CPAN特選Perlモジュールで

1歩進んだサーバープログラミング

第1回

Text：情報技研

CPANを知り、CGI.pmを使ってみよう

CPAN（Comprehensive Perl Archive Network）は、Perlプログラムとドキュメントの膨大なコレクションである。プログラミングに便利な機能を寄せ集めたもので、さしずめ知恵の宝庫といえよう。本連載では、CPANで4,000個以上提供されているPerlモジュールから便利なものを厳選し、その機能を自分のPerlプログラムから使い、効率よくプログラミングする方法を解説する。

CPAN Powered by Sun

http://www.cpan.org/

モジュールを集めたCPAN

いろんなPerlプログラムを作っているうちに、自分がいつも使う機能は関数にしていつでも使えるようにする場合が多い。だれもが使うような機能ならば、プログラミングの練習ならともかく、わざわざ自分でプログラミングするのは時間の無駄だ。

そこで、Perlプログラミングでよく使う機能や便利な機能を「モジュール」（詳しくは後述）にしてだれもが使えるようにすることで、他人の労力や知恵を共有しようというのがCPANだ。

CPANのウェブページにアクセスすると、Comprehensive Perl Archive Networkと大きく書かれている。これは、総合的なPerlの蓄積を記録して保管するためのネットワークといった意味になるが、要するにみんなでPerlに関するいろいろな蓄積を共有することを目的としたもので、Perlのモジュールだけでなくドキュメント類などさまざまな情報がある。CPANも、もちろんオープンソースコミュニティーの精神で成り立っており、自由にCPANの蓄積を利用できるだけでなく、自らもCPANに貢献することもできる。

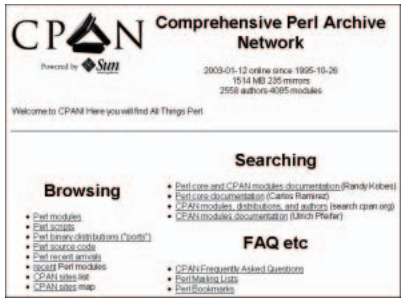
CPANからは、Perlのモジュール以外にも、便利なスクリプト、各種プラットフォームのバイナリー、ソースコードなどを入手できる。基本的なことはFAQにまとめられているので見ておくとよい。

CPANサイトの「Browsing」の中にある「Perl modules」からモジュールのリストを見ると膨大な数があることがわかる。この中から自分の使いたいモジュールを探すことは大変だと思うかもしれないが、実際には検索サイトが用意されているのでそちらを利用するとよいだろう。URL http://search.cpan.org/

モジュール名で検索したり、カテゴリー別にモジュールを一覧したりできる。このほか、CPANモジュールやPerlのドキュメントの検索はCPANサイトのトップページにあるSearchingの各ページで検索できる。

CPANから入手できるモジュールのほとんどはGPLまたはPerl本体と同じ条件で配布されているので、ほぼ自由に利用できるが、モジュールを使う際には、必ず各モジュールに付属のドキュメントを参照してライセンス条件などを確認してほしい。

図1　CPANサイト（URL http://www.cpan.org/）



そもそもPerlモジュールとは

Perlモジュールについて少し解説しておこう。「モジュール」とは、Perlにプログラム上便利な機能を追加するライブラリーで、プログラミングを楽に行うための仕組みといえる。特定の機能をモジュールという形式でインストールしておけば、プログラムから簡単に呼び出してその機能を利用できる。一部のモジュールは標準でインストールされているが、自分が使いたいモジュールはCPANなどから自由にインストールできる。

例をあげると、in-fileというファイルの中身をソートして、その結果をout-fileというファイルに書き込むとしよう。これをPerlのプログラムで書く場合、まずin-fileを開いてその内容を変数に読み込み、sort関数でソートしてから、out-fileを開き、ソートした変数の内容を書き込むのが普通だろう。

