




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

## LẬP TRÌNH WEB HƯỚNG JAVA

### Bài 13: JSP 2.0 Custom tags

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










## Nội dung



- 1. Trọng tâm của công nghệ JSP 2.0
- 2. Một số tính năng mới trong JSP 2.0
  - 2.1. Expression Language (EL)
  - 2.2. Xử lý thẻ với lớp tiện ích thực thi giao diện SimpleTag
  - 2.3. Tag files

2

## 1. Trọng tâm của công nghệ JSP 2.0

- Tăng tính dễ sử dụng

	User Class	HTML	XML	Java
JSP 1.2 JSP 2.0	Tag Library Developer			
	Advanced Page Author			
	Basic Page Author			

 = Basic Knowledge      = Expert

3

## Ví dụ cú pháp JSP 1.2 với Scriptlets

```

<%-- Output Shopping Cart --%>
<%@ page import="com.acme.util.*" %>
<%@ taglib prefix="util" uri="http://mytaglib" %>

<html>
<body>
  <util:getShoppingCart var="cart" />
  <table>
    <% for( int i = 0; i < cart.size(); i++ ) {
      CartItem item=(CartItem)cart.get(i);
    %>
    <tr>
      <td><%= item.getName() %></td>
      <td><%= item.getPrice() %></td>
    </tr>
    <% } %>
  </table>
</body>
</html>
  
```

4

## Ví dụ cú pháp JSP 2.0, không cần Scriptlets

```

<%-- Output Shopping Cart --%>
<%@ taglib prefix="util" uri="http://mytaglib" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<body>
  <util:getShoppingCart var="cart" />
  <table>
    <c:forEach var="item" values="${cart}">
      <tr>
        <td>${item.name}</td>
        <td>${item.price}</td>
      </tr>
    </c:forEach>
  </table>
</body>
</html>
  
```

5

## JSP 2.0 tuân thủ cú pháp XML

```

<!-- Output Shopping Cart -->
<html xmlns:util="http://mytaglib"
      xmlns:c="http://java.sun.com/jsp/jstl/core">
  <body>
    <util:getShoppingCart var="cart" />
    <table>
      <c:forEach var="item" values="${cart}">
        <tr>
          <td>${item.name}</td>
          <td>${item.price}</td>
        </tr>
      </c:forEach>
    </table>
  </body>
</html>
  
```

6

## 2.1. Expression language

DatTT-DSE-SOICT-HUT

7

## Expression Language

- Phát triển dựa trên "SPEL" từ JSTL 1.0
  - Simplest Possible Expression Language
- Cho phép truy cập giá trị thuộc tính của một JavaBean theo cú pháp đơn giản hơn:
  - Ví dụ: `${item.price}`
- Hỗ trợ các hàm EL functions:
  - Được định nghĩa với cơ chế tương tự như các custom tags
  - Ví dụ: `${fn:allCaps(lastName)}`
  - JSTL 1.1 cung cấp 16 EL functions chuẩn

DatTT-DSE-SOICT-HUT

8

## Ví dụ 1

- Sử dụng Scriptlets::

```
<center>
  <jsp:useBean id="foo" class="FooBean" />
  <%= foo.getBar() %>
</center>
```

- Sử dụng EL expression:

```
<center>
  ${foo.bar}
</center>
```

DatTT-DSE-SOICT-HUT

9

## Ví dụ 2

- Sử dụng scriptlets:

```
<% Map m = (Map)pageContext.getAttribute("state" );
   State s = ((State)m.get( "NY" ));
   if( s != null ) {
%>
      <%= s.getCapitol() %>
%>
```

- Sử dụng EL expression:

```
${state["NY"].capitol}
```

DatTT-DSE-SOICT-HUT

10

## 2.2. Giao diện xử lý thẻ SimpleTag

DatTT-DSE-SOICT-HUT

11

## So sánh các lớp xử lý thẻ

- Các lớp xử lý thẻ truyền thống (JSP 1.2)
  - Xử lý thẻ với API phức tạp
  - Chỉ viết được bằng ngôn ngữ Java
  - Chỉ được tạo bởi những LTV phát triển thư viện thẻ
- Các lớp xử lý thẻ mới (thực thi giao diện SimpleTag) (JSP 2.0)
  - Xử lý thẻ với API đơn giản hơn
  - Viết bằng ngôn ngữ Java hoặc sử dụng cú pháp của JSP (trong các file Tag)
  - Được tạo bởi các **page authors** hoặc LTV phát triển thư viện thẻ

