JSPベスト・プラクティス: JSPタグのカスタム属性を作る

ユーザー定義の属性でタグの機能を拡張する

Brett D. McLaughlin, Sr.
著作家
O'Reilly Media, Inc.

2003年 8月 05日

Brett McLaughlin氏のJSPベスト・プラクティス・シリーズ 今回は、カスタムのタイム・スタンプ・タグ（`lastModified`）を属性で拡張する方法について説明します。この属性は設計者独自のタイム・スタンプ・フォーマット設定を可能にするものです。

前回はJSPページでカスタム・タグ・ライブラリを使う上での基本を説明しました。簡単なタグの定義方法を学習し、次にタグ・ライブラリ・ディスクリプタ（TLD）により、他のJSP設計者がそのタグを使えるようにする方法について学習してきたわけです。今週はそのカスタム・タグに関する知識を元に、さらに説明を追加していきます。前回、例に使ったタグは限界まで単純化したものでしたが、今回はカスタム属性を導入することで、その機能を拡張してみます。

例についての注記:今回の記事のコードは前回作った、`lastModified`タグの上に構築していきます。前回の記事の作業が完了していないようでしたら、今回の作業を始める前に前回に戻り、セットアップをまず完了させてください。

"Hello, world"をカスタム化する

JSPタグに対する最も普通の要求は、ページ（つまりページ設計者）からのデータを受け取り、そのデータに対応できなければならない、ということです。タグ属性 で、この機能をカスタム・タグに追加することができるようになります。

非常に簡単な例として、おなじみの"Hello, world"アプリケーションを考えてみましょう。このスクリプトレットの機能を実行するカスタム・タグはごく簡単に思いつくと思いますが、では機能を拡張するのはどうでしょう。

リスト1はJSPページの一部で、"Hello, world!"タグを追加する部分です。おなじみのタグですが、`name`という属性を持っています。

リスト１　単純な"Hello, world!"タグ

```
<p>
    <examples:hello name="Reader" />
</p>
```

name属性は、ページ設計者がhelloタグにデータを入れられるような空間を生成します。この場合では、アプリケーションがメッセージを渡す、その相手の名前になります。つまり、これで"Hello, world"をカスタマイズしたわけです。・・でもどうしてそうなるんでしょう？　リスト2はhelloタグを実行するJavaコードを示します。

リスト2　helloタグのコード

```
package com.ibm.examples;
import java.io.IOException;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
public class HelloTag extends TagSupport {
    // The "person" to say hello to
    private String name;
    // Accept the attribute data
    public void setName(String name) {
      this.name = name;
    }
    public int doEndTag() {
      try {
   StringBuffer message = new StringBuffer("Hello, ");
   message.append(name)
          .append("!");
   pageContext.getOut().println(message.toString());
 } catch (IOException ignored) { }
 return EVAL_PAGE;
    }
```

リスト2のコードはその機能に比べて非常に簡単なものになっています。ここで追加したのはsetXXX()メソッドだけですが（XXXは属性名）、こうすることでこのタグの機能を大幅に拡張できたことになります。これでページ設計者は特定の目的のためにカスタム・データを設定し、必要に応じてそのデータを保存、操作し、有効化できるようになりました。doEndTag()メソッドで、タグ・データを自分の必要に応じてどのようにでも使えるようになるのです。

ではもう少し突っ込んで調べるために、lastModifiedタグに属性を追加したらどうなるかを見てみましょう。

lastModifiedを拡張する

ページ設計者に一つだけ表示オプションを与えるのではなく、彼らが好きなように出力（最終更新日付）のフォーマッティングができるようにしたいと思います。バックエンドから始めますが、lastModifiedTagクラスを拡張して出力フォーマッティングにjava.text.SimpleDateFormatを含むようにします。（リスト3）

リスト3　LastModifiedTagクラスでSimpleDateFormatを使う

```
package com.newInstance.site.tags;
import java.io.File;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.tagext.TagSupport;
public class LastModifiedTag extends TagSupport {
    private String format = "MMM d, yyyy";
    public int doEndTag() {
      try {
```

```
        HttpServletRequest request =
          (HttpServletRequest)pageContext.getRequest();
        String path = pageContext.getServletContext().getRealPath(
          request.getServletPath());
        File file = new File(path);
DateFormat formatter = new SimpleDateFormat(format);
        pageContext.getOut().println(
          formatter.format(new Date(file.lastModified())));
      } catch (IOException ignored) { }
      return EVAL_PAGE;
    }
}
```

リスト4ではタグ（フロントエンド）を属性とリンクすることで、タグに新しいフォーマッティング機能を追加しています。`format`属性の値が、リスト3で導入された新しい`format`メソッド変数に付加されていることに注意してください。

