



東京ITスクール

TOKYO IT SCHOOL

Struts タグライブラリ

目次

1. カスタムタグライブラリ	1
2. HTML タグライブラリ	3
3. Bean タグライブラリ	12
4. Logic タグライブラリ	21

1. カスタムタグライブラリ

1 カスタムタグライブラリとは

Struts の特徴として強力なカスタムタグライブラリが挙げられます。Struts では、標準で用意されているさまざまなカスタムタグライブラリを使用することで、JSP からスクリプトレットをできるだけ排除することができます。これにより機能の実装を簡略化し、コードの視認性を高めます。Struts に用意されているカスタムタグライブラリの種類は以下のとおりです。

タグライブラリ	機能
Bean タグライブラリ	JavaBean の操作を行うタグライブラリです。
HTML タグライブラリ	HTML のタグを生成するためのタグライブラリです。
Logic タグライブラリ	JSP 内で条件分岐、繰り返し等の制御を行うタグライブラリです。
Tiles タグライブラリ	Tiles フレームワークにより、レイアウトをテンプレート化する場合に使用します。
Nested タグライブラリ	Bean タグライブラリや Logic タグライブラリの記述を簡略化します。

本資料では、カスタムタグの中でも特に使用頻度が高い Bean タグ、HTML タグ、Logic タグについて解説します。

2 タグライブラリを使用するための準備

カスタムタグを使用する場合は、TLD ファイルの取得と、web.xml に設定の記述が必要です。Struts が標準で用意しているカスタムタグの TLD ファイルは、Struts をインストールしたディレクトリの lib ディレクトリにありますので、これを適当なディレクトリにコピーします。

例として、TLD ファイルを WEB-INF の直下にコピーした際の、web.xml の記述は以下のようになります。

web.xml

```
...
<taglib>
  <taglib-uri>/tags/struts-bean</taglib-uri>
  <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
</taglib>

<taglib>
  <taglib-uri>/tags/struts-html</taglib-uri>
  <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
</taglib>

<taglib>
  <taglib-uri>/tags/struts-logic</taglib-uri>
```

```
<taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
</taglib>

<taglib>
  <taglib-uri>/tags/struts-nested</taglib-uri>
  <taglib-location>/WEB-INF/struts-nested.tld</taglib-location>
</taglib>

<taglib>
  <taglib-uri>/tags/struts-tiles</taglib-uri>
  <taglib-location>/WEB-INF/struts-tiles.tld</taglib-location>
</taglib>
</web-app>
```

あとはタグライブラリを使用する JSP で、その使用を宣言します。宣言は taglib ディレクティブによって行います。

```
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<%@ taglib uri="/tags/struts-nested" prefix="nested" %>
<%@ taglib uri="/tags/struts-tiles" prefix="tiles" %>
```

uri 属性には web.xml の taglib-uri 要素の内容で指定した URI を指定し、prefix 属性にはタグライブラリの種類を指定します。

以上でカスタムタグライブラリを使用する準備は完了です。

2. HTML タグライブラリ

1 HTML タグライブラリとは

HTML タグライブラリは、HTML 入力フォームや HTML ベースの一般的なユーザインターフェースを作成するのに役立つカスタムタグを提供します。これらのタグは Struts フレームワークと密接に関連しており、ActionForm と併用することにより、よりその効果が発揮されるようになっています。以下に、HTML タグライブラリに含まれるタグの一覧を示します。

HTML タグライブラリのタグ一覧


タグ名	説明	生成する HTML 要素
base	base 要素を生成します。	<base>
button	ボタンを生成します。	<input type="button">
cancel	キャンセルボタンを生成します。	<input type="submit">
checkbox	チェックボックスを生成します。	<input type="checkbox">
errors	エラーがあれば、累積したエラーメッセージを表示します。	
file	ファイル選択入力フィールドを生成します。	<input type="file">
form	入力フォームを定義します。	<form>
frame	フレーム要素を生成します。	<frame>
hidden	隠しフィールドを生成します。	<input type="hidden">
html	html 要素を生成します。	<html>
image	イメージボタンを生成します。	<input type="image">
img	img 要素を生成します。	
javascript	Validator によりロードされる検証ルールに基づいて、JavaScript により妥当性検証を生成します。	
link	アンカーやハイパーリンクを生成します。	<a>
messages	メッセージがあれば、累積したメッセージを表示します。	
multibox	チェックボックスを生成します。	<input type="checkbox">
option	option 要素を生成します。	<option>
options	option 要素のコレクションを生成します。	<option>
optionsCollection	option 要素のコレクションを生成します。	<option>
password	パスワード入力フィールドを生成します。	<input type="password">
radio	ラジオボタンを生成します。	<input type="radio">
reset	リセットボタンを生成します。	<input type="reset">

rewrite	URI を生成します。	
select	select 要素を生成します。	<select>
submit	サブミットボタンを生成します。	<input type="submit">
text	テキスト型の入力フィールドを生成します。	<input type="text">
textarea	テキストエリアを生成します。	<textarea>
xhtml	HTML タグを XHTML として生成します。	<xhtml>

