





Bộ môn Công nghệ Phần mềm
Viện CNTT & TT
Trường Đại học Bách Khoa Hà Nội

LẬP TRÌNH WEB HƯỚNG JAVA

Bài 14: JSTL


Giảng viên: ThS. Trịnh Tuấn Đạt
Bộ môn CNPM
Email: trinhthuandat.bk@gmail.com/dattt@soict.hut.edu.vn



Nội dung


- 1. JSTL là gì, tại sao cần JSTL?
- 2. Các chức năng trong JSTL
 - 2.1. Các thẻ cơ bản (Core tags)
 - 2.2. Các thẻ truy cập CSDL
 - Phụ lục 1: Giới thiệu nhanh về XPath
 - 2.3. Các thẻ XML
 - 2.4. Các thẻ EL (Expression Language) functions
 - Phụ lục 2: Giới thiệu nhanh về Internationalization
 - 2.5. Internationalization và các thẻ định dạng Text

DatTT-DSE-SOICT-HUST 2



1. JSTL là gì, tại sao cần JSTL?


DatTT-DSE-SOICT-HUST 3



JSTL là gì?

- Tập chuẩn các thư viện thẻ
- Đóng gói các chức năng chung trong rất nhiều ứng dụng JSP
 - Điều kiện và lặp
 - XML
 - Truy cập database
 - internationalized formatting
- Vẫn đang được tiếp tục phát triển


DatTT-DSE-SOICT-HUST 4



Tại sao cần JSTL?

- LTV không cần phải tự viết các thẻ
- LTV học và sử dụng tập chuẩn các thư viện thẻ được cung cấp bởi Java EE platforms
- Các nhà cung cấp đã tối ưu hóa việc cài đặt
- Đảm bảo tính Portability cho ứng dụng

DatTT-DSE-SOICT-HUST 5



Các thư viện thẻ JSTL

- Core (prefix: c)
 - Variable support, Flow control, URL management
- XML (prefix: x)
 - Core, Flow control, Transformation
- Internationalization (i18n) (prefix: fmt)
 - Locale, Message formatting, Number and date formatting
- Database (prefix: sql)
 - SQL query and update
- Functions (prefix: fn)
 - Collection length, String manipulation

DatTT-DSE-SOICT-HUST 6

Khai báo các thư viện thẻ JSTL

- Core
 - `<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`
- XML
 - `<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>`
- Internationalization (i18n)
 - `<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>`
- Database (SQL)
 - `<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>`
- Functions
 - `<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>`

DatTT-DSE-SOICT-HUST

7

2.1. Các thẻ cơ bản (core tags)

DatTT-DSE-SOICT-HUST

8

Core Tags Types (1)

- Hỗ trợ về biến (Variable)
 - `<c:set>`
 - `<c:remove>`
- Điều kiện
 - `<c:if>`
 - `<c:choose>`
 - `<c:when>`
 - `<c:otherwise>`
- Lặp
 - `<c:forEach>`
 - `<c:forTokens>`

DatTT-DSE-SOICT-HUST

9

Core Tags Types (2)

- Quản lý URL
 - `<c:import>`
 - `<c:param>`
 - `<c:redirect>`
 - `<c:param>`
 - `<c:url>`
 - `<c:param>`
- Mục đích chung khác
 - `<c:out>`
 - `<c:catch>`

DatTT-DSE-SOICT-HUST

10

Hỗ trợ về biến, `<c:set>`

- Thiết lập giá trị của một EL variable hoặc thuộc tính của một EL variable qua thuộc tính "var"
 - Chỉ định tầm vực biến qua thuộc tính "scope": page, request, session, application
- Nếu biến chưa tồn tại, sẽ được tạo và lưu lại (theo đúng tầm vực chỉ định)
- Biến có thể được thiết lập theo 2 cách khác nhau:
 - `<c:set var="foo" scope="session" value="..."/>`
 - Ví dụ: `<c:set var="bookId" value="{param.BookID}"/>`
 - `<c:set var="foo">value to be set</c:set>`

DatTT-DSE-SOICT-HUST

11

Ví dụ: `<c:set>`

```
<c:set var="customerTable" scope="application">
<table border="1">
  <c:forEach var="customer" items="{customers}">
    <tr>
      <td>${customer.lastName}</td>
      <td><c:out value="{customer.address}" default="no address specified"/></td>
      <td>
        <c:out value="{customer.address}">
          <font color="red">no address specified</font>
        </c:out>
      </td>
    </tr>
  </c:forEach>
</table>
</c:set>
```

The content between `<c:set>` and `</c:set>` is saved as `$(customerTable)`.

