KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

EL, JSTL

[KB] IT's Your Life

앞에서 배운 jsp기억나능가?

<% %> 형식의 jsp 디렉티브태그가 있었고
<jsp: ,,, > 형식의 jsp 액션태그가 있었다.
하지만 좋지 않았다. model1에서 쓰이는 것들이고
model1은 실무에서 하지 않는다 불편해서
jstl이이를 커버해준다

앞서본 포워딩하여 jsp가 출력할 상황에서

필요한 기술이 el,jstl 다른 표현방법이다. 태그형식

디렉티브 표현은 웬만하면 쓰지 마라; 로직 표현은 하지말아야하고 애초에.

변수쵸현도 스코프에서 꺼내 쓸때 <%= request.getAttribute(key) %> 너무 길고 복잡
그리고 널 처리도 까다로움

KB국민은행

1 EL (Expression Language)

- **EL 개요** jsmp의 기본기능
  - EL은 데이터를 출력하기 위한 언어
  - 문법이 직관적, 사용이 용이
  - JSP에서 변수를 출력할 때 사용
  - 처리 가능한 데이터형
    - 프리미티브
    - Map, List, 배열, 자바빈 등

디렉티브의 표현식을 대체할
쉽게 사용할 수있는 Expression language

추가 library 필요X

EL 내장 객체

jsp와 별개로 EL이 사용하는 여러가지 내장 객체

내장 객체	설 명
pageScope	page 영역에 존재하는 변수 참조 시 사용
requestScope	request 영역에 존재하는 변수 참조 시 사용
sessionScope	session 영역에 존재하는 변수 참조 시 사용
applicationScope	application 영역에 존재하는 변수 참조 시 사용
param	파라미터 값을 참조 시 사용
paramValues	파라미터 배열 값을 참조 시 사용
header	헤더 정보 값을 참조 시 사용
headerValues	헤더 배열 정보 값을 참조 시 사용
cookie	쿠키 정보를 참조 시 사용
initParam	context 초기화 파라미터 참조 시 사용
pageContext	pageContext 참조 시 사용

나이스

req.getParameter와 대응

헤더 접근.

쿠키에 접근

EL 연산자

연산자	설명
.	자바빈 또는 Map에 접근할 때 사용
[]	배열 또는 List에 접근할 때 사용
()	우선 순위 연산자
empty	값이 null인지 판단하는 연산자로서 true 리턴
+, -, *, /, %	산술 연산자자 및 나머지 연산자
&&, ||, !	논리 연산자
==, >, >=, <, <=, !=	비교 연산자

멤버 접근용 (. 과 [])

검사 (empty)

연산자들 (산술 연산자자 및 나머지 연산자 ~ 비교 연산자)

EL은 작업대상이 스코프에 저장되어있다고 가정함.

1 EL (Expression Language)

EL 기본 문법

vue3에서 {{표현식 }}와 대응

${표현식} 표기 방법!

- 내장 객체이거나 scope에 저장된 속성을 지정

자바객체.프로퍼티명 O 객체.프로퍼티

객체.멤버명(속성명)X!!!!

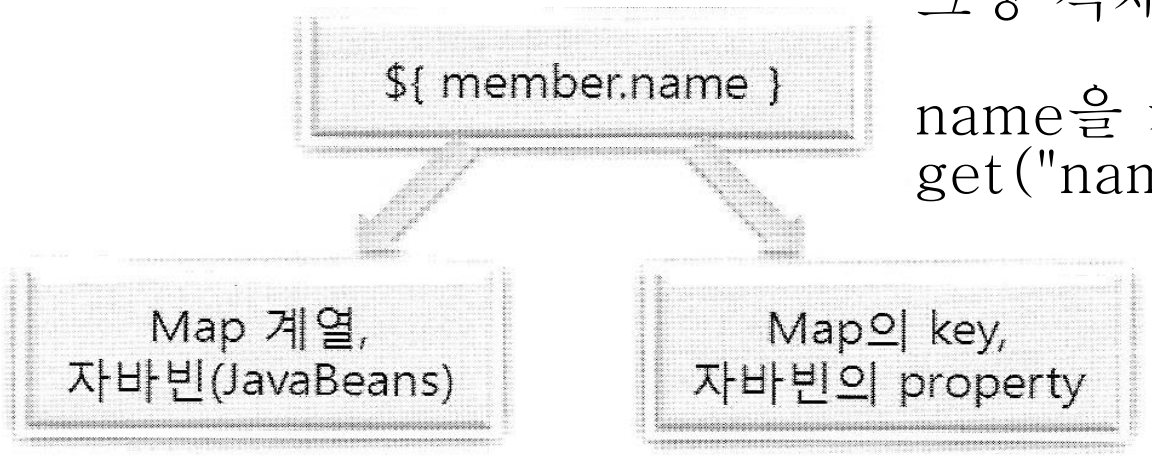
직관적이고 간단한 표현이지만
내부에서는 바쁨.
일단 member검사(1. 멤버검색. 문자열 키member
스코프에서 찾음. 검색스코프 순서

페이지부터 앱까지 순서대로 검색.작->큰범
있으면 검색 멈추고 사용
2) 하위. 찾기. 값의 유무를 검사함.