これらのタグはいくつかの共通属性を持ちます。以下の表に HTML タグライブラリの多くのタグに共通する属性の一覧を示します。

HTML タグライブラリの共通属性

属性名	記述
accesskey	この要素に、フォーカスをただちに移すために使用するキーボード文字を指定します。
alt	この要素の代替テキストを指定します。
altKey	この要素の代替テキストに対応するメッセージリソースのキーを指定します。
disabled	この入力フィールドを無効にする場合、true にセットします。
onblur	この要素が入力フォーカスを失ったときに実行される JavaScript イベントハンドラを指定します。
onchange	この要素が入力フォーカスを失い、その値が変更されている場合に実行される JavaScript イベントハンドラを指定します。
onclick	この要素がクリックされたときに実行される JavaScript イベントハンドラを指定します。
ondblclick	この要素がダブルクリックされたときに実行される JavaScript イベントハンドラを指定します。
onfocus	この要素がフォーカスされたときに実行される JavaScript イベントハンドラを指定します。
onkeydown	この要素がフォーカスされて、キーが押し下げられたときに実行される JavaScript イベントハンドラを指定します。
onkeypress	この要素がフォーカスされて、キーが押し下げられて放されたときに実行される JavaScript イベントハンドラを指定します。
onkeyup	要素がフォーカスされて、キーが放されたときに実行される JavaScript イベントハンドラを指定します。
onmousedown	要素がマウスポインタの下にあり、マウスボタンが押し下げられたときに実行される JavaScript イベントハンドラを指定します。
onmousemove	要素がマウスポインタの下にあり、ポインタが移動したときに実行される JavaScript イベントハンドラを指定します。
onmouseout	要素がマウスポインタの下にあり、ポインタが要素から外に移動したときに実行される JavaScript イベントハンドラを指定します。
onmouseover	要素がマウスポインタの下になく、ポインタが要素内に移動したときに実

	行される JavaScript イベントハンドラを指定します。
onmouseup	要素がマウスポインタの下にあり、マウスボタンが放されたときに実行される JavaScript イベントハンドラを指定します。
style	要素に適用される CSS スタイルを指定します。
styleClass 	要素に適用される CSS スタイルシートのクラス名を指定します。
styleId	要素に割り当てられる識別子 ("id"属性) を指定します。
tabindex	タブインデックスを指定します。
title	要素の補足説明 (タイトル) を指定します。
titleKey	要素の補足説明 (タイトル) に対応するメッセージリソースのキーを指定します。

2 フォームを作成する

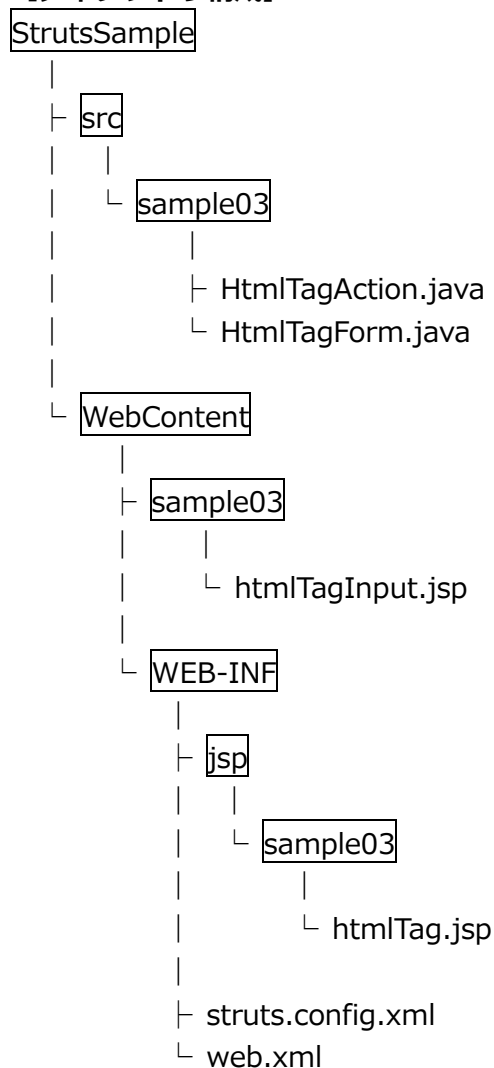
それでは、HTML タグの利用例としてフォームを作成してみましょう。HTML の form 要素を生成する際は、html:form タグを使用します。これにより ActionForm との連携が可能になります。

作成、および変更するファイルの一覧を以下に示します。

ファイル名	解説	ロケーション
struts-config.xml	Struts 設定ファイル	/WEB-INF/
HtmlTagAction.java	Action	/src/sample03/
HtmlTagForm.java	ActionForm	/src/sample03/
htmlTag.jsp	結果表示用 JSP	/WEB-INF/jsp/sample03/
htmlTagInput.jsp	入力フォーム	/sample03/

