

Wykład EL i JSTL

JAVA II

Dr inż. Damian Raczyński

EL

Expression Language (EL – język wyrażeń) – pozwala na znaczne uproszczenie dostępu do różnego rodzaju informacji (z zapytania, odpowiedzi, sesji...)

Wyrażenie musi rozpoczynać się od \$, po czym umieszczane są nawiasy klamrowe przechowujące kod wyrażenia

`${ KOD WYRAŻENIA }`

EL pozwala na:

- Wyświetlanie informacji z obiektów przechowywanych w kontekście strony, zapytania, sesji lub aplikacji
- Wykonywanie prostych operacji arytmetycznych,
- Wykonywanie porównań.

JSP EL pozwala w znaczny sposób uprościć kod JSP i ograniczyć wykorzystywanie kodu Java.

EL

Elementami możliwymi do wykorzystania w EL są:

- `pageScope`, `requestScope`, `sessionScope`, `applicationScope` - zbiory atrybutów o zasięgu strony, zapytania, sesji i kontekstu
- `cookie` - mapy ciasteczek,
- `header` - mapa nagłówków zapytania,
- `param` - parametry zapytania,
- `paramValues` - parametry zapytania, zwracające tablice łańcuchów a nie pojedyncze łańcuchy,
- `initParam` - parametry kontekstu.

Wszystkie elementy operują na mapach (tablica indeksowana obiektami)

EL

Możliwe jest posługiwanie się wyłącznie nazwami atrybutów.

W przypadku, gdy nazwy atrybutów pokrywają się (np. dla sesji i dla kontekstu występują takie same nazwy), wybierany jest element o większym zasięgu.

fragment kodu:

```
<%
pageContext.setAttribute("param1", "Strona", pageContext.PAGE_SCOPE);
pageContext.setAttribute("param1", "Sesja", pageContext.REQUEST_SCOPE);
%>
${pageScope.param1}<br/>
${requestScope.param1}<br/>
${param1}<br/>
```

JSP Page
Strona
Sesja
Strona

EL

Przykłady realizacji odwołań EL:

```
${header.host}
${initParam.wersja}
```

W przypadku, gdy nazwa elementu zawiera w sobie znaki specjalne, poprawne odwołanie EL powinno wyglądać:

```
${header[„Content-Type"]}
```

`${applicationScope[„ksiazka”][„rozdzial1"]}` – zostanie pobrany atrybut o nazwie ksiazka, po czym zostanie on potraktowany jako mapa – wyświetlony zostanie atrybut o nazwie rozdzial1

5

EL

fragment kodu:

```
<%
    int[] tab1= new int [3];
    tab1[0]=0; tab1[1]=1; tab1[2]=2;
    int[] tab2= new int [3];
    tab2[0]=10; tab2[1]=11; tab2[2]=12;
    Map<String, int[]> m = new HashMap<>();
    m.put("jeden", tab1);
    m.put("dwa", tab2);
    pageContext.setAttribute("m", m);
%>
${pageScope["m"]["jeden"][2]}
${pageScope["m"]["dwa"][1]}
```

EL

Notacja z nawiasem kwadratowym umożliwia użycie zmiennych – przykładowo:

```
${initParam[zmienna]}
```

Przykład:

Wpisy dodane do web.xml

```
<context-param>
    <param-name>parametr</param-name>
    <param-value>Java2</param-value>
</context-param>
</web-app>
```

EL

Strona jsp: Co zostanie wyświetlone po podaniu adresu: /WebApp.../?p=parametr ?

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>${initParam[param.p]}</h1>
    </body>
</html>
```

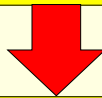
Java2

8

Ziarna + EL

Servlet:

```
ProduktBean x = new ProduktBean();
request.setAttribute („product“, x);
getServletConfig().getServletContext().getRequestDispatcher („/pr.jsp“)
.forward(request,response);
```



JSP z EL:

```
<p> Produkt id: ${product.id} </p>
```

Ziarna + EL

Zapis:

`${sessionScope.zalogowany}` – wyświetla atrybut o nazwie zalogowany, który znajduje się w sesji.

Zapis:

`${sessionScope.zalogowany.imie}` (alternatywnie `sessionScope.zalogowany[„imie“]`) – kontener spróbuje zlokalizować element imie w obiekcie zalogowany – NIE ZNAJĄC JEGO KLASY.

Po znalezieniu słowa specjalnego (zasięg sesji), serwer szuka atrybutu zalogowany. Po odnalezieniu go, serwer dostrzega, że wyrażenie wymaga znalezienia atrybutu dla obiektu zalogowany (imie). Kolejny serwer próbuje znaleźć metodę `getImie` (wyjątek, gdy nie znajdzie).

10

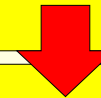
Ziarna + EL

Produkt.java:

```
public class Produkt {
    public Produkt(){}
    private String nazwa;
    private float cena;
    public String getNazwa() {
        return nazwa;
    }
    public void setNazwa(String nazwa) {
        this.nazwa = nazwa;
    }
    public float getCena() {
        return cena;
    }
    public void setCena(float cena) {
        this.cena = cena;
    }
}
```

index.jsp:

```
<form action="second.jsp" method="post">
    <input name="nazwa" type="text" placeholder="Podaj nazwe"/>
    <br/>
    <input name="cena" type="text" placeholder="Podaj cene"/>
    <input type="submit" value="OK"/>
</form>
```



second.jsp:

```
<body>
    <h2> DRUGA STRONA </h2>
    <jsp:useBean id="x"
                scope="application"
                class="pwsz.Produkt">
    </jsp:useBean>
    <jsp:setProperty name="x"
                    property="*" />

    ${x.cena}<br/>
    ${x["nazwa"]} <br/>
</body>
```

EL



W przypadku zastosowania notacji z nawiasami kwadratowymi mamy możliwość korzystania z dwóch dodatkowych elementów: list i tablic.

Tablice nie mogą zmieniać typu przechowywanych danych, natomiast listy mogą.

Indeksami tych elementów są liczby naturalne. Można je jednak podawać na dwa sposoby:

```
${sessionScope.elem[0].elem2}  
lub  
${sessionScope.elem[„0“].elem2}
```

13

EL – operatory arytmetyczne



Język EL umożliwia zastosowanie 5 operacji:

+, **-**, *****, **div**, **/**, **%**, **mod**



```
${8+2}           //10  
${8-2}           //6  
${3.5 div 2}     //1.75  
${2 mod 8}       //2
```

Wartość: `${Element.pole * Element2.pole}`

Operator trójargumentowy:

A?B:C

14

EL – operatory relacji



Język EL umożliwia zastosowanie następujących operatorów porównywania:

```
< lub lt,  
> lub gt,  
<= lub le,  
>= lub ge,  
== eq,  
!= ne.
```



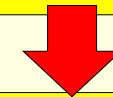
Elementy staną się użyteczne po wprowadzeniu JSTL np.
`<c:if test='${bean1.prop1<bean2.prop2}'>`

Język EL umożliwia zastosowanie następujących operatorów logicznych:
&& lub **and** (iloczyn logiczny),
|| lub **or** (suma logiczna),
! lub **not** (zaprzeczenie),

15

index.jsp:

```
<form action="jsp2.jsp" method="GET">  
  <h1>Wybierz opcje:</h1>  
  DODAJ<input type="radio" name="opcja" value="1"/>  
  ODEJMIJ<input type="radio" name="opcja" value="2"/><br/>  
  <h1>Wprowadź dane</h1>  
  op1:<input type="text" name="op1"/>  
  op2:<input type="text" name="op2"/><br/>  
  <input type="submit" value="OK"/>  
</form>
```



jsp2.jsp:

```
<h1> Wybrana operacja:  
${ (param["opcja"]=="1")?"dodawanie":"odejmowanie"}</h1>  
<h2> Wynik:  
${ (param.opcja eq 1) ? (param.op1+param.op2) : (param.op1-param.op2) }</h2>
```

16

index.jsp:

< http://localhost:8084/WebApplication4/

JSP Page

Wybierz opcje:

< DODAJ ☐ ODEJMIJ ☐

Wprowadź dane

op1: op2:

EL



Wartość null i kolekcje puste:

Aby sprawdzić, czy kolekcja jest pusta można użyć instrukcji empty, przykład:

```
${empty sessionScope}
```

Wyrażenie zwróci true, jeżeli sesja nie zawiera żadnych atrybutów.

- empty zwróci true, jeśli:
 - wyrażenie zwraca null,
 - wyrażenie jest pustym łańcuchem,
 - -||- pustą mapą,
 - -||- kolekcją (zbiorem, tablicą, listą).

18

index.jsp:

```
<%
    int [] tab= null;
    pageContext.setAttribute("tab", tab);
    String tekst="";
    pageContext.setAttribute("tekst", tekst);
    String tekst2=null;
    pageContext.setAttribute("tekst2", tekst2);
    request.setAttribute("atr1", null);
    int[] tab2= new int[1];
    tab2[0]=1;
    pageContext.setAttribute("tab2", tab2);
    String tekst3="tekst";
    pageContext.setAttribute("tekst3", tekst3);
    request.setAttribute("atr2", "atr2");
%>
tablica: ${ (empty tab)?"pusta":tab[0]}<br/>
tablica2: ${ (empty tab2)?"pusta":tab2[0]}<br/>
tekst ${ (empty tekst)?"pusty":tekst}<br/>
tekst2 ${ (empty tekst2)?"pusty":tekst2}<br/>
tekst3 ${ (empty tekst3)?"pusty":tekst3}<br/>
atrybut1 ${ (empty atr1)?"pusty":atr1}<br/>
atrybut2 ${ (empty atr2)?"pusty":atr2}<br/>
nieistniejacy ${ (empty nieist)?"pusty":nieist}<br/>
```

index.jsp:

```
<%
    i
    P
    S
    P
    S
    P
    r
    i
    t
    P
    S
    P
    r
%>
tablica: pusta
tablica2: 1
tekst pusty
tekst2 pusty
tekst3 tekst
atrybut1 pusty
atrybut2 atr2
nieistniejacy pusty
nieistniejacy ${ (empty nieist)?"pusty":nieist}<br/>
```

EL



W celu porównania z wartością null należy:

```
${sessionScope.koszyk == null}
```

21

JSTL



JSTL jest rozszerzeniem funkcji JSP. Zawiera dodatkowy zbiór funkcjonalności importowanych do strony JSP za pomocą dyrektywy `@taglib`.

Elementy wchodzące w skład JSTL:

- `c` (ang. core) – szereg istotnych funkcji omówionych w dalszej części,
- `fmt` (ang. format) – akcje pozwalające formatować daty, liczby, waluty,
- `sql` (ang. Structured Query Language) – podstawowe rodzaje zapytań w SQL,
- `xml` (ang. eXtensible Markup Language) – pozwala na przetwarzanie danych xml
- `fn` (ang. functions) – funkcje operujące na łańcuchach znaków i kolekcjach.

22

JSTL



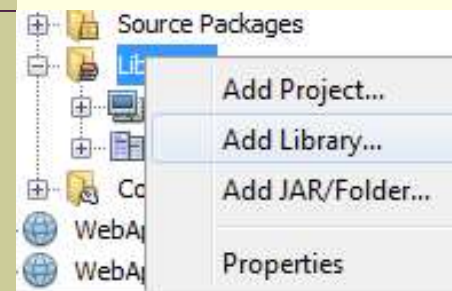
Dostęp do JSTL wymaga podania dyrektywy taglib:

```
<%@ taglib uri=„...” prefix=„...” %>
```

Tag	URI	Prefix
Core	http://java.sun.com/jsp/jstl/core	c
XML	http://java.sun.com/jsp/jstl/xml	xml
I18N	http://java.sun.com/jsp/jstl/fmt	fmt
Database	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

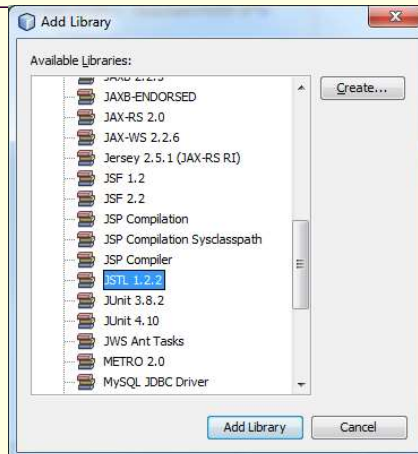
23

JSTL



24

JSTL



25

JSTL

```
<@page contentType="text/html" pageEncoding="UTF-8"%>
<@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
```

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

26

core

c:out – wyświetlanie wartości wyrażeń.

przykład:

```
<c:out value=„${uczen.numer}" default=„brak numeru" />
```

W przypadku, gdy wartość wyrażenia znajdującego się w atrybucie przyjmie wartość null, wtedy zostanie wyświetlony atrybut default.

Poprawna jest również składnia:

```
<c:out value=„${uczen.numer}">brak numeru </c:out>
```

27

core

Przykład w NetBeans

28

core

```
<@page contentType="text/html" pageEncoding="UTF-8"%>
<@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <c:out value="${param.number}">brak numeru </c:out>
  </body>
</html>
```

29

core

c:out – wyświetlanie wartości wyrażeń.

znacznik przybiera niezmiernie ważne znaczenie w przypadku zabezpieczenia stron internetowych przez atakami typu cross-site scripting.

