**CHƯƠNG 6. Generic tags**

1. **Giới thiệu**

* Như đã giới thiệu chương 5 “Form tags”, tag library của struts có 2 loại là UI tags và Non-UI tags. Trong chương này, ta sẽ tìm hiểu về Non-UI tags (generic tags).
* Genetic tags có 2 loại là data tag và control tag.
* Data tag gồm các tag sau:
  + a
  + action
  + bean
  + date
  + debug
  + i18n
  + include
  + param
  + push
  + get
  + text
  + url
  + property
* Control tag gồm các tag sau:
  + if
  + elseft
  + else
  + append
  + generator
  + iterator
  + merge
  + sort
  + subset

1. **Chi tiết các tag**
2. **Property tag**

* **Property tag** dùng để in ra các thuộc tính của action, gồm các attribute sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| default | String |  | Giá trị là default nếu value là null |
| escape | Boolean | true | Whether HTML special characters are escaped |
| value | String | <top on stack> | Giá trị hiển thị |

* Ví dụ về **tag property** xuất hiển thị ra giá trị **customerId** action

<s:property value =”customerId”/>

* Nếu giá trị của thuộc tính là null, thì giá trị đầu tiên trong value stack sẽ được hiển thị ra (<top on stack>).
* Mặc định**, property tag** sẽ không hiển thị khi gì khi gặp các kí tự đặc biệt HTML – bảng 6.2 **(chỗ này thực sự chưa hiểu rõ)**

|  |  |
| --- | --- |
| Character | Escaped Characters |
| “ | &quot; |
| & | &map; |
| < | &lt; |
| > | &gt; |

1. **a tag**

* **a tag** render thành HTML anchor, dạng link liên kết. Các thuộc tính của tag này giống như thuộc tính 1 element HTML. Ví dụ, tạo một anchor trỏ đến trang web [www.vnexpress.net](http://www.vnexpress.net)

<s:a hreft=<http://www.vnexpress.net>>Click đọc báo</s:a>

1. **action tag**

* **action tag** dùng để thực thi một action hoặc kết quả của action đó. Tag này cũng add thêm action vào Value Stack’s context map. Gồm các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| executeResult | Boolean | False | Cho biết action result có được execute/render |
| flush | Boolean | True | Cho biết bộ nhớ đệm (writer) có được xóa sạch khi kết thúc các tag thành phần. |
| ignoreContextParams | Boolean | False | Cho biết các request parameter được thêm khi action được gọi. |
| name\* | String |  | Tên của action được gọi, ko có hậu tố “.action” |
| namespace | String | Namespace của tag đang sử dụng | Namespace của action được gọi |
| var | String |  | Tên được sử dụng tham chiếu qua action được add trong context map. |

* Ví dụ, **action tag** được dụng cho action **MyAction**. Action object là biến **obj** trong Value Stack’s context map.

<s:action var=”obj” name =”MyAction” executeResult=”false”/>

1. **param tag**

* Param tag được dùng để pass một tham số chứa trong tag.Gồm các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| name | String |  | Tên của parameter được pass trong tag |
| value | String |  | Giá trị của parameter được pass trong tag |

* Thuộc tính value luôn được gán kể cả khi không có **%{** and **}**.Ví dụ:

<s:param name="userName" value="userName"/>

Tương đương:

<s:param name="userName" value="%{userName}"/>

1. **bean tag**

* **bean tag** tạo ra JavaBean và lưu nó trong Value Stack’s context map. Tag này tương tự như một hàm của JSP sử dụng action. Gồm các thuộc tính chính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default |  |
| name\* | String |  | Tên class đầy đủ của JavaBean được tạo. |
| var | String |  | Tên được sử dụng để tham chiếu giá trị đẩy vào Value Stack’s context map. |

* Ví dụ, trong class **DegreeConverter** trong **Listring 6.3**, có phương thức convert độ C qua độ F và ngược lại**. Bean.jsp** trong **Listing 6.4** sử dụng **bean tag** để thực hiện việc này