DatTT-DSE-SOICT-HUST

12

Ví dụ: sử dụng lại biến "customerTable" đã định nghĩa ở trên

<h4>Using "customerTable" application scope attribute defined in Set.jsp a first time</h4>

```
<c:out value="${customerTable}" escapeXml="false"/>
```

<h4>Using "customerTable" application scope attribute defined in Set.jsp a second time</h4>

```
<c:out value="${customerTable}" escapeXml="false" />
```

DatTT-DSE-SOICT-HUST

13

Ví dụ: sử dụng lại biến "customerTable" đã định nghĩa ở trên



DatTT-DSE-SOICT-HUST

14

Hỗ trợ về biến <c:remove>

- Bỏ một EL variable
 - <c:remove var="cart" scope="session"/>

DatTT-DSE-SOICT-HUST

15

Thẻ điều kiện

- Các thẻ điều khiển luồng (điều kiện, lặp) giúp loại bỏ các scriptlets
- <c:if test="...">
 - Thân thẻ sẽ được thực thi tùy theo giá trị của thuộc tính test
- <c:choose>
 - Nhúng thêm các thẻ con <c:when> và <c:otherwise>
 - Làm việc như if-then-else

DatTT-DSE-SOICT-HUST

16

Ví dụ: <c:if test="...">

```
<c:forEach var="customer" items="${customers}">
  <c:if test="${customer.address.country == 'USA'}">
    ${customer}<br>
  </c:if>
</c:forEach>
```

- Chỉ những customers có giá trị thuộc tính "address.country" là "USA" mới được hiển thị qua vòng lặp <c:forEach>

DatTT-DSE-SOICT-HUST

17

Ví dụ: <c:choose>, <c:when>

```
<c:forEach var="customer" items="${customers}">
  <c:choose>
    <c:when test="${customer.address.country == 'USA'}">
      <font color="blue">
    </c:when>
    <c:when test="${customer.address.country == 'Canada'}">
      <font color="red">
    </c:when>
    <c:otherwise>
      <font color="green">
    </c:otherwise>
  </c:choose>
  ${customer}</font><br>
</c:forEach>
```

DatTT-DSE-SOICT-HUST

18

Thẻ lặp: <c:forEach>

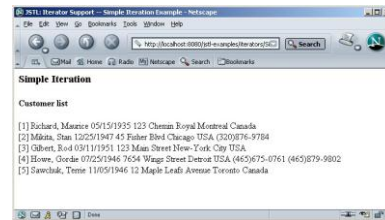
- Cho phép lặp trên 1 tập các objects (collection)
 - items: đại diện cho tập các đối tượng
 - var: item hiện tại (đối tượng đang được xét)
 - varStatus: trạng thái lặp
 - begin, end, step: phạm vi và bước lặp
- Các loại Collection
 - java.util.Collection
 - java.util.Map
 - Giá trị của thuộc tính var phải có kiểu là java.util.Map.Entry

DatTT-DSE-SOICT-HUST

19

Ví dụ: <c:forEach>

```
<c:forEach var="customer" items="${customers}">
  ${customer}<br>
</c:forEach>
```

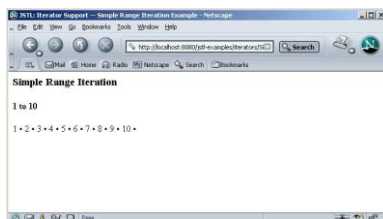


DatTT-DSE-SOICT-HUST

20

Ví dụ: <c:forEach> Phạm vi lặp

```
<c:forEach var="i" begin="1" end="10">
  ${i} •
</c:forEach>
```



DatTT-DSE-SOICT-HUST

21

Ví dụ: <c:forEach>, các kiểu dữ liệu

```
<c:forEach var="i" items="${intArray}">
  <c:out value="${i}"/> •
</c:forEach>

<c:forEach var="string" items="${stringArray}">
  <c:out value="${string}"/> <br>
</c:forEach>

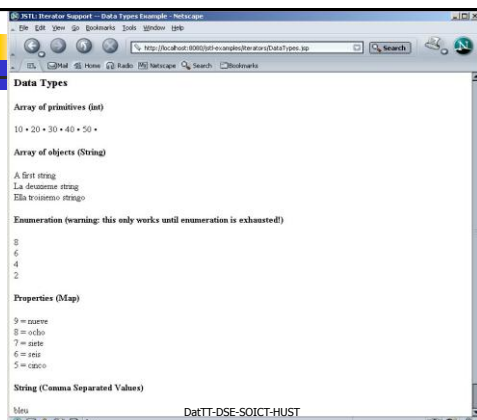
<c:forEach var="item" items="${enumeration}" begin="2" end="10" step="2">
  <c:out value="${item}"/> <br>
</c:forEach>

<c:forEach var="prop" items="${numberMap}" begin="1" end="5">
  <c:out value="${prop.key}"/> = <c:out value="${prop.value}"/> <br>
</c:forEach>

<c:forEach var="token" items="bleu,blanc,rouge">
  <c:out value="${token}"/> <br>
</c:forEach>
```

DatTT-DSE-SOICT-HUST

22



DatTT-DSE-SOICT-HUST

23

Ví dụ: <c:forEach>, trạng thái lặp

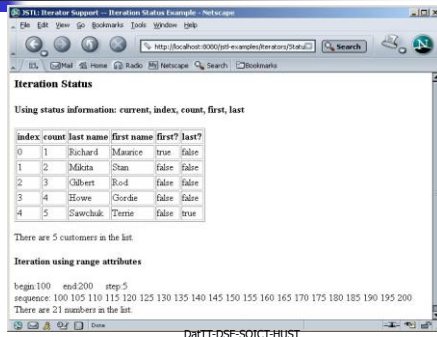
```
<c:forEach var="customer" items="${customers}" varStatus="status">
  <tr>
    <td><c:out value="${status.index}"/></td>
    <td><c:out value="${status.count}"/></td>
    <td><c:out value="${status.current.lastName}"/></td>
    <td><c:out value="${status.current.firstName}"/></td>
    <td><c:out value="${status.first}"/></td>
    <td><c:out value="${status.last}"/></td>
  </tr> ...
</c:forEach>

<c:forEach var="i" begin="100" end="200" step="5" varStatus="status">
  <c:if test="${status.first}">
    begin: <c:out value="${status.begin}"/> <br>
    end: <c:out value="${status.end}"/> <br>
    step: <c:out value="${status.step}"/> <br>
    sequence:
  </c:if> ...
</c:forEach>
```