- <%= %>과의 차이
  - Scope의 속성 값이 아닌 JSP 변수 및 표현식을 출력

있는 경우에만 출력함. 없으면 아무것도 출력X
속성일경우 (멤버(속성)가 아닌 프로퍼티로봄)
getName메서드 호출하여 얻음.
즉 하위 .name에서 name은 getter이름임.
getter와setter로 접근하는 요소를 프로퍼티라고함.

- Map 계열이거나 자바빈이 지정된 경우
  - 두 번째 값은 Map의 키 값이거나 자바빈의 프로퍼티

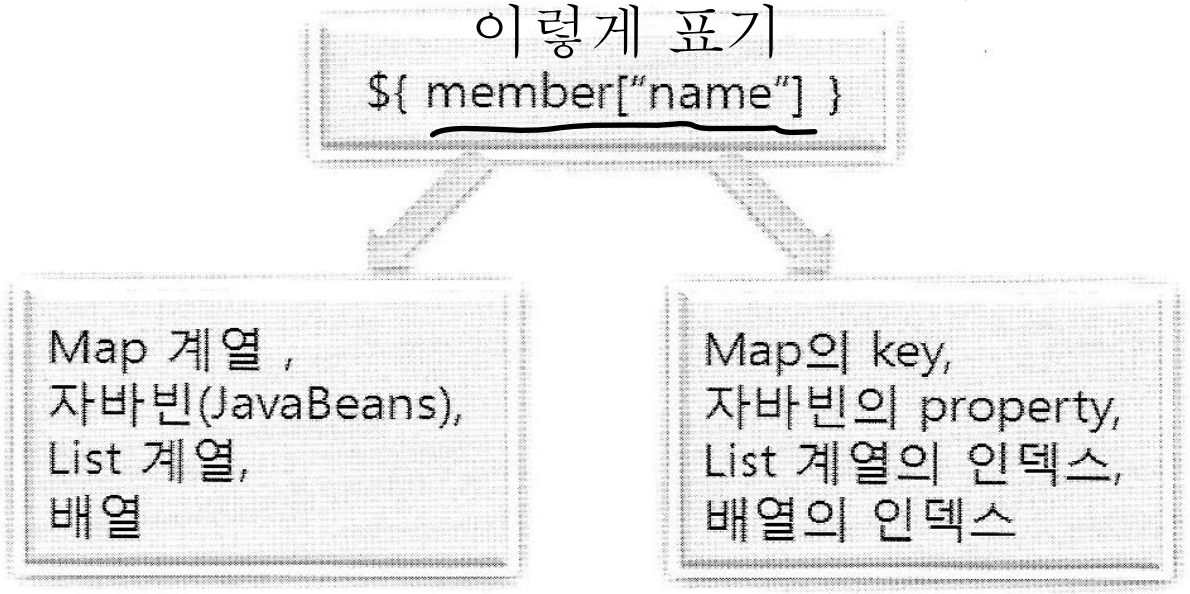


만약 member가
그냥 객체가 아닌 map객체라면

name을 키명으로 본다.
get("name")이 호출되어 얻음

- [] 배열 표기법

이렇게 쓸수도 있음.
내가 사용한 프로퍼티명이 변수에 담겨있을 때는
이렇게 표기



우리는 신경안쓰고 사용하면 됨

login_form.jsp

서블릿경로일 가능성 아주 높

```
<body>
<form action="login" method="get">
    <fieldset>
        <legend>로그인 폼</legend>
        <ul style="list-style: none;">
            <li>
                <label for="userid">아이디</label>
                <input type="text" id="userid" name="userid">
            </li>
            <li>
                <label for="passwd">비밀번호</label>
                <input type="password" id="passwd" name="passwd">
            </li>
            <li>
                <input type="submit" value="전송">
            </li>
        </ul>
    </fieldset>
</form>

</body>
```

겟 메서드

LoginServlet.java

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {

        String userid = req.getParameter("userid");
        String passwd = req.getParameter("passwd");

        req.setAttribute("userid", userid);
        req.setAttribute("passwd", passwd);

        req.getRequestDispatcher("login.jsp").forward(req, res);
    }
}
```

mvc모델에선 데이터 얻는 과정을
데이터 얻기 모델객체로 진행한다

검증 및
비즈니스 로직

리퀘스트 스코프에 저장

가공된 것도 모델로

해당 jsp로 포워딩

login.jsp 결과처리 출력 화면`

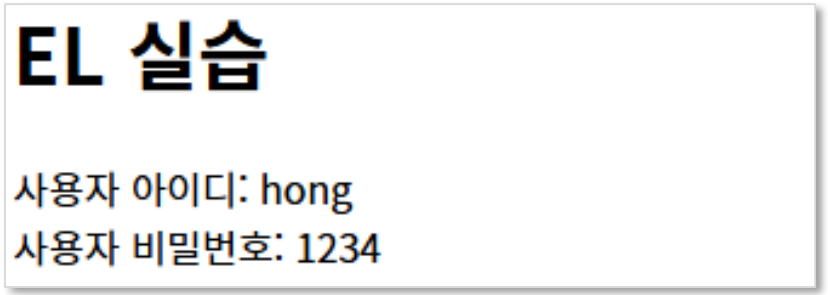
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>EL  실습</h1>
    사용자 아이디: ${userid}<br>
    사용자 비밀번호: ${passwd}<br>
</body>

</html>
```

