サンプルファイルには以下のページを収録しています。

・第 1 章　18～21 ページ
・第 2 章　62～65 ページ
・第 3 章　114～119 ページ
・第 4 章　174～177 ページ
・第 5 章　216～219 ページ
・第 6 章　287～290 ページ
・第 7 章　310～313 ページ
・第 8 章　350～353 ページ
・第 9 章　422～425 ページ

★PHPとMySQLの学習に必要な環境

まず、PHPとMySQLを学習する上で必要な環境から、話を始めることにしましょう。

■ローカルサーバー

PHPとMySQLを動作させるには、それらがインストールされたサーバー環境が必要になります。

一般的には、WordPressはレンタルサーバーで利用することが多いでしょう。レンタルサーバーには、Webサーバー（Apacheなど）や、PHP／MySQLなど、WordPressなどのさまざまなWebアプリケーションを動作させる環境が整っています。

ただ、PHPやMySQLに不慣れなうちから、レンタルサーバーでプログラム作りを学習するのは、お勧めできません。

まず、個人向けの安価なレンタルサーバーは、1台のサーバーを多数のユーザーで共有する形が取られています。そのため、皆様が作ったプログラムに不具合があった場合、同じサーバーを使っている他のユーザーに、迷惑をかけてしまう恐れがあります。最悪の場合、迷惑をかけてしまったために、レンタルサーバーのアカウントをはく奪されることもあり得ます。

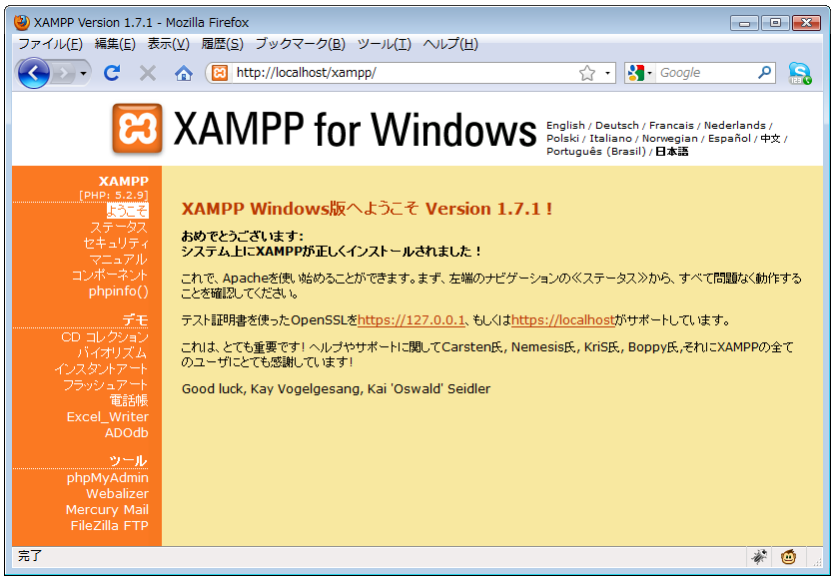
かといって、1台のサーバーを専有できるタイプのレンタルサーバーは、料金が非常に高価です。一般的な個人ユーザーなら、借りるのはまず無理だと言えるでしょう。

そこで、レンタルサーバーではなく、ご自分のパソコンにApache／PHP／MySQL等をインストールして、ご自分のパソコンの上で学習を進められるようにします。こうすれば、仮にプログラムに不具合があったとして、他の人に迷惑をかけることはありません。

ローカルサーバー環境を作るソフトは、いろいろとリリースされています。本書では、それらの中から、Windowsでは「XAMPP for Windows」(画面1.1)。MacOS Xでは「XAMPP for MacOS X」を使うことにします(※)。この後の節で、具体的なインストール手順を解説します。

(※) Macでは、ローカルサーバーのソフトとして、「MAMP」が有名でかつ使いやすいです。ただ、本書執筆時点では、MAMPは1年ほどバージョンアップされていなかったので、本書ではXAMPP for Macを取り上げました。

画面1.1　XAMPP for Windows



■統合開発環境

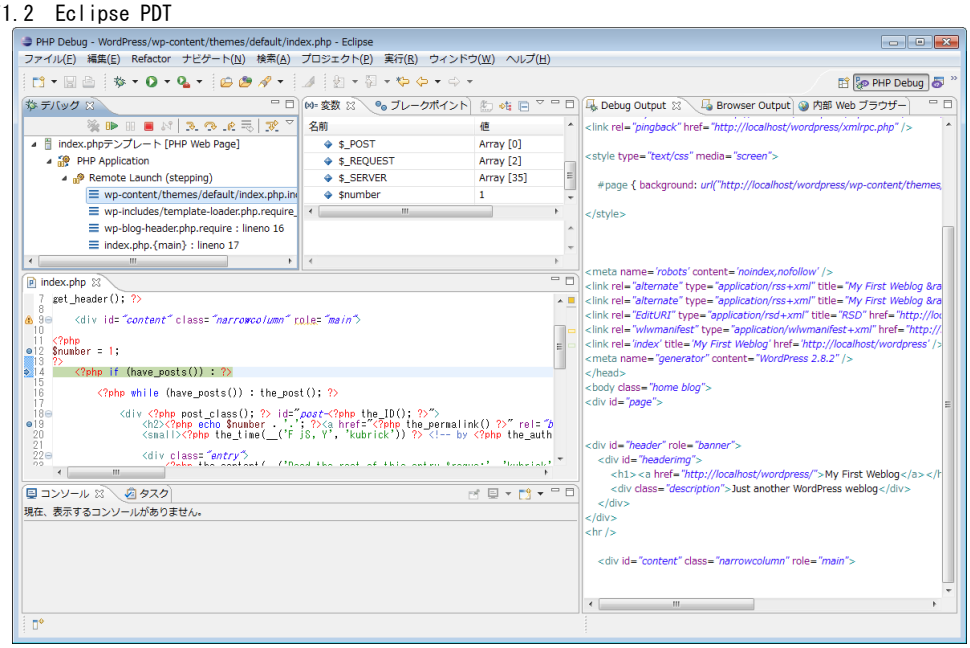
WordPressでは、テーマ（テンプレート）を管理画面上で編集することができます。ちょっとしたPHPのプログラムを作るだけなら、WordPressの管理画面上で操作を行っても、さほど難しくはないでしょう。