**Listing 6.3. Class DegreeConverter**

|  |
| --- |
| package demo;  public class DegreeConverter  {  private float doC;  private float doF;  public float getDoC() {  return (doF -32) \* 5/9;  }  public void setDoC(float c){  this.doC = c;  }  public float getDoF() {  return doC \* 9/5 + 32;  }  public void setDoF(float f){  this.doF = f;  }  } |

**Listing 6.4. Bean.jsp**

|  |
| --- |
| <%@ taglib prefix=”s” uri=”/struts-tags” %>  <html>  <head>  <title>bean tag Example</title>  <style type = “text/css”>@import url(css/main.css);</style>  </head>  <body>  <div id=”global” style=”width:250px”>  <s:bean name=”demo.DegreeConverter” id=”converter”>  <s:param name=”doF” value=”212”/>  </s:bean>  212&deg;F=<s:property value=”#converter.doC”/>&deg;C  </div>  </body>  </html> |

**(ví dụ này thực sự chưa hiểu, tag bean đã tự chay hàm getDoC??)**

* Kết quả là ta được như hình 6.2



1. **date tag**

* **date tag** dùng để định dạng lại một đối tượng date của java. Nó gồm các đối tượng sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| format | String |  | Định dạng ngày theo mẫu |
| name\* | String |  | Giá trị ngày format |
| nice | Boolean | False | Áp dụng nice formatting |
| var | String |  | Name được dùng tham chiếu giá trị đẩy vào value stack. |

* Thuộc tính **format** chỉnh lại date và time theo mẫu (đã được đinh nghĩa trong class **java.text.SimpleDateFormat**). Ví dụ, trang **Data.jsp** **trong Listing 6.5** sử dụng **date tag** để format dates.

**Listing 6.5. Date.jsp**

|  |
| --- |
| <%@ taglib prefix=”s” uri=”/struts-tags” %>  <html>  <head>  <title>date tag Example</title>  <style type = “text/css”>@import url(css/main.css);</style>  </head>  <body>  <div id=”global” style=”width:350px”>  <s:bean name=”java.util.Date” var =”today”/>  Today (original format): <s:property value =”#today”/>  <s:date name=”#today” var=”format1” format=”M/dd/yyyy”/>  <br/>Today (mm/dd/yyyy): <s:property value=”#format1”/>  <s:date name=”#today” var=”format2” format=”MMM d, yyyy”/>  <br/>Today (MMM d, yyyy): <s:property value=”#format2”/>  <s:date name=”#today” var=”format3” format=”MMM d, yyyy, hh:mm”/>  <br/>Today (MMM d, yyyy, hh:mm): <s:property value=”#format3”/>  </div>  </body>  </html> |

* Khi đó, kết quả sẽ là:

****

1. **include tag**

* Tag này bao gồm output của servlet hoặc jsp. Và chỉ có duy nhất một thuộc tính value

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| value\* | String |  | Ouput của servlet/JSP được thêm vào |

1. **set tag**

* **set tag** tạo ra một giá trị cho một biến trong một phạm vi chỉ định. Nó rất hữu ích khi bạn muốn tạo một biến được gán từ một biểu thức phức tạp và sau đó chỉ cần tham chiếu là biến mỗi lần thay vì biểu thức phức tạp. Nó tạo cặp key/value từ một trong các map sau:
  + Value Stack’s context map
  + Session map
  + Application map
  + Request map
  + Page map
* set tag gồm các attribute sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| name | String |  | Key của thuộc tính được tạo |
| value | String |  | Giá trị của key |
| scope | String | default | Phạm vi của biến, có thể là application, session, request, page hoặc default |

* Ví dụ:

|  |
| --- |
| <s:set name="personName" value="person.name"/>  Hello, <s:property value="#personName"/>. How are you? |