DatTT-DSE-SOICT-HUST

24

Ví dụ: <c:forEach>, trạng thái lặp



Iteration Status

Using status information: current, index, count, first, last

index	count	last	name	first name	last?
0	1	Richard	Monette	true	false
1	2	Mikita	Stan	false	false
2	3	Gilbert	Rod	false	false
3	4	Howe	Gordie	false	false
4	5	Sawchuk	Terrie	false	true

There are 5 customers in the list.

Iteration using range attributes

begin: 100 end: 200 step: 5
 sequence: 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200
 There are 21 numbers in the list.

25

Ví dụ: <c:forEach>

```
<c:forEach var="token" items="one,two,three" delims=",">
<c:out value="{token}" />
</c:forEach>
```

DatTT-DSE-SOICT-HUST

26

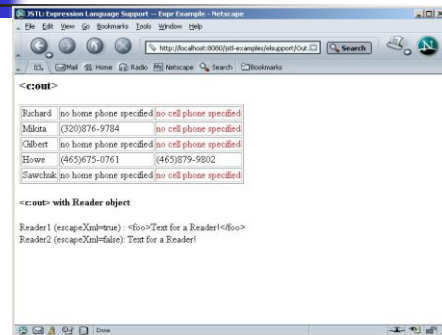
Ví dụ: <c:out>

```
<table border="1">
<c:forEach var="customer" items="{customers}">
<tr>
<td><c:out value="{customer.lastName}" /></td>
<td><c:out value="{customer.phoneHome}" default="no home
phone specified" /></td>
<td>
<c:out value="{customer.phoneCell}" escapeXml="false">
<font color="red">no cell phone specified</font>
</c:out>
</td>
</tr>
</c:forEach>
</table>
```

DatTT-DSE-SOICT-HUST

27

Ví dụ: <c:out>



<c:out>

Richard	no home phone specified	no cell phone specified
Mikita	(320)876-9784	no cell phone specified
Gilbert	no home phone specified	no cell phone specified
Howe	(465)675-0761	(465)879-9802
Sawchuk	no home phone specified	no cell phone specified

<c:out> with Reader object

Reader1 (escapeXml=true) <foo>Text for a Reader!</foo>
 Reader2 (escapeXml=false) Text for a Reader!

DatTT-DSE-SOICT-HUST

28

URL Import: <c:import>

- Cách thức tổng quát hơn để truy cập các resources theo URL (hơn là <jsp:include>)
 - URL tuyệt đối: truy cập các resources nằm ngoài ứng dụng Web
 - URL tương đối: truy cập các tài nguyên trong cùng ứng dụng Web
- Hiệu quả hơn (so với <jsp:include>)
 - Không cần buffering
- Thẻ <c:param> có thể được sử dụng để chỉ định các parameters (như thẻ <jsp:param>)

DatTT-DSE-SOICT-HUST

29

Ví dụ: <c:import> URL tuyệt đối

```
<blockquote>
<ex:escapeHtml>
<c:import url="http://www.cnn.com/cnn.rss"/>
</ex:escapeHtml>
</blockquote>
```

DatTT-DSE-SOICT-HUST

30

Ví dụ: <c:import> & <c:param>

```
<c:import url="header.jsp">
  <c:param name="pageTitle" value="newInstance.com"/>
  <c:param name="pageSlogan" value=" " />
</c:import>
```

DatTT-DSE-SOICT-HUST

31

URL Rewriting: <c:url>

- Được sử dụng trong URL rewriting
 - Tất cả các URL được trả về từ một trang JSP (cho trình duyệt) đều có kèm session ID nếu Cookie bị tắt
 - Có thể thêm các thẻ con để đính kèm các tham số vào trong URL trả về

DatTT-DSE-SOICT-HUST

32

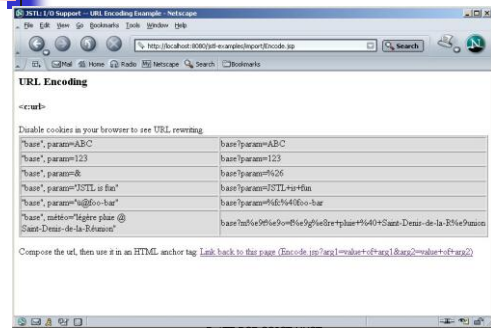
Ví dụ: <c:url>

```
<table border="1" bgcolor="#d4d4d4">
  <tr>
    <td>"base", param=ABC</td>
  </tr>
  <tr>
    <td><c:url value="base">
      <c:param name="param" value="ABC"/>
    </c:url>
  </td>
  </tr>
  <tr>
    <td>"base", param=123</td>
  </tr>
  <tr>
    <td><c:url value="base">
      <c:param name="param" value="123"/>
    </c:url>
  </td>
  </tr>
</table>
```