しかし、ある程度本格的にプログラムを書くとなると、WordPressの管理画面上ですべての作業を行うのは、難しくなってきます。特に、プログラムに不具合が発生した時に、不具合の原因を探る（＝デバッグ）際には、そのためのツールがないと非常に困難です。

また、プログラム作りに不慣れな間は、プログラムを1行ずつじっくり実行してみて、プログラムの動作を目で追った方が分かりやすいです。その際にも、デバッグの機能を活用することができます。

そこで、PHPの開発やデバッグを行いやすくするために、PHPの統合開発環境もインストールします。本書では、「Eclipse PDT」という、フリーの統合開発環境を使います（画面1.2）。Eclipse PDTのインストール手順は、後の45ページで解説します。

画面1.2　Eclipse PDT



■テキストエディタ

プログラムを作る際には、テキストファイルを編集する機会も出てきます。WindowsやMacOS Xには、テキストファイルを編集するソフトも付属していますが、使い勝手が良くありません。そこで、テキストファイルを編集するためのソフト（テキストエディタ）も、インストールしておくようにします。

テキストエディタはいろいろな製品があります。Windows用の無料で使えるテキストエディタとしては、「MKEditor」などがあります（画面1.3）。以下のアドレスからダウンロードすることができます。

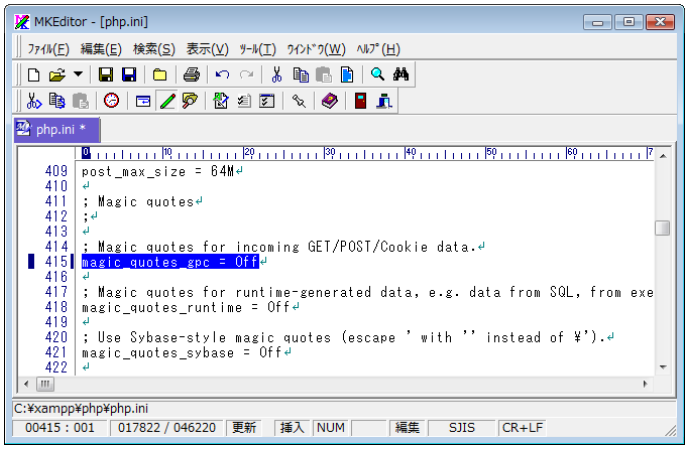
http://www.mk-square.com/

また、Mac用だと、「CotEditor」や「mi」（ミミカキエディタ）などが有名です。それぞれ、以下のアドレスからダウンロードすることができます。

・CotEditor
http://www.aynimac.com/p_blog/files/article.php?id=41

・mi
http://www.mimikaki.net/

画面1.3　MKEditor



■パソコンの設定等

XAMPPを動作させる上で、前もって設定しておくべき点があります。

●Skypeとのバッティングを避ける

XAMPPのインストール先のパソコンに、Skypeをインストールしている場合、SkypeとXAMPPがバッティングして、正しく動作しないことがあります。

Skypeをインストールされている方は、XAMPPをインストールする前に、以下の作業を行っておいてください。

①Skypeを起動します。
②「ツール」→「設定」メニューを選び、「設定」のダイアログボックスを開きます。
③ダイアログボックス左端のメニューで「詳細」→「接続」をクリックします。
④「上記のポートに代わり、ポート80を使用」のチェックがオンになっていたら、オフにして設定を保存します（画面1.4）。

★初めてのPHPスクリプト

ここでは、PHPを使って簡単なスクリプトを作ってみることにしましょう。また、作ったページを表示する方法も解説します。ここでは、サンプルとして「現在の日付と時刻を表示する」というものを作ることにします。

■スクリプトの入力

PHPのスクリプトは、HTMLファイルの中に埋め込む形をとります。PHPのスクリプトは、通常は「<?php」と「?>」で囲んだ範囲に書きます。

それでは、実際にスクリプトを入力しましょう。HTMLファイルの中に、「現在の日付と時刻を表示する」というスクリプトを入れると、リスト2.1のようになります。

リスト2.1　現在の日付と時刻を表示する（datetime.php）

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
   TR/xhtml1/DTD/xhtml1-transitional.dtd">
02 <html xmlns="http://www.w3.org/1999/xhtml">
03 <head>
04 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
05 <title>現在の日付と時刻を表示する</title>
06 </head>
07 <p>現在の日付と時刻は
08 <?php
09     echo date("Y/m/d H:i:s");
10 ?>
11 です。
12 </p>
13 <body>
14 </body>
15 </html>
```

リストの8行目から10行目がPHPのスクリプトです。8行目と10行目はスクリプトの最初と最後を表すタグなので、実質的なスクリプトは