Eclipse 上から見たディレクトリ構成を以下に示します。

【ディレクトリ構成】



【htmlTagInput.jsp】

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html:html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>入力画面</title>
  </head>
  <body>
    <h2>入力画面</h2>
    <html:form method="POST" action="/sample03/htmlTag.do">
      ID : <html:text property="userId"></html:text><br/>
      パスワード : <html:password
property="password"></html:password><br/>
      性別 : <html:radio property="gender" value="0"></html:radio>男
<html:radio property="gender" value="1"></html:radio>女<br/>
      出身地 : <html:select property="birth">
        <html:option value=""></html:option>
        <html:option value="1">北海道・東北</html:option>
        <html:option value="2">関東</html:option>
        <html:option value="3">信越</html:option>
        <html:option value="4">北陸</html:option>
        <html:option value="5">東海</html:option>
        <html:option value="6">近畿</html:option>
        <html:option value="7">中国</html:option>
        <html:option value="8">四国</html:option>
        <html:option value="9">九州・沖縄</html:option>
      </html:select><br/>
      趣味 : <html:multibox property="hobbies" value="1"></html:multibox>旅行
<html:multibox property="hobbies" value="2"></html:multibox>買い物
<html:multibox property="hobbies" value="3"></html:multibox>スポーツ
<html:multibox property="hobbies" value="4"></html:multibox>音楽鑑賞
<html:multibox property="hobbies" value="5"></html:multibox>その他
<br/>
      備考 : <html:textarea property="memo"></html:textarea><br/>
      <html:submit>送信</html:submit>
    </html:form>
  </body>
</html>
```


【HtmlTagForm.java】

```
package sample03;

import org.apache.struts.action.ActionForm;

public class HtmlTagForm extends ActionForm {

    private String userId;
    private String password;
    private String gender;
    private String birth;
    private String[] hobbies;
    private String memo;

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getBirth() {
        return birth;
    }

    public void setBirth(String birth) {
```

```
        this.birth = birth;
    }

    public String[] getHobbies() {
        return hobbies;
    }

    public void setHobbies(String[] hobbies) {
        this.hobbies = hobbies;
    }

    public String getMemo() {
        return memo;
    }

    public void setMemo(String memo) {
        this.memo = memo;
    }
}
```

【HtmlTagAction.java】

```
package sample03;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class HtmlTagAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        return mapping.findForward("success");
    }
}
```

【htmlTag.jsp】

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>確認画面</title>
</head>
  <body>
    <h2>確認画面</h2>
    ID : <%= (String)request.getParameter("userId") %><br/>
    パスワード : <%= (String)request.getParameter("password") %><br/>
    性別 : <%= (String)request.getParameter("gender") %><br/>
    出身地 : <%= (String)request.getParameter("birth") %><br/>
    趣味 : <%= (String)request.getParameter("hobbies") %><br/>
    備考 : <%= (String)request.getParameter("memo") %><br/>
  </body>
</html>
```

【struts-config.xml】

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
  <form-beans>
    <form-bean name="sample01_loginForm" type="sample01.LoginForm"/>
    <form-bean name="sample02_loginDynaForm"
      type="org.apache.struts.action.DynaActionForm">
      <form-property name="id" type="java.lang.Integer" initial="0"/>
      <form-property name="password" type="java.lang.String" />
    </form-bean>
    <form-bean name="sample03_htmlTagForm"
type="sample03.HtmlTagForm"/>
  </form-beans>

  <action-mappings>
    <action path="/sample01/login"
      type="sample01.LoginAction"
      name="sample01_loginForm"
      scope="request">
```

```
<forward name="success" path="/WEB-INF/jsp/sample01/success.jsp"/>
</action>
<action path="/sample02/login"
        type="sample02.LoginAction"
        name="sample02_loginDynaForm"
        scope="request">
    <forward name="success" path="/WEB-INF/jsp/sample02/success.jsp"/>
</action>
<action path="/sample03/htmlTag"
        type="sample03.HtmlTagAction"
        name="sample03_htmlTagForm"
        scope="request">
    <forward name="success"
path="/WEB-INF/jsp/sample03/htmlTag.jsp"/>
    </action>
</action-mappings>
</struts-config>
```

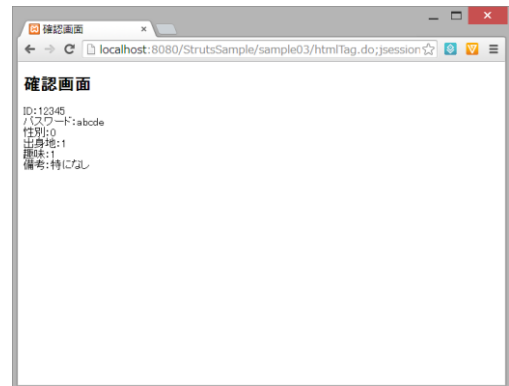
それでは、動作を確認しましょう。Tomcat を起動し、下記 URL をブラウザで表示します。

<http://localhost:8080/StrutsSample/sample03/htmlTagInput.jsp>

■ 入力画面



■ 確認結果



HTML タグを利用してフォームの表示、および入力データの送信が確認できました。

3. Bean タグライブラリ

1 Bean タグライブラリとは

Struts では、JavaBean の取り扱いを容易にする Bean タグライブラリが提供されています。Bean タグライブラリを利用することで、スコープに登録されている JavaBean が保持するデータへのアクセスが容易になります。また Cookie、リクエストヘッダ、リクエストパラメータの値に基づく新しい JavaBean を生成するメカニズムも提供されています。

2 Bean のプロパティ値を書き出す

スコープに登録されている JavaBean のプロパティ値を出力するには、bean:write タグを使用し、name 属性に Bean の登録名を、property 属性にプロパティ名を記述します。以下に例を示します。

