게시글에 html과 script 코드를 입력해서 해당 스크립트를 실행시키는 것이다.
    - 사이트 이용자가 작성하는 컨텐츠는 반드시 <c:out />태그를 사용해서 출력하자.
  - 사용법

```
  <c:out value="${표현식}" />
```

  - 주요 속성
    - value
      - 출력할 값을 지정한다.

        - 생략할 수 없음

```
<c:out value="${book.title}" />
```

    - escapeXml
        - 특수문자(< > " ' &)를 변환할지 여부를 지정한다.
        - 기본값은 true다.
        - 생략가능
    - default
        - value에서 지정한 값이 null인 경우 표현할 값을 지정한다.
        - 생략가능

- **<c:if>** 태그
    - if문과 동일한 역할을 수행한다.
    - else에 해당하는 태그는 없다.
    - 사용법

```
<c:if test="${표현식}">
  HTML 컨텐츠
</c:if>
<!-- test에서 제시한 조건이 true면 HTML 컨텐츠가 화면에 출력된다. --->
```

    - 주요 속성
        - test
            - 검사조건을 정의한다.
            - 결과값이 boolean 타입이어야 한다.
            - 생략할 수 없다.
- **<c:choose> <c:when> <c:otherwise>** 태그
    - if ~ else if ~ else if ~ else 와 동일한 역할을 수행한다.
    - 사용법

```
  <c:choose>
    <c:when test="${조건식1}">
      HTML 컨텐츠
      <!-- test에서 제시한 조건식1이 true면 HTML 컨텐츠가 화면
에 출력된다. --->
    </c:when>
    <c:when test="${조건식2}">
      HTML 컨텐츠
      <!-- test에서 제시한 조건식2이 true면 HTML 컨텐츠가 화면
에 출력된다. --->
    </c:when>
    <c:when test="${조건식3}">
      HTML 컨텐츠
      <!-- test에서 제시한 조건식3이 true면 HTML 컨텐츠가 화면
에 출력된다. --->
    </c:when>
    <c:otherwise>
      HTML 컨텐츠
      <!-- test에서 제시한 조건식1, 조건식2, 조건식3이 전부 fal
se면 HTML 컨텐츠가 화면에 출력된다. --->
    </c:otherwise>
  </c:choose>
```

- <c:when>과 <c:otherwise>는 반드시 <c:choose>안에 위치해야 한다.
- <c:when>은 조건식을 다르게 해서 여러 번 정의할 수 있다.
- <c:otherwise>는 생략할 수 있고, 맨 마지작 <c:when> 다음에 정의해야 한다.
- <c:when>으로 제시한 조건식이 true가 되면 남아있는 <c:when>은 검사하지 않는다.
- <c:otherwise>는 <c:when>으로 제시한 조건이 모두 false로 판정될 때만 실행된다.

- **<c:forEach>** 태그
  - for문과 같은 역할을 수행한다.
  - 배열 혹은 콜렉션(List, Set)에 대해서 그 항목의 갯수만큼 반복작업을 수행한다.
  - 사용법

```
<c:forEach var="변수명" items="${표현식}">
  <p>${변수명}<p>
</c:forEach>
```

- items의 표현식으로 찾은 값이 반드시 배열 혹은 콜렉션이어야 한다.
- 검색된 배열 혹은 콜렉션의 요소 갯수만큼 컨텐츠를 반복 출력한다.
- var는 배열 혹은 콜렉션에서 반복수행시 해당 요소가 저장되는 변수의 이름을 지정한다.

```
<c:forEach var="변수명1" items="${표현식}" varStatus="변수명2">
  <p>${변수명2.count} ${변수명1}<p>
</c:forEach>
```

- varStatus에 지정된 변수에는 현재 반복상태정보를 담고 있는 객체가 전달된다.
- 반복상태 정보
  - index : 현재 추출한 요소의 인덱스번호(0부터 시작)
  - count : 현재 반복횟수(1부터 시작)
  - first : 첫번째 요소인 경우 true
  - last : 마지막번째 요소인 경우 false

```
<c:forEach var="변수명" begin="${표현식}" end="${표현식}">
  <a href="list.hta?pageno=${변수명}">${변수명}</a>
</c:forEach>
```

- begin은 시작값, end는 끝값을 지정한다.
- begin과 end는 정수값이어야 한다.
- begin, end의 구간만큼 컨텐츠를 반복출력한다.
- var는 현재 숫자값을 저장하는 변수의 이름을 지정한다.