```
echo date("Y/m/d H:i:s");
```

の1行だけです。

「echo」は「文字を出力する」という意味の命令です。また、括弧の中にある「date」は「指定した書式で、日付と時刻を文字に変換する」という命令です(※)。これら2つの命令によって、日付と時刻が表示されるわけです。

スクリプトを入力する際には、テキストエディタを使うことができます（20ページ参照)。また、市販のWebページ作成ソフトで入力することもできます。

Webページ作成ソフトを使えば、ワープロソフト感覚でHTMLファイルを一通り作った後で、HTMLのタグを直接に編集するモードに切り替えて、PHPのスクリプトを入力することができます。

なお、リスト2.1のサンプルファイルは、「part02」→「datetime」フォルダの「datetime.php」です。

■ファイルの保存

スクリプトの入力が終わったら、それをファイルに保存します。

●ファイルの保存先

XAMPPを使ってスクリプトの動作をテストしたい場合、自分のパソコンの特定のフォルダにファイルを保存します。

XAMPP for Windowsの場合、標準のインストール先はCドライブの「xampp」フォルダになります。そのフォルダの中に「htdocs」というフォルダがありますが、そのフォルダの中にファイルを保存すれば、Webブラウザで開くことができるようになります。

また、XAMPP for MacOS Xでは、ハードディスクの「アプリケーション」のフォルダの中に「xampp」のフォルダがあり、さらにその中に「htdocs」のフォルダがあります。そこにファイルを保存します。

Windows／MacOS Xとも、「htdocs」フォルダ内にサブフォルダを作って、そこにファイルを保存しても構いません。

一方、外部のサーバーを利用する場合は、ファイルをいったん保存しておき、後でそのファイルをサーバーにアップロードします。アップロードの手順は後で解説します。

(※) 関数について

ここではdateを「命令」と呼びましたが、厳密にはdateは「関数」です。関数については、第6章の248ページで解説します。

●ファイルの文字コード

WordPressでは、文字コードとしてUTF-8が使われています。そこで本書でも、ファイルの文字コードをUTF-8にします。ファイルを作成する際には、UTF-8で保存することができるテキストエディタ等を使うか、文字コード変換ツールを使ってUTF-8に変換するなどします。

なお、パソコンではShiftJISが良く使われていますが、PHPではShiftJISを使うと文字列の処理時に不具合が起きることがあります。したがって、ShiftJISは基本的に使わないようにします。

●ファイルの拡張子

一般のHTMLファイルでは、拡張子を「.htm」や「.html」にしますが、PHPのファイルでは一般に拡張子を「.php」にします。

ただし、レンタルサーバーを使う場合、業者の設定によっては拡張子が「.php」ではなく、「.php5」などになっていることもあります。拡張子の指定については、レンタルサーバーのヘルプやマニュアル等を参照してください。

■ファイルのアップロード

レンタルサーバー等の外部のサーバーを利用する場合は、FTPを使ってファイルをそのサーバーにアップロードします。

その際に、転送モードは「ASCIIモード」にします（「テキストモード」と呼ぶ場合もあります）。もう1つのモードとして「BINARYモード」がありますが、こちらは使わないようにします。

なお、FTPソフトの設定方法については、それぞれのソフトのヘルプ等を参照してください。

■ファイルのパーミッションについて

Perl等でCGIを作る場合、スクリプトのファイルをアップロードした後でパーミッションを設定して、ファイルを実行可能にする必要がありました。

一方、PHPには「モジュール版」と「CLI（コマンドラインインターフェース）版」があります。モジュール版PHPの場合は、ファイルのパーミッションを変える必要はありません。XAMPPのPHPもモジュール版PHPになっています。一方、CLI版ではPHPもCGIとして動作しますので、パーミッションの設定が必要になる場合があります。

モジュール版／CLI版のどちらが使われているかは、サーバーの設定によって異なります。お使いのサーバーのヘルプ等を参照してください。

■ファイルを表示する

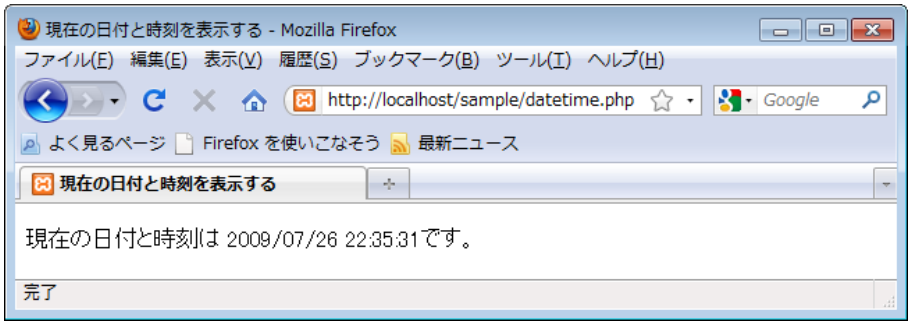
ファイルをアップロードしたら、一般のHTMLファイルと同様に、Webブラウザで表示することができます。URLの指定方法も、HTMLファイルの場合と同じです。ただし、ファイルの拡張子が「.php」などになる点が異なります。

XAMPPを使っている場合、「http://localhost/htdocs以下のフォルダ名/ファイル名.php」にアクセスすると、PHPのページが表示されます。

たとえば、62ページの「datetime.php」ファイルを、XAMPPの「htdocs」フォルダの中の「sample」フォルダを作って保存したとします。この場合、「http://localhost/sample/datetime.php」にアクセスします（画面2.1）。

また、外部のサーバーにアップロードした場合は、そのサーバーに通常のHTMLファイルをアップロードしたときと同様に、PHPのページのアドレスが決まります。

画面2.1　リスト2.1のスクリプトを実行したところ



★条件によって処理を変えるーーーif文

一連の処理の中では、条件が成立するかどうかによって、その後の処理を変えることもよくあります。このようなことを行うには、「if」という文を使います。

■if文の基本型

まず、条件が1つだけの基本的なif文から説明しましょう。図3.1のように、条件が成立するかどうかで処理を分ける場合、その構文はリスト3.1のようになります。

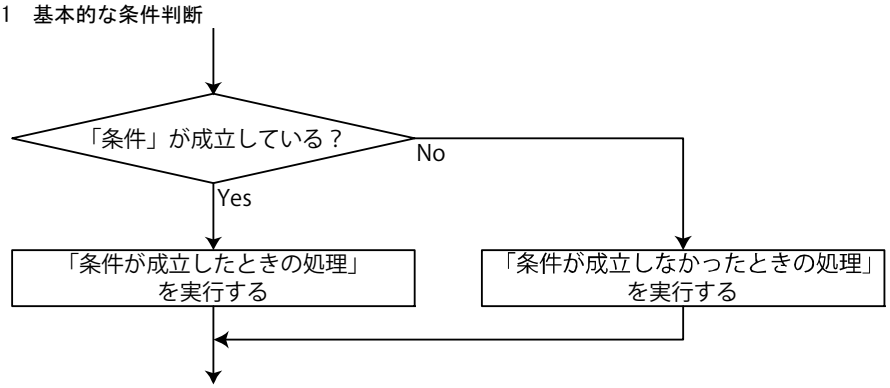
「条件」の部分には、条件を判断するための式を入れます。条件式の書き方はいくつかありますが、表3.1の記号を使って2つの値を比較することがよくあります。比較に使う記号のことを「比較演算子」と呼びます。

ifとelseの間には、条件が成立したときに行う処理を入れます。複数の文からなる処理を入れてもかまいません。また、elseの後には条件が成立しなかったときの処理を入れます。

「{」と「}」の間の文は、リスト3.1のように字下げ（インデント）して、「ここは条件によって行われる処理だ」ということが分かりやすくなるようにすることをお勧めします。

なお、条件が成立しなかったときには何もしない場合は、else以降の部分を省略します。

図3.1　基本的な条件判断



リスト3.1　if文の基本型

