JSPベスト・プラクティス:JavaBeansコンポーネントとJSPテクノロジーの組み合わせ

JavaBeansとJSPのパラメーターを使用したWebページ間のデータの受け渡し

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

2003年 5月 13日

Web設計者であるBrett McLaughlin氏が、JavaBeansコンポーネントとJSPテクノロジーを組み合わせてデータの保存やWebページ間のデータの受け渡しを行う方法を説明するとともに、より動的なサイトの設計につながる組み合わせについて詳しく説明します。

これまでのJSPベスト・プラクティス・シリーズでは、基本的なトピックを中心に話を進めてきました。前の2回の記事では、JSPのインクルードのしくみを使用してWebサイトやWebアプリケーションに外部コンテンツを組み込む方法を学習しました。具体的には、2種類の異なるincludeディレクティブとして、静的な`include`コマンドと動的な`jsp:include`タグについて学習しました。

これまでは、親ページ (これまでの例で取り上げたWebサイトのメイン・ページなど) とそこにインクルードされたコンテンツの間に情報伝達の機能を作成する必要はありませんでした。しかし、このようなシナリオは単純すぎます。現実のWebサイトやWebアプリケーション・インターフェースのプログラムを作成するときには、一般に親ページとそこにインクルードされたファイルとの間でデータの受け渡しを行う情報伝達のしくみが必要になります。たとえば、Webサイト上で、メイン・ページで指定されたタイトルやメッセージをヘッダーまたはフッター・ページに渡さなければならない場合があります。今回の記事では、ページから別のページへデータを渡す方法と、そのようにして渡されたデータを、インクルードされたページで使用する方法について説明します。

データを保存するためのJavaBeanコンポーネント

各ページに短い「スローガン」("Books: A shelf full of learning" や "CDs: Music worth listening to" など) とタイトルを表示するWebサイトを考えてみましょう。各ページのスローガンは親ページ (マスター・ページと呼ばれることもあります) で決定されますが、そのスローガンを出力するためのHTMLは、インクルードされるページであるヘッダーで処理されます。このシナリオどおりの動作を行うには、マスター・ページからヘッダーにスローガンを渡すことができ、ヘッダーはページ・タイトルを受け取って、要求に応じてそれを表示することができなければなりません。

必要なもの

このシリーズのすべてのベスト・プラクティスは、JavaServer Pagesテクノロジーを基盤としています。これらを実行するためには、JSP対応のWebコンテナーをローカル・マシンとテスト・サーバーのどちらかに設定する必要があります。また、JSPページのコードを作成するには、テキスト・エディターまたはIDEが必要です。

まず最初に必要なものは、受け渡しするデータを保存するための何らかのオブジェクトです。JavaBeansコンポーネントは、ちょうど (偶然ではありません) このような目的に適しており、JSPテクノロジーとの組み合わせにも最適です。このBeanでデータを処理するために必要なものは、accessorメソッドとmutatorメソッドだけです。他のJavaプログラミング経験からご存知かもしれませんが、`get()` はデータにアクセスすることからaccessorメソッドの1つであり、`set()` はデータを変化させることからmutatorメソッドの1つです。

リスト1は、このシナリオで必要となる種類のBeanのコードを示しています。`PageHeaderInfo` Beanには、Webサイトのページ・ヘッダーに関する情報が含まれています。

リスト1. PageHeaderInfo JavaBean

```
<![CDATA[
package com.newInstance.site.beans;
import java.io.Serializable;
public class PageHeaderInfo implements Serializable {
     /** The title of the page */
     private String pageTitle;
     /** The slogan of the page */
     private String pageSlogan;
     public String getPageTitle() {
       return pageTitle;
     }
     public void setPageTitle(String pageTitle) {
       this.pageTitle = pageTitle;
     }
     public String getPageSlogan() {
       return pageSlogan;
     }
     public void setPageSlogan(String pageSlogan) {
       this.pageSlogan = pageSlogan;
     }
}
]]>
```

最初の演習用に、このファイルをPageHeaderInfo.javaとして保存し、コンパイルします。次に、コンパイル結果のクラス・ファイル、PageHeaderInfo.classを、皆さんのWebアプリケーションのWEB-INF/classesディレクトリーに配置します。パッケージのパスも必ず含めるようにしてください。ただし、パッケージ名は自由に変更してかまいません。以下は、このコンパイルされたクラスのパスの例です。

```
$<TOMCAT-ROOT>/webapps/$<WEB-APP-NAME>/WEB-INF/classes/com/newInstance/
  site/beans/PageHeaderInfo.class
```