DatTT-DSE-SOICT-HUT

12

## Các lớp xử lý thẻ trong JSP 2.0

- Dễ sử dụng hơn các lớp xử lý thẻ truyền thống
- Thực thi giao diện SimpleTag
- Thường kế thừa lớp SimpleTagSupport
- Phương thức doTag() trong giao diện SimpleTag sẽ được gọi khi bắt gặp thẻ đóng
  - LTV cần override phương thức này

DatTT-DSE-SOICT-HUT

13

## Cách thức container gọi lớp xử lý thẻ SimpleTag (Các lớp thực thi giao diện SimpleTag)?

```

ATag t = new ATag();
t.setJSPContext(...);
t.setParent(...);
t.setAttribute1(value1);
t.setAttribute2(value2);
...
t.setJspBody(new JspFragment(...))
t.doTag();

```

DatTT-DSE-SOICT-HUT

14

## JspFragment

- Đóng gói mã nguồn JSP vào một đối tượng để sau đó gọi ra với số lần tùy ý
- JSP Fragments được biên dịch từ đoạn code JSP.

DatTT-DSE-SOICT-HUT

15

## Lớp xử lý thẻ không xử lý phần body

- Nếu một tag handler chỉ đơn thuần lấy về nội dung của phần body
  - Lấy qua phương thức getJspBody() của SimpleTag (trả về đối tượng JspFragment)
  - Sau đó, đọc phần body với phương thức invoke()
- Phương thức invoke()
  - Sử dụng lời gọi invoke(null) khi không cần xử lý phần body
  - Ngược lại, gọi invoke(StringWriter writer)

DatTT-DSE-SOICT-HUT

16

## Ví dụ: Lớp xử lý thẻ không xử lý phần body

```

public class IfSimpleTag extends SimpleTagSupport {
    private boolean test;
    public void setTest(boolean test) {
        this.test = test;
    }
    public void doTag() throws JspException, IOException {
        if(test){
            getJspBody().invoke(null);
        }
    }
}

```

DatTT-DSE-SOICT-HUT

17

## Lớp xử lý thẻ xử lý phần body

- Nếu lớp xử lý thẻ cần xử lý phần body, phải đặt phần body vào trong một StringWriter

```

public class SimpleWriter extends SimpleTagSupport {
    public void doTag() throws JspException, IOException {
        StringWriter sw = new StringWriter();
        jspBody.invoke(sw);
        jspContext().getOut().println(sw.toString().toUpperCase());
    }
}

```

DatTT-DSE-SOICT-HUT

18

## Cài đặt thẻ lặp (Repeat Tag) với các lớp trong JSP 1.2

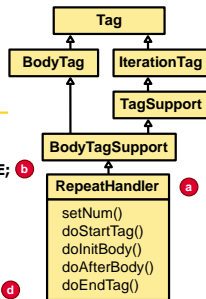
**Usage**

```
<%@ taglib prefix="my"
    uri="/mytags" %>
<my:repeat num="3">
    tag body
</my:repeat>
```

**Implementation**

```
int doStartTag() {
    this.count = this.num;
    return Tag.EVAL_BODY_INCLUDE;
}

int doAfterBody() {
    this.count--;
    return (this.count > 0) ?
        Tag.EVAL_BODY_AGAIN :
        Tag.SKIP_BODY;
}
```



19

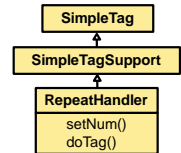
## Cài đặt thẻ lặp với các lớp trong JSP 2.0

**Usage**

```
<%@ taglib prefix="my"
    uri="/mytags" %>
<my:repeat num="3">
    tag body
</my:repeat>
```

**Implementation**

```
void doTag() {
    for (int i = 0; i < num; i++) {
        getJspBody().invoke( null );
    }
}
```



