Apache モジュール mod_include

説明:	サーバがパースする html ドキュメント (Server Side Includes)
ステータス:	Base
モジュール識別子:	include_module
ソースファイル:	mod_include.c

概要

このモジュールはファイルがクライアントに送られる前に処理するフィルタを 提供します。処理の内容は要素と呼ばれる特別な形式の SGML コメントにより 制御されます。これらの要素は条件分岐や、他のファイルや プログラムの出力の取り込み、環境変数の設定や表示を行なうことが できます。

トピック

ディレクティブ

参照

- Options
- SetOutputFilter
- AcceptPathInfo

Server-Side Includes を有効にする

Server Side Includes は INCLUDES フィルタ [1] により実装されています。 Server-side include のディレクティブを含むドキュメントの拡張子が .shtml の場合、以下のディレクティブでは Apache がそれらを パースして、その結果できるドキュメントに text/html の MIME タイプを割り当てます:

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

以下のディレクティブは shtml ファイルのあるディレクトリで指定されている 必要があります (通常は <Directory> セクション で指定しますが、AllowOverride Options が設定されていると、 .htaccess ファイルに書くこともできます):

```
Options +Includes
```

互換性を保つために、server-parsed ハンドラ [2] も INCLUDES フィルタを 有効にします。MIME タイプ text/x-server-parsed-html や text/x-server-parsed-html3 のドキュメントに対しても Apache は INCLUDES フィルタを有効にします（出力されるものは MIME タイプ text/html になります)。

詳しい情報は Server Side Includes のチュートリアル [3] を読んでください。

基本要素

ドキュメントは、SGML のコメントとして特別なコマンドが埋め込まれた HTML ドキュメントとしてパースされます。コマンドの構文は次のように なっています:

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

値（訳注: value）は二重引用符で囲まれることがよくあります。多くのコマンドは 属性-値（訳注: attribute-value）の組を一つだけ指定できます。 コメントの終わり (-->) の前には、SSI の句の一部だと解釈されないようにするために空白を 入れてください。

要素（訳注: element）には以下のものを指定することができます:

config
: このコマンドはパース時の振る舞いを制御します。指定可能な属性は:

  errmsg
  : 値はドキュメントのパース時にエラーが発生した場合に クライアントに送られるメッセージです。

  sizefmt
  : この値はファイルの大きさを表示するときに使われるフォーマットを 設定します。有効な値はバイト単位で表示する bytes と、 状況に応じて Kb や Mb を使う abbrev です。

  timefmt
  : 値は日付を表示するときに strftime(3) ライブラリ関数が 使う文字列です。

echo
: このコマンドは以下で定義されている include 変数 を表示します。変数が設定されていない場合は (none) と表示されます。日付はその時点での timefmt に従って 表示されます。

  属性:

  var
  : 値は表示する変数の名前です。

  encoding
  : Apache が変数を出力する前に、特別な文字をエンコードする 方法を指定します。"none" に設定されていると、エンコードは行なわれません。 "url" に設定されていると、URL エンコード (%-エンコードとも 呼ばれています。これはリンクの URL での使用に適しています) が 行なわれます。echo 要素の開始時は、デフ

ォルトは "entity" に設定されています。これはエンティティエンコード（段落やテキストなどのブロックレベルの HTML エレメントのコンテキストに 適しています）を行ないます。これは encoding 属性 を加えることで変更できます。変更は次の encoding 属性か、 要素の終了まで効力を持ちます。encoding 属性は エンコードの変更をしたい var の前に ある 必要があることに注意してください。また、ISO-8859-1 エンコーディングで 定義されている特別な文字だけがエンコードされます。 別の文字のエンコーディングが使われている場合、このエンコーディングは 望みの結果をもたらさないかもしれません。これは、Apache 1.3.12 以降 で使用可能です。以前のバージョンはエンコードを行ないません。

exec

exec コマンドは指定されたシェルコマンドや CGI スクリプトを 実行します。mod_cgi がサーバに組み込まれていることが 必要です。IncludesNOEXEC Option はこのコマンドを無効にします。 使用可能な属性は:

cgi

値は（%-エンコードされた）URL を指定します。パスが スラッシュ (/) で始まらないときは、ドキュメントからの 相対パスとして扱われます。このパスで参照されているドキュメントは サーバが CGI スクリプトとして扱っていなくても CGI スクリプトとして 起動されます。ただし、スクリプトのあるディレクトリでは（ScriptAlias や ExecCGI Option によって）CGI スクリプトの使用が許可されている必要があります。

CGI スクリプトには PATH_INFO とクライアントからの 元々のリクエストのクエリー文字列が渡されます。スクリプトは標準 CGI [4] 環境に加えて、include 変数を 使用することができます。