- <c:set> 태그
  - EL로 표현할 수 있는 값을 가진 변수를 선언한다.
  - 사용법

```
<c:set var="변수명" value="${표현식}" />
<c:set var="변수명" value="값" />
<c:set var="변수명" value="<%=값 %>" />
```

- var 지정된 이름으로 값이 저장된다.
- ${변수명}으로 저장된 값을 표현할 수 있다.
- scope는 값이 저장될 객체를 지정한다.

```
<c:set var="a" value="길동" />
<%-- pageContext.setAttribute("a", "길동"); --%>
<p>${a}</p>

<c:set var="b" value="길동" scope="page"/>
<%-- pageContext.setAttribute("b", "길동"); --%>
<p>${b}</p>

<c:set var="c" value="유신" scope="request"/>
<%-- request.setAttribute("c", "유신"); --%>
<p>${c}</p>

<c:set var="d" value="감찬" scope="session"/>
<%-- session.setAttribute("d", "감찬"); --%>
<p>${d}</p>

<c:set var="e" value="순신" scope="application"/>
<%-- application.setAttribute("e", "순신"); --%>
<p>${e}</p>
```

- <c:remove> 태그
  - 지정된 이름의 값(객체)을 속성에서 삭제한다.
  - 사용법

```
<c:remove var="변수명" />

<c:remove var="a" />
<%-- pageContext.removeAttribute("a"); --%>
```

```
  <c:remove var="b" scope="page"/>
  <%-- pageContext.removeAttribute("b"); --%>

  <c:remove var="c" scope="request"/>
  <%-- request.removeAttribute("c"); --%>

  <c:remove var="d" scope="session"/>
  <%-- session.removeAttribute("d"); --%>

  <c:remove var="e" scope="application" />
  <%-- application.removeAttribute("e"); --%>
```

- <c:url> 태그
  - url을 생성한다.
  - 사용법

```
  <c:url var="변수명" value="경로">
    <c:param name="이름1" value="값1" />
    <c:param name="이름2" value="값2" />
    <c:param name="이름3" value="값3" />
  </c:url>

  <a href="${변수명}">링크</a>
  <%-- <a href="경로?이름1=값1&이름2=값2&이름3=값3">링크</a> -
-%>
```

- <c:import> 태그
  - 지정된 파일을 include한다.
  - 사용법

```
  <c:import url="포함시킬파일의 경로" />
```

  - url에는 프로젝트 내부 파일 및 외부파일(다른 웹서버의 파일) 포함 가능

국제화 태그

- fmt:formatNumber 태그

    - 숫자를 지정된 형식에 맞게 출력한다.
    - 사용법

```
<fmt:formatNumber value="${표현식}" />
```

    - 숫자에 콤마가 포함되어서 출력된다.

```
<fmt:formatNumber value="${표현식}" pattern="패턴문자열" />
```

    - 숫자가 지정된 패턴형식에 맞게 출력된다.
    - value에서 지정된 표현식으로 검색되는 값은 반드시 숫자값이어야 한다.
- fmt:formatDate 태그
    - 날짜를 지정된 형식에 맞게 출력한다.
    - 사용법

```
<fmt:formatDate value="${표현식}" />
```

    - 날짜가 년월일시분초 형태로 출력된다.

```
<fmt:formatDate value="${표현식}" pattern="패턴문자열" />
```

    - 날짜가 지정된 패턴형식에 맞게 출력된다.
    - value에서 지정된 표현식으로 검색되는 값은 반드시 Date 타입이어야 한다.
- fmt:bundle, fmt:message 태그
    - 국제화처리를 지원하는 태그다.
    - 사용법

```
<fmt:bundle basename="패키지경로.기본메세지번들파일명"></fmt:bundle>
```

    - JSP에서 사용할 메세지 번들 파일을 지정한다.
    - 국제화처리를 지원받기 위해서는 HTML 컨텐츠가 fmt:bundle와 </fmt:bundle> 안에 위치해야 한다.

- <fmt:message var="별칭" key="번들파일에 정의한 키" />
  - 번들파일에서 키에 해당하는 텍스트를 읽어서 출력한다.
  - 사용예

```
  <h1><fmt:message key="label.home.title"/></h1>

  <fmt:message var="submit_message" key="label.home.btn.
submit"/>
  <input type="submit" value="${submit_message}" />
```

- 국제화처리 절차
  - 지원할 각 언어별로 메세지 번들 파일을 만든다.
    - /src/resources/messages.properties 기본 메세지번들 파일
    - /src/resources/messages_ko.properties 한국어용 메세지번들 파일
    - /src/resources/messages_en.properties 영어용 메세지번들 파일
    - /src/resources/messages_ja.properties 일본어용 메세지번들 파일
      - 언어별 코드
        - 한국어 ko, 영어 en, 중국어 zh, 일본어 ja, 프랑스어 fr, 독일어 de, 스페인어 es
      - 국가별 코드
        - 한국 kr, 미국 us, 영국 gb, 중국 cn, 일본 jp, 프랑스 fr, 독일 de, 스페인 es
  - JSP 파일에서 메세지번들 파일의 기본 경로 및 이름을 설정한다.

```
  <fmt:bundle basename="패키지경로.기본메세지번들파일명" />
```