```
<c:out value="${param.par}" escapeXml="true" />
```

przykład w NetBeans

30

core

```
<body>
  <p>
    ${param.par}
  </p>
</body>
```

localhost:8084/WebApplication21/?par=<h1> Zaatakowana strona Internetowa </h1>

Zaatakowana strona Internetowa

/?par=<h1> Zaatakowana strona Internetowa </h1>

```
<c:out value="${param.par}" escapeXml="true" />
```

core

c:set – znacznik ustawia wartości zarówno typów prymitywnych, jak i zmiennych (gdy zmienna nie istnieje, to zostaje utworzona).

parametry:

value – wartość dla elementu,

scope – zasięg (atrybut opcjonalny, w przypadku, gdy nie zostanie podany zostaną przeszukane wszystkie zasięgi – od strony → żądanie → sesja → kontekst)

var – nazwa elementu

przykład:

```
<c:set var="jezyk" scope="context" value="${param.jezyk}" />
```

32

core



c:set umożliwia również modyfikowanie wartości map

Przykład:

```
<c:set target=„${sessionScope.uczen}"
property=„pesel" value=„${param.pesel}" />
```

zmianie ulegnie właściwość pesel przechowywana w sesji obiektu uczen.

Alternatywna składnia:

```
<c:set target=„${sessionScope.uczen}"
property=„pesel"> ${param.pesel}</c:set>
```

33

core



c:set umożliwia również modyfikowanie wartości

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
  </head>
  <body>
    <c:set var="x" value="przykładowa wartość"/>
    <h1><c:out value="${x}"> ??????</c:out></h1>
  </body>
</html>
```

?????

core



c:set umożliwia również modyfikowanie wartości

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
  </head>
  <body>
    <c:set var="x" value="przykładowa wartość"/>
    <h1><c:out value="${x2}"> ??????</c:out></h1>
  </body>
</html>
```

przykładowa wartość

core



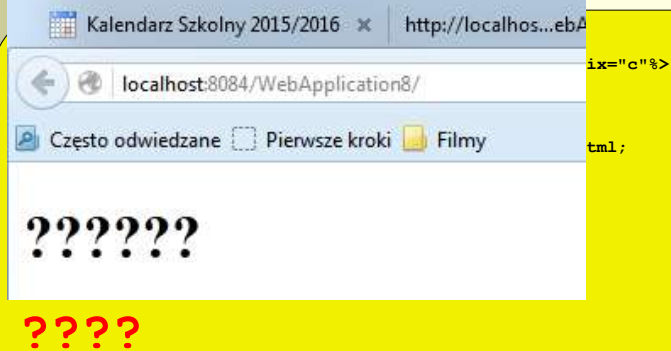
c:set umożliwia również modyfikowanie wartości

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
  </head>
  <body>
    <c:set var="x" value="przykładowa wartość"/>
    <h1><c:out value="${x2}"> ??????</c:out></h1>
  </body>
</html>
```

?????

core

c:set umożliwia również modyfikowanie wartości



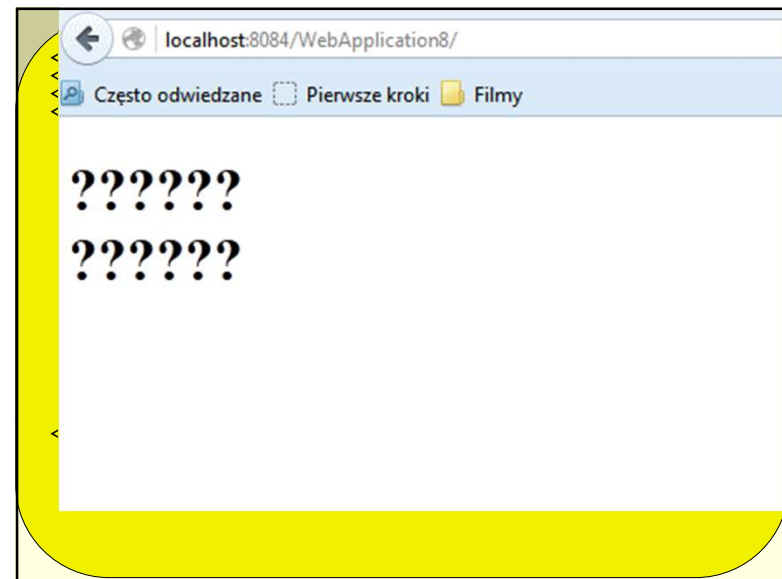
```
package pl.nysz.pwsz;

import java.io.Serializable;

public class ziarnoClass implements Serializable {
    private String imie;
    private String nazwisko;
    public void setImie(String imie){this.imie=imie;}
    public String getImie(){return imie;}
    public void setNazwisko(String
nazwisko){this.nazwisko=nazwisko;}
    public String getNazwisko(){return nazwisko;}
    public ziarnoClass ()
    {
        imie="X"; nazwisko="Y";
    }
}
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<jsp:useBean id="ziarno"
scope="page"
class="pl.nysz.pwsz.ziarnoClass">
</jsp:useBean>
<c:set target="${ziarno}" property="imie" value="${param.x}"/>
<c:set target="${ziarno}" property="nazwisko" value="${param.y}"/>
<h1><c:out value="${ziarno.imie}"> ??????</c:out><br/>
<c:out value="${ziarno.nazwisko}"> ??????</c:out>
</h1>

</body>
</html> ?????
```



```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<%
String imie="Jan";
String nazwisko="Kowalski";
%>
<c:set var="imie" value="${param.x}"/>
<c:set var="nazwisko" value="${param.y}"/>
<h1><c:out value="${imie}"> ??????</c:out><br/>
<c:out value="${nazwisko}"> ??????</c:out>
</h1>

</body>
</html>

```

????

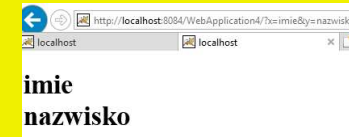
```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<%
String imie;
String nazwisko;
%>
<c:set var="imie" value="${param.x}"/>
<c:set var="nazwisko" value="${param.y}"/>
<h1><c:out value="${pageScope.imie}"> ??????</c:out><br/>
<c:out value="${pageScope.nazwisko}"> ??????</c:out>
</h1>

</body>
</html>

```

????



core

c:remove – służy do usuwania atrybutów

Przykład:

```
<c:remove var="uczen" scope="session" />
```

W przypadku pominięcia parametru scope, usunięciu ulegnie atrybut o jak najmniejszym zasięgu.

