JSPベスト・プラクティス: Webサイトへコンテンツをインポートする

外部サイトからコンテンツを取り込むために改善されたJSTLタグ

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

2003年 6月 17日

JSTLの`c:import`タグは、表面上はJSPのインクルード・メカニズムを模倣しています。しかし、`c:import`(あるいはJSTLタグのうちのいずれか)についてもう少し深く調べてみると、さらに付加的な機能に気づくことでしょう。パラメーターを渡し、ローカルファイルのコンテンツを操作することに加えて、`c:import`は外部サイトからコンテンツを取り込むためにも使用することができます。JSPベスト・プラクティスの今回の記事では、ベテランWeb開発者であるBrettMcLaughlin氏がこの方法について解説します。

JSPベスト・プラクティスの前回の記事は、JSTLの簡単な入門編でした。私は、WebコンテナにJSTLをロードする方法、JSTLタグを使用してJSPファイルを変換する方法についての説明をし、`c:import`タグについても簡単に紹介をしておきました。`jsp:include`と同様、`c:import`はウェブサイト上にコンテンツを載せるためにパラメーターを使用します。しかし、includeディレクティブとは異なり、`c:import`はその範囲をローカルファイルに限定していません。

今回は、`jsp:params`を`c:params`に変換する方法を説明します。この変換により、パラメーター渡しにimportタグを使ったり、includeタグで行ったかのようにローカル・コンテンツを操作できるようになります。この基本的な機能に加えて、さらに他のWebサイトからコンテンツを取り込むために`c:import`を使用する方法についても説明いたします。これにより、サイト上にファイルを取得すれば、簡単にサイトのルック・アンド・フィールに取り込むことが可能となります。

少し戸惑いを覚えますか?

JSPベスト・プラクティスシリーズを始めからお読みになられていない場合は、いくつかの項目は理解し難いかもしれません。リンクがあるほとんどのベスト・プラクティスや例は、最初の記事から現在のものへと継続してきているものです。リストのあるJSPベスト・プラクティス・シリーズを参照してみていただければ、すぐに内容を把握することができるでしょう。

JSTL変換再び

最新記事の「JSTLでJSPページを書き換える」の最後において、`jsp:include`を`c:import`に変更してWebサイトのメイン・ページを変換しました。しかしながら、私は、ヘッダーをインクルード

するコードを変更することを避けてきていました。それは、このヘッダー・コードにはあるリクエスト・データを渡すパラメーターが含まれているためです。要点の復習として、前回扱ったJSPページをリスト1に示します。

リスト1. 大部分が変換されたindexページ(jsp:includeエレメントに注意してください)

```
<%@ page language="java" contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<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=" " />
</jsp:include>
<%@ include file="/navigation.jsp" %>
<c:import url="bookshelf.jsp" />
<c:import url="/mt-blogs/index.jsp" />
<%@ include file="/footer.jsp" %>
</body>
</html>
```

パラメータ・タグの変換

`jsp:include`ではなく`c:import`を使用してindexファイルのほぼ全体を変換するために、最後にやるべき事は、一つだけ残っているパラメータ・パスの`jsp:include`エレメントを変換して、その動作を`c:import`タグにエミュレートすることです。リスト2に示されるように、行うべきことは`c:param`と`jsp:param`を交換することです。これでOKです。なぜなら、`c:param`はJSPにおける相当物(`jsp:param`)と同様に機能するからです。

完全にJSTLに変換されたリスト1のindex ファイルをリスト2に示します。

リスト2. 完全なJSTL変換

```
<%@ page language="java" contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<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>
<c:import url="header.jsp">
     <c:param name="pageTitle" value="newInstance.com"/>
     <c:param name="pageSlogan" value=" " />
</c:import>
<%@ include file="/navigation.jsp" %>
<c:import url="bookshelf.jsp" />
<c:import url="/mt-blogs/index.jsp" />
<%@ include file="/footer.jsp" %>
</body>
</html>
```

JSPページへのパラメーター渡しに関する詳細は、このシリーズの3番目の記事「JavaBeansコンポーネントとJSPテクノロジーの組み合わせ」を参照してください。c:paramおよびjsp:paramはほぼ同じように機能することを思い出してください。思い出せれば、結構です。では、c:importのさらにクリエイティブな用途について調べていきましょう。