값이 생략되었거나
{ } 안 표현식이
잘못됐으면
출력이 안된다.
아무것도

문자열 값 "userid", "passwd"를
스코프들에서 검색을 함.
스코프 지정 안해놓음.
일반적으로 여러스코프에 동일한 이름
으로 정보를 저장할 일이 별로 없음.

http://localhost:8080/login_form.jsp

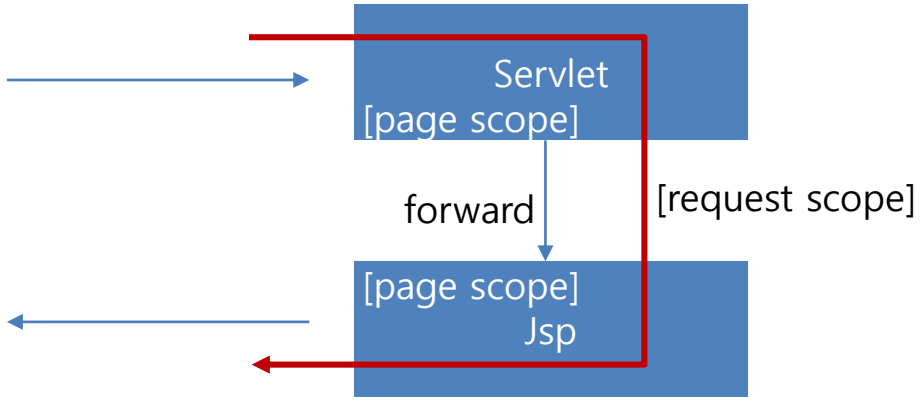
http://localhost:8080/login?userid=hong&passwd=1234

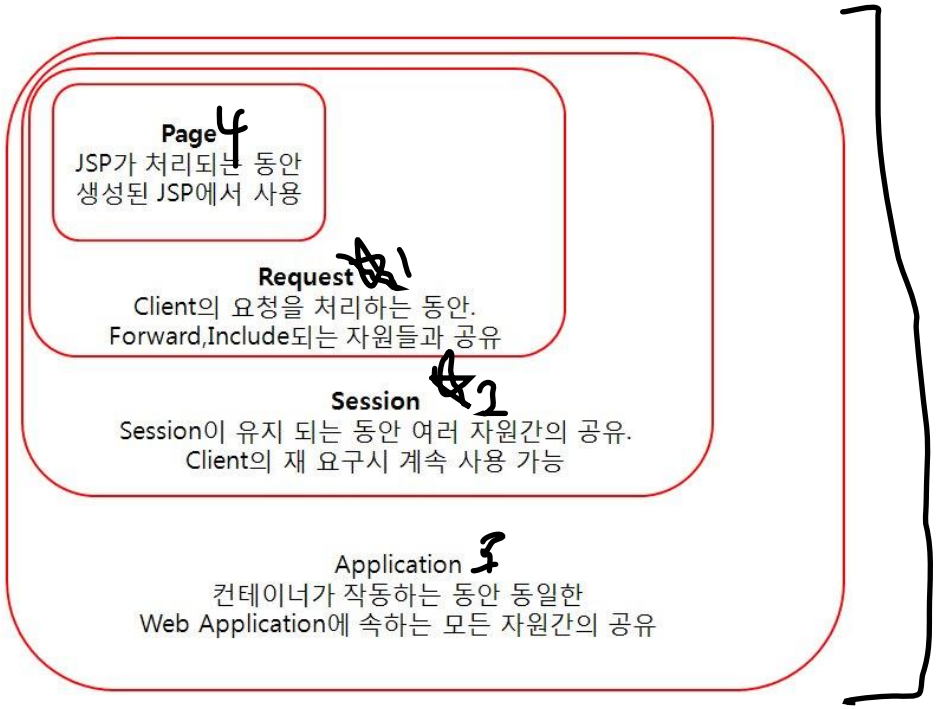


1 EL (Expression Language)

Scope 어느 스코프를 쓸 것인가

- page scope
  - 한 페이지(servlet 또는 jsp 파일) 내에서만 가능
  - 일종의 지역 변수
- request scope
  - request 객체에 저장되는 속성
  - 접속이 끊길 때까지 유지
- session scope
  - session 객체에 저장되는 속성
  - 세션이 만기되기 전 까지 유지
- application scope
  - application contex에 저장되는 속성
  - 어플리케이션이 종료할 때까지 유





스코프 객체

각각스코프에 대응하는 스코프객체

스코프 명	EL(jsp)	Servlet
page scope	pageScope	지역변수
request scope	requestScope	HttpServletRequest
session scope	sessionScope	HttpSession
application scope	applicationScope	ServletContext