```
<bean:write name="user" property="userName" scope="request"/>
```

例えば上の例の場合、user という名前で request スコープに登録されている JavaBean のプロパティ userName の値が出力されます。スクリプトレットを用いて下記のように記述した場合と、同様の結果となります。

```
<% User user = (User)request.getAttribute("user"); %>
<%= user.getUserName() %>
```

bean:write タグの属性を以下に示します。

bean:write タグの属性

属性	必須	説明
bundle		application スコープに登録されている MessageResources オブジェクトを識別する名前を指定します。 デフォルトは Globals.MESSAGES_KEY です。
filter		この属性が true の場合、出力される文字列が HTML 文書への出力に適した形に変換されます。「<」→「<」、「>」→「>」、「&」→「&」などの変換が行われます。デフォルトは true です。
format		この属性が指定されていると、ここで指定したフォーマットで文字列を出力します。もし指定されていない場合、formatKey に指定されたキーで、メッセージリソースからそのフォーマットが検索されます。
formatKey		出力フォーマットをメッセージリソースから検索するためのキーを指定します。

ignore		name 属性で指定した JavaBean が、scope 属性で指定したスコープに存在しない場合、この属性に true がセットされていると何も出力されません。false にセットされている場合は、例外がスローされます。デフォルトは false です。
locale		session スコープに登録されている Locale オブジェクトを識別する名前を指定します。デフォルトは Globals.LOCALE_KEY です。
name	○	スコープに登録されている JavaBean の名前を指定します。
property		name 属性で指定された JavaBean のプロパティを指定します。この属性が指定されている場合はこのプロパティ値が出力されますが、指定されていない場合は、JavaBean そのものの値が出力されます。
scope		name 属性で指定された JavaBean を検索するスコープを指定します。この属性が指定されない場合、page→request→session→application の順で検索されます。

3 スクリプティング変数を定義する

bean:define タグを使用すると、request スコープや session スコープに登録されている JavaBean のプロパティ値を、スクリプティング変数として定義することができます。同時にスコープへのプロパティ値の登録も行います。

使用の際は、name 属性に読み込む Bean の登録名、property 属性にプロパティ名、id 属性にスクリプティング変数名を記述します。

例を以下に示します。

```
<bean:define id="regDate" name="user" property="regDate"/>
```

例えば上の例の場合、スクリプティング変数 regDate に user.getRegDate() の値がセットされます。定義した変数は、以下のように読み込むことができます。

```
<%= regDate %>
```

また、JavaBean を参照せずに直接値を定義する場合は、name 属性、property 属性は記述せずに value 属性に直接値を記述します。

bean:define タグの属性を以下に示します。

bean:define タグの属性

属性	必須	説明
id	○	タグが生成するスクリプティング変数名を指定します。
name		スコープに登録されている JavaBean を識別する名前を指定します。
property		name 属性で指定された JavaBean のプロパティを指定します。この属性が指定されている場合はこのプロパティ値が定義されますが、指定されていない場合は、JavaBean そのものの値が定義されます。
scope		name 属性で指定された JavaBean を検索するスコープを指定します。この属性が指定されない場合、page→request→session→application の順で検索されます。
toScope		スクリプティング変数をどのスコープに登録するのかを指定します。page、request、session、application のいずれかを指定します。デフォルトは page スコープです。
type		生成されるスクリプティング変数に適用される完全限定名を指定します。デフォルトは java.lang.String か java.lang.Object です。

4 メッセージリソースの値を書き出す

Struts では多言語で書かれたメッセージを簡単に取得できるように、メッセージの内容が書かれたメッセージリソースファイルが提供されています。

メッセージリソースは「.properties」という拡張子で作成します。Struts への登録は、Struts 設定ファイルに message-resources タグを追加し、parameter 属性にリソースファイルのファイル名を記述することで可能となります。

以下に MessageResources.properties という名前のリソースファイルを登録する例を示します。

【struts-config.xml】