パスをこのように使用すると、Webアプリケーション内のサーブレット、JSPページ、および他のクラスでこのクラスを使用できるようになります。ここまでの手順が完了したら、`PageHeaderInfo` Beanにデータを設定して、さまざまなJSPページでその内容を取得することができます。

保存したデータの受け渡し

今回のWebサイトのシナリオでは、複数の異なるページにさまざまなスローガンを渡すためのコードは、ページ・ヘッダーに含まれます。前回の記事を思い出してください。ヘッダー(`header.jsp`) は`jsp:include` 要素によって管理されるインクルード・ファイルです。静的な`include` ディレクティブではなく、動的な`jsp:include` タグを使用しているため、ヘッダーへのデータの受け渡しは簡単です。このデータの受け渡し機能は、`jsp:param` 要素によって提供されます。この要素は、`jsp:include` 要素内にネストすることができます。リスト2は、サイトのメイン・ページ用の`header.jsp` ファイルにデータを渡すため、これらの要素がどのように使用されるかを示しています。

リスト2. JSPページ間のデータの受け渡し

```
<![CDATA[
<%@ page language="java" contentType="text/html" %>
<html>
<head>
     <title>newInstance.com</title>
     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
     <link href="/styles/default.css" rel="stylesheet" type="text/css" />
</head>
<body>
<jsp:include page="header.jsp" flush="true">
     <jsp:param name="pageTitle" value="newInstance.com"/>
     <jsp:param name="pageSlogan" value="Java and XML :: Turning theory into practice" />
</jsp:include>
<%@ include file="/navigation.jsp" %>
<jsp:include page="bookshelf.jsp" flush="true" />
<jsp:include page="/mt-blogs/index.jsp" flush="true" />
<%@ include file="/footer.jsp" %>
</body>
</html>
]]>
```

ご覧のように、ここではスローガンとしてタイトルが渡されています。

既にお気づきの方もいらっしゃると思いますが、JavaBeanコンポーネントはデータを受け取るためだけに使用されるので、データを渡すページを設定する際、このBeanはまだなくてもかまいません。ただし、私は、ある好ましい理由から、いつもBeanのコードを先に作成しています。それは、JSPパラメーターの名前がJavaBeanプロパティーの名前と一致しなければならないことから、Beanのコードを先に作成することによって、JSPページのコードを作成するときにこれらのパラメーター名を確実に正しく設定できるためです。

データの受け取り

JSPパラメーターとJavaBeanプロパティーのコードが作成されており、`header.jsp` ページにデータが渡されれば、このページでデータの受け取りを開始することができます。リスト3は、`header.jsp` ページを示しています。コードのほとんどはHTMLですが、JSPステートメントに注目してください。これらのJSPステートメントについては、コードを見た後で説明します。

リスト3. 他のJSPページへのデータの受け渡し

```
<![CDATA[
<!-- Begin header section -->
<%@ page language="java" contentType="text/html" %>
<jsp:useBean id="pageHeaderInfo"
```

```
class="com.newInstance.site.beans.PageHeaderInfo">
     <jsp:setProperty name="pageHeaderInfo" property="*" />
</jsp:useBean>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
     <tr>
       <td width="91" height="50" align="right" valign="top"
           bgcolor="#330066"><font color="#FFFFFF"><img
     src="/images/header-lions.gif" width="90" height="60"></font></td>
       <td colspan="3" align="left" valign="top"
           bgcolor="#000000"><table width="100%" height="60" border="0"
           cellpadding="0" cellspacing="0">
           <tr>
             <td width="261" rowspan="2"><img
         src="/images/header-title.gif" width="261" height="60"></td>
             <td class="pagetitle" width="249" height="55" align="right"
               valign="bottom"><jsp:getProperty name="pageHeaderInfo"
               property="pageSlogan"/></td>
             <td width="10" height="55"> </td>
           </tr>
           <tr>
             <td height="5"><img src="/images/spacer.gif" width="1"
               height="5"></td>
             <td height="5"><img src="/images/spacer.gif" width="1"
               height="5"></td>
           </tr>
         </table></td>
       <td width="141" bgcolor="#000000">
         <font color="#FFFFFF"> </font>
       </td>
     </tr>
<!-- End header section -->
]]>
```