EL에서 스코프 객체를 지정하지 않은 경우 속성 찾는 순서

앞서 말했듯

작은 단위에서 큰 단위로

- 예) ${username}

  page → request → session → application

  특정 스코프에서 속성이 발견되면 그 값을 출력

스코프 관련 실습 시작

Member.java

```
package org.scoula.ex05.domain;

public class Member {
    private String name;
    private String userid;

    public Member() {
    }

    public Member(String name, String userid) {
        this.name = name;
        this.userid = userid;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getUserid() {
        return userid;
    }

    public void setUserid(String userid) {
        this.userid = userid;
    }
}
```

ScopeServlet.java

```
@WebServlet("/scope")
public class ScopeServlet extends HttpServlet {
    ServletContext sc;

    @Override
    public void init(ServletConfig config) throws ServletException {
        sc = config.getServletContext();
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {

        sc.setAttribute("scopeName", "applicationScope 값"); // Application Scope

        HttpSession session = req.getSession(); // Session Scope
        session.setAttribute("scopeName", "sessionScope 값");

        req.setAttribute("scopeName", "requestScope 값"); // Request Scope
        Member member = new Member("홍길동", "hong");
        req.setAttribute("member", member);

        req.getRequestDispatcher("scope.jsp").forward(req, res);
    }
}
```

서블릿에서는 이 타입이 application scope

앱스코프
scopeName-apscope값

세션스코프
scopeName-seccope값

리퀘스코프
scopeName-reqscope값
member-멤버객체

scope.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>scope 데이터 보기 </h1>

    pageScope의 속성값은 : ${pageScope.scopeName}<br>
    requestScope의 속성값은 : ${requestScope.scopeName}<br>
    sessionScope의 속성값은 : ${sessionScope.scopeName}<br>
    applicationScope의 속성값은 : ${applicationScope.scopeName}<br>

    scopeName 자동 찾기: ${scopeName} <br>
    member: ${member.name}(${member.userid})<br>
</body>
</html>

</html>
```

∨ 데이터 없음

작은스코프부터 키 있는지 검색 있으면 값객체 사용

값객체 있으면 프로퍼티getter 호출하여 최종 값 가져옴

EL은 간단하고 편하지만

표현식만을 표현하느데 한계가 있음

로직을 아예 안슬순 없음 jsp에서도.

출력전용 로직이 필요함 가령 배열객체의 정보를 다 출력해야할대 처럼 => jstl사용

http://localhost:8080/scope

scope 데이터 보기

pageScope의 속성값은 :
requestScope의 속성값은 : requestScope 값
sessionScope의 속성값은 : sessionScope 값
applicationScope의 속성값은 : applicationScope 값
scopeName 자동 찾기: requestScope 값
member: 홍길동(hong)

태그로써 로직처리해야할게 필요함

2 JSTL(Jsp Standard Tag Library)

JSTL 개요와 환경 설정

- 액션 태그를 사용자가 직접 제작 가능 → 커스텀 태그
- 커스텀 태그들 중에서 자주 사용되는 태그들을 묶어서 아파치 그룹에서 제공하는 것 → JSTL
- EL과 JSTL을 함께 사용

2 JSTL(Jsp Standard Tag Library)

jar형식다운

KB국민은행

- **Jstl 다운로드**
  - mvnrepository 에서 jstl 검색

톰캣에 설치하면
모든 프로젝트에
적용된다.
gradle말고
직접 다운로드
받아서 톰캣
디렉에 옮기기

JSTL » 1.2

JavaServer Pages Standard Tag Library (JSTL) pro
common tasks like iteration, conditionals, format
JSP.

License	CDDL GPL 2.0
Categories	Java Specifications
Tags	standard servlet javax jstl
Date	Jun 23, 2011
Files	pom (356 bytes) jar (404 KB)

프로젝트마다 적용하기
gradle사용하여.
거의 필수여서 매번해야함

Sort: **relevance** | popular | newest

1. **JSTL**
javax.servlet.jsp.jstl » jstl
JSTL
Last Release on May 14, 2015

2. **JSTL** — **2,066** usages
javax.servlet » jstl
GPL CDDL
JSTL
Last Release on Jun 23, 2011

3. **JSTL** — **291** usages
jstl » jstl
JSTL
Last Release on May 12, 2006

톰캣에 설피하여 전체 적용 하든
프로젝에만 설치하여 적용하든
해야하는 작업 하는이유는 지정도 있지만
인텔리제이에게도 알리는것

build.gradle

```
dependencies {
    compileOnly('javax.servlet:javax.servlet-api:4.0.1')

    implementation 'javax.servlet:jstl:1.2'