```
<struts-config>
  . . . (略) . . .
  <message-resources parameter="MessageResources"/>
</struts-config>
```

メッセージリソースファイルには、エントリというキーとメッセージの組み合わせを

(キー) = (メッセージ)

という形で記述します。アプリケーションは、このキーを元にしてメッセージを取得します。

bean:message タグを利用すると、メッセージリソースからメッセージを検索して、出力することができます。メッセージリソースのメッセージを検索するときに、直接その検索キーを指定することもできますし、スコープに登録されている JavaBean のプロパティを検索キーとすることもできます。

bean:message タグの属性を以下に示します。

bean:message タグの属性

属性	必須	説明
arg0 ~arg4		メッセージに渡すパラメータを指定します。
bundle		application スコープに登録されている MessageResources オブジェクトを識別する名前を指定します。 デフォルトは Globals.MESSAGES_KEY です。
key		目的のメッセージを検索するキーの値を指定します。対応する値がメッセージリソースに存在する必要があります。この属性を指定しない場合は、name 属性と property 属性から得られる値をキーとします。
locale		session スコープに登録されている Locale オブジェクトを識別する名前を指定します。 デフォルトは Globals.LOCALE_KEY です。
name		プロパティの値がメッセージを検索するキーとなる JavaBean の名前を指定します。プロパティが指定されていない場合 (property 属性が指定されていない場合)、この JavaBean そのものの値が検索キーとなります。

property		name 属性で指定された JavaBean のプロパティ名を指定し ます。
scope		name 属性で指定された JavaBean を検索するスコープを指定します。 この属性が指定されない場合、page→request→session→application の順で検索されます。

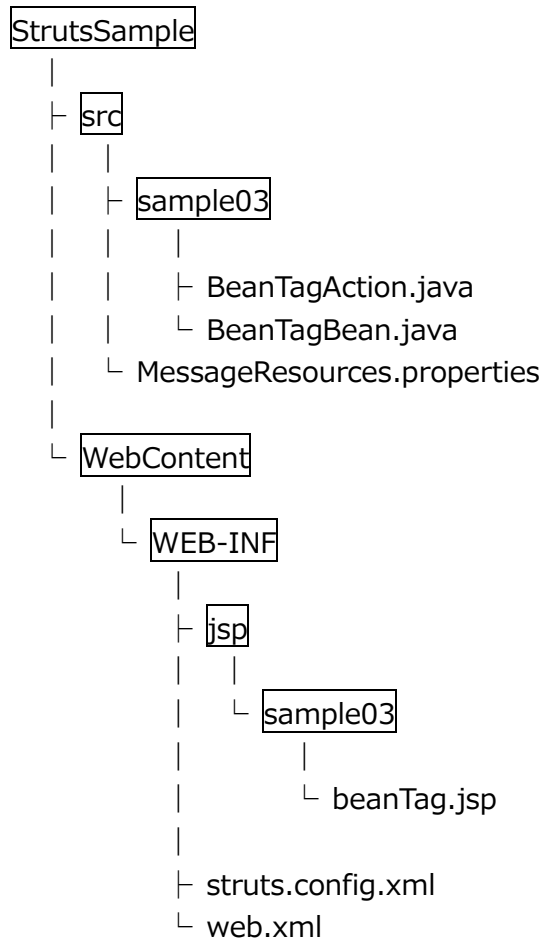
5 実際に Bean タグを利用する

それでは、ここまで紹介した Bean タグを利用してみましょう。作成、および変更するファイルの一覧を以下に示します。

ファイル名	解説	ロケーション
struts-config.xml	Struts 設定ファイル	/WEB-INF/
BeanTagAction.java	Action	/src/sample03/
BeanTagBean.java	JavaBean	/src/sample03/
beanTag.jsp	結果表示用 JSP	/WEB-INF/jsp/sample03/
MessageResources.properties	リソースファイル	/src/

Eclipse 上から見たディレクトリ構成を以下に示します。

【ディレクトリ構成】



【MessageResources.properties】

info=本日、10 時よりタイムセールを実施します

【BeanTagBean.java】

```
package sample03;

public class BeanTagBean {

    private String userId;
    private String userName;
    private String gender;

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
}
```

【BeanTagAction.java】

```
package sample03;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class BeanTagAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        BeanTagBean bean = new BeanTagBean();
        bean.setUserId("12345");
        bean.setUserName("システム太郎");
        bean.setGender("男性");
        request.setAttribute("user", bean);
        return mapping.findForward("success");
    }
}
```

【BeanTag.jsp】

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>表示画面</title>
</head>
<body>
    <h2>表示画面</h2>
    <p><bean:message key="info" /></p>
    ID : <bean:write name="user" property="userId" scope="request" /><br/>
    氏名 : <bean:write name="user" property="userName" scope="request" /><br/>
    性別 : <bean:write name="user" property="gender" scope="request" /><br/>
    登録日 : <bean:define id="regDate"
value="2014/10/10"></bean:define><%=regDate%>
```