Perlの機能だけで作った場合

```
open(INPUT,'in-file');
while(<INPUT>){
    push (input,$_);
}
close(INPUT);
@output = sort @input;
open(OUTPUT,'>out-file');
forearch $i (@output){
    print OUTPUT $_;
}
close(OUTPUT);
```

それでは、CPANにある「File::Sort」というモジュールを使った場合に、このプログラムはどうなるだろうか。その例を次に示す。先ほど11行で作ったのと同じ処理が2行で済んでしまうのだ。

File::Sortモジュールを使った場合

```
use File::Sort qw(sort_file);
sort_file('in-file','out-file');
```

この場合、モジュールを呼び出したうえで、入力と出力のファイルを指定してモジュールのメソッドを使うだけで、in-fileの各行をソートした結果をout-fileに書き込んでくれる。

このように、あたり前だが毎回書かなければいけない面倒なプログラミングはモジュールを使えばどんどん省略できる。ここで示したのはほんの一例で、プログラミングを効率化してくれる便利なモジュールが数多く存在する。

CPAN利用を便利にするCPAN.pmモジュールを使ってみよう

CPANサイトにはかなりの数のモジュールがある。カテゴリー分けされて整理されているとはいえ、特定のモジュールを探し、ダウンロード、解凍、インストールといった作業を手で行うのは面倒だろう。それを楽に行うために、CPAN.pmというPerlのモジュールが用意されているので使ってみよう。

CPANモジュールはAndreas Konig氏の作で、Perl標準モジュールとなっているため、Perlがインストールされていればそのまま利用できる。CPANモジュールからはモジュールの各種検索、自動ダウンロードとインストールが行えるようになっており、これがあれば簡単にPerlモジュールを追加できる。

CPANモジュールの初期化

CPANを使ってモジュールを追加する前に、まずCPANモジュールを初期化して、ダウンロードサイトを設定しておこう。CPANモジュールを呼び出すには、perlコマンドが使える環境で次のようにコマンドを実行すればよい。

```
# perl -MCPAN -e shell
```

これで、CPANモジュールがインタラクティブモードで起動する。画面に各種質問事項が表示されるので、それに従って選択入力をしていく(図2)。質問に答えていくと初期設定が完了してCPANモジュールのインタラクティブモードになるので、exitと入力して終了する。

```
cpan> exit
```

図2　CPAN.pmの初期化の手順

```
まずメッセージが表示される(省略)

Are you ready for manual configuration? [yes]
←[Enter]を入力

この後しばらく質問事項は[Enter]キーを押す

If you're accessing the net via proxies, you can specify
them in the
CPAN configuration or via environment variables. The
variable in
the $CPAN::Config takes precedence.

Your ftp_proxy?  ← FTPのプロキシがある場合には入力
Your http_proxy?  ← HTTPのプロキシがある場合には入力
Your no_proxy?

You have no /root/.cpan/sources/MIRRORED.BY
  I'm trying to fetch one

この後 MIRRORED.BY ファイルをダウンロードする
続けてしばらく質問事項は[Enter]キーを押す

(1) Africa
(2) Asia
(ほかにも地域の選択肢が表示されるが省略)
Select your continent (or several nearby continents) [] 2
←地域の選択なのでアジア(2)を選ぶ

(1) China
(2) India
(3) Indonesia
(4) Israel
(5) Japan
(ほかにも国名の選択肢が表示されるが省略)
Select your country (or several nearby countries) [] 5
←国の選択なので日本(5)を選ぶ

(1) ftp://ftp.ayamura.org/pub/CPAN/
(2) ftp://ftp.cpan.jp/CPAN/
(ほかにもダウンロードサイトの選択肢が表示されるが省略)
Select as many URLs as you like [] 2
← 自分の環境で高速につながるサイトを選ぶ
  わからなければftp.cpan.jp/CPAN/を選ぶ

Enter another URL or RETURN to quit: []
←[Enter]キーを押して続行する

New set of picks:
  ftp://ftp.cpan.jp/CPAN/

WAIT support is available as a Plugin. You need the
CPAN::WAIT module
to actually use it.  But we need to know your favorite
WAIT server. If
you don't know a WAIT server near you, just press ENTER.

Your favorite WAIT server?
   [wait://ls6.informatik.uni-dortmund.de:1404]
←[Enter]キーを押す。メッセージが表示されて初期化が終了する

cpan>
```