    testImplementation("org.junit.jupiter:junit-jupiter-api:${junitVersion}")
    testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:${junitVersion}")
}
```

컴파일할때만 써라. 배포할때는 빼라. 라는 뜻
톰캣이 갖고있기 때문에 배포할때는 괜찮아서

impl은 컴파일,배포할때도 써라 라는 뜻. 지정된
라이브러리는 webapp-lib에 설치되어서 배포됨.

→ 추가 후 Sync!!

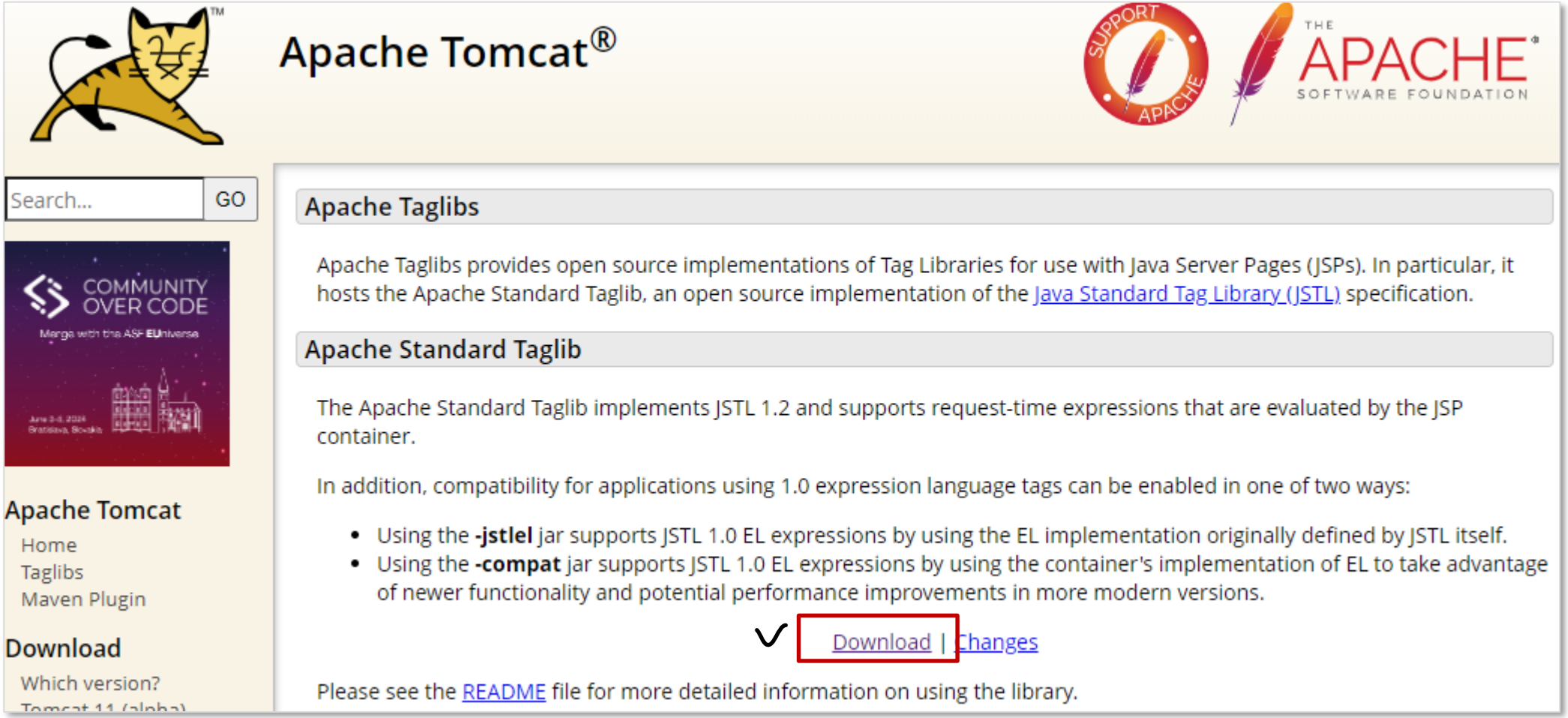
그라들의 의해서 사용 지정된 라이브러리는
webapp-lib아래에 모여있다. 해당 앱에서만 쓰는 라이브러리.

톰캣자체에서 가지고있는 라이브러리랑은 또 다른 의미.

그렇다면 아까 우리는 jstl을 톰캣자체 라이브러리에 설치했으니.
comopileOnly로 jstl을 지정해도 되겠네?

그렇다!. 톰캣자체가 갖고 ㅗ있으니 배포할때 앱자체에서 안갖고 있어도 되니까.
하지만 돌아갈 톰캣 자체에 있는지 확인이 안되면 저렇게 하는게 안심되지

- **taglibs** 톰캣 홈ㅁ페이지에서도 다운로드가 가능하다
  - https://tomcat.apache.org/taglibs.html



taglibs

Jar Files

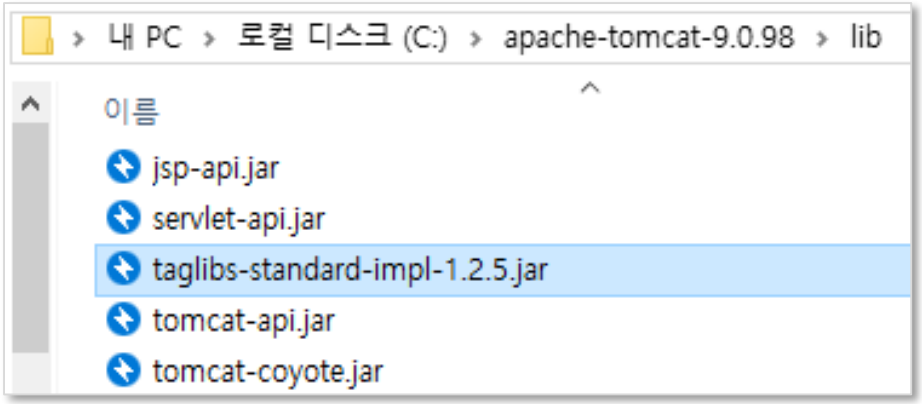
- Binary README
- Impl:
  - taglibs-standard-impl-1.2.5.jar (pgp, sha512)
- Spec:
  - taglibs-standard-spec-1.2.5.jar (pgp, sha512)
- EL:
  - taglibs-standard-jstlel-1.2.5.jar (pgp, sha512)
- Compat:
  - taglibs-standard-compat-1.2.5.jar (pgp, sha512)

앞서 한거랑
동일한 효과

- taglibs-standard-impl-1.2.5.jar 배치
  - C:₩apache-tomcat-9.0.89₩lib
    모든 프로젝트에 적용

  또는

  - WEB-INFO/lib
    이번 프로젝트에만 사용



JSTL 라이브러리

기능별로 5가지로구분

보편적인 관례 prefix

라이브러리	URI	Prefix	예제
Core	http://java.sun.com/jsp/jstl/core	c	<c:tag ...> 프로그램로직처리
I18N formatting	http://java.sun.com/jsp/jstl/fmt	fmt	<fmt:tag ...> 포매팅, 국제화, 문자셋
Functions	http://java.sun.com/jsp/jstl/functions	fn	fn:function(...) 유틸리티
SQL	http://java.sun.com/jsp/jstl/sql	sql	<sql:tag ...> 디비처리
XML processing	http://java.sun.com/jsp/jstl/xml	x	<x:tag ...> xml 데이터 처리

국제서비스시 만히 활용

서블릿과 겹침

taglib 지정자로 설정

이건 페이지디렉티브로 어디에 정의 되어있는지

태그명이 중복될때 어느 소속인지 알기위해

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

prefix==namespace

굳이,, 필요하면 서블릿 자바파일에서ㅡ,,

태그에서의 namespace == prefoix
<prefix:태그명>

JSTL Core 라이브러리

- 기본적이고 핵심적인 기능들을 구현해 놓은 라이브러리
- 문자열 출력, if문, for문과 같은 제어문 기능 포함

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

JSTL Core 라이브러리에 포함된 태그 목록

스코프에 있는 속성에 대한 작업들을 함. 출력 set 제거

스코프를 대상으로 하는

태그	설명	사용예
out	지정된 값을 출력할 때 사용	<c:out ...>
set	JSP의 setAttribute(name, key) 기능과 동일. scope에 따른 바인딩 처리가 가능	<c:set ...>
remove	JSP의 removeAttribute(name) 기능과 동일. scope에 따른 속성 제거가 가능	<c:remove ...>

하지만 잘 안씀. 특히 <c:out ,,,> 은 el의 ${,,,}로 해결가능하니

JSTL Core 라이브러리에 포함된 태그 목록

로직처리

뒤에는 여러가지 속성

태그	설명	사용예
if	조건 처리를 사용할 때 사용한다.	<c:if ...>
forEach	반복 처리를 하고자 할 때 사용한다.	<c:forEach ...>
choose	자바의 switch문과 비슷하다.	<c:choose ...>
when	choose의 서브 태그로 사용한다. 조건을 만족한 경우에 사용된다.	<c:when>
otherwise	choose의 서브 태그로 사용한다. 조건을 만족하지 못한 경우에 사용된다.	<c:otherwise>
url	URL을 생성하는 기능이다.	<c:url>
forTokens	자바의 StringTokenizer 클래스 기능이다.	<c:forTokens>

<c:set> 태그

- 지정된 scope에 데이터를 바인딩하는 태그
- JSP의 setAttribute(name, value) 메서드와 동일한 기능 제공