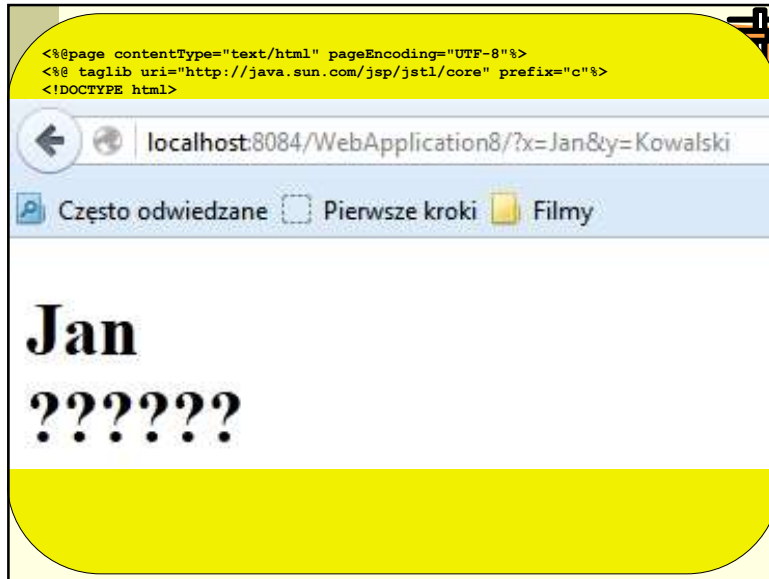
```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<c:set var="imie" value="${param.x}"/>
<c:set var="nazwisko" value="${param.y}"/>
<c:remove var="nazwisko" scope="page"/>
<h1><c:out value="${pageScope.imie}"> ??????</c:out><br/>
<c:out value="${pageScope.nazwisko}"> ??????</c:out>
</h1>

</body>
</html>

```

????



core

`c:if` – instrukcja warunkowa.

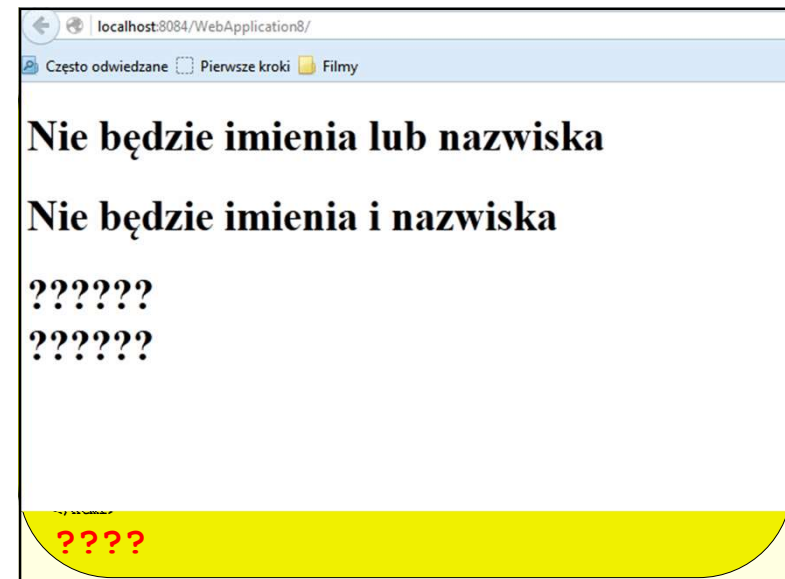
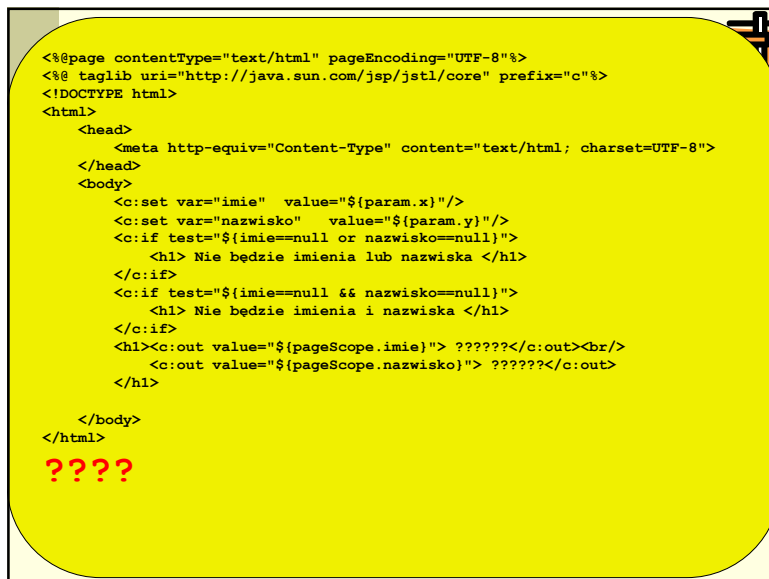
obowiązkowym parametrem jest test, który powinien zawierać wyrażenie EL zwracające `true/false`.

W przypadku, gdy wyrażenie EL zwróci `true`, wszystkie znaczniki wewnętrzne zostaną wykonane.

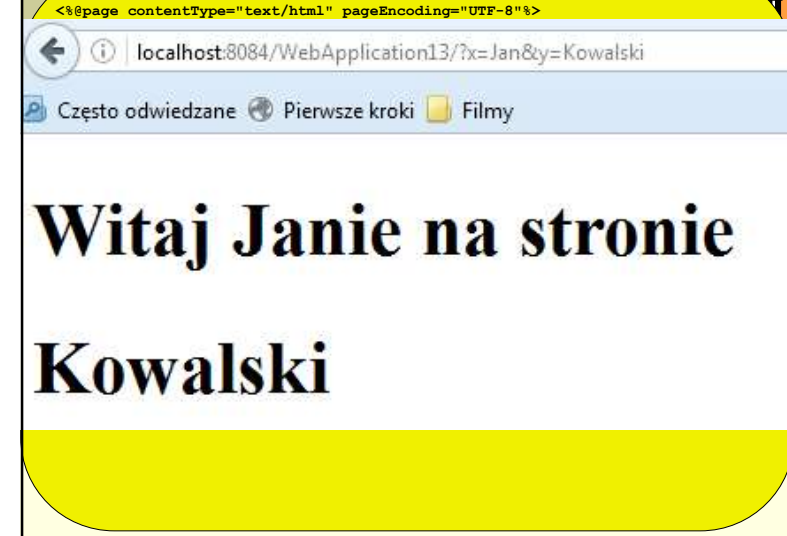
Znacznik `c:if` nie udostępnia bloku `else`.

```
<c:if test=" ${param.par == jan}">
<p> Witaj Janie na stronie Internetowej </p>
</c:if>
```