DatTT-DSE-SOICT-HUST

33

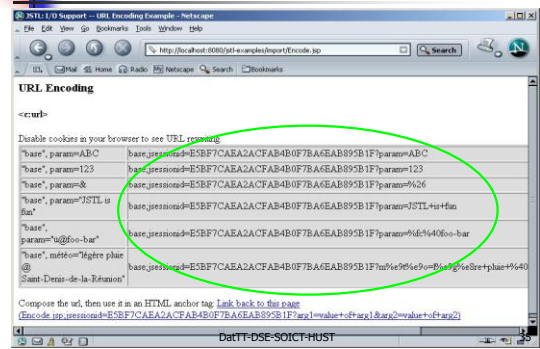
Ví dụ: <c:url> - Có Cookie



DatTT-DSE-SOICT-HUST

34

Ví dụ: <c:url> - Tắt Cookie



DatTT-DSE-SOICT-HUST

Điều hướng: <c:redirect>

- Gửi lại điều hướng HTTP (HTTP redirect) cho client
- Có thể thêm các thẻ con <c:param> để đính kèm các tham số vào trong URL trả về

DatTT-DSE-SOICT-HUST

36

<c:out>

- Tính giá trị biểu thức và đưa kết quả vào đối tượng JspWriter để hiển thị
- Nếu kết quả tính được là một đối tượng java.io.Reader, dữ liệu trước tiên sẽ được đọc từ đối tượng Reader, sau đó được ghi vào đối tượng JspWriter
 - Tăng hiệu năng
- Cú pháp
 - `<c:out value="value" [escapeXml="{true|false}"] [default="defaultValue"] />`
 - Nếu escapeXml là true, các ký tự đặc biệt được chuyển đổi (Ví dụ: `< -> <`; `& -> &`; ...)

37

Ví dụ: <c:out>

```
<table border="1">
<c:forEach var="customer" items="{customers}">
  <tr>
    <td><c:out value="{customer.lastName}" /></td>
    <td><c:out value="{customer.phoneHome}" default="no home
phone specified" /></td>
    <td>
      <c:out value="{customer.phoneCell}" escapeXml="false">
        <font color="red">no cell phone specified</font>
      </c:out>
    </td>
  </tr>
</c:forEach>
</table>
```

DatTT-DSE-SOICT-HUST

38

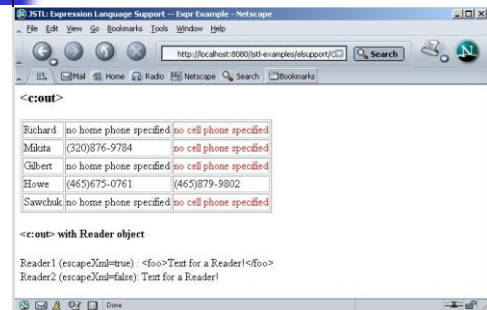
Ví dụ: <c:out>

```
<h4><c:out> with Reader object</h4>
<%
java.io.Reader reader1 = new java.io.StringReader("<foo>Text for a
Reader!</foo>");
pageContext.setAttribute("myReader1", reader1);
java.io.Reader reader2 = new java.io.StringReader("<foo>Text for a
Reader!</foo>");
pageContext.setAttribute("myReader2", reader2);
%>
Reader1 (escapeXml=true) : <c:out value="{myReader1}" /><br>
Reader2 (escapeXml=false): <c:out value="{myReader2}"
escapeXml="false" /><br>
```

DatTT-DSE-SOICT-HUST

39

Ví dụ: <c:out>



DatTT-DSE-SOICT-HUST

40

2.2. Các thẻ truy cập CSDL (SQL tags)

DatTT-DSE-SOICT-HUST

41

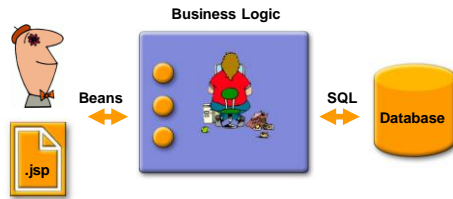
RAD/Prototyping/Simple Apps



DatTT-DSE-SOICT-HUST

42

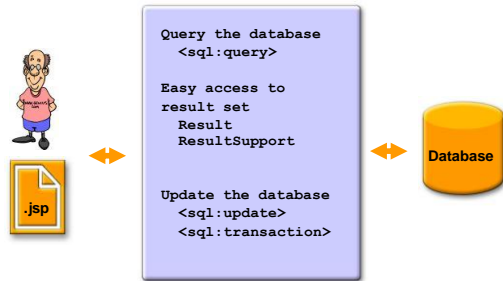
MVC Architecture



DatTT-DSE-SOICT-HUST

43

SQL Tags



DatTT-DSE-SOICT-HUST

44

DataSource

- Tất cả các thao tác với DB thực hiện trên một DataSource
- Các cách truy cập tới DataSource
 - Đối tượng cung cấp từ application logic
 - Đối tượng cung cấp từ <sql:dataSource>
- <sql:dataSource var="dataSource"


```
driver="org.gjt.mm.mysql.Driver"
url="jdbc:..."/>
<sql:query dataSource="${dataSource}" .../>
```

DatTT-DSE-SOICT-HUST

45

Ví dụ: <sql:setDataSource>

```
<sql:setDataSource
var="example"
driver="com.pointbase.jdbc.jdbcUniversalDriver"
url="jdbc:pointbase:server://localhost:1092/jstlsample;create=true"
/>
```