モジュールをインストールしてみよう

次にCPANモジュールを使ってモジュールをインストールしよう。まず、CPANモジュールをインタラクティブモードで起動する。

```
# perl -MCPAN -e shell
```

たとえばFTPモジュールであるNet::FTPモジュールをインストールする場合には、CPANインタラクティブモードで次のように入力すればよい。

```
cpan> install Net::FTP
```

これだけで、初期設定で指定したCPANサイトに自動的に接続して調べてくれる。指定したモジュールがシステムにインストールされている場合には、その旨表示される。システムにインストールされていない場合や、またはさらに新しいバージョンのモジュールがみつかった場合には、ダウンロードからインストールまでを自動的に行ってくれる。インストールが終了するとプロンプトが表示されるので、exitと入力して終了する。

```
cpan> exit
```

これでNet::FTPモジュールがインストールできた。実際にCPANからモジュールを入手してみよう。次のページでは、まず手始めに、CGIプログラミングに便利なCGI.pmを紹介する。

CPANのインストールは基本的にはPerlのインストールと同様にシステム管理者のrootで行った方がよい。root権限がなくてCPANを利用したい場合には、個人のホームディレクトリーの下にCPANをインストールすることになるので、利用者が限定されることに注意しておきたい。

CGI.pmモジュール

【カテゴリー】CGI
【バージョン】2.89（2002/10/16）
【作者】Lincoln D. Stein 氏
【URL】
http://search.cpan.org/author/LDS/CGI.pm-2.89/CGI.pm
http://stein.cshl.org/WWW/software/CGI/

HTMLによるCGIフォームを作り、データを受け取って処理するメソッドを多数持っている

CGI.pmモジュールは、CGIフォームの作成と受け取りを行うためのモジュールで、PerlでCGIプログラムを作成するうえで、だれもが行っているような操作を効率よくプログラミングする機能を提供している。Perlのバージョン5.004以降では標準モジュールとなっているため、改めてインストールする必要はないが、最新版はもちろんCPANからダウンロードすることができる。原稿執筆時点での最新バージョンは2.89で、2002年10月16日にリリースされたものである。

CGI.pmのココがスゴイ!

CGI.pmは、PerlでCGIプログラムを作成するときに使うような処理が簡単に行えるような機能を提供する。

ウェブページでよく使うフォームで送られたデータの処理なども、面倒なエンコード／デコードやタグ出力の処理を自分でいちいち書かなくても、CGI.pmを使えば簡単にできてしまうのだ。

また、CGIプログラミングにつきものの、HTMLタグを出力する機能もあり、CGIプログラミング全般に役に立つ。

CGI.pmを使ったサンプル1：フォームデータの処理と出力

たとえば、フォーム処理を行うCGIのプログラムを作成する場合、まずURLエンコードされた入力をデコードすることから始めなければならない。これを、CGI.pmのようなモジュールを使用しない場合には、一連の正規表現を使って変換したうえで変数に代入するという処理を自分で書かなければならない。しかしこれだけでは、GETメソッドを使ってフォームから呼び出されたCGIが、各パラメーターを%stinrgという変数に格納しただけだ。実際に「name」というパラメーターの値を表示したい場合には、次のように、おきまりの「content-type」行を出力してから、値を出力する必要がある（図3）。

さて、これをCGI.pmで行うとどうなるだろうか。説明はともかくコードを見てみよう。これだけで済んでしまうのである。

```
#!/usr/bin/perl
use CGI qw(:standard);
$name = param("name");

print header;
print start_html;
print p("Your Name is $name")
print end_html;
```