```
<c:set var="변수명" value="변수값" />
```
원시 데이터 타입은 변수명과 변수값으로 지정

or

```
<c:set target="객체" property="객체의 프로퍼티" />
```
객체는 객체명과 객체의 프로퍼티명(setter이름)을 지정해야함

```
scope="scope값" />
```
어느 스코프에 대한 작업인지 지정

<c:out> 태그

- 지정된 값을 웹 브라우저에 출력하는 태그

```
<c:out value="출력값" default="기본값" escapeXml="true|false" />
```
default: 없을경우

escapeXml: true면 데이터 false면 태그로 보겠다

출력값중에서 태그형태일경우 유무 선택
"<h1> ... </h1>" 이런 문자열이면,,,

<c:remove> 태그

- 지정된 scope에 설정한 변수 값을 제거하는 태그

```
<c:remove var="변수명" scope="scope값" />
```

<c:if> 태그

- 조건 처리를 할 때 사용하는 태그
- 자바의 if문과 동일

속성명으로 지정 스코프에 test결과 값 저장됨

참이면 출력

```
<c:if test="조건식" var="변수명" scope="scope값">
  문장
</c:if>
```

- 조건식이 참인 경우 출력
- var가 지정되어 있다면 조건식 검사 결과를 변수에 저장
- else는 없음 → 부정표현으로 if 추가

<c:choose>, <c:when>, <c:otherwise> 태그

- 자바의 switch문과 동일한 기능을 제공하는 태그