DatTT-DSE-SOICT-HUST

46

Ví dụ: <sql:transaction> & <sql:update>

```
<sql:transaction dataSource="${example}">
  <sql:update var="newTable">
    create table mytable (
      nameid int primary key,
      name varchar(80)
    )
  </sql:update>
  <sql:update var="updateCount">
    INSERT INTO mytable VALUES (1,'Paul Oakenfold')
  </sql:update>
  <sql:update var="updateCount">
    INSERT INTO mytable VALUES (2,'Timo Maas')
  </sql:update>
  ...
  <sql:query var="deejays">
    SELECT * FROM mytable
  </sql:query>
</sql:transaction>
```

DatTT-DSE-SOICT-HUST

47

SQL Query Execution using an iterator

Iterating over each Row of the result

```
1 Paul Oakenfold
2 Timo Maas
3 Paul van Dyk
```

Iterating over Columns without knowing the index

```
Name: 1 Value: Paul Oakenfold
Name: 2 Value: Timo Maas
Name: 3 Value: Paul van Dyk
```

Putting it all together

NAMEID	NAME
1	Paul Oakenfold
2	Timo Maas
3	Paul van Dyk

DatTT-DSE-SOICT-HUST

48

Phụ lục 1: Giới thiệu nhanh về XPath

DatTT-DSE-SOICT-HUST

49

Ví dụ 1 tài liệu XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<games>
  <country id="Luxembourg">
    <athlete>
      <name>Lux 1</name>
      <sport>swimming</sport>
      <age>23</age>
      <gender>M</gender>
    </athlete>
  </country>
  <country id="Denmark">
    <athlete>
      <name>Den 1</name>
      <sport>cycling</sport>
      <age>18</age>
      <gender>F</gender>
    </athlete>
    <athlete>
      <name>Den 2</name>
      <sport>sailing</sport>
      <age>27</age>
      <gender>M</gender>
    </athlete>
  </country>
</games>
```

DatTT-DSE-SOICT-HUST

50

Xpath là gì?

- XPath là một **Expression Language** để tham chiếu tới một phần cụ thể nào đó của tài liệu XML
- Biểu thức XPath sử dụng mô hình cây để biểu diễn một tài liệu XML
 - Ví dụ: Biểu thức XPath `/games/country/athlete` tương ứng với một tập các node, chứa tất cả các **athlete** của tất cả các **country** của **games** trong một tài liệu XML

DatTT-DSE-SOICT-HUST

51

4 kiểu dữ liệu trả về từ biểu thức XPath

- Node set
 - Kiểu phổ biến nhất, sẽ được tập trung trình bày ở phần sau
- Boolean
- Number
- String

DatTT-DSE-SOICT-HUST

52

Node Set, Location Path, Location Step, Predicate

- Một **node-set** là tập gồm 0 hoặc nhiều nodes trong tài liệu XML
- Một node-set là kết quả trả về của một **location path expression**
- Một **location path expression** bao gồm nhiều **location steps**
- Một **Location step** có thể có thêm **predicate**
 - `/games/country/athlete[sport="sailing"]` (predicate là `[sport="sailing"]`, tức là phần tử athlete phải có phần tử con **sport**, có giá trị là xâu "sailing")
 - `/games/country[@id="Denmark"]/athlete`

DatTT-DSE-SOICT-HUST

53

Các ví dụ về Node Set

- `/games/country/athlete`
 - Tất cả các phần tử **athlete** là con của phần tử **country** nằm trong phần tử **games**
- `/games/country[1]/athlete[2]`
 - Phần tử **athlete** số 2 trong phần tử **country** thứ nhất
- `/games/country/athlete[sport="sailing"]`
 - Tất cả các phần tử **athlete** có phần tử con **sport** có giá trị là xâu **sailing**
- `/games/country[@id="Denmark"]/athlete`
 - Tất cả các phần tử **athlete** trong phần tử cha là **country** có thuộc tính **id** nhận giá trị là **Denmark**

DatTT-DSE-SOICT-HUST

54

Các ví dụ về Node Set (2)

- `/games/country/*`
 - Tất cả các phần tử con trong `/games/country`
- `/games/country//sport`
 - Tất cả các phần tử `sport` trong một cây con (subtree) bắt đầu bằng `/games/country`

DatTT-DSE-SOICT-HUST

55

XPath Type Coercion (Conversion)

- Đặc tả XPath định nghĩa các luật chuyển đổi giữa 4 kiểu dữ liệu: **node-set**, **boolean**, **number**, **string**
- Node-set được chuyển thành
 - boolean: true nếu node-set khác rỗng, false nếu ngược lại
 - string: giá trị string của node đầu tiên trong node-set
 - Ví thể, `<x:out select="$doc//sport"/>` cho kết quả là "swimming"
 - number: node-set được chuyển thành string trước, sau đó chuyển string về number

DatTT-DSE-SOICT-HUST

56

XPath Functions

- Biểu thức XPath có thể chứa các functions
- Ví dụ
 - `count(node-set)`: trả về số lượng các node trong một node-set
 - `count(/games/country)` trả về 2 vì có 2 country nodes trong node-set này
 - `id(object)`: lấy ra 1 node theo id đầu vào
 - `last()`: trả về kiểu number: chỉ số cuối của node-set
 - string functions
 - `string substring(/games/country, 1, 3)`
 - boolean functions
 - `boolean not(/games/country)`

DatTT-DSE-SOICT-HUST

57

2.3. Các thẻ XML-XML Tags

DatTT-DSE-SOICT-HUST

58

XML Tags

- Điều khiển luồng:
 - `<x:choose>`, `<x:when>`, `<x;if>`, `<x:otherwise>`
- Lặp:
 - `<x:forEach>`
- Mục đích chung khác
 - `<x:out>`
 - `<x:set>`
- Parsing & Transformation
 - `<x:parse>`
 - `<x:transform>` với các thẻ con `<x:param>`

DatTT-DSE-SOICT-HUST

59

XML Tags

- Sử dụng để truy cập các thông tin lưu trữ trong tài liệu XML
- Để truy cập tới thông tin mong muốn, sử dụng **biểu thức XPath** qua thuộc tính **select**
 - `<x:set var="d" select="$a//d"/>`
 - `<x:out select="$d/e"/>`
- Các thẻ khác: điều khiển luồng, lặp, ... tương tự như trong **Core tags**

DatTT-DSE-SOICT-HUST

60

<x:parse>

- Phân tích tài liệu XML và lưu thành 1 biến

DatTT-DSE-SOICT-HUST

61

Ví dụ: <x:parse>