CGI.pmモジュールを使わない最初のサンプルコードの場合には、前処理としてURLエンコードされた入力をデコードすることから始め、それから目的とした処理（ここではパラメーター「name」の表示）を行っていた。ところがCGI.pmを使うことで、前処理のことを考える必要もなく、ちょっとした宣言文の後に目的の処理を書くだけで同じことができてしまう。

図3　フォームのデータを受け取って処理するコード

```
#!/usr/bin/perl
foreach $i (split(/&/, $ENV{QUERY_STRING})){
  ($key, $value) = split(/=/, $i);
  $key   =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
  $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
  $strings{$key} = $value;
}
print "Content-type: text/html\n\n";
print "<HTML><BODY>\n";
print "<P>Your Name is $strings{name}.</P>\n";
print "</BODY></HTML>\n";
```

また、このサンプルコードはGETメソッドを使う場合だが、POSTメソッドでは、環境変数$ENV{QUERY_STRING}で読み込んでいるデータを標準入力より受け取るように書き換えなければならない。こういった細かいところもCGI.pmモジュールを使えばプログラマーが気を使う必要はなくなる。

CGI.pmの代表的なメソッド

CGI.pmにはたくさんのメソッドがあるが、サンプルプログラムで利用したparam、header、start_html、p、end_htmlメソッドをそれぞれ簡単に説明しておこう。

paramメソッド

パラメーター名の取得もしくは値の設定

【構文】

```
param([parameter,[value1,value2,...]]);
```

【引数】

・parameter
取得するパラメーター名。省略の場合には既知のすべてのパラメーター名のリストを返す。

・value1, value2, ...
新たにパラメーターに値を代入する場合に指定する。

Headerメソッド

HTTPヘッダーの生成

【構文】

```
header([content_type,status,headers]);
```

【引数】

・content_type

出力されるメディアタイプの指定。デフォルトは "text/html"。

・status

出力されるHTTPのステータスコードおよび説明。デフォルトは "200 OK"。

・headers

追加したいヘッダーの指定。指定できるのは-type=>、-nph=>、-status=>、-expires=>、-cookie=>、-target=>、-header=>

start_htmlメソッド

パラメーター名の取得もしくは値の設定

【構文】

```
param([parameter,[value1,value2,...]]);
```

【引数】

・parameter

取得するパラメーター名。省略の場合には既知のすべてのパラメーター名のリストを返す。

・value1, value2, ...

新たにパラメーターに値を代入する場合に指定する。

Pメソッド

pタグの設定

【構文】

```
p(parameter);
```

【引数】

```
parameter
```

pタグで表示する文字列

end_htmlメソッド

HTML文書の終了

【構文】

```
end_html;
```

その他のCGI.pmのメソッド

ここまでに出てきたCGI.pmのメソッドはほんの一部に過ぎない。CGI.pmのメソッドには、CGI処理に必要なもの、HTMLやFORMのフォーマットのために必要なものなど数多くのものが存在する。数多くのメソッドはグループ分けされて名前がつけられている。各グループに含まれるメソッドを図4に示す。

また、:html2、:html3、:html4、 :netscapeをまとめた「:html」や、:html2、 :html3、:html4、:cgi、:formをまとめた「:standard」と、全部まとめた「:all」というグループが存在する。実際に利用する場合には「:standard」でほぼ問題ないだろう。使用するグループの宣言は最初の「use CGI qw(:standard)」の部分で行う。

このように、CGI.pmでは数多くのメソッドが用意されているので、CGIの基本的な処理やHTMLの出力といった自分で書くとめ

図4　CGI.pmのメソッドグループ

■:html2グループ（HTML 2.xタグの生成）

a address b base blockquote body br charset cite code comment dd dfn dl dt em end_html escapeHTML h1..h6 head hr html i img input kbd li Link menu meta nextid ol option p pre samp Select start_html strong title tt u ul var