46



```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<c:set var="imie" value="${param.x}"/>
<c:set var="nazwisko" value="${param.y}"/>
<c:if test="${imie=="Jan"}">
<h1> Witaj Janie na stronie </h1>
</c:if>
<c:if test="${nazwisko=="Kowalski"}">
<h1>Kowalski </h1>
</c:if>
</body>
</html> ????
```



core

c:choose

akcja c:choose jest kompletnym odpowiednikiem instrukcji switch.

Przykład:

```
<c:choose>
  <c:when test="${sessionScope.param==null}">
    <p> parametr pusty </p>
  </c:when>
  <c:when test="${sessionScope.param==jeden}">
    <p> parametr jest równy jeden</p>
  </c:when>
</c:choose>
```

51

core

c:choose

W przypadku znalezienia prawidłowego testu dla któregośkolwiek ze znaczników <c:when ... >, pozostałe są pomijane.

Możliwe jest również zastosowanie znacznika <c:otherwise ... > pełniącego rolę bloku else.

```
<c:choose>
  <c:when test="${not (empty sessionScope.pesel)}">
    pesel jest równy ${sessionScope.pesel}
  </c:when>
  <c:otherwise>
    nie zdefiniowano pola pesel
  </c:otherwise> ...
```

52

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html><html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <c:set var="imie" value="{param.x}"/>
    <c:set var="nazwisko" value="{param.y}"/>
    <h1>
      <c:choose>
        <c:when test="{imie==null && nazwisko==null}">
          NIE MA NAZWISKA I IMIENIA
        </c:when>
        <c:when test="{imie==null}">
          NIE MA IMIENIA
        </c:when>
        <c:when test="{nazwisko==null}">
          NIE MA NAZWISKA
        </c:when>
        <c:otherwise>
          BĘDZIE I IMIĘ I NAZWISKO !
        </c:otherwise>
      </c:choose>
    </h1>
    <h1><c:out value="{pageScope.imie}"> ??????</c:out><br/>
    <c:out value="{pageScope.nazwisko}"> ??????</c:out>
    </h1>
  </body>

```

core

c:forEach - pętla

przykład (dla kolejnych wartości numerycznych):

```

<c:forEach var="x" begin="1" end="20" step="1">
  ${x}
</c:forEach>

```

Przejsię przez elementy struktury (array, string, collection):

```

<c:forEach var="wiadomosc" items="{messages}">
  <li>${wiadomosc}</li>
</c:forEach>

```

54

core

Porównanie (do bez JSTL):

Przejsię przez elementy struktury (array, string, collection):

```

<c:forEach var="wiadomosc" items="{messages}">
  <li>${wiadomosc}</li>
</c:forEach>

```

```

<% for (int x=0; x<messages.length; x++){
String wiadomosc=messages[i]; %>
  <li> <%= message %> </li>
<% } %>

```

55

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <ul>
      <c:forEach var="i" items="{header}">
        <li>${i.key}, ${i.value}</li>
      </c:forEach>
    </ul>
  </body>
</html>

```

?????

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
```

localhost:8084/WebApplication8/

Często odwiedzane Pierwsze kroki Filmy

- accept-language, pl,en-US;q=0.7,en;q=0.3
- cookie, csrftoken=8QYNode1WbNsTmfqilLrN5MayCP35yu3
- host, localhost:8084
- connection, keep-alive
- accept-encoding, gzip, deflate
- user-agent, Mozilla/5.0 (Windows NT 6.1; WOW64; rv:42.0) Gecko/20100101 Firefox/42.0
- accept, text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <table>
      <c:forEach var="i" begin="1" end="100" step="1">
        <tr>
          <c:forEach var="j" begin="1" end="100" step="1">
            <td>
              <c:out value="${i*j}"?></c:out>
            </td>
          </c:forEach>
        </tr>
      </c:forEach>
    </table>
  </body>
</html> ????
```

localhost:8084/WebApplication3/

Często odwiedzane Pierwsze kroki Filmy

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60	63	66	69	72	75
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	105	110	115	120	125
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120	126	132	138	144	150
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133	140	147	154	161	168	175
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160	168	176	184	192	200
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171	180	189	198	207	216	225
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200	210	220	230	240	250
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187	198	209	220	231	242	253	264	275
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240	252	264	276	288	300
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247	260	273	286	299	312	325
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238	252	266	280	294	308	322	336	350
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270	285	300	315	330	345	360	375
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320	336	352	368	384	400
17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289	306	323	340	357	374	391	408	425
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360	378	396	414	432	450
19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323	342	361	380	399	418	437	456	475
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400	420	440	460	480	500
21	42	63	84	105	126	147	168	189	210	231	252	273	294	315	336	357	378	399	420	441	462	483	504	525

core

<c:catch ...> - obsługa wyjątków

Akcja **c:catch** przechwytuje wyjątki związane z obiektami klas dziedziczących po **Throwable**, które powstają wewnątrz niej.

W atrubucie strony określonym parametrem **var** zostanie umieszczony obiekt wygenerowanego wyjątku.