```
<c:set var="xmlText">
  <a>
    <b>
      <c>
        foo
      </c>
    </b>
    <d>
      bar
    </d>
  </a>
</c:set>
```

```
<x:parse var="a" doc="{xmlText}" />
```

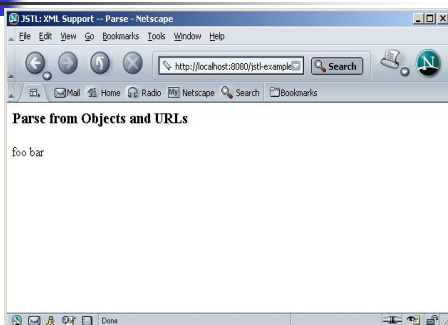
```
<x:out select="$a/c"/>
```

```
<x:out select="$a/a/d"/>
```

DatTT-DSE-SOICT-HUST

62

Ví dụ: <x:parse>



DatTT-DSE-SOICT-HUST

63

<x:out>

- Tương tự như thẻ <c:out>
- Thẻ <x:out> chuyển kiểu node-set thành String
 - Giá trị chuyển đổi được là giá trị String của node đầu tiên trong node-set
 - Giá trị String của 1 phần tử là nối của các text trong tất cả các node con của nó, ở độ sâu bất kỳ
- Example element String value

```
<athlete>
  <name>Lux 1</name>      Lux 1
  <sport>swimming</sport> swimming
  <age>23</age>           23
  <gender>M</gender>      M
</athlete>
```

DatTT-DSE-SOICT-HUST

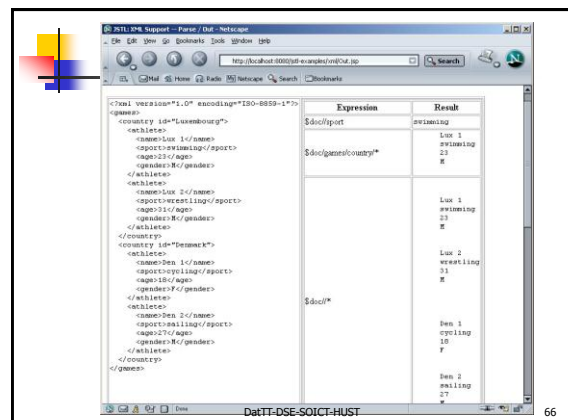
64

Ví dụ 1: <x:out>

```
<tr>
  <td>$doc//sport</td>
  <td><pre><x:out select="$doc//sport"/></pre></td>
</tr>
<tr>
  <td>$doc/games/country/*</td>
  <td><pre><x:out select="$doc/games/country/*"/></pre></td>
</tr>
<tr>
  <td>$doc/*</td>
  <td><pre><x:out select="$doc/*"/></pre></td>
</tr>
<tr>
  <td>$doc/games/country</td>
  <td><pre><x:out select="$doc/games/country"/></pre></td>
</tr>
```

DatTT-DSE-SOICT-HUST

65



DatTT-DSE-SOICT-HUST

66

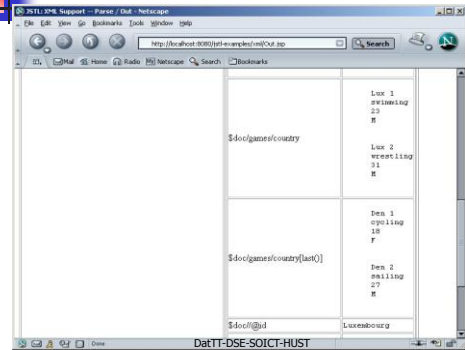
Ví dụ 2: <x:out>