リスト4　新しい属性を処理する

```
package com.newInstance.site.tags;
import java.io.File;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.tagext.TagSupport;
public class LastModifiedTag extends TagSupport {
    private String format = "MMM d, yyyy";
    public void setFormat(String format) {
      this.format = format;
    }
    public int doEndTag() {
      try {
        HttpServletRequest request = (HttpServletRequest)pageContext.getRequest();
        String path = pageContext.getServletContext().getRealPath(
          request.getServletPath());
        File file = new File(path);
        DateFormat formatter = new SimpleDateFormat(format);
        pageContext.getOut().println(
          formatter.format(new Date(file.lastModified())));
      } catch (IOException ignored) { }
      return EVAL_PAGE;
    }
}
```

実行の詳細

`format`属性により、ページ設計者は日付 / 時間の出力フォーマットを好きなように設定できるようになります。ただし、この新しい属性を使えるようにするにはTLDファイル（WEB-INF/tldsディレクトリのsite-utils.tldのはずです）を、リスト5のように少し手直しする必要があります。

リスト5　TLDを手直しする

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE taglib

    PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2/EN"
           "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd" >
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.2</jsp-version>
    <short-name>site-utils</short-name>
    <uri>http://www.newInstance.com/taglibs/site-utils</uri>
    <tag>
      <name>lastModified</name>
      <tag-class>com.newInstance.site.tags.LastModifiedTag</tag-class>
      <body-content>empty</body-content>
<attribute>
        <name>format</name>
      </attribute>
    </tag>
</taglib>
```

デフォルト値

TLDが更新されたので、新しい属性が使えるようになったはずです。ではテストしてみましょう。新しいタグとTLDの変更が確実に反映されるように、サーブレット・コンテナを再起動し、`lastModified`タグのあるページを起動してみます。

動きました。タイム・スタンプが現れます。しかし読者が私と同じものを見ているなら、出力のフォーマッティングは前と同じままのはずです。これは新しい`format`に値がまだ入っていないので、前と同じものを見ているだけだからです。こんな結果でも無駄ではなく、このちょっとしたテスト動作で、`format`属性にデフォルトの値（この例では、今読者が見ているフォーマット）を設定しておくことの大事さが分かっていただけると思います。

カスタム属性にデフォルト値を設定しておくのは良い考えと言えます。ページ設計者が自分で値を設定したくなければしないでも大丈夫だからです。ページ設計者がデフォルト値でも構わないと思う場合もあるでしょうし、新しい属性やフォーマットを決めるまでに単純に時間がかかり、決めるまでの暫定値としてデフォルト値が入っているのは便利なものです。いずれにせよ、（タグにカスタム属性を付加することで）オプションを用意しておくことや機能ページにデフォルトの振る舞いを用意しておくのは、プログラミングする上で大事な習慣と言えます。

まとめ

もちろんこんなことを色々、わけ無しにやってきた訳ではありません。リスト6は元のタイム・スタンプの例（タイム・スタンプの力参照）のfooter.jspですが、`format`属性に値が入っています。

リスト6　format属性を使う

```
<%@ taglib prefix="site-utils"
             uri="http://www.newInstance.com/taglibs/site-utils"%>
          </td>
          <td width="16" align="left" valign="top"> </td>
    </tr>
    <!-- End main content -->
<!-- Begin footer section -->
    <tr>
      <td width="91" align="left" valign="top" bgcolor="#330066"> </td>
      <td align="left" valign="top"> </td>
      <td class="footer" align="left" valign="top"><div align="center"><br>
          &copy; 2003 <a href="mailto:webmaster@newInstance.com">Brett McLaughlin</a><br>
          Last Updated: <site-utils:lastModified
format="HH:mm a zz :: MM/dd/yyyy"/>
        </div></td>
          <td align="left" valign="top"> </td>
      <td width="141" align="right" valign="top" bgcolor="#330066"> </td>
    </tr>
</table>
<!-- End footer section -->
```

カスタム・タグの属性の扱いに関して順調に滑り出しましたが、まだほんの表面をなぞったにすぎません。次回はカスタムで拡張可能なタイム・スタンプ・タグを作る上で読者が突き当たる可能性のある、より複雑な状況、例えばエラー処理、異なる時間帯への対応などを見ていく予定です。それまで、違う形式の機能に対するカスタム属性で練習してみてください。ではまたお会いしましょう。

著者について

Brett D. McLaughlin, Sr.

Brett McLaughlin氏は、Logo (小さな三角形を覚えていますか?) の時代からコンピューターの仕事をしています。現在の専門は、JavaおよびJava関連のテクノロジーを使ったアプリケーション・インフラストラクチャーの構築です。ここ数年は、Nextel Communications and Allegiance Telecom, Inc. でこれらのインフラストラクチャーの実装に携わっています。Brett氏は、Javaサーブレットを使ってWebアプリケーション開発のための再利用可能なコンポーネント・アーキテクチャーを構築するJava Apacheプロジェクトturbineの共同設立者の1人です。同氏はまた、オープン・ソースのEJBアプリケーション・サーバーであるEJBossプロジェクトと、オープン・ソースのXML Web公開エンジンであるCocoonにも貢献しています。