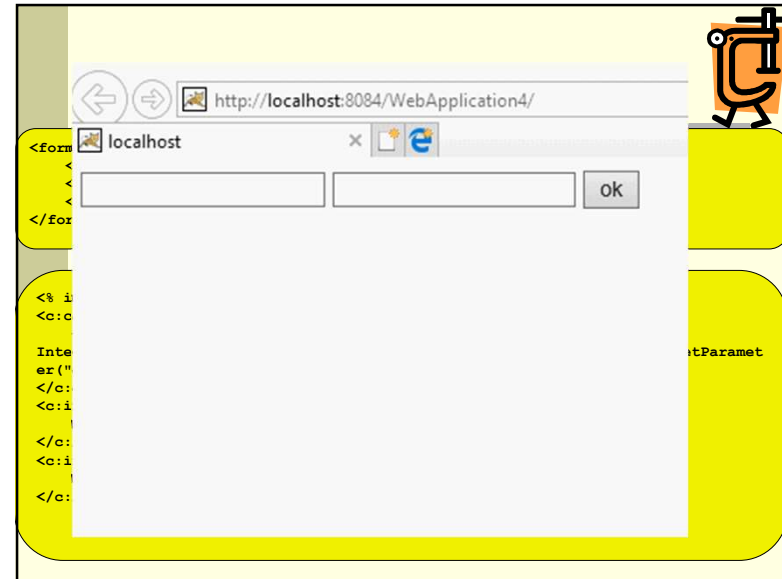
```
<c:catch var="wyjatek">
  ${10/"dzielenie przez tekst ?"}
</c:catch>
<c:if test=",$${not (empty wyjatek)}">
  Wygenerowano wyjatek ${wyjatek}
</c:if>
```

60

core

```
<form action="jsp2.jsp" method="get">
  <input type="text" name="op1"/>
  <input type="text" name="op2"/>
  <input type="submit" value="ok"/>
</form>
```

```
<% int wynik=0; %>
<c:catch var="wyjatek">
  <% wynik=
Integer.parseInt(request.getParameter("op1"))/Integer.parseInt(request.getParamet
er("op2")); %>
</c:catch>
<c:if test="${not empty wyjatek}">
  Wygenerowano wyjatek ${wyjatek.message}
</c:if>
<c:if test="${empty wyjatek}">
  Wynik= <%= wynik %>
</c:if>
```



core

c:url – akcja odpowiada za przetwarzanie podanych adresów URL.

Pozwala na podanie adresów bez ścieżki kontekstu (ścieżka automatycznie pobierana z kontekstu aplikacji)

```
<c:url value="/strony/strona.jsp" var="mainPage" />
<a href="<c:out value="${mainPage}" />"> Strona
glowna </a>
```

```
12 <body>
13
14 <a href="/WebApplication4/strony/strona.jsp"> Strona glowna </a>
15
```

core

c:url

Możliwe jest również określenie parametrów strony w „elegancki sposób”:

```
<c:url value="/strony/strona.jsp" var="mainPage" >
<c:param name="imie" value="jerzy" />
<c:param name="nazwisko"
value="${applicationScope.nazwisko}" />
</c:url>
<a href= '<c:out value="${mainPage}" />'> Strona
glowna </a>
```

```
13
14 <a href= '/WebApplication4/strony/strona.jsp?imie=jerzy&nazwisko='> Strona glowna </a>
15
```

64

core



`c:import` – akcja pozwala na umieszczanie na stronach JSP treści z zewnętrznych źródeł.

Przykład:

```
<c:import url=„strona.jsp” />
```

Akcja ta pozwala na importowanie elementów wchodzących w skład samej aplikacji web'owej, jak i innych elementów dostępnych za pomocą URL.

```
<c:import url=„http://other_server.com/plk.txt” />
```

Możliwe jest również przekazywanie parametrów za pomocą znaczników `c:param`

65

core



Za pomocą akcji `c:import` możliwe jest wczytanie treści zawartej pod wskazanym adresem do atrybutu:

```
<c:import url=„http://serv.com/tresc.txt” var=„tresc”  
scope=„page/session/...” />
```

Możliwe jest również wykorzystanie czytnika (obiekt klasy Reader):

```
<c:import url=„...” varReader=„czytnik”  
...  
</c:import>
```

66

SQL

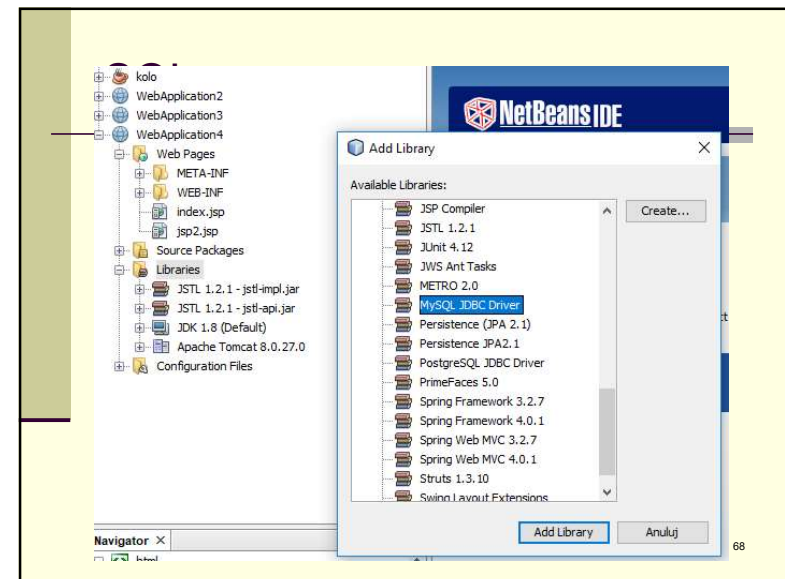
Biblioteka znaczników SQL umożliwia interakcję z relacyjnymi bazami danych (MS SQL SERVER, MySQL, Oracle).

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

Znaczniki

Znacznik	opis
<code>sql:setDataSource</code>	Tworzenie i ustawienia połączenia
<code>sql:query</code>	tworzenie zapytania
<code>sql:update</code>	aktualizacja bazy
<code>sql:param</code>	określenie parametru w wyrażeniu SQL
<code>sql:dateParam</code>	określenie parametru w postaci daty

67



68

```

PLIK index.jsp:
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</head>
<body>
<form
  action="${pageContext.request.contextPath}/index.jsp"
  method="POST">
  <table border="0">
    <tr>
      <td>imie:</td>
      <td><input type="text" name="imie" />
    </tr>
    <tr>
      <td>nazwisko:</td>
      <td><input type="text" name="nazwisko" />
    </tr>
  </table>
  <input type="submit" value="zapisz"/>
</form>
  . . .

```

```

PLIK index.jsp:
  . . .
  <br/><br/>
  <sql:setDataSource var="myDS" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/jee" user="root" password="" />
  <c:choose>
  <c:when test="${not empty param.imie && not empty param.nazwisko}">
    <sql:update dataSource="${myDS}" var="nowy">
      INSERT INTO jstl (imie,nazwisko) VALUES (?, ?)
      <sql:param value="${param.imie}" />
      <sql:param value="${param.nazwisko}" />
    </sql:update>
  </c:when>
  <c:otherwise>
    <font color="#cc0000">Wprowadź dane!</font>
  </c:otherwise>
  </c:choose>

  <br/><br/>
  <sql:query dataSource="${myDS}" var="tabela">
    SELECT * from jstl;
  </sql:query>
  <table border="1">
    <c:forEach var="row" items="${tabela.rows}">
      <tr>
        <td><c:out value="${row.imie}" /></td>
        <td><c:out value="${row.nazwisko}" /></td>
      </tr>
    </c:forEach>
  </table>
</body>

```

PLIK

http://localhost:8084/Wet localhost

imie:

nazwisko:

Wprowadź dane!