```
<tr>
<td>${doc/games/country[last()]}</td>
<td><pre><x:out
select="${doc/games/country[last()]}"/></pre></td>
</tr>
<tr>
<td>${doc}/@id</td>
<td><pre><x:out select="${doc}/@id"/></pre></td>
</tr>
<tr>
<td>${doc}/country[@id='Denmark']</td>
<td><pre><x:out
select="${doc}/country[@id='Denmark']"/></pre></td>
</tr>
</table>
</td>
</tr>
```

DatTT-DSE-SOICT-HUST

67

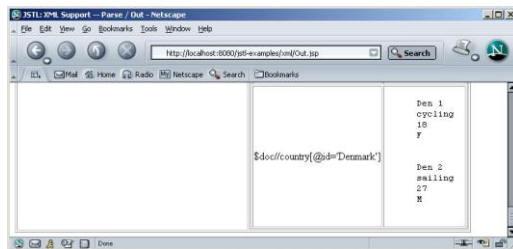
Ví dụ 2: <x:out>



DatTT-DSE-SOICT-HUST

68

Ví dụ 2: <x:out> (2)



DatTT-DSE-SOICT-HUST

69

Truy cập đến các biến trên các tầm vực trong biểu thức XPath

- JSTL XPath hỗ trợ các tầm vực sau để truy cập vào dữ liệu của ứng dụng Web trong biểu thức XPath

- \$foo
- \$param:
- \$header:
- \$cookie:
- \$initParam:
- \$pageScope:
- \$requestScope:
- \$sessionScope:
- \$applicationScope:

DatTT-DSE-SOICT-HUST

70

Ví dụ

```
<%-- Declare the core library --%>
<%@ taglib uri="/WEB-INF/tld/c.tld" prefix="c" %>

<%-- Save data in scoped variables --%>
<c:set var="name1" value="value1" scope="page" />
<c:set var="com_mycompany_name2" value="value2"
scope="request" />
<c:set var="com_mycompany_name3" value="value3"
scope="session" />
<c:set var="com_mycompany_name4" value="value4"
scope="application" />

<%-- Show the saved values --%>
<c:out value='${pageScope.name1}' />
<c:out value='${requestScope.com_mycompany_name2}' />
<c:out value='${sessionScope.com_mycompany_name3}' />
<c:out value='${applicationScope.com_mycompany_name4}' />
```

DatTT-DSE-SOICT-HUST

71

2.4. EL functions

DatTT-DSE-SOICT-HUST

72

Các hàm EL trong JSTL 1.1

- `<fn:length>`
- `<fn:toUpperCase>`, `<fn:toLowerCase>`
- `<fn:substring>`, `<fn:substringBefore>`, `<fn:substringAfter>`
- `<fn:trim>`
- `<fn:replace>`
- `<fn:indexOf>`, `<fn:startsWith>`, `<fn:endsWith>`, `<fn:containsIgnoreCase>`
- `<fn:split>`, `<fn:join>`
- `<fn:escapeXml>`

DatTT-DSE-SOICT-HUST

73

Ví dụ: EL Functions

```
<!-- truncate name to 30 chars and display it in uppercase -->
${fn:toUpperCase(fn:substring(name, 0, 30))}

<!-- Display the text value prior to the first '*' character -->
${fn:substringBefore(text, '*')}

<!-- Scoped variable "name" may contain whitespaces at the
beginning or end. Trim it first, otherwise we end up with '+'s in the URL -->
<c:url var="myUrl" value="${base}/cust/${fn:trim(name)}/"/>

<!-- Display the text in between brackets -->
${fn:substring(text, fn:indexOf(text, '[')+1, fn:indexOf(text, ']'))}

<!-- Display the name if it contains the search string -->
<c:if test="${fn:containsIgnoreCase(name, searchString)}">
  Found name: ${name}
</c:if>

<!-- Display the last 10 characters of the text value -->
${fn:substring(text, fn:length(text)-10)}

<!-- Display text value with bullets instead of '-' -->
${fn:replace(text, '-', '&#149;')}
```

DatTT-DSE-SOICT-HUST

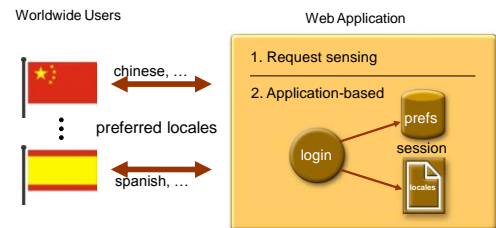
74

Phụ lục 2: Giới thiệu nhanh về Internationalization (I18N)

DatTT-DSE-SOICT-HUST

75

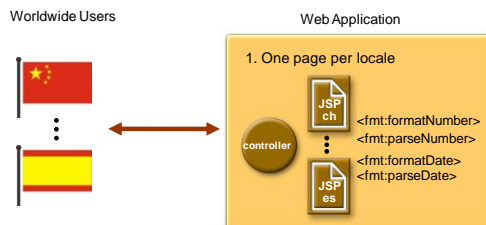
Locale được thiết lập như thế nào trong ứng dụng Web



DatTT-DSE-SOICT-HUST

76

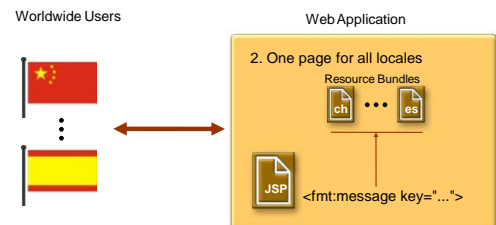
I18N Architecture: lựa chọn 1



DatTT-DSE-SOICT-HUST

77

I18N Architecture: Lựa chọn 2



DatTT-DSE-SOICT-HUST

78

2.5. Internationalization (i18n) & các thẻ định dạng Text

DatTT-DSE-SOICT-HUST

79

I18N và các thẻ định dạng

- Thiết lập locale
 - <fmt:setLocale>
 - <fmt:requestEncoding>
- Messaging (Hiển thị các thông tin)
 - <fmt:bundle>
 - <fmt:message> với thẻ con <fmt:param>
 - <fmt:setBundle>
- Định dạng Number và Date
 - <fmt:formatNumber>, <fmt:parseNumber>
 - <fmt:formatDate>, <fmt:parseDate>
 - <fmt:setTimeZone>, <fmt:timeZone>

DatTT-DSE-SOICT-HUST

80

Thiết lập Locales

- <fmt:setLocale>
 - Thiết lập locale của client cho 1 page (ghi đè lên thiết lập đã có)
- <fmt:requestEncoding>
 - Thiết lập kiểu encoding cho request

DatTT-DSE-SOICT-HUST

81

Messaging Tags

- <fmt:bundle>
 - Xác định **resource bundle** (file properties) cho trang
- <fmt:message key="...">
 - Sử dụng để trả về một chuỗi theo locale hiện tại
 - Có thể sử dụng thẻ con <fmt:param> để cung cấp tham số điền vào thông điệp

DatTT-DSE-SOICT-HUST

82

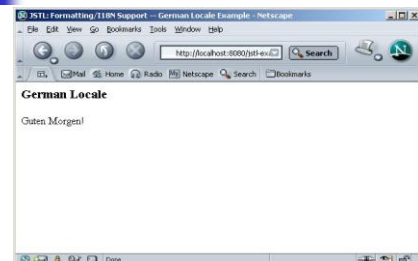
Ví dụ