1. **push tag**

* **push tag** tương tự như **set tag**, nhưng điểm khác là **push** đẩy đối tượng vào Value Stack, ko phải context map. Do đó, các tag bên trong **tag push**, khi cần xài các thuộc tính đối tượng này thì không cần gõ lại tên đối tượng, mà gõ trực tiếp tên thuộc tính
* push tag chỉ có duy nhất một attribute là value

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| value\* | String |  | Value được đẩy vào value stack |

* Ví dụ, class **PushTestAction** ở Listing 6.8, có phương thức **execute** đẩy một đối tượng **Employee** vào **HttpSession**

**Listing 6.8. Class PushTestAction**

|  |
| --- |
| package demo;  import java.util.Map;  import org.apache.struts2.interceptor.SessionAware;  import com.opensymphony.xwork2.ActionSupport;  public class PushTestAction extends ActionSupport  implements SessionAware {  private Map sessionMap;  public void setSession (Map sessionMap) {  this.sessionMap = sessionmap;  }  public String execute() {  Employee employee = new Employee();  employee.setId(“1”);  employee.setFirstName(“Karl”);  employee.setLastName(“Popper”);  sessionMap.put(“employee”,employee);  return SUCCESS;  }  }  Class Employee {  private int id;  private String firstName;  private String lastName;  //Các phương thức getter và setter không được hiển thị ở đây.  } |

* Trang **Push.jsp** ở **Listing 6.9** sử dụng push tag để đẩy đối tượng **Employee** vào Value Stack

**Listing 6.9: Trang Push.jsp**

|  |
| --- |
| <%@ taglib prefix=”s” uri=”/struts-tags” %>  <html>  <head>  <title>push tag Example</title>  <style type = “text/css”>@import url(css/main.css);</style>  </head>  <body>  <div id=”global” style=”width:250px”>  <h3>Employee Details</h3>  <s:push value =”#session.employee”>  Employee Id: <s:property value=”id”/>  <br/>First Name: <s:property value=”firstName”/>  <br/>Last Name: <s:property value=”lastName”/>  </s:push>  </div>  </body>  </html> |

* Kết quả ta được như sau:



1. **url tag**

* **url tag** tạo một url động. Gồm các attribute chính sau đây:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| action | String |  | Action tạo ra url |
| anchor | String |  | Anchor của url **(có phải tên để hiển thị cho url ???)** |
| encode | Boolean | True | **Where the encode parameters** |
| escapeAmp | Boolean | True | Chỉ rõ có thoát khi gặp dấu & hay không |
| includeContext | Boolean | True | **Indicates whether the actual context shoule be included** |
| includeParams | String | Get | **One of these values:one, get, all ? what for?** |
| method | String |  | Phương thức của action |
| namespace | String |  | Namespace được dùng |
| portletMode | String |  | The resulting prolet mode |
| porletUrlType | String |  | Chỉ rõ URL được tạo là một porlet render hay một action URL |
| scheme | String |  | **Set scheme attribute** |
| value | String |  | Giá trị được gán trực tiếp nếu không xài action |
| var | String |  | Name dùng tham chiếu giá trị được đẩy vào value stack |
| windowState | String |  | Khi sử dụng môi trường portlet, chỉ rõ portlet window state |

* Ví dụ:

|  |
| --- |
| <-- Example 1 -->  <s:url value="editGadget.action">  <s:param name="id" value="%{selected}" />  </s:url>  <-- Example 2 -->  <s:url action="editGadget">  <s:param name="id" value="%{selected}" />  </s:url>  <-- Example 3-->  <s:url includeParams="get">  <s:param name="id" value="%{'22'}" />  </s:url> |

1. **if, else, và elseIf tag**

* Có 3 tag dùng để thực hiện các việc kiểm tra điều kiện như trong java là if, else, và if else. Tag if và elseif phải có thuộc tính test

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| Test\* | Boolean |  | Kiểm tra điều kiện |