  - 메세지를 표현할 곳에 메세지번들파일에서 정의한 키를 지정한다.

```
  <fmt:message key="번들파일에 정의한 키" />
```

모델2 MVC 패턴

Front Controller

-서블릿-

HTTP 요청접수 컨트롤러 실행

Controller(유효성체크, 값 가져오기)

-JAVA 클래스 -

HTTP 요청처리

-JSP-

View

데이터 표현

Model

Service

-JAVA클래스-

업무로직수행

DAO

-JAVA클래스-

DB엑세스

@WebServlet(urlPatterns = "*.do")

- **단일 진입점을 만듬 .do를 넣으면 들어가짐**

META-INF

WEB-INF 는 보안 폴더

모델2의 키포인트 같은 요청객체를 공유한다

요청객체는 속성이란것을 담을 수 있다.

컨트롤과 뷰 사이에

프론트 컨트롤러가 요청 받아서 적절한 컨트롤러한테줌

적절한 컨트롤러는 값을 요청객체에 담아서 다시 프론트컨트롤러한테줌

프론트 컨트롤러는 view에 주고 요청객체는 사라짐

JSP내장객체

requset 객체가 적절한 객체임

단일 진입점 프론트 컨트롤러

setAttribute 사용 요청객체에 값을 담음

보여주러가는 과정이 내부이동 forward

모델2 메커니즘

요청을 접수받는 프론트컨트롤러

분석해서 실행 →

컨트롤러 필요하면 값을 담는다 →

어느 jsp에 이동해야할지 알려줌

jsp는 값을 표현한다 .

model 방식의 요청처리 흐름

// 사용자

- 1. 클라이언트가(웹브라우저)가 http://locallhost/model2/home.do을 서버(Tomcat)로 보낸다.
- 22. 클라이언트(웹브라우저)가 응답으로 받은 HTML 컨텐ㅊ트를 화면에 표시한다.

// 톰켓

- 2. 톰캣을 클라이언트 요청을 받고, HttpServletRequest,HttpServletResponse객체를 생성한다.
- 3. *.do로 끝나는 요청을 처리한 FrontController객체의 service(request, response)를 호출한다.
- 21. 응답이 완료되면 HttpServletRequest,HttpServletResponse 객체를 폐기한다.

// FrontController

- 4. 요청 URL을 분석한다
- 5. 요청 URL에 해당하는 요청을 처리하는 컨트롤러 객체를 생성한다.
- 6. 컨트롤러 객체의 exe(request.response)를 호출한다.
- 16. 컨트롤러가 반환하는 경로를 분석한다.
- 17. 내부이동이 필요한 경우, 해당 JSP로 내부이동시키는 RequestDispatcher객체를 획득한다.
- 18. RequestDispatche객체의 forward(request, response)를 호출해서 jsp로 내부이동시킨다.

// XXX JSP

- 19. 컨트롤러가 요청객체에 속성으로 저장한 데이터를 표현한다
- 20. 데이터가 표현된 HTML 컨텐츠를 응답으로 보낸다.

// XXX 컨트롤러 (적절한 컨트롤러)

- 7. 전달받은 요청객체에서 요청파라미터를 조회한다.
- 8. 요청 파라미터값을 적절한 객체에 담는다.
- 9. 업무로직을 수행하는 서비스객체의 업무로직 메소드를 호출한다.
- 14. 서비스가 반환하는 데이터를 요청객체에 속성으로 저장한다.
- 15. 이동할 경로를 프론트 컨트롤러로 반환한다.

// XXXX 서비스

- **10. 업무로직 수행에 필요한 정보를 데이터베이스에서 조회한다.**
- **11. 업무로직을 수행한다.**
- **12. 업무로직 수행결과를 데이터베이스에 반영한다.**
- **13. 업무로직 수행 결과를 컨트롤러에 반환한다.**

오후수업

EL(Expression Language)

EL 표현식은 오류를 잘 안보여줌 찾기 힘듬 null값도 화면에 안 보여줌

- **요청파라미터값**, 초기화파라미터값의 **표현**
- PageContext, **요청객체**, **세션객체**, 애플리케이션객체의 **속성값 표현 ← 이거로 많이씀**
- 요청헤더정보, 쿠키값의 표현
- **사칙연산, 비교연산, 논리연산자, 기타 연산자 제공**
- 메소드 호출 기능 제공

```
${EL표현식}
  <p>${표현식}</p>
  <img src="resources/images/${표현식}" />
  <input type="text" name="username" value="${표현식}" />
  <a href="list.hta?pageNo=${표현식}&cateNo=${표현식}">링크</a
>
  <div id="box-${표현식}"> ... </div>
```

