# Apache チュートリアル: Server Side Includes 入門

サーバサイドインクルードによって、既存の HTML ドキュメントに動的なコンテンツを追加することができます。

## トピック



## はじめに

| 関連モジュール | 関連ディレクティブ |
|---|---|
| mod_include | Options |
| mod_cgi | XBitHack |
| mod_expires | AddType |
| | SetOutputFilter |
| | BrowserMatchNoCase |

この記事は、通常は単に SSI と呼ばれる Server Side Includes を扱います。この記事においては、サーバでの SSI を許可するための設定と、 現在の HTML ページに動的なコンテンツを加えるためのいくつかの基本的な SSI 技術を紹介します。

記事の後半では、SSI ディレクティブで SSI と共に実行することができる条件文のような幾分高度な事柄について述べています。

## SSI とは ?

SSI (Server Side Includes) は、HTML ページ中に配置されるディレクティブであり、 サーバでページを提供する時に評価されます。SSI は、CGI プログラムやその他の動的な技術で全てのページを提供せずに、 動的に生成されたコンテンツを現在の HTML ページに加えます。

どういう場合に SSI を使い、どういう場合にプログラムで ページを完全に生成するかは、ページのうちどの程度が静的であり、 ページが提供されるたびに再計算する必要がどの程度あるかで通常は決定します。 SSI は現在時刻のような小さい情報を加えるにはうってつけの方法です。 しかし、そのページのほとんどの部分が提供時に生成される場合は、 他の方法を探す必要があります。

## SSI を許可するためのサーバの設定

サーバで SSI を許可するには、httpd.conf ファイルまたは .htaccess ファイルに次のディレクティブを指定する必要があります:

```
Options +Includes
```

この指定は、ファイルを SSI ディレクティブで解析させることを許可するということを Apache に伝えます。ほとんどの設定ではお互いを上書きできる、複数の Options があることに 注意してください。おそらく、設定が最後に評価されることを 保証されるために、SSI を使用したいディレクトリに Options ディレクティブを適用する必要があるでしょう。

全てのファイルが SSI ディレクティブで解析されるというわけではありません。 どのファイルが解析されるかを Apache に伝える必要があります。 これを行なうには二つ方法があります。 次のディレクティブを使うことで、例えば .shtml のような特別なファイル拡張子を持つファイルを解析するよう Apache に伝えることができます:

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

この方法の欠点は、もし現在のページに SSI ディレクティブを加えたい場合、 それらのディレクティブが実行されるように .shtml 拡張子にするため、そのページの名前と、 そのページへの全てのリンクを変更しなければならないことです。

もう一つの方法は、XBitHack ディレクティブを使用することです:

```
XBitHack on
```

XBitHack は、ファイルの実行ビットが立っている場合、 SSI ディレクティブにより解析することを Apache に伝えます。 従って、SSI ディレクティブを現在のページに加えるためには、 ファイル名を変更しなくてもよく、単に chmod を使用してファイルを実行可能にするだけで済みます。

```
chmod +x pagename.html
```

行なうべきではないことに関する短いコメント。時々誰かが、全ての .html ファイルを SSI で解析するよう Apache に伝えれば、 わざわざ .shtml というファイル名にする必要がないといって 薦めるのを見ることでしょう。こういう人たちは、おそらく XBitHack について聞いたことがないのでしょう。 この方法について注意することは、たとえ SSI ディレクティブを全く含まない場合でも、Apache がクライアントに 送る全てのファイルを最後まで読み込ませることになります。 この方法はかなり処理を遅くするものであり、良くないアイデアです。

もちろん、Windows ではそのような実行ビットをセット するようなものはありませんのでオプションが少し制限されています。

デフォルトの設定では、Apache は SSI ページについて最終変更時刻や コンテンツの長さを HTTP ヘッダに送りません。 動的なコンテンツであるため、それらの値を計算するのが難しいからです。 このためドキュメントがキャッシュされなくなり、 結果としてクライアントの性能が遅くなったように感じさせることになります。 これを解決する方法が二つあります:

1. XBitHack Full 設定を使用する。 この設定により、もともと要求されたファイルの時刻を参照し、 読み込まれるファイルの変更時刻を無視して最終変更時刻を決定するよう

Apache に伝えます。

2. mod_expires [1] で提供されているディレクティブを使用して、 ファイルが無効になる時刻を明示します。これにより、 ブラウザとプロキシにキャッシュが有効であることを通知します。

## 基本的な SSI ディレクティブ

SSI ディレクティブは以下の文法で記述します:

```
<!--#element attribute=value attribute=value ... -->
```

HTML のコメントのような書式をしているので、もし SSI を正しく動作可能にしなければ、ブラウザはそれを無視するでしょう。 しかし、HTML ソース中では見えます。もし SSI を正しく設定したなら、 ディレクティブはその結果と置き換えられます。

element はたくさんあるものから一つ指定することができます。 指定できるものの大多数については、次回もう少し詳しく説明します。 ここでは、SSI で行なうことができる例をいくつか示します。