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

【struts-config.xml】

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
    <form-beans>
        <form-bean name="sample01_loginForm" type="sample01.LoginForm"/>
        <form-bean name="sample02_loginDynaForm"
            type="org.apache.struts.action.DynaActionForm">
            <form-property name="id" type="java.lang.Integer" initial="0"/>
            <form-property name="password" type="java.lang.String" />
        </form-bean>
        <form-bean name="sample03_htmlTagForm"
type="sample03.HtmlTagForm"/>
    </form-beans>

    <action-mappings>
        <action path="/sample01/login"
            type="sample01.LoginAction"
            name="sample01_loginForm"
            scope="request">
            <forward name="success" path="/WEB-INF/jsp/sample01/success.jsp"/>
        </action>
        <action path="/sample02/login"
            type="sample02.LoginAction"
            name="sample02_loginDynaForm"
            scope="request">
            <forward name="success" path="/WEB-INF/jsp/sample02/success.jsp"/>
        </action>
        <action path="/sample03/htmlTag"
            type="sample03.HtmlTagAction"
            name="sample03_htmlTagForm"
            scope="request">
            <forward name="success"
path="/WEB-INF/jsp/sample03/htmlTag.jsp"/>
        </action>
        <action path="/sample03/beanTag"
            type="sample03.BeanTagAction"
```

```
scope="request">
  <forward name="success"
path="/WEB-INF/jsp/sample03/beanTag.jsp"/>
  </action>
</action-mappings>
<message-resources parameter="MessageResources"/>
</struts-config>
```

それでは、動作を確認しましょう。Tomcat を起動し、下記 URL をブラウザで表示します。

<http://localhost:8080/StrutsSample/sample03/beanTag.do>

■ 表示画面



Bean タグを利用して JavaBean の表示、スクリプティング変数の定義、リソースファイルの読み込みが確認できました。

4. Logic タグライブラリ

1 Logic タグライブラリとは

Logic タグライブラリは、スクリプトレットを使用することなく、タグで条件分岐や繰り返しなどの論理的な処理を実現するものです。

2 繰り返し処理を行う

コレクションや配列から要素を取り出し、繰り返し処理を記述するためのタグとして、`logic:iterate` タグがあります。スクリプティング変数に格納された要素は、このタグにネストされている範囲で使うことが可能です。`logic:iterate` タグで取り扱うことができるコレクションや配列は次のようなものです。

- Java オブジェクトかプリミティブ型の配列
- `java.util.Collection`、`java.util.Enumeration`、`java.util.Iterator`、`java.util.Map` のいずれかを実装したクラスのインスタンス

`bean:define` タグの属性を以下に示します。

`logic:iterate` タグの属性

属性名	必須	説明
<code>collection</code>	※	スクリプトレットでコレクションや配列を指定します。
<code>id</code>	○	コレクションや配列から取り出した要素を格納する変数名を指定します。この変数は page スコープに登録されます。
<code>indexId</code>		現在取り出している要素が、何回目に取り出したのかを格納する変数名を指定します。この変数は page スコープに登録されます。
<code>length</code>		取り出す要素の最大数を指定します。
<code>name</code>	※	スコープに登録されているコレクションや配列の名前、 <code>property</code> 属性が指定されている場合は、そのプロパティを持つ <code>JavaBean</code> の名前を指定します。
<code>offset</code>		コレクションや配列から要素を取り出す開始位置を指定します。指定しない場合は 0 となり、先頭から取り出されることになります。指定の方法は、数値を書くか、スコープに登録されている <code>java.lang.Integer</code> オブジェクトの名前を書きます。
<code>property</code>		<code>name</code> 属性で指定した <code>JavaBean</code> のプロパティ名を指定します。このプロパティにより得られるオブジェクトは、コレクションや配列でなければなりません。
<code>scope</code>		<code>name</code> 属性に指定されたオブジェクトをどのスコープから検索するのかを指定します。
<code>type</code>		取り出した要素のクラス名を完全限定名で指定します。

※いずれか 1 つ必須

取り扱うコレクションや配列を指定するには次の3つの方法があります。

- ・ collection 属性で、スコープに登録されているコレクションや配列を指定する
- ・ name 属性でスコープに登録されているコレクションや配列を指定する
- ・ name 属性と property 属性を指定して、スコープに登録されている JavaBean のコレクションや配列であるプロパティを指定する

例えば、1 から 5 まで数を順番に表示させたい場合は以下のように記述することができます。