■:html3グループ（HTML 3.xタグの生成）

applet basefont big caption div embed font frame frameset ilayer layer Param script small span strike style Sub sup table td th TR Tr

■:html4グループ（HTML 4.xタグの生成）

abbr acronym bdo col colgroup del fieldset iframe ins label legend noframes noscript object optgroup Q tbody tfoot thead

■:formグループ（フォーム要素の生成）

autoEscape button checkbox_group checkbox defaults end_form end_multipart_form endform filefield hidden image_button isindex MULTIPART password_field popup_menu radio_group reset scrolling_list start_form start_multipart_form startform submit textarea textfield tmpFileName uploadInfo URL_ENCODED

■:cgiグループ（CGIの処理）

Accept auth_type cgi_error content_type cookie Delete_all Delete Dump header http import_names param_fetch param path_info path_translated put query_string raw_cookie redirect referer remote_addr remote_host remote_ident remote_user request_method restore_parameters save_parameters script_name self_url server_name server_port server_protocol server_software upload url_param url user_agent user_name virtual_host

■:netscapeグループ（HTML 3.xに含まれてないネットスケープ拡張タグの生成）

blink center fontsize

CGI.pmはオブジェクト指向モジュール

CGI.pmはオブジェクト指向のモジュールとなっている。実際に使う分には特に怖がることもなければ、気にすることもなく利用できるが、念のため説明をしておこう。

例であげた処理を正式に記述すると次のようになる。実際の処理では「use」でモジュールを呼んだ後、コンストラクタ「new()」で$queryというCGIオブジェクトを生成し、$queryのメソッドを呼び出すことでパラメーターや各種の処理を行うことになる。

実際のプログラミングでは記述の手間を減らすために、本文中のサンプルプログラムのようなプログラミングを行ってしまえばよい。

```
use CGI;
$query = CGI::new();
$name = $query->param("name");
print $query->header();
print $query->start_html();
print $query->p("Your Name is $name")
```

んどうな処理を簡単に処理できてしまう。これらのメソッドについては、CGI.pmのサイトで解説されている。

CGI.pmを使ったサンプル2：HTMLタグの出力

それでは、実際にCGI.pmモジュールのメソッドを使ってみる。手軽に使えるものとしてはHTMLタグの生成メソッドがあるので、これを使ってみよう。

CGIプログラムを書くときに面倒だと感じることの1つにHTMLの出力があるだろう。いちいちテンプレートとなるHTMLを先に書いておいたうえで、それをPerlのプログラムにprint文で埋め込んでいくといった作業を行っていることも多々あるだろう。なにより、Perlのコードの中にHTMLのタグが混在すると、書きづらいうえに読みづらい。CGI.pmによるHTMLタグの生成は、他のメソッドと比べて何かが非常に楽になるというあっと驚くものはないが、HTMLタグを生成するPerlコードを書きやすくするという点では地味ではあるが便利なメソッドだといえる。

簡単なページを表示させるサンプルプログラムを図5に示す。このCGIを実行すると図6のようなページを出力する。

CGI.pmモジュールを使わないで同じような画面を表示させようとする場合、Perlコードの中にベタでHTMLを書いておくことになる。そうなると、HTMLタグとPerlコードが入り乱れて美しくないのは想像のつくとおりである。かといってコードを綺麗にするために、HTMLの部分だけを別ファイルにしておいて読み込むというのもそれはそれで面倒といえる。このサンプルプログラムを見ると、HTMLのタグを出力するコードがPerlのコードに自然に組み込まれていて読みやすいことがわかるだろう。

ここで、HTMLタグの生成メソッドについて説明しておく。基本的にメソッド名はHTMLのタグ名と同じだと考えてよい。TABLEタグであればtableメソッド、H1タグであればh1メソッドといった具合である。基本的な書式の対応は図7のようになると考えればよい。単純に、指定した文字列をメソッドと同じ名前のタグで囲んで出力する。