* Ví dụ**, if tag** kiểm trả nếu tham số **ref** là null

<s:if test=”#parameters.ref == null”>

* Thêm một ví dụ khác, if tag được dùng để kiểm tra thuộc tính loggedIn của session tồn tại. Nếu thuộc tính này không có, thì loginform sẽ hiển thị. Ngược lại, hiện bảng welcome. Ví dụ này gồm class IfTestAction ở Listing 6.10 và If.jsp ở Listing 6.11

**Listing 6.10. Class IfTestAction**

|  |
| --- |
| package demo  import org.apache.struts2.ServletactionContext;  import com.opensymphony.xwork2.ActionSupport;  public class IfTestAction extends ActionSupport {  private String userName;  private String password;  //Các phương thức getter và setter không được hiển thị ở đây  Public String execute() {  if (userName != null && username.length() > 0  && password != null  && password.length > 0 {  ServletActionContext.getContext().  getSession().put(“loggedIn”, true);  }  Return SUCCESS;  }  } |

**Listing 6.11 Trang If.jsp**

|  |
| --- |
| <%@ taglib prefix=”s” uri=”/struts-tags” %>  <html>  <head>  <title>push tag Example</title>  <style type = “text/css”>@import url(css/main.css);</style>  </head>  <body>  <div id=”global” style=”width:350px”>  <s:if test=”#session.loggedIn == null”>  <h3>Login</h3>  <s:form>  <s:textfield name =”userName” label=”User Name”/>  <s:password name=”password” label=”Password”/>  <s:submit value=”Login”/>  </s:form>  </s:if>  <s:else>  Welcome <s:property value=”userName”/>  </s:else>  </div>  </body>  </html> |

* Kết quả ta được hình sau: (khi chưa login lần nào)



1. **interator tag**

* **interator tag** là một tag quan trọng. Nó được dùng để lặp một mảng, một Collection, hoặc một Map và thêm và lấy từng phần tử trong đối tượng lặp ở Value Stack. Tag này gồm các thuộc tính sau

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| value | String |  | Đối tượng iterable để lặp |
| status | Org.apache.struts2.views.jsp.IteratorStatus |  |  |
| var | String |  | Biến tham chiếu đến phần tử hiện tại trong mảng |

* Khi thực hiện, **iterator tag** đẩy một **IteratorStatus** vào context map và cập nhật nó sau mỗi vòng lặp. Thuộc tính **status** có thể được gán một biến trỏ đến đối tượng **IteratorStatus**.
* Các tính chất của **InteratorStatus** gồm:

|  |  |  |
| --- | --- | --- |
| Name | Type | Mô tả |
| index | interger | Giá trị củban đầu của một vòng lặp |
| count | Integer | Số vòng lặp hiện tại |
| first | Boolean | Là true nếu phần tử hiện tại là giá trị đầu vòng lặp |
| last | Boolean | Là true nếu phần tử hiện tại là giá trị cuối vòng lặp |

* Ví dụ: lấy giá trị các ngày trong tuần qua hàm getDays() (days là mảng một tuần). Các tag <s:property/> in ra giá trị hiện tại của các biến lặp

|  |
| --- |
| <s:iterator value="days">  <p>day is: <s:property/></p>  </s:iterator> |

* Ví dụ: lặp có chọn lựa

|  |
| --- |
| <s:iterator value="{1,2,3,4,5}" begin="2" end="4" >  <!-- current iteration value (2,3,4) -->  <s:property value="top" />  </s:iterator> |

1. **append tag**

* **append tag** dùng để ghép nối tiếp các iterators (mảng, vòng lặp..) với nhau. Ví dụ, nếu bạn có 2 list với 3 phần tử mỗi list, bạn sẽ có một mảng mới như sau:
  + List 1, phần tử 1
  + List 1, phần tử 2
  + List 1, phần tử 3
  + List 2, phần tử 1
  + List 2, phần tử 2
  + List 2, phần tử 3
* **append tag** chỉ có một thuộc tính var