```
01 if（条件）{
02      条件が成立したときの処理
03 }
04 else {
05      条件が成立しなかったときの処理
06 }
```

表3.1　比較演算子の種類

演算子	意味	例とその意味
A == B	AとBが等しい	if（$count == 10） （変数$countの値が10の場合）
A === B	AとBの値と型が等しい	if（$count === 10） （変数$countの値が10で、型が数値の場合）
A != B	AとBが異なる	if（$username != ″tanaka″） （変数$usernameの値が「tanaka」ではない場合）
A !== B	AとBの値か型が異なる	if（$username !== ″tanaka″） （変数$usernameの値が「tanaka」ではないか、型が文字列でない）
A > B	AがBより大きい	if（$count > $oldcount） （変数$countの値が、変数$oldcountの値より大きい場合）
A < B	AがBより小さい	if（$username < ″yamada″） （変数$usernameの値が文字コード順で「yamada」より前の場合）
A >= B	AがB以上	if（$count >= 10） （変数$countの値が10以上の場合）
A <= B	AがB以下	if（$count <= 20） （変数$countの値が20以下の場合）

●「{」「}」の省略

「{」と「}」の間に行う処理が1つの文だけで済む場合は、「{」と「}」を省略することができます。

ただ、後で処理を追加すると「{」と「}」を入れなければならなくなりますので、1つの文だけの場合も、はじめから「{」と「}」を入れておくことをお勧めします。

●テンプレート的なif文の書き方

WordPressのテンプレートのように、PHPファイルのあちこちに、「<?php」と「?>」

が部分的にでてくるような書き方をすることもあります。その場合、リスト3.1の書き方でif文を表すとすると、リスト3.2のようになります。ただ、この書き方だとif文の最後が「<?php } ?>」で終わっていて、非常に分かりにくいです。

そこで、リスト3.2の代わりに、リスト3.3のように書くこともできます。「{」を「:」に置き換え、elseの前の「}」を削除し、最後の「}」を「endif;」に置き換えます。リスト3.3の書き方は、リスト3.2と比べると、if文の最初と最後の対応が分かりやすくなります。

なお、WordPressのデフォルトテーマでは、リスト3.3のような書き方を使っています。

リスト3.2　if文を分けた書き方

```
01 <?php if (条件) { ?>
02     条件が成立した時に出力する内容
03 <?php } else { ?>
04     条件が成立しなかった時に出力する内容
05 <?php } ?>
```

リスト3.3　リスト3.2の別の書き方

```
01 <?php if (条件) : ?>
02     条件が成立した時に出力する内容
03 <?php else : ?>
04     条件が成立しなかった時に出力する内容
05 <?php endif; ?>
```

●文字列の比較

if文では文字列どうしを比較することもできます。等しい（==）／異なる（!=）だけでなく、大小を比較することもできます。

文字列の大小は、「文字コード順で並べ替えたときに、コードが後ろにある方が大きい」というように判断します。文字コードというのは、1つひとつの文字につけられているコード番号のことです。

アルファベットはABCの順に文字コードがつけられていますので、英単語どうしを比較すれば、辞書順で後ろに来るものほど大きいと判断されます。

ただ、大文字と小文字を混在させた場合、小文字の方が文字コードが大きいので、辞書順にならない場合もあります。たとえば、「SUZUKI」と「sasaki」を比較すると、「sasaki」の方が大きいと判断されます。

一方、漢字の文字コードは必ずしもあいうえお順で並んでいるわけではないので、

漢字どうしを比較した場合の結果は、比較する漢字によって異なります。

●「=」と「==」の間違いに注意

「等しい」を表す比較演算子は「==」ですが、つい「=」と間違えることがよくありますので、注意が必要です。「=」にしてまうと、代入が行われて比較にはならないので、スクリプトが思わぬ動作をすることになります。

また、このような間違いを防ぐ方法として、「==の左辺には、代入できないもの（計算式など）を書く」ということも、広く行われています。

例えば、「変数$aの値が10に等しい」という条件判断をするとします。この場合、以下のように書くと、「==」を「=」に間違えると、「変数$aに10を代入した上で、その変数$aが『true』である」という条件判断を行うことになってしまいます（「true」については後述）。

```
if ($a == 10)
```

一方、この条件判断を以下のように書いた場合、「==」を「=」に間違えると、「Parse error」（文法解釈エラー）というエラーが発生し、プログラムの実行がそこでストップします。

```
if (10 == $a)
```

ちなみに、WordPressのコアのプログラムも、if文はここで述べたような書き方を使っています。

■if文の例

if文を使った簡単な例として、現在の時刻を表示する例を紹介します。その際に、if文を使って、現在の時刻が午前か午後かを判断し、表示するようにします。

●プログラムの組み方

実際に、if文を使ってプログラムを組むと、以下リスト3.4のようになります。各行の意味は以下の通りです。

①時刻を読み込む（1～4行目）

変数$nowtimeに、現在の時刻を読み込みます。「getdate」という命令(※)を使って、現在の時刻を読み込んでいます。

そして、変数$nowtimeから、現在の時／分／秒を、それぞれ変数$hour24／$min／$secに代入しています。

なお、「$nowtime['hours']」など、変数名の後に「[」を付ける書き方は、「配列」と呼びます。配列については、後の第4章で解説します。

②午前か午後かを判断する（5行目）

時刻を取り出した後、24時制の時を表す変数（$hour24）が12未満かどうかを調べて（リストの5行目）、処理を分けます。

③午前の場合の処理（6～7行目）

$hour24の値が12未満の場合は、現在の時刻が午前であることを意味します。そこで、変数$ampmに「午前」の文字を代入します（6行目）。

また、午前では12時制の時は24時制の時と同じなので、12時制の時を表す変数（$hour12）に$hour24の値をそのまま代入しています（7行目）。

④午後の場合の処理（10～11行目）

一方、$hour24の値が上記以外の場合は午後ですので、変数$ampmに「午後」を代入しています（10行目）。

また、午後では12時制の時は24時制の時から12を引いた値になりますので、$hour12には$hour24から12を引いた値を代入しています（11行目）。

⑤時刻の表示（13～17行目）

最後に、echo命令を使って、12時制と24時制で時刻を表示します。

リスト3.4　現在の時刻を24時制と12時制で表示する