コードの最初の部分は、このページがJSPページであることを示しています。その後、`jsp:useBean` タグにより、このページがPageHeaderInfo Beanにアクセスする必要があることが記述されています。`id` は、このJSPページで使用可能なBeanの別名を設定し、`class` はそのBeanの完全に修飾されたJavaクラス名を示します。この`jsp:useBean` タグの中には`jsp:setProperty` タグがネストされています。このタグは、該当のBean (別名で識別) に、任意の要求データを使用してすべての使用可能なプロパティーが設定されるように指定しています。これは、Bean内のすべてのプロパティー (`pageTitle` や`pageSlogan` など) について、一致する要求パラメーターが検索されることを意味します。これらの要求パラメーターは、Webブラウザーを使用してクライアントから渡されたものでも、このJSPページをインクルードする他のページから渡されたものでもかまいません。この例では、唯一の要求データは親ページで作成されたものです。この例のサイトでは、メイン・ページ (`index.jsp`) が`pageTitle` と`pageSlogan` を送信し、これらはそれぞれ "`newInstance.com`" と "Java and XML: Turning theory into practice" に設定されています。

Beanのプロパティーに値が設定されたら、このページでそのデータを使用することができます。`header.jsp` の場合、このようなデータの使用はこのページのずっと下の方で、`jsp:getProperty` タグを使用することによって発生します。`jsp:getProperty` は、データの取得先となるオブジェクトを示す`name` パラメーターと、そのオブジェクトのどのプロパティーをポーリングするかを示す`property` パラメーターを取ります。その後、プロパティーの値はページの出力に挿入され、それが親ページに挿入された結果、動的なページ・スローガンがシームレスに表示されます。このように、JSPページ間でのデータの受け渡しは、簡単に実行できます。1つのJSPページに好きなだけBeanを追加し、各Beanに無限の個数のプロパティーを持たせることができるため、非常に複雑な要求データにも対応することができます。

変更の管理

変更は、すべての開発者にとって大きな悩みの種です。Beanのコードを思い通りに作成し、すべてのプロパティーの設定が完了して、JSPページでそれらを使用する準備が整ったというときになって、アプリケーションやサイトの要件が変更されるということは避けられないことのように見えます。その変更によって新しいプロパティーを追加する必要が出てきた (変更の多くはそうです) 場合は、JavaBeanのソース・コードを編集、再コンパイルし、JSPがその新しいBeanクラスにアクセスできることを確認しなければなりません。しかし、場合によっては、そのような作業を実行せずに済むことがあります。そのページが、今は使用されなくなったプロパティーを保存および取得している (つまり、あるページがサイトのディレクトリーに存在していても、そのページ・コンポーネントの参照を止めている) 場合には、コードをそのままにしておくこともできます。実際、私にはこの経験があります。

私の個人的なWebページでは、HTMLサイトの冒頭部分を`header.jsp` ページで出力していました。当時、このページは、私のページの`head` の`title` 要素で使用するタイトルの組み込みに使用されていました。しかし、その後このしくみを変更し、今はもうヘッダーのJSPページでページ・タイトルを使用していません。しかし、`PageHeaderInfo` Beanからわざわざ`pageTitle` プロパティーを削除しませんでした。実際、ヘッダー・ページにタイトルを渡しているほとんどのJSPページから`jsp:param` 要素を削除することさえもしませんでした。私は、これを実行するまでもない作業だと判断し、データをそのままにしておいても何の害もないこと (加えて、いつかまたそのデータを必要とする日が来るかもしれないこと) を知っていました。このことから、もし皆さんが同じ状況に直面しても、心配はいりません。このような些細なことを想像して心配するよりも、素晴らしい新機能を追加することに時間を使う方がずっと楽しみです。

次回までに

JSPページ間でのデータの受け渡しに慣れてきたら、今度は皆さん自身で便利なJavaBeansを作成し、それを皆さんのサイトで実行することができるかどうか確認してみてください。これらのBeanを、`jsp:useBean`、`jsp:param`、および`jsp:get/setProperty` のしくみとともにいろいろ試してみることにより、何か素晴らしいものを作り出すことができるはずです。次回のベスト・プラクティスでは、JSPを使用してマスター・サイトに外部コンテンツを取り込む別の方法をご紹介します。JSTLタグは、私たちがよく知っているincludeタグのような動作で、JSPにさらに柔軟性と機能性を追加します。それでは、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。