연산자

구분	연산	연산자	사용사례
사칙연산	덧셈	+	${표현식 + 표현식}
사칙연산	뺄셈	-	${표현식 - 표현식}
사칙연산	곱셈	*	${표현식 * 표현식}
사칙연산	나눗셈	div	${표현식 div 표현식}

사칙연산	나머지	mod	${표현식 mod 표현식}
비교연산	크다	gt	${표현식 gt 표현식}
비교연산	크거나 같다	ge	${표현식 ge 표현식}
비교연산	작다	lt	${표현식 lt 표현식}
비교연산	작거나 같다	le	${표현식 le 표현식}
비교연산	같다	eq	${표현식 gt 표현식}
비교연산	같지않다	ne	${표현식 ne 표현식}
논리연산	논리곱	and	${표현식 gt 10000 && 표현식 eq 'VIP'}
논리연산	논리합	or	${표현식 gt 50000 or 표현식 eq 5}
논리연산	논리부정	not	${not 표현식 eq 'VIP'}
삼항연산	삼항연산	? :	${표현식 비교연산자 비교값 ? 값1 : 값2}
기타연산	기타	empty	${ empty 표현식} 혹은 ${not empty 표현식}

httpSession 클라이언트마다 객체 생성

Servletcontext

요청마다 만드는 객체

EL 내장객체

param	${param.파라미터명}	요청파라미터값을 조회할 수 있다.request.getParameter("파라미터명")와 동일하다.

JSTL(JSP Standard Tag Library)

- **JSP 표준태그 라이브러리**
- **스크립틀릿을 사용해서 작성했던 자바코드를 대체할 수 있다.**
  - **변수 선언/삭제, 값 출력, 제어문 처리, 반복문 처리, 숫자나 날짜에 대한 포맷팅, 국제화처리, URL처리**

사용법

- **JSTL 파일을 다운받아서 WEB-INF/lib에 복사한다.**

- **JSP 파일에 사용할 태그라이브러리를 지시어를 사용해서 정의한다.**

```
<%@ taglib prefix="별칭" uri="태그라이브러리식별자" %>
<별칭:태그명 value="${EL표현식}" />
```

<c:forEach> 태그

중요함 많이 사용함 반복처리

- for문과 같은 역할을 수행한다.
- 배열 혹은 콜렉션(List, Set)에 대해서 그 항목의 갯수만큼 반복작업을 수행한다.
- 사용법

```
<c:forEach var="변수명" items="${표현식}">
  <p>${변수명}<p>
</c:forEach>
```

- items의 표현식으로 찾은 값이 반드시 배열 혹은 콜렉션이어야 한다.
- 검색된 배열 혹은 콜렉션의 요소 갯수만큼 컨텐츠를 반복 출력한다.
- var는 배열 혹은 콜렉션에서 반복수행시 해당 요소가 저장되는 변수의 이름을 지정한다.

```
<c:forEach var="변수명1" items="${표현식}" varStatus="변수명
2">
  <p>${변수명2.count} ${변수명1}<p>
</c:forEach>
```

- varStatus에 지정된 변수에는 현재 반복상태정보를 담고 있는 객체가 전달된다.
- 반복상태 정보
  - index : 현재 추출한 요소의 인덱스번호(0부터 시작)
  - count : 현재 반복횟수(1부터 시작)
  - first : 첫번째 요소인 경우 true
  - last : 마지막번째 요소인 경우 false

```
<c:forEach var="변수명" begin="${표현식}" end="${표현식}">
  <a href="list.hta?pageno=${변수명}">${변수명}</a>
</c:forEach>
```

- begin은 시작값, end는 끝값을 지정한다.
- begin과 end는 정수값이어야 한다.
- begin, end의 구간만큼 컨텐츠를 반복출력한다.
- var는 현재 숫자값을 저장하는 변수의 이름을 지정한다.

fmt 태그

- 숫자나 날짜에 대한 포맷팅을 지원한다. 숫자 3개마다 ,찍히는것 등
- 국제화처리를 지원한다. 사용자가 입력한건 국제화처리가 안됨.
- JSP에 아래의 지시어를 정의한다.

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
```

fmt:formatDate 태그

- 날짜를 지정된 형식에 맞게 출력한다.
- 사용법

```
<fmt:formatDate value="${표현식}"/>
```

- 날짜가 년월일시분초 형태로 출력된다.

```
<fmt:formatDate value="${표현식}" pattern="패턴문자열" /> yyyy년 M월 d일 EEEE요일 a오전,오후 h시 m분 s초
```

- 날짜가 지정된 패턴형식에 맞게 출력된다.
- value에서 지정된 표현식으로 검색되는 값은 반드시 Date 타입이어야 한다.