```
01 $nowtime = getdate();
02 $hour24 = $nowtime['hours'];
03 $min = $nowtime['minutes'];
04 $sec = $nowtime['seconds'];
05 if ($hour24 < 12) {
06     $ampm = '午前';
07     $hour12 = $hour24;
08 }
09 else {
10     $ampm = '午後';
11     $hour12 = $hour24 - 12;
12 }
13 echo <<< HERE
```

(※)　getdateは、厳密には「関数」の一種です。関数については、後の第6章で解説します。

```
14 現在の時刻：<br />
15 24時制では${hour24}時${min}分${sec}秒です<br />
16 12時制では${ampm}${hour12}時${min}分${sec}秒です
17 HERE;
```

●サイトのトップページに現在時刻を表示する

WordPressのテンプレートにリスト3.4を組み込めば、サイトのページに、そのページがアクセスされた時点の時刻を表示することができます。

例えば、WordPressのデフォルトテーマで、index.phpテンプレートに、リスト3.5の中で行番号の左に網掛けがされている部分を追加すると、トップページの記事一覧の前に現在時刻が表示されるようになります（画面3.1）。

なお、リスト3.5を含むindex.phpテンプレートのサンプルは、「part03」→「if1」フォルダの「index.php」ファイルにあります。

リスト3.5　WordPressのデフォルトテーマでindex.phpにリスト3.4を追加した例

```
01 ・
02 ・（以前略）
03 ・
04 get_header(); ?>
05
06     <div id="content" class="narrowcolumn" role="main">
07
08 <?php
09 $nowtime = getdate();
10 $hour24 = $nowtime['hours'];
11 $min = $nowtime['minutes'];
12 $sec = $nowtime['seconds'];
13 if ($hour24 < 12) {
14     $ampm = '午前';
15     $hour12 = $hour24;
16 }
17 else {
18     $ampm = '午後';
19     $hour12 = $hour24 - 12;
20 }
21 echo <<< HERE
22 現在の時刻：<br />
23 24時制では${hour24}時${min}分${sec}秒です<br />
24 12時制では${ampm}${hour12}時${min}分${sec}秒です
```

★配列変数の基本

同じ性質を持った大量のデータがある場合には、「配列変数」という変数を使って処理すると便利です。この節では、配列変数の基本を解説します。

■普通の変数では一連のデータを処理しにくい

たとえば、あなたのWebサイトについて、「どのページが一番好きですか」というアンケートを取るようなページを作るとしましょう。

ここで、各ページの得票数を変数で管理することを考えてみてください。これまでの知識で作るとすると、それぞれのページの得票数に対応して、$point01、$point02、$point03・・・のような変数を作って、投票されるごとに、ページに対応する変数の値を1つ増やす、というような手順になるでしょう。

フォームのselectを使って好きなページを選んでもらうようにし、フォームで「送信」ボタンがクリックされたときに、ページに対応した番号が送信されてくるようにするとします（リスト4.1）。また、送信されたページ番号は、PHPのスクリプトで変数$selpageに代入されるものとします。この場合、リスト4.2のようなスクリプトで投票数をカウントすることができます。

ただ、見ても分かるように、このスクリプトは非常に面倒です。それぞれのページに対応してif文を書くことが必要で、ページが増えれば増えるほど面倒さが増します。

このように、同じような意味を持つ一連のデータを処理する場合、普通の変数を使うと非常に不便なことになります。

リスト4.1　好きなページを選んでもらうフォーム

```
01 <form name="anquete" method="post" action="anquete.php">
02     <p>
03         <select name="selpage">
04             <option value="1">○○のページ</option>
05             <option value="2">□□のページ</option>
06             <option value="3">△△のページ</option>
07 ・
08 ・（途中省略）
09 ・
```

```
10          </select><br />
11          <input type="submit" value="送信" />
12      </p>
13 </form>
```

リスト4.2　投票数をカウントする

```
01 <?php
02      ・
03      ・（各種の処理）
04      ・
05      // 番号1のページが選ばれた場合
06      if ($selpage == 1) {
07          $point01++;
08      }
09      // 番号2のページが選ばれた場合
10      elseif ($selpage == 2) {
11          $point02++;
12      }
13      // 番号3のページが選ばれた場合
14      elseif ($selpage == 3) {
15          $point03++;
16      }
17      ・
18      ・（各種の処理）
19      ・
20 ?>
```

■配列変数で一連のデータを効率よく処理する

このようなときに「配列変数」を使うと、処理を非常にすっきりさせることができます。配列変数は、一連のデータ全体に1つの名前をつけて、個々のデータを番号などで指定できるようにするものです。

●配列変数のイメージ

配列変数は以下のような形で表します。「キー」とは、要素の番号を表す数値のことです。

```
変数名[キー]
```

たとえば、前述の投票の場合、得票数を「$point」という配列変数で表すものとす

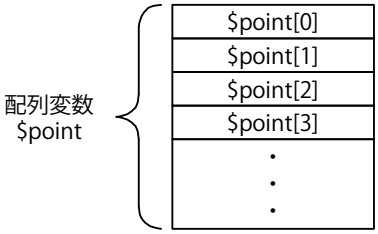
ると、1ページ目の得票数は「$point[1]」、2ページ目の得票数は「$point[2]」のように表すことができます（図4.1）。

配列変数内の個々の要素は、通常の変数と同じように扱うことができます。たとえば、配列変数$pointの1番の要素に10という値を代入するには、

```
$point[1] = 10;
```

のように書きます。

図4.1　配列変数のイメージ



●キーを変数で指定する

キーは変数で指定することもできます。たとえば、変数$iの値が5の時に、以下の文を実行すると、配列変数$pointの5番の要素の値が1つ増えます。

```
$point[$i]++;
```

この「変数で要素を指定できる」ことから、配列変数を使うと一連のデータを効率よく処理することができます。

たとえば、先ほどの投票の処理を配列変数で書き直すことを考えて見ましょう。得票数を配列変数$pointで表すとすると、ページが選ばれたときには、そのページの番号（変数$selpage）に対応する要素の値を1つ増やせばOKです。

したがって、リスト4.2の長いif文は、配列変数を使って書き直すと以下の1行になり、非常に簡潔になります。