```
<% int[] nums = {1, 2, 3, 4, 5};
    request.setAttribute("nums", nums); %>
<logic:iterate id="num" name="nums">
    <bean:write name="num"/><br>
</logic:iterate>
```

3 値を比較する

Logic タグライブラリには、変数の値を比較するために equal タグ、notEqual タグ、greaterEqual タグ、greaterThan タグ、lessEqual タグ、lessThan タグが提供されています。これらのタグはいずれも、リクエストパラメータの値（parameter 属性）やスコープに登録された JavaBean のプロパティ値（property 属性）と指定した値（value 属性）の間で比較を行い、true を返す場合にボディ部を評価します。

いずれのタグもその名前から推測できますが、どのような場合に true を返すのかを次の表に示します。

値を比較する Logic タグ

タグ名	説明
equal	対象となる変数の値が指定した値と等しいときに true を返す。
notEqual	対象となる変数の値が指定した値と等しくないときに true を返す。
greaterEqual	対象となる変数の値が指定した値より大きいか、等しいときに true を返す。
greaterThan	対象となる変数の値が指定した値より大きいときに true を返す。
lessEqual	対象となる変数の値が指定した値より小さいか、等しいときに true を返す。
lessThan	対象となる変数の値が指定した値より小さいときに true を返す。

これらのタグはいずれも同じ属性を持ちます。

値を比較する Logic タグの属性

属性名	必須	説明
cookie	※	比較する Cookie の名前を指定します。
header	※	比較する HTTP ヘッダの名前を指定します。
name	※	スコープに登録されている JavaBean を識別する名前を指定します。
parameter	※	比較するリクエストパラメータの名前を指定します。
property		name 属性で指定された JavaBean のプロパティを指定します。

scope		name 属性で指定された JavaBean をどのスコープから検索するのかを指定します。この属性が指定されていない場合、全スコープから検索されます。
value	○	比較の基準となる値を指定します。ここで指定する値は定数でなければなりません。

※いずれか 1 つ必須

これらのタグによる値の比較は、次に示すルールに基づいて行われます。

- value 属性で指定した値は、まず double 型か long 型に変換されます。その上で、数値としての比較が行われます。もしこの変換に失敗したとき、文字列として比較が行われます。
- cookie、header、name、parameter、property などの属性の値も、value 属性で指定した値と同じ変換が行われます。
- cookie、header、name、parameter、property などの属性の値が null の場合、比較が行われる前に、長さがゼロの文字列に変換されます。
- 文字列の比較は、java.lang.String クラスの compareTo メソッドの結果に基づいて行われます。

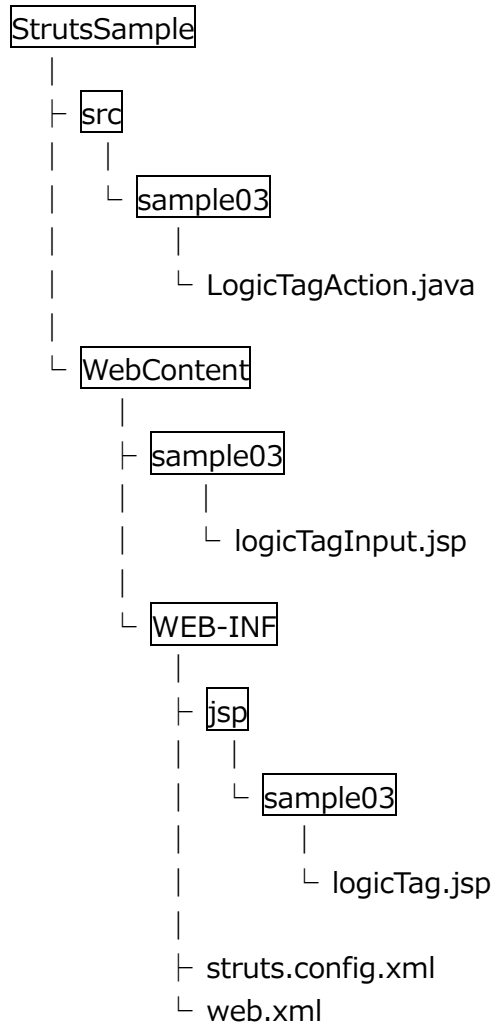
4 実際に Logic タグを利用する

それでは、ここまで紹介した Logic タグを利用してみましょう。作成、および変更するファイルの一覧を以下に示します。今回のサンプルは HTML タグのサンプルで使用した入力画面やアクションフォームを流用します。

ファイル名	解説	ロケーション
struts-config.xml	Struts 設定ファイル	/WEB-INF/
LogicTagAction.java	Action	/src/sample03/
LogicTag.jsp	結果表示用 JSP	/WEB-INF/jsp/sample03/
LogicTagInput.jsp	入力フォーム	/sample03/

Eclipse 上から見たディレクトリ構成を以下に示します。

【ディレクトリ構成】



【logicTagInput.jsp】

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html:html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>入力画面</title>
  </head>
  <body>
    <h2>入力画面</h2>
    <html:form method="POST" action="/sample03/logicTag.do">
      ID : <html:text property="userId"></html:text><br/>
      パスワード : <html:password
property="password"></html:password><br/>
      性別 : <html:radio property="gender" value="0"></html:radio>男
<html:radio property="gender" value="1"></html:radio>女<br/>
      出身地 : <html:select property="birth">
        <html:option value=""></html:option>
        <html:option value="1">北海道・東北</html:option>
        <html:option value="2">関東</html:option>
        <html:option value="3">信越</html:option>
        <html:option value="4">北陸</html:option>
        <html:option value="5">東海</html:option>
        <html:option value="6">近畿</html:option>
        <html:option value="7">中国</html:option>
        <html:option value="8">四国</html:option>
        <html:option value="9">九州・沖縄</html:option>
      </html:select><br/>
      趣味 : <html:multibox property="hobbies" value="1"></html:multibox>旅
行
        <html:multibox property="hobbies" value="2"></html:multibox>買い物
        <html:multibox property="hobbies" value="3"></html:multibox>スポーツ
        <html:multibox property="hobbies" value="4"></html:multibox>音楽鑑賞
        <html:multibox property="hobbies" value="5"></html:multibox>その他
      <br/>
      備考 : <html:textarea property="memo"></html:textarea><br/>
      <html:submit>送信</html:submit>
    </html:form>
  </body>
</html>
```