### 今日の日付

```
<!--#echo var="DATE_LOCAL" -->
```

echo 要素は単に変数の値を出力します。 CGI プログラムに利用可能な環境変数の全てのセットを含む多くの標準変数があります。また、set 要素を用いることで、独自の変数を定義することができます。

出力される日付の書式が好きではない場合、その書式を修正するために、 config 要素に timefmt 属性を使用することができます。

```
<!--#config timefmt="%A %B %d, %Y" -->
Today is <!--#echo var="DATE_LOCAL" -->
```

### ファイルの変更日

```
This document last modified <!--#flastmod file="index.html" -->
```

この要素も timefmt フォーマットの設定に従います。

### CGI プログラムの結果を取り込む

これは、全ての人のお気に入りである ``ヒットカウンタ'' のような CGI プログラムの結果を出力する SSI のより一般的な使用のうちの一つです。

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```

## 追加の例

以下は、SSI を使用して HTML ドキュメントにおいてできることのいくつかの特別な例です。

### いつこのドキュメントは修正されたのか ?

先に、ドキュメントが最後に変更されたのはいつかを ユーザに通知するために SSI を使用することができることを述べました。 しかしながら、実際の方法は、いくぶん問題のままにしておきました。 HTML ドキュメントに配置された次のコードは、ページにそのようなタイムスタンプを入れるでしょう。もちろん、上述のように、 SSI を正しく動作可能にしておく必要があります。

```
<!--#config timefmt="%A %B %d, %Y" -->
This file last modified <!--#flastmod file="ssi.shtml" -->
```

もちろん、ssi.shtml の部分を実際の当該ファイル名と置き換える必要があります。 もし、あらゆるファイルに張ることができる一般的なコードを探しているなら、 これは不便であるかもしれません。おそらくその場合は、 そうする代わりに変数 LAST_MODIFIED を使用したいと考えるでしょう:

```
<!--#config timefmt="%D" -->
This file last modified <!--#echo var="LAST_MODIFIED" -->
```

timefmt 書式についてのより詳細については、お好みの検索サイトに行き、 strftime で検索してみてください。文法は同じです。

### 標準のフッタを挿入する

もし数ページを超えるページを持つサイトを管理しているならば、 全ページに対して変項を行なうことが本当に苦痛となり得ることが 分かるでしょう。全てのページに渡ってある種の標準的な外観を 維持しようとしているならば特にそうでしょう。

ヘッダやフッタ用の挿入用ファイルを使用することで、 このような更新にかかる負担を減らすことができます。 一つのフッタファイルを作成し、それを include SSI コマンドで各ページに入れるだけで済みます。include 要素は、file 属性または virtual 属性のいずれかを使用してどのファイルを挿入するかを決めることができます。 file 属性は、カレントディレクトリからの相対パスで示された ファイルパスです。 それは / で始まる絶対ファイルパスにはできず、また、そのパスの一部に ../ を含むことができないことを意味します。virtual 属性は、おそらくより便利だと思いますが、提供するドキュメントからの相対 URL で指定すべきです。それは / で始めることができますが、 提供するファイルと同じサーバ上に存在しなくてはなりません。

```
<!--#include virtual="/footer.html" -->
```

私は最後の二つを組み合わせて、LAST_MODIFIED ディレクティブをフッタファイルの中に置くことがよくあります。 SSI ディレクティブは、挿入用のファイルに含ませたり、 挿入ファイルのネストをしたりすることができます。すなわち、 挿入用のファイルは他のファイルを再帰的に挿入することができます。

## 他に何が設定できるのか ?

時刻書式を config で設定できることに加えて、 更に二つ config で設定することができます。

通常、SSI ディレクティブで何かがうまくいかないときは、 次のメッセージが出力されます。

```
[an error occurred while processing this directive]
```

このメッセージを他のものにしたい場合、config 要素の errmsg 属性で変更することができます:

```
<!--#config errmsg="[It appears that you don't know how to use SSI]" -->
```

おそらく、エンドユーザはこのメッセージを決して見ることはありません。 なぜなら、そのサイトが生きた状態になる前に SSI ディレクティブに関する 全ての問題を解決しているはずだからです。(そうですよね?)