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| var | String |  | Biến được tạo để tham chiếu đến mảng được nối. |

* Ví dụ, **Listing 6.14** sử dụng append tag để nối 2 list:

**Listing 6.14 Sử dụng append**

|  |
| --- |
| <s:set var=”list1” value=”{‘one’, ‘two’}”/>  <s:set var=”list2” value=”{‘1’, ‘2’, ‘3’}”/>  <s:append var=”allLists”>  <s:param value=”#list1”/>  <s:param value=”#list2”/>  </s:append>  <s:iterator value=”#allLists”>  <s:property/><br/>  </s:iterator> |

* Kết quả ta sẽ được mảng sau:
  + one
  + two
  + 1
  + 2
  + 3
* Nếu sử dụng merge tag, cũng tương tư như append tag, thì ví dụ trên kết quả là:
  + one
  + 1
  + two
  + 2
  + 3

1. **merge tag**

* **merge tag** trộn các list bằng cách đọc từng phần tử từ mỗi list.Ví dụ, ta có 2 list với 3 phần tử mỗi list, thì list mới sẽ là:
  + List 1, phần tử 1
  + List 2, phần tử 1
  + List 1, phần tử 2
  + List 2, phần tử 2
  + List 1, phần tử 3
  + List 2, phần tử 3
* **merge tag** chỉ có một thuộc tính var

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| var | String |  | Biến được tạo để tham chiếu đến mảng được nối. |

* Ví dụ, action class **MergeTestAction** có 3 thuộc tính list là: myList1, myList2, myList3. Chi tiết class này ở Listing 6.15

**Listing 6.15. Class MergeTestAction**

|  |
| --- |
| public class MergeIteratorTagAction extends ActionSupport {  private List myList1;  private List myList2;  private List myList3;  public List getMyList1() {  return myList1;  }  public List getMyList2() {  return myList2;  }  public List getMyList3() {  return myList3;  }  public String execute() throws Exception {  myList1 = new ArrayList();  myList1.add("1");  myList1.add("2");  myList1.add("3");  myList2 = new ArrayList();  myList2.add("a");  myList2.add("b");  myList2.add("c");  myList3 = new ArrayList();  myList3.add("A");  myList3.add("B");  myList3.add("C");  return "done";  }  } |

**Listing 6.16 Trang Merge.jsp**

|  |
| --- |
| <s:merge var="myMergedIterator1">  <s:param value="%{myList1}" />  <s:param value="%{myList2}" />  <s:param value="%{myList3}" />  </s:merge>  <s:iterator value="%{#myMergedIterator1}">  <s:property />  </s:iterator> |

* Kết quả ta được list là: 1,a,A,2,b,B,3,c,C

1. **generator tag**

* **generator tag** dùng để sinh ra một mảng và đẩy vào Value Stack. Gồm các attribute sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| converter | Converter |  | Chuyển đổi sang dạng String từ **val** thành một object |
| count | Integer |  | Số lượng tối đa một phần tử trong mảng |
| separator\* | String |  | Phân cách từ val thành các giá trị nhỏ của mảng |
| val\* | String |  | Giá trị cần chuyển thành mảng |
| var | String |  | Biến đế lưu mảng kết quả |

* Khi dùng thuộc tính **converter** thì phải thiết lập một thuộc tính action dạng **Converter** (định nghĩa trong org.apache.struts2.util.IteratorGenerator)
* Ví dụ: Trang Generator.jsp ở Listing 6.17 sử dụng generator tag để tạo một list từ String

**Listing 6.17 Trang Generator.jsp**

|  |
| --- |
| <s:generator val=”%{‘Honda, Toyota, Ford, Dodge’}” separator=”,”>  <ul>  <s:iterator>  <li><s:property/></li>  </s:iterator>  <ul>  </s:generator>  <s:generator id=”cameras” count=”3”  val=”%{‘Canon,Nikon,Pentax,FujiFilm’}” separator=”.”>  </s:generator>  <s:iterator value=”#attr.cameras”>  <s:property/>  </s:iterator> |