```
$point[$selpage]++;
```

■配列変数への代入方法

前述したように、配列変数を作って値を代入するには、以下のように変数名とキーを指定します。キーは0以上の整数で指定します。

```
$point[1] = 10;
```

また、配列を初期化する場合など、たくさんの要素に一度に値を代入する時は、「array」という命令を使って以下のように書くこともできます。

```
配列変数名 = array(0番の要素の値, 1番の要素の値, ・・・ n番目の要素の値);
```

たとえば、リスト4.3のスクリプトは、リスト4.4のように書き直すことができます。

なお、リスト4.3のように、配列の要素の番号は、通常は0番から順につけていくようにします。

リスト4.3　配列変数の1つひとつの要素に順に代入する

```
01 $ar[0] = 10;
02 $ar[1] = 20;
03 $ar[2] = 30;
```

リスト4.4　array命令を使って一度に代入する

```
$ar = array(10, 20, 30);
```

●array命令でキーを指定して代入する

array命令で普通に代入すると、0番の要素から順に値が代入されていきます。しかし、キーを指定して値を代入することもできます。それには、以下のような書き方をします。

```
配列変数名 = array(キーA => 値A, キーB => 値B, ・・・ ,キーX => 値X);
```

たとえば、以下の文を実行すると、$ar[1]に「yamada」、$ar[3]に「tanaka」、$ar[6]に「suzuki」が代入されることになります。

★URLを使ってページ間でデータを受け渡す（GET）

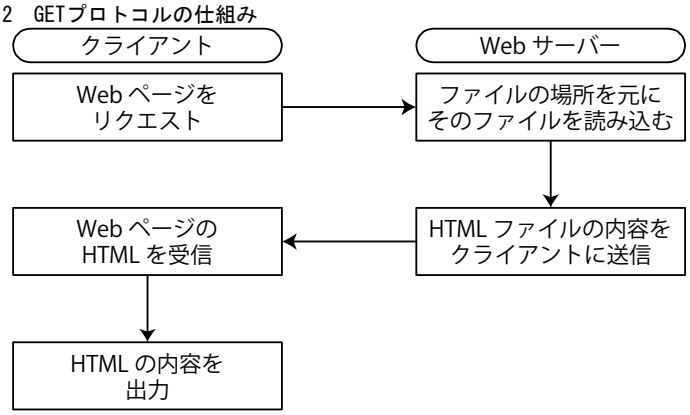
複数のWebページを連携して処理を進める場合、ページ間でデータを受け渡すことが必要になります。その方法はいくつかありますが、その1つとして、URLを使う方法があります。

■URLでのデータのやり取りの仕組み

WebサーバーとWebブラウザとの間のデータのやり取りは、「HTTP」（Hypertext Transport Protocol）というプロトコルに沿って行われます。

HTTPのプロトコルはいくつかありますが、その中で特によく使われるものが、「GET」というプロトコルです。GETでは、Webブラウザから Webサーバーに対し、URLに基づいて、サーバー上のファイルの場所を送信します。そして、サーバーがそれに反応してファイルの内容を送り返します（図5.2）。

図5.2　GETプロトコルの仕組み



GETプロトコルでは、URLでWebページのアドレスを指し示しますが、それとともにWebサーバーに各種のデータを送信することもできます。

アドレスの後に「?」を書き、そのあとに「データ名=値」のように続けると、データを渡すことができます。複数のデータを渡したい場合は、データと値の組を複数ならべ、それぞれの間を「&」で区切ります。ただし、XHTMLファイルの中で「&」を使っ

てデータを区切る場合は、文字実体参照の「&」を使います。
たとえば、以下のリンクがクリックされたとします。

```
<a href="http://www.abc.com/somepage.php?var1=5&amp;var2=10">次のページ</a>
```

すると、「http://www.abc.com/sompage.php」に対応するスクリプトが実行され、「var1」と「var2」という名前のデータが渡されます。データの値はそれぞれ5と10になります。
なお、値として半角英数字以外を渡す場合は、「URLエンコード」という方法を使って半角英数字に変換することが必要です。URLエンコードの方法は後述します。

■PHPでのデータのやりとりの方法

URLを使ってデータをやり取りする場合、以下のような手順を取ります。

●他のページにデータを渡す

URLでデータを渡す場合、前述したように、一般にはリンクを使います。
また、リンクでデータを渡す場合、渡すデータに応じて、リンクのURLを変化させることになります。リンクのタグをスクリプトで出力すれば、状況に応じて渡す値を変えることができます。
たとえば、以下のような文を作ると、somepage.phpに移動し、変数$pageの値を情報として渡すリンクが出力されます。

```
echo "<a href=¥"somepage.php?page=$page¥">$pageページへ移動</A>";
```

●他のページからデータを受け取る

URLでデータが渡された場合、「$_GET」という名前の連想配列に値がセットされるようになっています。連想配列の個々のキーが、渡されたデータの名前になります。
たとえば、以下のリンクがクリックされて、somepage.phpファイルが開かれたとします。この場合、somepage.phpのスクリプトの中では、「$_GET['foo']」の値が100になり、「$_GET['bar']」の値がabcになります。