たとえば、メソッドh1やcommentを例にとってみると、メソッドと出力されるHTMLの関係は図8のようになる。

これらのように単純なものでなくオプションを指定できるようなタグの場合には、「{-オプション名=>'指定値'}」のようにしてオプションとその値を指定する。たとえば、リンクを作るAタグであれば、図9のようになる。

出力されるHTMLコードを見やすく

HTMLタグの生成メソッドを使うことでHTMLを生成するPerlのコードは見やすくなったが、実際に生成されるHTMLのコードが見やすいというわけではない。

実際にサンプルプログラムが生成するHTMLコードは、次ページの図10に示すような見づらいコードとなっている。ヘッダー部分はともかく、HTMLファイルのほとんどの内容が、改行されずに1行にひたすら続いている。これでは生成したHTMLファイルを

図5　CGI.pmのHTMLタグ生成メソッドを使ったサンプルプログラム

```
#!/usr/bin/perl
use CGI qw(:standard);

print
  header(-charset=>'euc-jp'),
  start_html(-lang=>'japanese', -title=>'CGI.pmで作るHTML'
             -head=>meta({-http_equiv=>'Content-Type',
                          -content=> 'text /html; charset=euc-jp'})),
  h1('さまざまなHTMLタグの出力')
  comment('ここから表スタート')
  table({-border=>1}, caption('月の名前')
        Tr({-aglign=>CENTER},
           [th(['月','異名','英語']), td([1,'睦月',January]),
            td([2,'如月',February]), td([3,'弥生',March]),]
        ),
  ),
  hr,
  a({-href=>'http://www.google.com/', -target=>'_new'},
  'Googleの検索ページ'),
  end_html;
```

図7　HTMLタグメソッドと出力の関係

```
メソッド名('文字列')
        ↓
<HTMLタグ>文字列</HTMLタグ>
```

図8　HTMLタグメソッドの使用例と出力の例

```
h1('さまざまなHTMLタグの出力')
        ↓
<H1>さまざまなHTMLタグの出力</H1>

comment('ここから表スタート')
        ↓
<!-- ここから表スタート -->
```

図9　オプションを特定するHTMLタグメソッドと出力の例

```
a({-href=>'http://www.google.com/', -target=>'_new'}, 'Googleの検索ページ
')
                              ↓
<A HREF="http://www.google.com/" TARGET="_new">Googleの検索ページ</A>
```

図6　サンプルプログラムの実行結果

さまざまなHTMLタグの出力

月の名前

月	異名	英語
1	睦月	January
2	如月	February
3	弥生	March

Googleの検索ページ

図10　CGI.pmが出力した非常に見づらいHTMLコード

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
   "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="japanese"><head><title> CGI.pmで作るHTML </title>
<meta content="text/html; charset=euc-jp" http-equiv="Content-Type" />
</head><body><h1>さまざまなHTMLタグの出力</h1><!-- ここから表スタート --><table border="1"><caption>月の名前</caption> <tr
aglign="CENTER"><th>月</th> <th>異名</th> <th>英語</th></tr> <tr aglign="CENTER"><td>1</td> <td>睦月</td>
<td>January</td></tr> <tr aglign="CENTER"><td>2</td> <td>如月</td> <td>February</td></tr> <tr
aglign="CENTER"><td>3</td> <td>弥生</td> <td>March</td></tr></table><hr /><a href="http://www.google.com/"
target="_new">Googleの検索ページ</a></body></html>
```

後から別の用途で加工して使おうとしても手間がかかる。

HTMLタグの生成メソッドを使わずに自分でインデントしながらHTMLコードを書くことで、出力するHTMLを見やすくすることはできるが、それは非常に手間がかかる。

CGI.pmモジュールには、便利なことに、出力するHTMLコードを自動的にインデントするモジュールが存在する。このモジュールはCGIモジュール用のPrettyモジュールで、比較的最近のCGIモジュールがインストールされていれば標準で利用可能なモジュールである。

Prettyモジュールを使って自動的にインデントするには、先ほどのPerlのサンプルコードに次の行を付け足すだけでよい。