例えば:

```
<!--#exec cgi="/cgi-bin/example.cgi" -->
```

スクリプトが出力の代わりに Location: ヘッダを返すと、 HTML のアンカーに変換されます。

exec cgi よりも、 include virtual の方を使うようにしてください。特に、CGI への追加の引数を クエリー文字列を使って渡すことは exec cgi は できませんが、include virtual は以下のようにして 可能です。

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value" -->
```

cmd

サーバは指定された文字列を /bin/sh を使って 実行します。コマンドは通常の CGI 変数に加えて include 変数も使うことができます。

ほとんどいつも、#include virtual を使う方が #exec cgi や #exec cmd を使うよりも良い方法です。前者（#include virtual）は標準の Apache のサブリクエスト機構を使ってファイルやスクリプトの 出力を取り込みます。その方法の方がよりたくさん試され、 よく維持されています。

さらに、Win32 のようないくつかのプラットフォームや、suexec を使っている unix では、exec ディレクティブのコマンドに 引数を渡したり、コマンドに空白

を入れることはできません。 ですから、以下のものは unix の suexec でない設定では動作しますが、 Win32 や suexec を使っている unix では期待した結果にはなりません:

```
<!--#exec cmd="perl /path/to/perlscript arg1 arg2" -->
```

fsize
このコマンドは指定されたファイルの大きさを sizefmt の 書式指定に基づいて出力します。属性は:

file
値は解析されているドキュメントの存在するディレクトリからの 相対パスです。

virtual
値は (% エンコードされた) URL-path です。スラッシュ (/) で 始まらないときはドキュメントからの相対パスとして扱われます。

flastmod
このコマンドは指定されたファイルの最終修正時刻を timefmt 書式指定に従って表示します。 指定可能な属性は fsize コマンドと同じです。

include
このコマンドは別の文書やファイルのテキストを解析しているファイルに 挿入します。挿入されるファイルはアクセス制御の管理下にあります。 解析しているファイルの存在するディレクトリに Option[5] IncludesNOEXEC が 設定されていて、文書の挿入によりプログラムが実行されるような場合は、 その文書は挿入されません。その Option の設定は CGI スクリプトの実行を 禁止するからです。その他の場合は、 クエリー文字列も含め、コマンドで指定された 完全な URL を使って普通に CGI スクリプトが呼び出されます。

属性が文書の位置を指定します。include コマンドに与えられたそれぞれの 属性に対して挿入作業が行なわれます。有効な属性は:

file
値は解析されているドキュメントの存在するディレクトリからの 相対パスです。../ を含んでいたり、絶対パスを指定したりはできません。 ですから、ドキュメントルートの外にあるファイルや、ディレクトリ構造で 上位にあるファイルを挿入することはできません。 常にこの属性よりは、virtual 属性を使うようにしてください。

virtual
値は解析されているドキュメントからの (% エンコードされた) URL です。URL にはスキームやホスト名を含めることはできません。パスと、 もしあればクエリー文字列を指定できるだけです。スラッシュ (/) から 始まらない場合は、ドキュメントからの相対パスとして扱われます。

URL は属性から作られ、その URL をクライアントがアクセスしたときに 出力される内容が解析後の出力に含められます。ですから、挿入される ファイルは入れ子構造にすることができます。

指定された URL が CGI プログラムであった場合は、 プログラムが実行され、その出力が解析しているファイル中の ディレクティブがあった位置に挿入されます。CGI の url に クエリー URL を入れることもできます:

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value" -->
```

HTML ドキュメントに CGI プログラムの出力を含める方法としては、 include virtual の方が exec cgi よりも 好ましい方法です。

printenv
これは、存在するすべての変数とその値を表示します。Apache 1.3.12 から、 特別な文字は出力される前にエンティティエンコード（詳細は echo 要素を参照）されるようになりました。 属性はありません。

例えば:

```
<!--#printenv -->
```

printenv 要素は Apache 1.2 以降でのみ使用可能です。

set
これは変数の値を設定します。属性は:

var
設定する変数の名前。

value
変数に設定する値。

例えば:

```
<!--#set var="category" value="help" -->
```

set 要素は Apache 1.2 以降でのみ使用可能です。

Include 変数

標準 CGI 環境の変数に加えて、echo コマンドや、 if や elif, それにドキュメントから呼び出される すべてのプログラムから使用できる変数があります。

DATE_GMT
グリニッジ標準時による現在時刻。

DATE_LOCAL
ローカルの標準時による現在時刻。

DOCUMENT_NAME
ユーザがリクエストした（ディレクトリを除いた）ファイル名。

DOCUMENT_URI
ユーザがリクエストした（% エンコードされた）URL-path。 挿入ファイルが入れ子になっている場合は、解析されている ドキュメントの URL ではないことに注意してください。