外部コンテンツのインポート

c:importを使用することの実際の利点の1つは、外部のWebサイトあるいはWebアプリケーションからコンテンツを取り込むことができることにあります。まだjsp:includeについて学んでいた頃、静的なコンテンツを指定するためにfile属性を使用していたことにお気付きになられていたかもしれません。file属性は、単にその名前が意味することを行います。つまり、ローカルファイルのコンテンツを取り込むことができます。c:importに対応する属性はurlです。これもまた名前が示唆することを行います。つまり、任意のURLを取り込むことができます。ローカルファイルのコンテンツだけを備えたサイトページではなく、c:importによって任意のURLからコンテンツを取り込ませることができます。これは、自身のルック・アンド・フィールに他のサイトからのコンテンツを含める実に巧妙な方法です。

単純な例を考えてみましょう。ウェブサイトに豪華なMadagascar Rosewoodギターの写真を何枚か配置したいとします。イメージ・ファイルおよび相対リンクを備えた自分のページを構築することもできますが、別のサイトからコンテンツをインポートし、それを自分のサイトのルック・アンド・フィールに取り込むことは、もっと簡単なことです。リスト3は、c:importのurl属性を使用すればどれ程簡単にお気に入りのギターのサイトからイメージ・ファイルを読み込めるかを示しています。

リスト3.外部コンテンツのインポート

```
<%@ page language="java" contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<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>
<c:import url="header.jsp">
     <c:param name="pageTitle" value="newInstance.com :: True North Guitars"/>
     <c:param name="pageSlogan" value="...building it from scratch" />
</c:import>
<%@ include file="/navigation.jsp" %>
<c:import url="bookshelf.jsp" />
<c:import url="http://www.truenorthguitars.com/Clients/Richman/index.htm" />
<%@ include file="/footer.jsp" %>
</body>
</html>
```

このコードは十分なように見えます。しかし、これを実際に使ってみれば、すぐに問題があるのが分かるでしょう。イメージが全く表示されず、相対リンクもすべて失敗しています。もちろん、考えてみれば原因は明らかです。外部のリソース(この場合はイメージ・ファイル)が解釈され、出力ストリームに直接挿入された解釈の結果を見ているためです。そのため、/images/guitar-01-24.jpgのような外部イメージリンクは、見当たらないものとして表示されます。この

問題を解決するただ一つの方法は、サイトにイメージをコピーすることでしょう。これは単純なインポートと比較して、全く異なる(より多くの時間を消費する)テクニックです。

外部コンテンツのインポートが実際に意味をなすのは、コンテンツが純粋にテキストベースの場合のみです。例えば、ヘッダー、フッター、およびルック・アンド・フィールを備えた、システム管理者向けのサイトを考えてみてください。サイトの1ページにおいて、ユーザーがどのようにサイトを使用すべきかを知らせるREADMEファイルを掲載する方法を詳述します。次のように既存のFTPサイトの説明を記述します。

```
<c:import url="ftp://ftp.oreilly.com/pub/README.ftp" />
```

HTTPコンテンツを取り込むのと同じくらい、FTPサーバーのコンテンツをインポートするのは単純であることに注目してください。自分のサイトとターゲットサイトの双方が理解するプロトコルのように、HTTPSのコマンドを使用することができます。

なぜJSTLを使うのか？

JSTLについて知識を深めてください。

JSTLの入門 シリーズを参照してください
Part 1, 「式言語」（2003年、2月）
Part 2, 「核心(core)に触れる」（2003年、3月）
Part 3, 「プレゼンテーションがすべて」（2003年、4月）
Part 4, 「SQLおよびXMLコンテンツへのアクセス」（2003年、5月）

JSPのコア・タグを超えてJSTLを使用することには、いくつかの大きな利点があります。第一に、現在JSTLはJSPタグとは別の仕様で定義されています。これは、JSPの仕様が変更されても、JSTLタグは整合性を保つであろうことを保証しています。JSPコンテナの変化によってもたらされる影響に対処する余裕がないアプリケーションは、JSTLの仕様により利益を得ることでしょう。第二に、JSTLタグは、JSPのコア・タグの能力以上の機能を提供しています。今回のベスト・プラクティスで見てきたように、c:importは、複雑なコンテンツのみならず他のサイトからURLをインポートすることができます。これはjsp:includeタグでは不可能なことです。そして最後に、JSTLはそれ自身の式言語(通常ELと略されます)を提供しています。JSTLのELはJSPコードを書く際にかなりの柔軟性を提供し、コアJSP言語には大きな追加機能です。

私たちは今回1つのJSTLタグを学びましたが、みなさんが知っておくべきタグはまだたくさんあります。JSTLの詳細に関心がある方は、参考文献を参照してください。みなさんが知識を増やされている間、私は次回の記事へと取り組んでいることでしょう。次回の記事では、JSPページへのタイムスタンプの追加を扱います。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。