```
<fmt:setLocale value="de"/>
<fmt:bundle
  basename="org.apache.taglibs.standard.examples.i18n.Resources">
  <fmt:message>
    greetingMorning
  </fmt:message>
</fmt:bundle>
```

DatTT-DSE-SOICT-HUST

83

Ví dụ: <fmt:setLocale>



http://localhost:8080/webapps-jstl/format/GermanLocale.jsp

DatTT-DSE-SOICT-HUST

84

Các thẻ định dạng (Formatting Tags)

- `<fmt:formatNumber>`, `<fmt:formatDate>`
 - Được sử dụng để lấy ra kiểu dữ liệu numbers hoặc dates theo locale hiện tại
- `<fmt:parseNumber>`, `<fmt:parseDate>`
 - Được sử dụng để phân tích các kiểu dữ liệu numbers và dates
- `<fmt:setTimeZone>`, `<fmt:timeZone>`
 - Thiết lập/truy cập timezone

DatTT-DSE-SOICT-HUST

85

Ví dụ

```
<jsp:useBean id="now" class="java.util.Date" />
<fmt:setLocale value="en-US" />

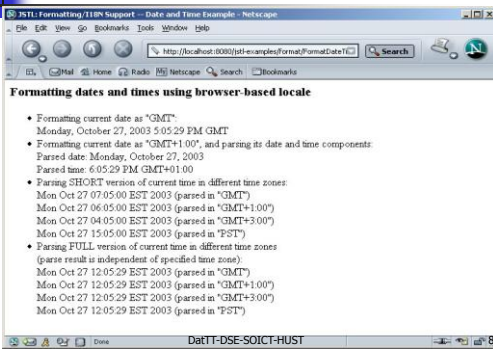
<ul>
<li> Formatting current date as "GMT": <br>
<fmt:timeZone value="GMT">
  <fmt:formatDate value="${now}" type="both" dateStyle="full" timeStyle="full"/>
</fmt:timeZone>

<li> Formatting current date as "GMT+1:00", and parsing
its date and time components: <br>
<fmt:timeZone value="GMT+1:00">
  <fmt:formatDate value="${now}" type="both" dateStyle="full"
timeStyle="full" var="formatted"/>
  <fmt:parseDate value="${formatted}" type="both" dateStyle="full"
timeStyle="full" timeZone="PST" var="parsedDateTime"/>
  Parsed date: <fmt:formatDate value="${parsedDateTime}" type="date"
dateStyle="full"/> <br>
  Parsed time: <fmt:formatDate value="${parsedDateTime}" type="time"
timeStyle="full"/>
</fmt:timeZone>
```

DatTT-DSE-SOICT-HUST

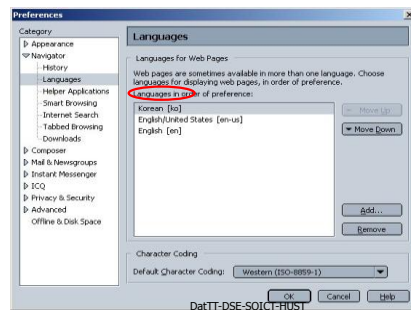
86

Ví dụ: sử dụng trình duyệt có thiết lập locale en-us



DatTT-DSE-SOICT-HUST

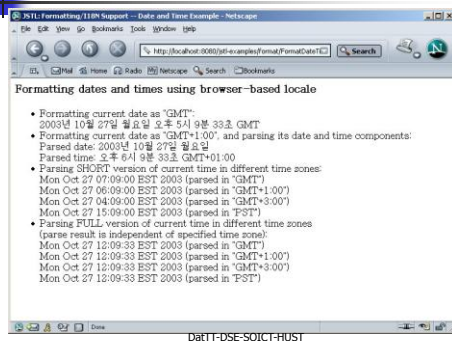
Thay đổi Locale của trình duyệt thành Korean



DatTT-DSE-SOICT-HUST

88

Truy cập lại trang web theo locale mới



DatTT-DSE-SOICT-HUST

89