LAST_MODIFIED
ユーザがリクエストしたドキュメントの最終修正時刻。

変数置換

変数置換はたいていの場合 SSI ディレクティブの引数として妥当な場所にある 引用符で囲まれた文字列中で行なわれます。これに該当するものには、 config, exec, flastmod, fsize, include, echo, set の 各ディレクティブと、条件分岐用のオペレータへの引数があります。 ドル記号はバックスラッシュを使うことで使うことができます:

```
<!--#if expr="$a = ¥$test" -->
```

変数名としてみなされる文字列の中で変数への参照を置換する必要があるときは、 シェルでの変数置換のように、中括弧で括ることで区別することができます:

```
<!--#set var="Zed" value="${REMOTE_HOST}_${REQUEST_METHOD}" -->
```

この例では、REMOTE_HOST が "X" で REQUEST_METHOD が "Y" のときに変数 Zed を "X_Y" に設定します。

例: 以下の例では、DOCUMENT_URI が /foo/file.html のときに "in foo" を、/bar/file.html のときに "in bar" を、どちらでもないときには "in neither" を表示します:

```
<!--#if expr="¥"$DOCUMENT_URI¥" = ¥"/foo/file.html¥"" -->
   in foo
<!--#elif expr="¥"$DOCUMENT_URI¥" = ¥"/bar/file.html¥"" -->
   in bar
<!--#else -->
   in neither
<!--#endif -->
```

フロー制御用要素

Apache 1.2 以降で使用できます。基本フロー制御要素は:

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

if 要素はプログラミング言語の if 文と同じように動作します。条件が評価され、結果が真であれば elif か else か endif までの文字列が出力に挿入されます。

elif や else 文は test_condition が偽のときにテキストを出力に挿入するために使われます。 これらの要素はあってもなくても構いません。

endif 要素は if 要素を終了させます。この要素は必須です。

test_condition は以下のどれかです:

string
    string が空でない場合に真です

string1 = string2
string1 != string2
string1 < string2
string1 <= string2
string1 > string2
string1 >= string2
    string1 と string2 を比較します。string2 が /string/ という形式であれば、正規表現として比較されます。正規表現は Unix の egrep コマンドと同じ構文です。

(test_condition)
    test_condition が真のときに真です

! test_condition
    test_condition が偽のときに真です

test_condition1 && test_condition2
    test_condition1 と test_condition2 の両方が真のときに真です

test_condition1 || test_condition2
    test_condition1 と test_condition2 のどちらかが真のときに真です

"=" と "!=" の方が "&&" より きつく束縛します。"!" の束縛が一番きつくなっています。 ですから以下の二つは等価です:

```
<!--#if expr="$a = test1 && $b = test2" -->
<!--#if expr="($a = test1) && ($b = test2)" -->
```

変数やオペレータとして認識されないものはすべて文字列として 扱われます。文字列は引用符で囲むこともできます: 'string' のように。引用符で囲まれていない文字列には空白 (スペースとタブ) を含めることはできません。それらは変数などの句を分離するために 使われているからです。複数の文字列が続いているときは、 空白を間に入れて一つにくっつけられます。ですから、

```
string1    string2  は string1 string2 になります

'string1    string2' は string1    string2 になります
```

ErrorDocuments で Server Side Includes を使う

mod_include の機能を使って、国際化され、カスタマイズされたエラー ドキュメントを提供するための方法を説明した 文書[6]があります。

Server Side Includes での PATH_INFO

server-side includes で処理されるファイルは PATH_INFO (後に付いたパス名の情報) 付きのリクエストを受け付けなくなりました。 PATH_INFO の付いたリクエストを受け付ける

ように設定するために、AcceptPathInfo ディレクティブを 使うことができます。

SSIEndTag ディレクティブ

説明:	include 要素を終了させる文字列
構文:	SSIEndTag tag
デフォルト:	SSIEndTag "-->"
コンテキスト:	サーバ設定ファイル, バーチャルホスト
ステータス:	Base
モジュール:	mod_include
互換性:	バージョン 2.0.30 以降で使用可能。

このディレクティブは mod_include が探す、include 要素の終了を示す 文字列を変更します。

例
```
SSIEndTag "%>"
```

参照

- SSIStartTag

SSIErrorMsg ディレクティブ

説明:	SSI のエラーがあったときに表示されるエラーメッセージ
構文:	SSIErrorMsg message
デフォルト:	SSIErrorMsg "[an error occurred while processing this directive]"
コンテキスト:	サーバ設定ファイル, バーチャルホスト, ディレクトリ, .htaccess
上書き:	All
ステータス:	Base
モジュール:	mod_include
互換性:	バージョン 2.0.30 以降で使用可能

