JSPベスト・プラクティス: taglib入門

JSPコード改良してスクリプトレットをカスタム・タグに変換する

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

2003年 7月 23日

スクリプトレットを使用すればコーディングは簡単ですが、コードは分かりづらくなります。しかし長期的にみれば、JSPページは分かりやすいソリューションであるべきです。JSPベスト・プラクティスの今回の記事では、Brett McLaughlin氏が、スクリプトレットをJSPのカスタム・タグへ変換する方法、カスタム・タグをカスタム・タグ・ライブラリーへ追加する方法、JSPの開発でカスタム・タグを使用する方法について説明いたします。

JSPベスト・プラクティス の前回の記事では、JavaServer Pages(JSP)ファイルに最終更新日付を加えるスクリプトレット・ベースのテクニックを学習しました。残念ながら、スクリプトレットは短期的な観点からは利益をもたらしますが、長期的に見ると複雑さをもたらすことになりがちです。Javaコードが加えられたHTMLは、デバッグやオーサリングが困難になります。また、スクリプトレットは再利用可能ではないため、開発者はしばしばJSPページ間でカット&ペーストをおこなうようになり、複数のバージョンで同一コードを使用するという事態に陥りがちです。さらに、JSPページはスクリプト・エラーをはくための明確な方法を持っていないため、エラー報告も困難になっています。

したがって、今回はソリューションの改良を考えてみることにしましょう。この記事で、スクリプトレットをカスタム・タグに変換し、JSPでの開発プロジェクトでカスタム・タグを使用するための基本を学んでください。

なぜtaglib？

Atag library はJSPページで使用することができるタグ・ライブラリーです。JSPコンテナーは小さなデフォルト・タグ・ライブラリーを備えています。custom tag libraryは特定の用途もしくは目的のためにまとめられたライブラリーです。チームで共同に作業している開発者は、継続して使われている共通タグ・ライブラリーだけでなく、個々のプロジェクト向けのカスタム・タグ・ライブラリーを作成することでしょう。

JSPタグをスクリプトレットに代えることで、スクリプトレット共にもたらされていた頭痛を和らげることができます。JSPタグとは次のようなものです。

```
<store:shoppingCart id="1097629"/>
```

もしくはこちら。

```
<tools:usageGraph />
```

タグはそれぞれJavaクラスを参照しますが、コードはページや、コンパイル済みのクラス・ファイルに存在します。

スクリプトレットからタグへ

カスタム・タグ作成の第一歩は、タグの使用用途、タグの呼び名、(もしあれば)タグの属性を決定することです。タイム・スタンプ・タグの場合、ニーズはかなり単純で、ページの最終更新日付を出力するシンプルなタグとなります。

属性は必要ないため、次にようになります。

```
<site-utils:lastModified />
```

タグの名前とプレフィックスは同じで、`site-utils`です。子エレメントがないということでエレメントのコンテンツは空になります。タグを定義したら、次のステップは振る舞いの実装です。

振る舞いの実装

タグの振る舞いを実装するための最初のステップは、前回の記事のスクリプトレット・コードをリスト1のようなJavaクラス(`LastModifiedTag`)へ変更することです。

リスト1. タイム・スタンプ・タグの作成

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

このメソッドのコードには見覚えがあるはずです。というのも、これは前回の記事で扱ったタイム・スタンプのコードと基本的には同じだからです。ユーザーの入力が必要なく、タグには属

性もネストされたコンテンツもないので、気をつけるべきメソッドは`doEndTag()`だけです。これは、タグがJSPページにコンテンツ(この場合最終更新日付)を出力するメソッドです。

リスト1の他の変更点は、ページ内のスクリプトレットよりもむしろJSPタグに関係しています。例えば、すべてのJSPタグはJSPタグに基本的なフレームワークを提供しているJSPクラス`javax.servlet.jsp.tagext.TagSupport`を継承するべきです。また、タグが`EVAL_PAGE`を返すことに気づかれたもしれません。`EVAL_PAGE`は、残りのページを処理するようにコンテナーに命じるあらかじめ定義されている数値定数です。他のオプションには`SKIP_PAGE`があります。これは、残りのページ処理を異常終了させます。ユーザーをフォワードまたはリダイレクトする際のような別ページに制御を移す場合は、`SKIP_PAGE`を使用するべきです。残りの詳細はタイム・スタンプに特有なものです。