```
<c:choose>
    <c:when test="조건식">문장</c:when>
    <c:when test="조건식">문장</c:when>
    <c:otherwise>문장</c:otherwise>
</c:choose>
```

앞조건에 만족되지 않은 경우

<c:forEach> 태그

- 반복 처리시 사용하는 태그, for문과 비슷

for(a : list)

택 1

```
<c:forEach var="변수명"  items="객체명"스코프에 저장된 속성객체명
  begin="시작인덱스" end="끝인덱스"  step="증가값"
  varStatus="other변수">
    문장
</c:forEach>
```

이건 for(i=?;i<?;i?)용

${ var에 해당하는 이름 }

- items 속성: 배열이나 List 형태의 반복할 객체 지정
- var : 반복문에서 사용할 아이템 변수

for(PhoneInfo pi : list) 의 형태와 동일

- varStatus : 반복문의 상태에 대한 참조 변수 루프 상황

몇번째 루프인지, 끝인지 시작인지,
루프 횟수인지 varStatus변수의 속성값으로
들어있다

<c:forEach> 태그

- varStatus 속성은 루프 정보를 담는 LoopTagStatus 객체를 이용
  - index – 루프 실행에서 현재 인덱스
  - count – 루프 실행 회수
  - begin – begin 속성의 값
  - end – end 속성의 값
  - step – step 속성의 값
  - first – 현재 실행이 첫 번째 실행인 경우 true
  - last – 현재 실행이 루프의 마지막 실행인 경우 true
  - current – 콜렉션 중 현재 루프에서 사용할 객체

- 현재 사용항목의 인덱스 값은 varStatus 속성을 이용

```
<c:forEach var="item" items="<%=someItemList%>"
      vasrStatus="status">
${status.index +1}번째 항목 : ${item.name}
</c:forEach>
```

JstlServlet.java

```
@WebServlet("/jstl_ex")
public class JstlServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {

        List<Member> members = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
            Member member = new Member("홍길동_" + i, "hong_" + i);
            members.add(member);
        }
        req.setAttribute("members", members);
        req.setAttribute("role", "ADMIN");

        req.getRequestDispatcher("jstl_ex.jsp")
           .forward(req, res);
    }
}
```

members 컬렉션객체 리퀘스트 속성에 넣기

보통 로그인 정보는 세션스코프에

jstl_ex.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"  prefix="c" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>JSTL 테스트</h1>
                    jstl안에 el표기
    <c:if test="${role == 'ADMIN'}">관리자</c:if>
    <c:if test="${role != 'ADMIN'}">일반회원</c:if>

    <table >
        <c:forEach var="member" items="${members}" varStatus="state">
            <tr>
                <td>${state.index}</td>
                <td>${member.name}</td>
                <td>${member.userid}</td>
            </tr>
        </c:forEach>
    </table>
    <br>
</body>
</html>
```

http://localhost:8080/jstl_ex

JSTL 테스트

관리자
0 홍길동_0 hong_0
1 홍길동_1 hong_1
2 홍길동_2 hong_2
3 홍길동_3 hong_3
4 홍길동_4 hong_4
5 홍길동_5 hong_5
6 홍길동_6 hong_6
7 홍길동_7 hong_7
8 홍길동_8 hong_8
9 홍길동_9 hong_9

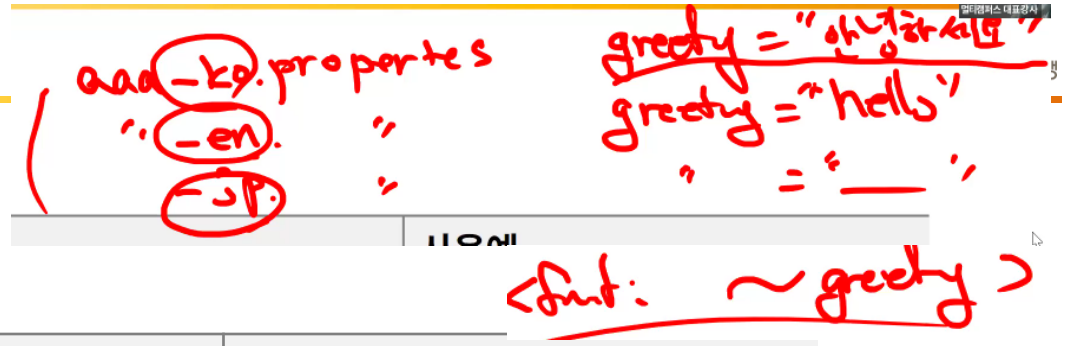
JSTL formatting 라이브러리

- 국제화/ 지역화 및 데이터 포맷과 관련된 기능 제공
- 국제화/지역화 → 다국어 처리
- 데이터 포맷 → 날짜와 숫자와 관련된 기능

언제 수정, 등록 등등 ㅇ한 데이터이냐

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

2 JSTL(Jsp Standard Tag Library)



JSTL formatting 라이브러리

태그	설명	사용예
requestEncoding	setCharacterEncoding(enc) 메서드와 동일한 기능	<fmt:requestEncoding>
setLocale	다국어 페이지 사용 시 언어 지정	<fmt:setLocale ...>
timeZone	지정한 지경 값으로 시간을 설정	<fmt:timeZone ...>
setBundle	*.properties 확장자의 리소스 번들 파일 접근 시 사용	<fmt:setBundle ...>
message	번들에서 설정한 값 사용	<fmt:message ...>
formatNumber	숫자형식의 포맷 지정	<fmt:formatNumber ...>
parseNumber	문자열을 숫자로 변환	<fmt:parseNumber ...>
formatDate	날짜형식의 포맷 지정	<fmt:formatDate ...>
parseDdate	문자열을 날짜로 변환	<fmt:parseDate ...>