【LogicTagAction.java】

```
package sample03;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class LogicTagAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        HtmlTagForm htmlTagForm = (HtmlTagForm) form;
        request.setAttribute("hobbies", (String[])htmlTagForm.getHobbies());
        return mapping.findForward("success");
    }
}
```

【LogicTag.jsp】

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>確認画面</title>
</head>
<body>
<h2>確認画面</h2>
ID : <%= (String)request.getParameter("userId") %><br/>
パスワード : <%= (String)request.getParameter("password") %><br/>
性別 : <logic:equal parameter="gender" value="0">男性</logic:equal>
<logic:equal parameter="gender" value="1">女性</logic:equal><br/>
出身地 : <logic:equal parameter="birth" value="1">北海道・東北</logic:equal>
<logic:equal parameter="birth" value="2">関東</logic:equal>
<logic:equal parameter="birth" value="3">信越</logic:equal>
<logic:equal parameter="birth" value="4">北陸</logic:equal>
```

```
<logic:equal parameter="birth" value="5">東海</logic:equal>
<logic:equal parameter="birth" value="6">近畿</logic:equal>
<logic:equal parameter="birth" value="7">中国</logic:equal>
<logic:equal parameter="birth" value="8">四国</logic:equal>
<logic:equal parameter="birth" value="9">九州・沖縄</logic:equal> <br/>
趣味 : <logic:iterate id="hobby" name="hobbies">
  <logic:equal name="hobby" value="1">旅行</logic:equal>
  <logic:equal name="hobby" value="2">買い物</logic:equal>
  <logic:equal name="hobby" value="3">スポーツ</logic:equal>
  <logic:equal name="hobby" value="4">音楽鑑賞</logic:equal>
  <logic:equal name="hobby" value="5">その他</logic:equal>
</logic:iterate> <br/>
備考 : <%= (String)request.getParameter("memo") %> <br/>
</body>
</html>
```

【struts-config.xml】

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
  "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
  <form-beans>
    <form-bean name="sample01_loginForm" type="sample01.LoginForm"/>
    <form-bean name="sample02_loginDynaForm"
      type="org.apache.struts.action.DynaActionForm">
      <form-property name="id" type="java.lang.Integer" initial="0"/>
      <form-property name="password" type="java.lang.String" />
    </form-bean>
    <form-bean name="sample03_htmlTagForm"
type="sample03.HtmlTagForm"/>
  </form-beans>

  <action-mappings>
    <action path="/sample01/login"
      type="sample01.LoginAction"
      name="sample01_loginForm"
      scope="request">
      <forward name="success" path="/WEB-INF/jsp/sample01/success.jsp"/>
    </action>
    <action path="/sample02/login"
      type="sample02.LoginAction"
```

```

        name="sample02_loginDynaForm"
        scope="request">
        <forward name="success" path="/WEB-INF/jsp/sample02/success.jsp"/>
    </action>
    <action path="/sample03/htmlTag"
        type="sample03.HtmlTagAction"
        name="sample03_htmlTagForm"
        scope="request">
        <forward name="success"
path="/WEB-INF/jsp/sample03/htmlTag.jsp"/>
    </action>
    <action path="/sample03/beanTag"
        type="sample03.BeanTagAction"
        scope="request">
        <forward name="success"
path="/WEB-INF/jsp/sample03/beanTag.jsp"/>
    </action>
    <action path="/sample03/logicTag"
        type="sample03.LogicTagAction"
        name="sample03_htmlTagForm"
        scope="request">
        <forward name="success"
path="/WEB-INF/jsp/sample03/logicTag.jsp"/>
    </action>
</action-mappings>
<message-resources parameter="MessageResources"/>
</struts-config>

```

それでは、動作を確認しましょう。Tomcat を起動し、下記 URL をブラウザで表示します。

<http://localhost:8080/StrutsSample/sample03/logicTagInput.jsp>

■入力画面



■確認結果



Logic タグを利用して JSP 上での条件分岐や繰り返し処理が確認できました。