SSIErrorMsg ディレクティブは mod_include がエラーが起こったときに 表示するメッセージを変更します。プロダクションサーバでは メッセージがユーザに表示されないようにするために デフォルトエラーメッセージを "<!-- Error -->" に変えるというようなことを考えるかもしれません。

このディレクティブは <!--#config errmsg=message --> 要素と同じ効果になります。

例
```
SSIErrorMsg "<!-- Error -->"
```

SSIStartTag ディレクティブ

説明:	include 要素を開始する文字列
構文:	SSIStartTag tag
デフォルト:	SSIStartTag "<!--"
コンテキスト:	サーバ設定ファイル, バーチャルホスト
ステータス:	Base
モジュール:	mod_include
互換性:	バージョン 2.0.30 以降で使用可能。

このディレクティブは mod_include が探す、include 要素の開始を 示す文字列を変更します。

二つのサーバが（もしかすると違うときに）ファイルの出力を解析していて、 それぞれに違うコマンドを処理させたい、 というようなときにこのオプションを使います。

例
```
SSIStartTag "<%"
```

上の例と、それに対応する SSIEndTag を使うと、 下の例の様にして SSI ディレクティブを使うことができます:

違う開始と終了のタグを使った SSI ディレクティブ
```
<%#printenv %>
```

参照

- SSIEndTag

SSITimeFormat ディレクティブ

説明:	日付けを現す文字列の書式を設定する
構文:	SSITimeFormat formatstring
デフォルト:	SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"
コンテキスト:	サーバ設定ファイル, バーチャルホスト, ディレクトリ, .htaccess
上書き:	All
ステータス:	Base
モジュール:	mod_include
互換性:	バージョン 2.0.30 以降で使用可能。

このディレクティブは DATE 環境変数を echo して日付を現す文字列が 表示されるときの書式を変更します。formatstring は C 標準ライブラリの strftime(3) と同じ形式です。

このディレクティブは <!--#config timefmt=formatstring --> 要素と同じ効果になります。

例

```
SSITimeFormat "%R, %B %d, %Y"
```

上のディレクティブでは、日付は "22:26, June 14, 2002" という 形式で表示されます。

SSIUndefinedEcho ディレクティブ

説明:	未定義の変数が echo されたときに表示される文字列
構文:	SSIUndefinedEcho tag
デフォルト:	SSIUndefinedEcho "(none)"
コンテキスト:	サーバ設定ファイル, バーチャルホスト
ステータス:	Base
モジュール:	mod_include
互換性:	バージョン 2.0.34 以降で使用可能。

このディレクティブは変数が定義されていないにも関わらず "echo" されたときに mod_include が表示する文字列を 変更します。

例

```
SSIUndefinedEcho "<!-- undef -->"
```

XBitHack ディレクティブ

説明:	実行ビットが設定されたファイルの SSI ディレクティブを 解析する
構文:	XBitHack on|off|full
デフォルト:	XBitHack off
コンテキスト:	サーバ設定ファイル, バーチャルホスト, ディレクトリ, .htaccess
上書き:	Options
ステータス:	Base
モジュール:	mod_include

XBitHack ディレクティブは通常の HTML ドキュメントの解析を 制御します。このディレクティブは MIME タイプ text/html と関連付けられているファイルにのみ影響します。XBitHack は 以下の値をとることができます:

off
: 実行可能ファイルに対して特別な扱いをしません。

on
: ユーザの実行ビットが設定されている text/html ファイルが SSI html ドキュメントとして扱われます。

full
: on と同様ですが、グループ実行ビットもテストします。 もしそれが設定されていれば、返されるファイルの Last-modified の 日付をファイルの最終修正時刻にします。それが設定されていないときは、 last-modified の日付は送られません。このビット

を設定すると、 クライアントやプロキシがリクエストをキャッシュできるようになります。

> 注意: 他の CGI を #include するかもしれないものや、各アクセスに対して違う出力を生成する（もしくは後のリクエストで変わるかもしれないもの）すべての SSI スクリプトに対してグループ実行ビットが 設定されていないことを確認できない場合は、full は使わない方が良い でしょう。

URI References

[1] http://httpd.apache.org/docs-2.1/filter.html
[2] http://httpd.apache.org/docs-2.1/handler.html
[3] http://httpd.apache.org/docs-2.1/howto/ssi.html
[4] http://httpd.apache.org/docs-2.1/mod/mod_cgi.html
[5] http://httpd.apache.org/docs-2.1/mod/core.html#options
[6] http://httpd.apache.org/docs-2.1/misc/custom_errordocs.html