次は、このクラスをコンパイルし、適切なパス階層内のWEB-INF/classesディレクトリーにLastModifiedTag.classファイルを配置します。このパスは、パッケージ名のドット(`.`)をスラッシュ(`/`)に変更したタグのパッケージ名と一致しているべきです。今回のケースでは、基本ディレクトリー・パス(WEB-INF/classes)に、階層のcom/newInstance/site/tagsをつなげたものとなります。もしタグがfoo.bar.tag.MyTagであれば、WEB-INF/classes/foo/bar/tagとなります。このパス階層により、タグをロードするためにクラスが呼び出される場合はWebコンテナーは必ずクラスを見つけることができます。

TLDの作成

次のステップは、tag library descriptor (TLD)ファイルの作成です。TLDはそれを使用するコンテナーとJSPページへのタグ・ライブラリーの定義です。リスト2は、1つのタグだけを含む標準的なTLDです。ライブラリーに、多くのタグを追加するほど、TLDファイルの長さと複雑さは増すことになります。

リスト2. タグ・ライブラリー記述子

```
<?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>
    </tag>
</taglib>
```

TLDファイルの最初の情報はタグ・ライブラリー全体に適用されます。ここでは、以下のものを定義しています。version(ライブラリーをしばしば修正している場合などに、ライブラリーのバージョンを指定しておくと役に立ちます)、タグ・ライブラリーが機能するためのJSP仕様のバージョン、ライブラーの推奨プレフィックス、ライブラリーが識別される URIです。プレフィックスとライブラリーURIを視覚的に簡単に関連付けできるよう、URIの一部としてプレフィックスの`short-name`を使用していることに注目してください。

残りの情報は## ・エレメントで表わされて、特定のタグに適用されます。タグの名前、(コンパイルされてからコンテナー・ロードの際に配置されるべき)タグのクラス、タグにネストされたコンテンツがあるかを指定します。この場合、ネストされたコンテンツはないので、「empty」値が使用されています。

このファイルを保存して、WEB-INF/tldsディレクトリー(コンテナーにこのディレクトリーを作成する必要があるかもしれません)にファイルを置きます。ライブラリーのURI、推奨プレフィックス、およびTLDファイル間の明瞭なリンクを作成し、ファイルをsite-utils.tldという名前にして再び保存します。ライブラリーを利用可能にするための最後のステップは、タグ・ライブラリーの使用を要求するJSPページのURIと特定ライブラリーの関連をWebアプリケーションに知らせることです。これはweb.xmlファイルで行われます。リスト3は非常に簡単なweb.xmlの一部で、タグ・ライブラリーをURIと関連付けています。

リスト3. URIとタグ・ライブラリーのリンク

```
    <taglib>
      <taglib-uri>http://www.newInstance.com/taglibs/site-utils</taglib-uri>
      <taglib-location>/WEB-INF/tlds/site-utils.tld</taglib-location>
    </taglib>
```

まとめ

もしこれまでのステップを行ってこられたのであれば、JSPページで新しいタグを参照することができるはずです。リスト4は、スクリプトレットやスクリプトレットがあるJSPページへの参照を行っていない新しく改善されたfooter.jspです。

リスト4. 新しいタグ・ライブラリーの使用

```
<%@ 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 />
        </div></td>
      <td align="left" valign="top"> </td>
      <td width="141" align="right" valign="top" bgcolor="#330066"> </td>
    </tr>
</table>
<!-- End footer section -->
```

以前の記事(「JSTLでJSPページを書き換える」、「Webサイトへコンテンツをインポートする」を参照)で、JSTLはどのように動作したのか確かめておけば、次のステップはクリアになることでしょう。web.xmlファイルのURIを使用してタグ・ライブラリーを参照し、それにプレフィックス(TLDのshort-nameが最適)を割り当ててから、他のJSPタグのようにライブラリーを使用しています。結果は、スクリプトレットが含まれているときと同じ動作を行うことができる分かりやすいJSPページとなっています。

次回は、今回の機能を拡張した、やや簡単なカスタム・タグについてです。では、次回をお楽しみに。

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。