```
<a href="somepage.php?foo=100&amp;bar=abc">1ページへ移動</a>
```

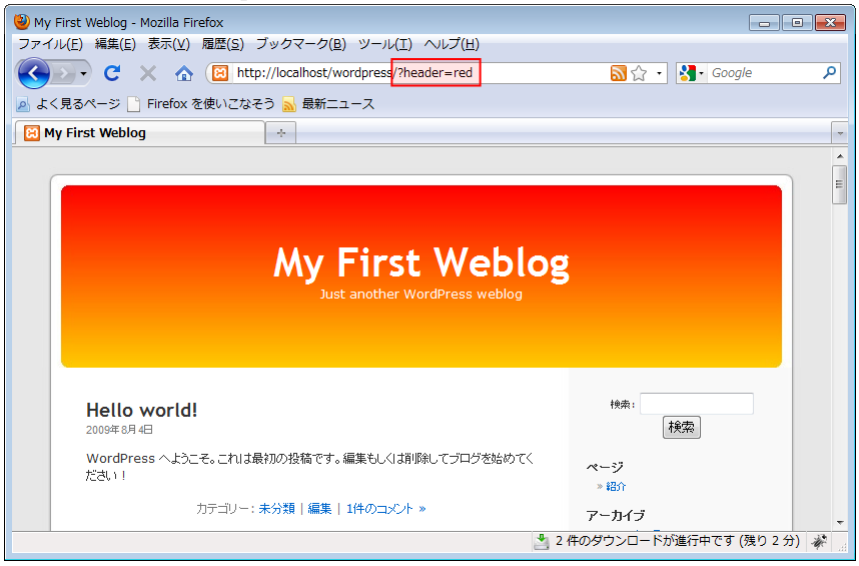
■URLに指定された値でヘッダーの背景画像を切り替える

URLでデータを受け渡しする例として、WordPressのデフォルトテーマを対象に、URLに「?header=red」などと指定することで、ページのヘッダー部分の背景画像の色を変える例を紹介します（画面5.3）。

なお、色は「red」「green」「yellow」「pink」「cyan」の5色を指定できるようにします。

また、各色の背景画像は、サンプルファイルの「part05」→「header」→「images」フォルダに用意してあります。これらの画像を、デフォルトテーマのディレクトリにある「images」ディレクトリにアップロードして使います。

画面5.3　URLに「?header=red」と付加するとヘッダー画像が赤系になる



●カスタマイズの考え方

WordPressのデフォルトテーマでは、ページのヘッダー部分は、「header」というIDがついたdiv要素になっています。また、スタイルシートで、このdiv要素に対して背景画像を指定するようになっています。

スタイルシートを組み込む部分（link要素）は、テーマの「ヘッダー」のテンプレ

ート（header.php）にあります。ヘッダーテンプレートを書き換えて、link要素の直後に、ヘッダー部分の背景画像を上書きするためのstyle要素を出力すれば、画面5.3のように背景画像を変えることができます。例えば、ヘッダーの背景画像を赤にするなら、リスト5.2のようにします。

ただ、リスト5.2では、画像のアドレスが決めうちになっています。この部分をPHPに置き換え、ページのURLの「header=色」から色の情報を読み込んで、色に応じた背景画像を出力するようにします。

リスト5.2　背景画像を赤にする

```
・
・（以前略）
・
<link rel="stylesheet" href="http://WordPressのインストール先/wp-content/themes/headerimg/style.css" type="text/css" media="screen" />
<style type="text/css">
#header {
    background-image : url(http://WordPressのインストール先/wp-content/themes/headerimg/images/header_red.jpg);
}
</style>
・
・（以後略）
・
```

●ヘッダーテンプレートの書き換え

ここまでの話に基づいて、ヘッダーのテンプレートを実際に書き換えると、リスト5.3のようになります。白抜き文字の部分を、ヘッダーのテンプレートに追加します。

リスト5.3　ヘッダーのテンプレートの書き換え方

```
・
・（以前略）
・
<title><?php wp_title('&laquo;', true, 'right'); ?> <?php bloginfo('name'); ?></title>

<link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>" type="text/css" media="screen" />
<?php
$color = $_GET['header'];
$colors = array('red', 'green', 'yellow', 'cyan', 'pink');
```

★ユーザー定義関数を作る

スクリプトのあちこちで同じような処理を行う場合、それをユーザー定義関数にすることで、スクリプト作りの効率を上げることができます。

■ユーザー定義関数の基本

ユーザー定義関数を使うには、まずその関数を定義し、別の箇所にその関数を呼び出す処理を書きます。

●関数の定義

ユーザー定義関数を定義するには、リスト6.6のような書き方をします。「関数名」で関数の名前を決めます。関数名のつけ方は変数名と同じで、英数字およびアンダースコア（_）を使います。

ユーザー定義関数に引数を渡せるようにするには、括弧の中に引数名を順に並べます。引数名は、変数名と同じように「$」から始まる名前でつけます。ユーザー定義関数の中では、渡された引数は変数と同じように使うことができます。

そして、「{」から「}」の間に、関数で行う処理を書きます。この部分は、一般のスクリプトとまったく同じように書くことができます。PHP組み込み関数や他のユーザー定義関数を呼び出したり、ifやforなどの文を使ったりすることができます。

リスト6.6　ユーザー定義関数の書き方

```
01 function 関数名(引数1, 引数2, ･･･ , 引数n) {
02     関数の処理内容
03 }
```

●戻り値を返す

関数から呼び出し元に戻り値を返したい場合、関数の中に「return 戻り値;」の文を入れます。関数の実行中にreturn文があると、関数がそこで終了して、呼び出し側に戻り値が返されます。

return文は、関数の最後に書くことが多いです。ただ、条件によって別々の戻り値

を返すこともあります。その場合は、if文等の条件判断の中にreturn文を入れます。

なお、単に「return;」とだけ書くと、戻り値を返さずに、関数から抜けることができます。

●関数を呼び出す

ユーザー定義関数は、PHP組み込みの関数とまったく同じように使うことができます。

例えば、「my_func」という名前の関数を定義してあるとします。また、その関数は1つの引数を取るとします。また、戻り値も返されるとします。この場合、以下のように書くと、変数$xの値を引数としてmy_func関数に渡し、戻り値を変数$yに代入することになります。

```
$y = my_func($x);
```

■ユーザー定義関数の例

ユーザー定義関数を使った簡単な例として、「投稿の文章から先頭部分のテキストを抜き出して出力する」という関数を作ってみます。

●例の関数の動作

WordPressのデフォルトテーマで、メインインデックスのテンプレート（index.php）に、「the_content_first」という名前の関数を追加します。そして、the_contentタグをthe_content_first関数に置き換えて、メインページには投稿の先頭部分を一覧表示するようにします。

例えば、以下のように書くと、投稿の文章から先頭20文字だけを出力できるようにします。また、投稿の元々の文章が20文字を超えている場合は、切り落とした文章の代わりに「...」の文字を出力するようにします（画面6.6）。