そして、config において sizefmt 属性を使用することで、 返されるファイルサイズの書式を設定することができます。 バイト数には bytes を、適当に Kb や Mb に短縮させるには abbrev を指定することができます。

## コマンドの実行

今後数ヶ月のうちに、小さな CGI プログラムと SSI を使用する記事を出したいと考えています。ここではそれとは別に、 exec 要素によって行なうことができることを示します。SSI にシェル (正確には /bin/sh。Win32 ならば DOS シェル) を使用してコマンドを実行させることができます。 下記の例では、ディレクトリリスト出力を行ないます。

```
<pre>
<!--#exec cmd="ls" -->
</pre>
```

Windows 上では、

```
<pre>
<!--#exec cmd="dir" -->
</pre>
```

Windows 上では、このディレクティブによっていくつかの奇妙な 書式に気づくでしょう。なぜなら dir の出力が文字列 ``<dir>'' を含み、ブラウザを混乱させるからです。

この機能は非常に危険であり、どんなコードでも exec タグに埋め込まれてしまえば実行することに注意してください。例えば `` ゲストブック '' のように、もし、 ユーザがページの内容を編集できる状況にあるならば、 この機能を確実に抑制してください。Options ディレクティブの IncludesNOEXEC 引数を指定することで、 SSI は許可するけれど exec 機能は許可しないようにすることができます。

## 高度な SSI テクニック

コンテンツを出力することに加え、Apache SSI は変数を設定し、 そして比較と条件分岐にその変数を使用できる機能を提供しています。

### 警告

この記事で述べた大部分の機能は、Apache 1.2 以降を使用している場合のみ利用可能です。もちろん、もし Apache 1.2 以降を使用してない場合、直ちにアップグレードする必要があります。 さぁ、今それを行ないなさい。それまで待っています。

## 変数を設定する

set ディレクティブを使用して、 後で使用するために変数を設定することができます。 これは後の説明で必要になるので、ここでそれについて述べています。 文法は以下のとおりです:

```
<!--#set var="name" value="Rich" -->
```

このように単純に文字どおりに設定することに加え、 例えば環境変数や前の記事で述べた変数（例えば LAST_MODIFIED のような）を含む他のあらゆる変数を値を設定するのに使用することができます。 変数名の前にドル記号（$）を使用することで、 それがリテラル文字列ではなくて変数であることを示します。

```
<!--#set var="modified" value="$LAST_MODIFIED" -->
```

ドル記号（$）を文字として変数の値に入れるには、 バックスラッシュによってドル記号をエスケープする必要があります。

```
<!--#set var="cost" value="¥$100" -->
```

最後になりますが、長い文字列の中に変数を置きたい場合で、 変数名が他の文字とぶつかる可能性があり、 それらの文字について混乱してしまう場合、この混乱を取り除くため、変数名を中括弧で囲むことができます（これについての良い例を示すのは難しいのですが、おそらく分かっていただけるでしょう）。

```
<!--#set var="date" value="${DATE_LOCAL}_${DATE_GMT}" -->
```

## 条件式

さて、変数を持っていて、 それらの値を設定して比較することができるのですから、 条件を表すためにそれらを使用することができます。これにより SSI はある種の小さなプログラミング言語になっています。 mod_include は条件を表現するために if, elif, else, endif 構造を提供しています。これによって、 一つの実際のページから複数の論理ページを効果的に生成することができます。

条件構造は以下のとおりです:

```
<!--#if expr="test_condition" -->
<!--#elif expr="test_condition" -->
<!--#else -->
```

```
<!--#endif -->
```

test_condition はあらゆる種類の論理的比較をすることができます。 値を比較したり、その値が ``真'' かどうかを評価します (空でないなら与えられた文字列は真です)。 利用可能な比較演算子の全てのリストについては、 mod_include ドキュメンテーションを参照してください。 ここでは、この構造をどう使用するかの例をいくつか示します。

設定ファイルで次の行を記述します:

```
BrowserMatchNoCase macintosh Mac
BrowserMatchNoCase MSIE InternetExplorer
```

これはクライアントが Macintosh 上でインターネットエクスプローラが動いている場合、環境変数 ``Mac'' と ``InternetExplorer'' を真と設定します。

次に、SSI が可能になったドキュメントで以下を行ないます:

```
<!--#if expr="${Mac} && ${InternetExplorer}" -->
Apologetic text goes here
<!--#else -->
Cool JavaScript code goes here
<!--#endif -->
```

Mac 上の IE に対して何か思うところがあるわけでありません。 他では実行できているいくつかの JavaScript を Mac 上の IE で実行させるのに、先週数時間苦労したというだけのことです。 上の例はその暫定的な対処方法です。

他のどんな変数 (あなたが定義するもの、 または普通の環境変数のいずれか) も、条件文に使用することができます。 Apache は SetEnvIf ディレクティブや他の関連 ディレクティブを使用して環境変数を設定することができます。 この機能により、CGI に頼ることなくかなり複雑な動的なことをさせることができます。

## 終わりに

SSI は確かに CGI や動的なウェブページを生成する他の技術に代わるものではありません。 しかし、たくさんの余分な作業をせずに、 少量の動的なコンテンツを加えるにはすぐれた方法です。

## URI References

[1] http://httpd.apache.org/docs-2.1/mod/mod_expires.html