```
#!/usr/bin/perl

use CGI qw(:standard);
use CGI::Pretty;  ←この行を付け足す。
```

これで、生成されるHTMLコードが自動的にタブでインデントされるようになる。また、次のようにすれば、インデントを明示的に指定することも可能である。

インデントをタブ2つに

```
$CGI::Pretty::INDENT = "\t\t";
```

インデントをスペース2つに

```
$CGI::Pretty::INDENT = "  ";
```

ここでは、空白2つでインデントするオプションをつけてHTMLコードを生成させてみよう。実際のPerlコードの頭の部分を次のように変更する。

```
#!/usr/local/bin/perl
use CGI qw(:standard);
#次の2行を付け足す。
use CGI::Pretty;
$CGI::Pretty::INDENT = "  ";
```

この結果生成されるHTMLコードの一部を図11に示す（長くなるので後半は省略）。非常に見やすくなっていることがわかるだろう。モジュールを利用すれば、Perlコードを数行追加するだけでここまでできてしまうのだ。

なお、使用しているPerlのバージョンが比較的古くて、標準のCGIモジュールにPrettyが付属していない場合はCPANモジュールを使ってPrettyをインストールしよう。

```
# perl -MCPAN -e shell

cpan> install CGI::Pretty
```

図11　Prettyによって整形されたHTML

```
Content-Type: text/html; charset=euc-jp

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-
basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
lang="japanese"><head><title> CGI.pmで作るHTML
</title>
<meta content="text/html; charset=euc-jp" http-
equiv="Content-Type">
</head><body>
<h1>
  さまざまなHTMLタグの出力
</h1>
<!--
  ここから表スタート
 -->
<table border="1">
  <caption>
    月の名前
  </caption>
   <tr aglign="CENTER">
    <th>
      月
    </th>
    <th>
      異名
    </th>
    <th>
      英語
    </th>
  </tr>
  <tr aglign="CENTER">
    <td>
      1
    </td>
    <td>
      睦月
    </td>
    <td>
      January
    </td>
  </tr>
```

［インターネットマガジン　バックナンバーアーカイブ］ ご利用上の注意

このPDFファイルは、株式会社インプレスR&D（株式会社インプレスから分割）が1994年～2006年まで発行した月刊誌『インターネットマガジン』の誌面をPDF化し、「インターネットマガジン　バックナンバーアーカイブ」として以下のウェブサイト「All-in-One INTERNET magazine 2.0」で公開しているものです。

http://i.impressRD.jp/bn

このファイルをご利用いただくにあたり、下記の注意事項を必ずお読みください。

- 記載されている内容（技術解説、URL、団体・企業名、商品名、価格、プレゼント募集、アンケートなど）は発行当時のものです。
- 収録されている内容は著作権法上の保護を受けています。著作権はそれぞれの記事の著作者（執筆者、写真の撮影者、イラストの作成者、編集部など）が保持しています。
- 著作者から許諾が得られなかった著作物は収録されていない場合があります。
- このファイルやその内容を改変したり、商用を目的として再利用することはできません。あくまで個人や企業の非商用利用での閲覧、複製、送信に限られます。
- 収録されている内容を何らかの媒体に引用としてご利用する際は、出典として媒体名および月号、該当ページ番号、発行元（株式会社インプレスR&D）、コピーライトなどの情報をご明記ください。
- オリジナルの雑誌の発行時点では、株式会社インプレス R&D（当時は株式会社インプレス）と著作権者は内容が正確なものであるように最大限に努めましたが、すべての情報が完全に正確であることは保証できません。このファイルの内容に起因する直接的および間接的な損害に対して、一切の責任を負いません。お客様個人の責任においてご利用ください。

このファイルに関するお問い合わせ先
株式会社インプレスR&D
All-in-One INTERNET magazine 編集部
im-info@impress.co.jp