DatTT-DSE-SOICT-HUT

20

## 2.3. Tag files

DatTT-DSE-SOICT-HUT

21

## Tag Files

- Là một file mã nguồn chứa các đoạn code JSP, được tái sử dụng như một custom tag
- Cho phép tạo các custom tags sử dụng cú pháp JSP
- Được dịch thành lớp xử lý thẻ tương ứng, sau đó được biên dịch tự động bởi container
  - "JSP → Servlet"
  - "Tag File → Tag Handler"
- Không cần các file TLD

DatTT-DSE-SOICT-HUT

22

## Tag Files

- Dễ dàng đóng gói
  - Chỉ cần đặt các file .tag vào thư mục **/WEB-INF/tags/**
  - Các thư viện thẻ tương ứng sẽ được sinh ra tự động
  - Nếu muốn linh động hơn, có thể viết thêm file .tld

DatTT-DSE-SOICT-HUT

23

## Khai báo 1 thư viện thẻ: thuộc tính tagdir trong taglib directive

- Xác định vị trí của các **tag files**
- Giá trị thuộc tính tagdir bắt đầu bằng **/WEB-INF/tags/**
- Cú pháp
  - `<%@ taglib prefix="tt" tagdir=/WEB-INF/tags/dir %>`

DatTT-DSE-SOICT-HUT

24

## Các Directives được sử dụng trong 1 Tag file

- **taglib**
- **include**
- **tag**
  - Tương tự như các **page directive** của trang JSP, nhưng áp dụng cho các tag files
- **attribute**
  - Khai báo các thuộc tính của thẻ được định nghĩa trong tag file
- **variable**
  - Khai báo biến EL của thẻ cung cấp cho trang JSP sử dụng thẻ

DatTT-DSE-SOICT-HUT

25

## Các thuộc tính trong "attribute" Directive

- description
- name
- required
- rtexprvalue
- type
- fragment (mặc định là false)
  - Nếu là true, container sẽ thiết lập
    - Giá trị thuộc tính rtexprvalue là true
    - Giá trị của thuộc tính type là `javax.servlet.jsp.tagext.JspFragment`
  - Nếu là false, attribute của Tag chỉ là attribute thường, cho phép container truyền tham số cho lớp xử lý thẻ

DatTT-DSE-SOICT-HUT

26

Ví dụ 1: Attribute thường  
(shipDate.tag)

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ attribute name="shipping" required="true" %>

<jsp:useBean id="now" class="java.util.Date" />
<jsp:useBean id="shipDate" class="java.util.Date" />
<c:choose>
<c:when test="${shipping == 'QuickShip'}">
<c:set var="days" value="2" />
</c:when>
<c:when test="${shipping == 'NormalShip'}">
<c:set var="days" value="5" />
</c:when>
<c:when test="${shipping == 'SaverShip'}">
<c:set var="days" value="7" />
</c:when>
</c:choose>
<jsp:setProperty name="shipDate" property="time"
value="${now.time + 86400000 * days}" />
<fmt:formatDate value="${shipDate}" type="date"
dateStyle="full"/> <br> <br>
</c:choose>
```

DatTT-DSE-SOICT-HUT

27

Ví dụ 1: Attribute thường  
(bookreceipt.jsp - trang sử dụng thẻ)

[illegible]

DatTT-DSE-SOICT-HUT

28

## "variable" Directive

- Khai báo các **EL variables**
- **EL variables**
  - EL variables giống như ĐẦU RA (từ tag file truyền cho trang JSP gọi nó còn Tag attributes giống như đầu vào (từ trang JSP truyền vào tag file) type
  - Không được khởi tạo từ trang JSP sử dụng thẻ
  - Được thiết lập giá trị trong tag file

DatTT-DSE-SOICT-HUT

29

## Các thuộc tính của "variable" Directive

- `description`
- `name-given|name-from-attribute`
  - Định nghĩa một biến EL sẽ được sử dụng trong trang JSP gọi đến thẻ
- `alias`
- `variable-class`
- `declare`
- `scope`

DatTT-DSE-SOICT-HUT

30

## Lấy các đoạn code (Fragments) được truyền cho Tag Files

- Khi thực thi, Web container truyền 2 loại fragments (các đoạn code) cho tag file
  - **fragment attribute**
  - tag body
- Bên trong tag file, có thể sử dụng
  - **jsp:invoke** để lấy về loại **fragment attribute**
  - **jsp:doBody** để lấy về body của tag file

DatTT-DSE-SOICT-HUT

31

## Lấy các đoạn code (Fragments) được truyền cho Tag Files

- Fragments lấy về có thể
  - Được gửi cho đối tượng response hoặc
  - Được lưu trữ thành một EL variable để xử lý về sau. Muốn vậy, cần chỉ định 1 trong 2 kiểu thuộc tính (tên biến là giá trị của 2 thuộc tính này):
    - var: EL variable có kiểu là String
    - varReader: EL variable có kiểu là java.io.Reader
  - Thuộc tính scope (tùy chọn) chỉ định phạm vi của biến kết quả nhận được
    - page (mặc định)
    - request
    - session
    - application

DatTT-DSE-SOICT-HUT

32

## Ví dụ 2: Attributes thường, Attributes Fragment và Variables (catalog.tag - 1)

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

<%@ attribute name="bookDB" required="true" type="database.BookDB" %>
<%@ attribute name="color" required="true" %>

<%@ variable name-given="price" %>
<%@ variable name-given="salePrice" %>

<%@ attribute name="normalPrice" fragment="true" %>
<%@ attribute name="onSale" fragment="true" %>
```