<fmt:formatNumber> 태그

- 수치 데이터를 특정 포맷으로 설정 시 사용되는 태그

처리결과를 여기에 저장하겠다는 의미
해당스코프에 저 이름으로

```
<fmt:formatNumber value="값" typ="타입" pattern="패턴" var="값" scope="값"
    currencySymbole="값" minIntegerDigits="값" maxIntegerDigits="값"
    minFractionDigits="값" maxFractionDigits="값" />
```

- value : 실제 수치 값을 지정
- type : number, currency, percent 값 중에서 설정
- pattern : 사용자가 지정한 형식 패턴을 설정
- currentcySymbol : 통화 기호 지정
- maxIntegerDigits : 정수의 최대자리 수를 지정
- minIntegerDigits : 정수의 최소 자릿수를 지정
- maxFractionDigits : 소수점 이하 최대 자리수를 지정
- minFractionDigits : 소수점 이하 최소 자릿 수를 지정

<fmt:formatDate> 태그

날짜 데이터를 특정 포맷으로 설정 시 사용

결과를 여기에 저장해라 해당스코프에 저이름으로

```
<fmt:formatDate value="값" typ="타입" var="값" scope="값"
    dateStyle="날짜 스타일" timeStyle="시간스타일" />
```

- value : 실제 날짜와 시간을 설정
- type : time, date, both 중 하나
- dateStyle : 미리 정의된 날짜 스타일 형식
- timeStyle : 미리 정의된 시간 스타일 형식

미리정의된 스타일

- pattern : 사용자가 지정한 형식 스타일
  - "YYYY-MM-dd HH:mm:ss" HH: 24시간제
  - "YYYY-MM-dd a hh:mm:ss" a: 오전/오후, hh:12시간제

우리가 정의하는 스타일

JstlServlet.java

```
@WebServlet("/jstl_ex")
public class JstlServlet extends HttpServlet {

    protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {

        List<Member> members = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
            Member member = new Member("홍길동_" + i, "hong_" + i);
            members.add(member);
        }
        req.setAttribute("members", members);
        req.setAttribute("role", "ADMIN");

        req.setAttribute("today", new Date());

        req.getRequestDispatcher("jstl_ex.jsp")
           .forward(req, res);
    }
}
```

jstl_ex.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"  prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>JSTL 테스트</h1>
    …
    <br>

    ${today}<br>
    <fmt:formatDate value="${today}" type="date"/><br>
    <fmt:formatDate value="${today}" type="time"/><br>
    <fmt:formatDate value="${today}" type="both"/><br>
    <fmt:formatDate value="${today}" type="both" dateStyle="short" timeStyle="long"/><br>
    <fmt:formatDate value="${today}" type="both" dateStyle="long" timeStyle="short"/><br>
    <fmt:formatDate value="${today}" pattern="YYYY-MM-dd HH:mm:ss"/><br>
    <fmt:formatDate value="${today}" pattern="YYYY-MM-dd a hh:mm:ss"/><br>
</body>
</html>
```

디렉티브 사용X
모델2에서는
EL JSTL사용

여러개로 운영되어 효율적이지 않은
방법을 1개로 운영되는 서블릿을 통해서
해결.
이때 1개인 서블릿을 프론트컨틀로러
라고 부름

여러가지기술이
접목됨.

일종의
프레임웍화됨

여기서 기능이
더 추가된게
스프링

http://localhost:8080/jstl_ex

7 홍길동_7 hong_7
8 홍길동_8 hong_8
9 홍길동_9 hong_9

Mon Jan 13 12:40:07 KST 2025
2025. 1. 13.
오후 12:40:07
2025. 1. 13. 오후 12:40:07
25. 1. 13. 오후 12시 40분 7초 KST
2025년 1월 13일 오후 12:40
2025-01-13 12:40:07
2025-01-13 오후 12:40:07

번외: 요청1개당 – 서블릿 1개 정의됨. 우리의 웹앱사이트가 요청이 100개 있다.–서블릿100개정의?!?!
통제하기 힘들어짐. 자동화하기도 힘듬. 서블릿들에서 모델객체를통해서 데이터를 가져오는 중복코드가
많아. 에러처리할때도 에러.jsp로 이동해라하는 코드가 많이 중복됨. => 비효율적인 문제.
자동화와 통제하기가 힘듬. 이처리를 어떻게 개선할까.=>원인:서블릿이많다..해결 서블릿1개로
운영. how? 공통적으로 보이는 패턴이 있따. 서블릿들의 진행과정 단계가 비슷하다