* Kết quả ta được là:



1. **sort tag**

* **sort tag** sắp xếp các phần tử của iterator. Gồm các attribute sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| comparator\* | java.util.Comparator |  | Tiêu chí săp xếp |
| source | String |  | Iterator được sắp xếp |
| var | String |  | Biến chứa iterator kết quả |

* Ví dụ, **Class SortTestAction** ở Listing 6.20 có thuộc tính dạng **Comparator** dùng trong sort tag ở trang Sort.jsp

**Listing 6.20 Class SortTestAction**

|  |
| --- |
| package demo;  import java.util.Comparator;  import com.opensymphony.xwork2.ActionSupport;  public class SortTestAction extends ActionSupport {  public Comparator getMyComparator() {  return new Comparator() {  public in compare (Object o1, Object o2) {  return o1.toString().compareTo(o2.toString());  }  };  }  } |

**Listing 6.21 Trang Sort.jsp**

|  |
| --- |
| ….  <h4>Computer></h4>  <s:generator id=”computers” val=”%{‘HP, Dell, Asus, Fujitsu, Toshiba’ }”  separator=”,”>  <s:sort comparator=”myComparator”>  <s:iterator>  <s:property/>  </s:iterator>  </s:sort>  </s:generator>  <hr/>  <h4>Cameras></h4>  <s:generator id=”cameras” val=”%{‘Canon, Nikon, Pentax, FujiFilm’ }”  separator=”,”>  </s:generator>  <s:sort source=”#attr.cameras” id=”sortedCameras” comparator=”myComparator”>  </s:sort>  <s:iterator value=”#attr.sortedCameras”>  <s:property/>  </s:iterator> |

* Khi đó ta được kết quả sau:



1. **subset tag**

* **subset** tag dùng để tạo một mảng con từ một iterator. Gồm các attribute sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Mô tả |
| count | Integer |  | Số lượng phần tử trong mảng kết quả |
| decider | Decider |  | Dùng để quyết định có thêm phần tử nào thỏa mãn vào mảng kết quả |
| source | String |  | Iterator ban đầu |
| start | Integer |  | Vị trí bắt đầu của iterator ban đầu để thêm vào mảng kết quả |
| var | String |  | Biến dùng để chứa mảng kết quả |

* Ví dụ, class SubsetTestAction ở **Listing 6.22** có một **Decider**, dùng để khi xài **subsettag** để quyết định thêm vào mảng con các phần tử có độ dài kí tự lớn hơn 4. Trang Subset.jsp ở **Listing 6.23** thể hiện **subsetag** có sử dụng **Decider**

**Listing 6.22 Class SubsetTestAction**

|  |
| --- |
| package demo;  import java.apache.struts2.util.SubsetIteratorFilter;  import com.opensymphony.xwork2.ActionSupport;  public class SubsetTestAction extends ActionSupport {  public SubsetIteratorFilter.Decider getMyDecider() {  return new SubsetIteratorFilter.Decider() {  public boolean decide (object o1) {  return o1.toString().length() > 4;  }  };  }  } |

**Listing 6.23 Trang Subset.jsp**

|  |
| --- |
| ….  <h4>Computers</h4>  <s:generator id=”computers”  val=”%{‘HP, Dell, Asus, Fujitsu, Toshiba’ }” separator=”,”>  </s:generator>  <s:subset source=”#attr.computers” decider=”myDecider”>  <s:iterator>  <s:property/>  </s:iterator>  </s:subset>  … |

* Khi đó ta được kết quả:



1. **Tóm lại**

Struts tag library cung cấp non-UI tags, còn gọi là generic tags. Các tags này được phân biệt thành data tags và control tags, mà mỗi tag trong 2 nhóm này đã được giới thiệu ở chương này.