DatTT-DSE-SOICT-HUT

33

## Ví dụ 2: Attributes thường, Attributes Fragment và Variables (catalog.tag - 2)

```
<center>
<table>
<c:forEach var="book" begin="0" items="${bookDB.books}">
<tr>
<c:set var="bookId" value="{book.bookId}" />
<td bgcolor="${color}">
<c:url var="url" value="/bookdetails">
<c:param name="bookId" value="{bookId}" />
</c:url>
<a href="{url}"><strong>${book.title}&nbsp;</strong></a></td>
<td bgcolor="${color}" rowspan=2>
<c:set var="salePrice" value="{book.price * .85}" />
<c:set var="price" value="{book.price}" />
</td>
</tr>
<c:choose>
<c:when test="{book.onSale}">
<jsp:invoke fragment="onSale" />
</c:when>
<c:otherwise>
<jsp:invoke fragment="normalPrice" />
</c:otherwise>
</c:choose> &nbsp;</td>
```

Attributes passed from calling page

Variables

Attributes with fragments

DatTT-DSE-SOICT-HUT

34

## Ví dụ 2: Attributes thường, Attributes Fragment và Variables (catalog.tag - 3)

```
<td bgcolor="{color}" rowspan=2>
<c:url var="url" value="/bookcatalog" >
<c:param name="Add" value="{bookId}" />
</c:url>
<p><strong><a href="{url}">&nbsp;</a><fmt:message
key="CartAdd"/>&nbsp;</td></tr>

<tr>
<td bgcolor="#ffffff">
&nbsp;&nbsp;&nbsp;<fmt:message key="By"/>
<em>${book.firstName}&nbsp;&nbsp;&nbsp;${book.surname}</em></td></tr>
</c:forEach>

</table>
</center>
```

DatTT-DSE-SOICT-HUT

35

## Ví dụ 2: Attributes thường, Attributes Fragment và Variables (bookcatalog.jsp - 1)

```
<%@ taglib prefix="sc" tagdir="/WEB-INF/tags" %>

<jsp:useBean id="bookDB" class="database.BookDB" scope="page" >
<jsp:setProperty name="bookDB" property="database"
value="{bookDBAO}" />
</jsp:useBean>

<c:if test="{!empty param.Add}">
<c:set var="bid" value="{param.Add}" />
<jsp:setProperty name="bookDB" property="bookId" value="{bid}" />
<c:set var="addedBook" value="{bookDB.bookDetails}" />
<p><h3><font color="red" size="+2">
<fmt:message key="CartAdded1"/> <em>${addedBook.title}</em>
<fmt:message key="CartAdded2"/> </font></h3>
</c:if>
```

DatTT-DSE-SOICT-HUT

36