</body>

fmt

Odpowiada za formatowanie

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<c:set var="liczba" value="123456789.987654321"/>
domyślnie :
<fmt:formatNumber value="${liczba}" type="currency"/>
<br/>
a teraz jak? :
<fmt:formatNumber type="number" maxFractionDigits="3"
value="${liczba}"/>
<br/>
a teraz jak?1 :
<fmt:formatNumber type="number" maxIntegerDigits="3"
value="${liczba}"/>
<br/>
a teraz jak?2 :
<fmt:formatNumber type="number" groupingUsed="false"
value="${liczba}"/>
<br/>
a teraz jak?3 :
<fmt:formatNumber type="number" pattern="###.###E0"
value="${liczba}"/>
</body>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<html>
<body>
```

domyślnie : 123 456 789,99
a teraz jak? : 123 456 789,988
a teraz jak?1 : 789,988
a teraz jak?2 : 123456789,988
a teraz jak?3 : 123,457E6

```
a teraz jak?3 :
<fmt:formatNumber type="number" pattern="###.###E0"
value="${liczba}"/>
</body>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<c:set var="data" value="<%=new java.util.Date()%>"/>
<br/>
typ 1:
<fmt:formatDate type="time" value="${data}"/>
<br/>
typ 2:
<fmt:formatDate type="date" value="${data}"/>
<br/>
typ 3:
<fmt:formatDate type="both" value="${data}"/>
<br/>
</body>
</html>
```

localhost:8084/WebApplication8/

Często odwiedzane ☐ Pierwsze kroki

typ 1: 21:20:08
typ 2: 2015-11-22
typ 3: 2015-11-22 21:20:08

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<!DOCTYPE html>
<fmt:setLocale value="us_US"/>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<c:set var="data" value="<%=new java.util.Date()%>"/>
<br/>
typ 1:
<fmt:formatDate type="time" value="${data}"/>
<br/>
typ 2:
<fmt:formatDate type="date" value="${data}"/>
<br/>
typ 3:
<fmt:formatDate type="both" value="${data}"/>
<br/>
</body>
</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<!DOCTYPE html>
<fmt:setLocale value="us_US"/>
```

localhost:8084/WebApplication8/

Często odwiedzane ☐ Pierwsze kroki ☐ Filmy

typ 1: Sun Nov 22 21:22:26 CET 2015
 typ 2: Sun Nov 22 21:22:26 CET 2015
 typ 3: Sun Nov 22 21:22:26 CET 2015

XML

<x:parse> - element umożliwiający parsowanie plików xml
 Wybrane atrybuty:

Atrybut	opis
var	zmienna, która będzie przechowywała wynik parsowania
xml	zmienna/obiekt reader'a, który przechowuje/przetwarza dane xml
filter	filtr stosowany do dokumentu oryginalnego
doc	dokument xml do parsowania

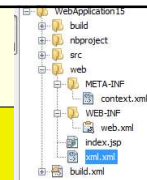
<x:out> - element umożliwiający wyświetlanie wybranych wyników parsowania.

attribut select umożliwia określenie ścieżki (xpath) określającej konkretną wartość/wartości - przykład:
 \$o/korzen/grupa/element[n]/attribut
 wyświetla atrybut n'tego elementu znajdującego się w znaczniku grupa, który z kolei jest umieszczony w znaczniku korzen

xml

Plik xml.xml:

```
<studenci>
  <student>
    <imie>Jan</imie>
    <nazwisko>Kowalski</nazwisko>
    <rok>2</rok>
  </student>
  <student>
    <imie>Anna</imie>
    <nazwisko>Nowak</nazwisko>
    <rok>1</rok>
  </student>
  <student>
    <imie>Sebastian</imie>
    <nazwisko>Nowak</nazwisko>
    <rok>2</rok>
  </student>
</studenci>
```



xml

Plik index.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<body>
  <c:import var="studenci"
url="http://localhost:8084/WebApplication15/xml.xml"/>
  <x:parse xml="${studenci}" var="o"/>
  <x:out select="$o/studenci/student[1]/imie" />
  <x:out select="$o/studenci/student[2]/nazwisko"/>
  <x:out select="$o/studenci/student[3]/rok"/>
</body>
</html>
```

xml

Plik index.jsp:

```
<x:out select="$o/studenci/student[3]/rok"/>
</body>
</html>
```

XML

<x:transform> - element stosuje transformację XSL do dokumentu XML

Wybrane atrybuty:

Atrybut	opis
doc	dokument xml do transformacji
xslt	plik zawierający zdefiniowane instrukcje transformacji
result	obiekt do przechowywania wyników transformacji
var	zmienna przechowująca wynik transformacji

83

```
PLIK style.xml:
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:output method="html" indent="yes"/>
  <xsl:template match="/">
    <html>
      <body>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="studenci">
    <table border="1" width="100%">
      <xsl:for-each select="student">
        <tr>
          <td>
            <xsl:value-of select="imie"/>
          </td>
          <td>
            <xsl:value-of select="nazwisko"/>
          </td>
          <td>
            <xsl:value-of select="rok"/>
          </td>
        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>
</xsl:stylesheet>
```

```

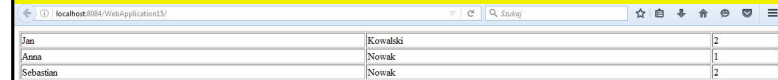
PLIK index.jsp:
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<body>
  <c:import var="studenci"
url="http://localhost:8084/WebApplication15/xml.xml"/>
  <c:import var="xslt"
url="http://localhost:8084/WebApplication15/style.xsl"/>
  <x:transform xml="${studenci}" xslt="${xslt}"/>
</body>
</html>

```

```

PLIK index.jsp:
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<body>
  <c:import var="studenci"
url="http://localhost:8084/WebApplication15/xml.xml"/>
  <c:import var="xslt"
url="http://localhost:8084/WebApplication15/style.xsl"/>
  <x:transform xml="${studenci}" xslt="${xslt}"/>
</body>
</html>

```



The screenshot shows a web browser window with the address bar displaying 'localhost:8084/WebApplication15/'. The browser content shows a table with three rows of student data. The table has three columns: Name, Surname, and Age.

Jan	Kowalski	2
Anna	Nowak	1
Sebastian	Nowak	2

Źródło strony:

```

<!DOCTYPE html>
<html>
<body>

  <html>
  <body>
  <table border="1" width="100%">
  <tr>
  <td>Jan</td><td>Kowalski</td><td>2</td>
  </tr>
  <tr>
  <td>Anna</td><td>Nowak</td><td>1</td>
  </tr>
  <tr>
  <td>Sebastian</td><td>Nowak</td><td>2</td>
  </tr>
  </table>
  </body>
</html>

```

```

PLIK style.xsl:
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:output method="html" indent="yes"/>
  <xsl:template match="/">
    <h1> Tabela studentow </h1>
    <xsl:apply-templates/>
    <hr/>
    <h2> Koniec tabeli studentow </h2>
  </xsl:template>
  <xsl:template match="studenci">
    <table border="1" width="100%">
      <xsl:for-each select="student">
        <tr>
          <td>
            <xsl:value-of select="imie"/>
          </td>
          <td>
            <xsl:value-of select="nazwisko"/>
          </td>
          <td>
            <xsl:value-of select="rok"/>
          </td>
        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>
</xsl:stylesheet>

```

PLIK style.xsl:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:output method="html" indent="yes"/>
  <xsl:template match="/">
    <h1> Tabela studentow </h1>
    <xsl:apply-templates/>
    <hr/>
    <h2> Koniec tabeli studentow </h2>
  </xsl:template>
</xsl:stylesheet>
```

Tabela studentow

Jan	Kowalski	2
Anna	Nowak	1
Sebastian	Nowak	2

Koniec tabeli studentow

```
<td>
  <xsl:value-of select="nazwisko"/>
</td>
<td>
  <xsl:value-of select="rok"/>
</td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

Źródło strony:

```
<!DOCTYPE html>
<html>
<body>

  <h1> Tabela studentow </h1><table border="1" width="100%">
<tr>
<td>Jan</td><td>Kowalski</td><td>2</td>
</tr>
<tr>
<td>Anna</td><td>Nowak</td><td>1</td>
</tr>
<tr>
<td>Sebastian</td><td>Nowak</td><td>2</td>
</tr>
</table><hr><h2> Koniec tabeli studentow </h2>

</body>
</html>
```

PLIK index.jsp - alternatywa:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<body>
  <c:set var="studenci">
    <studenci>
      <student>
        <imie>Jan</imie>
        <nazwisko>Kowalski</nazwisko>
        <rok>2</rok>
      </student>
      <student>
        <imie>Anna</imie>
        <nazwisko>Nowak</nazwisko>
        <rok>1</rok>
      </student>
      <student>
        <imie>Sebastian</imie>
        <nazwisko>Nowak</nazwisko>
        <rok>2</rok>
      </student>
    </studenci>
  </c:set>
  <c:import var="xslt"
url="http://localhost:8084/WebApplication15/style.xsl"/>
  <x:transform xml="{studenci}" xslt="{xslt}"/>
</body> </html>
```

PLIK index.jsp - alternatywa:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<body>
  <c:set var="studenci">
    <studenci>
      <student>
        <imie>Jan</imie>
      </student>
    </studenci>
  </c:set>
  <c:import var="xslt"
url="http://localhost:8084/WebApplication15/style.xsl"/>
  <x:transform xml="{studenci}" xslt="{xslt}"/>
</body> </html>
```

Tabela studentow

Jan	Kowalski	2
Anna	Nowak	1
Sebastian	Nowak	2

Koniec tabeli studentow

```
<imie>Sebastian</imie>
<nazwisko>Nowak</nazwisko>
<rok>2</rok>
</student>
</studenci>
</c:set>
<c:import var="xslt"
url="http://localhost:8084/WebApplication15/style.xsl"/>
<x:transform xml="{studenci}" xslt="{xslt}"/>
</body> </html>
```

fn



Przestrzeń nazw funkcji (fn) zawiera 16 elementów, z których większość jest związana z przetwarzaniem łańcuchów znaków.

Funkcja `length()` zwraca liczbę elementów kolekcji/łańcucha znaków.

Przykład:

```
...
liczba elementów: ${fn:length(sessionScope.zamowienia)}
```

Wewnątrz wyrażenia należy podać pełną nazwę funkcji (prefiks:nazwa), w nawiasach umieszczane są parametry funkcji

93

fn



`fn:indexOf` – indeks wystąpienia łańcucha znaków w innym

```
${(sessionScope.uczen.imie != null and
fn:indexOf(applicationScope.imieniny,
sessionScope.uczen.imie) gt -1)?"Wszystkiego
najlepszego":""}
```

Jeśli wewnątrz atrybutu kontekstu znajduje się łańcuch zawierający dzisiejsze imiona imieninowe, wyświetlone zostaną życzenia dla ucznia o imieniu znajdującym się w tym łańcuchu.

94

fn



`fn:join` – łączy elementy tablicy w jednolity tekst z wykorzystaniem separatorów.

```
Zalogowani: ${fn:join(sessionScope uzytkownicy, ", " )}
```

`fn:replace` – zamienia wystąpienia jednego ciągu znaków na inny

```
${fn:replace („jedenDwa”, „jeden”, „trzy”)}
```

wynikiem będzie trzyDwa

95

fn



`fn:substring` – zwraca fragment tekstu – pierwszym parametrem jest tekst, dwa kolejne określają indeks pierwszego i ostatniego znaku do pobrania

```
${fn:substring („PWSZ Nysa”, 0, 3)}
```

Wyświetlone zostanie PWSZ

`substringBefore()` – zwraca część łańcucha od początku do wystąpienia szukanego podłańcucha,

`substringAfter()` – zwraca część łańcucha od wystąpienia szukanego podłańcucha do końca.

96


```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <c:set var="tekst" scope="page" value="jeden dwa trzy cztery piec"/>
    <ul>
      <li><c:out value="${fn:length(pageScope.tekst)}"?</c:out></li>
      <li><c:out value="${fn:indexOf(tekst, \"trzy\")}"?</c:out></li>
      <li>${fn:replace(tekst, "cztery", "osiem")}</li>
      <li>${fn:substring(tekst, 10,21)}</li>
    </ul>
  </body>
</html>

```



```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

```

localhost:8084/WebApplication13/

Często odwiedzane Pierwsze kroki Filmy

- 26
- 10
- jeden dwa trzy osiem piec
- trzy cztery

inne

<c:redirect> - przekierowanie zapytania do innego URL:

```
<c:redirect url=„http://rolniknysa.pl” />
```