```
the_content_first(20);
```

画面6.6　投稿の文章の先頭部分だけを出力する

●関数の内容

実際に関数を作ると、リスト6.7のようになります。

まず、WordPressコアの「get_the_content」という関数で、投稿の内容を取り出し、変数$contentに代入します（3行目）。そして、PHPのstrip_tags関数を使って、その内容からHTMLのタグをすべて取り除きます（5行目）。

次に、mb_substr関数を使って、$contentの文字列の先頭から、引数の$countで示す文字数だけ、文字を取り出します（9行目）。そして、元の文字列の文字数が取り出した後の文字列より多い場合は、取り出した文字列の後に「...」を連結します（12～14行目）。

最後に、echo命令を使って、ここまででできた文字列（変数$content）を出力します（16行目）。

なお、リスト6.7の関数は、メインインデックステンプレートの最後に追加します。

リスト6.7　the_content_first関数

```
01 function the_content_first($count) {
02     // 投稿の文章を得る
03     $content = get_the_content();
04     // 文章からHTMLのタグを取り除く
05     $content = strip_tags($content);
06     // 文章の文字数を調べる
07     $len = mb_strlen($content, 'utf-8');
08     // 文章の先頭から、$count文字分を取り出す
09     $content = mb_substr($content, 0, $count, 'utf-8');
10     // 文章の文字数が$count文字より多い場合は、
11     // 取り出した先頭部分の後に「...」を追加する
12     if ($len > $count) {
13         $content .= ' ...';
14     }
15     // 取り出した先頭部分を出力する
16     echo $content;
17 }
```

●関数の呼び出し側の書き換え

次に、メインインデックステンプレートの中で、the_contentテンプレートタグの箇所を、上で作ったthe_content_first関数に置き換えます。

例えば、各投稿の先頭20文字だけを表示するようにするには、リスト6.8の7行目の網掛け部分を書き換えます。the_content_first関数に、引数として「20」を渡していますので、最初の20文字だけが出力されます。

リスト6.8　the_content_first関数を使うように書き換える

```
01 ・
02 ・（以前略）
03 ・
04 <small><?php the_time(__('F jS, Y', 'kubrick')) ?> <!-- by <?php the_author() ?> --></small>
05
06 <div class="entry">
07     <?php the_content_first(20); ?>
08 </div>
09
10 <p class="postmetadata"><?php the_tags(__('Tags:', 'kubrick') . ' ', ', ', '<br />'); ?>・・・途中略</p>
11 ・
12 ・（途中略）
13 ・
```

★オブジェクトの基本

まず、オブジェクト指向の基本的な考え方や、PHPでのオブジェクト指向プログラミングの方法など解説します。

■オブジェクトとは

プログラムの中では、いろいろな「物」を扱います。たとえば、WordPressのプログラムではデータベースを操作する機会がありますが、データベースも「物」の一種です。

「オブジェクト」(object) とは、日本語で言えばそのものずばり「物」です。そして、「オブジェクト指向」とは、物を中心にしてプログラムを作っていこうという考え方を指します。

●プロパティとメソッド

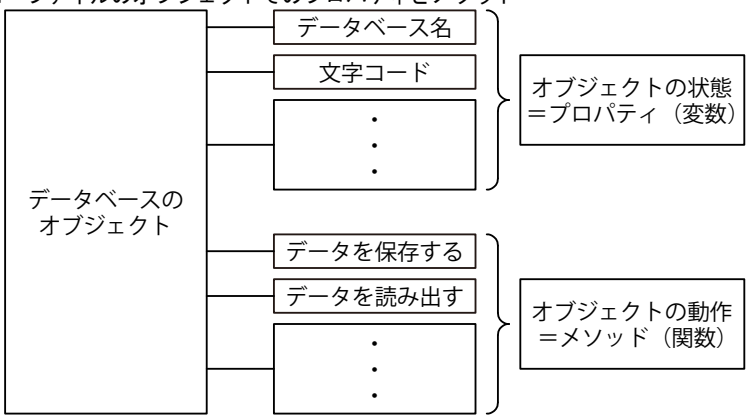
プログラムからデータベース等のオブジェクトを操作する場合、大きく分けて、以下のようなことを行います。

①オブジェクトの状態を調べたり、変えたりする
②オブジェクトに、それ自身が可能な動作を行わせる

たとえば、データベースの場合だと、状態にあたるものとして、「データベース名」や「データベースの文字コード」があげられます。また、動作として「データを保存する」「データを読み出す」といったことがあげられます。

オブジェクトの状態は、変数のようにして管理することができます。この変数のことを、オブジェクト指向の一般的な用語では、「プロパティ」(Property) と呼びます。また、オブジェクト自身が可能な動作は、関数として扱うことができます。これを「メソッド」(Method) と呼びます (図7.1)。

図7.1　ファイルのオブジェクトでのプロパティとメソッド



●オブジェクト指向プログラミングの流れ

ここまでで述べたように、オブジェクトはプロパティとメソッドを持った「物」です。そして、このオブジェクトを使ってプログラムを作っていくのが、オブジェクト指向プログラミングの考え方です。実際には、以下のような手順でプログラムを作っていくことになります。

①オブジェクトのひな型として、プロパティやメソッドを定義します。
②オブジェクトを初期化します。
③オブジェクトのプロパティ（変数＝状態）を調べたり変えたりします。
④オブジェクトのメソッド（関数＝動作）を呼び出して、処理を行わせます。

また、上では「ひな型」という言葉を使いましたが、オブジェクト指向の用語では、ひな型のことを「クラス」(Class) と呼びます。

たとえば、データベースをオブジェクトとして扱う場合、以下のような手順をとります。

①データベースオブジェクトのクラスを作り、「データベース名」や「文字コード」といったプロパティと、「データを書き込む」「データを読み出す」などのメソッドを定義しておきます。
②データベースオブジェクトを作成して初期化します。
③データベースオブジェクトの「データベース名」や「文字コード」などのプロパティを使って、データベースの状態を設定したり調べたりします。

④データベースオブジェクトの「データを書き込む」や「データを読み出す」などのメソッドを使って、ファイルを読み書きします。

■オブジェクトを使ってみる

ここまでで、オブジェクトを定義してプログラムを作る際の流れを述べました。ただ、いきなり自分でオブジェクトを定義するのは、簡単だとは言えません。

一方、オブジェクト指向プログラミングでは、既存のオブジェクトを利用して、プログラム（スクリプト）を作っていくこともできます。その場合、前述の「オブジェクト指向プログラミングの流れ」のうち、難しいクラス定義の作業は不要で、その後の作業だけを行えば良いです。

オブジェクト指向に慣れるために、まずはWordPressの「wp_query」というオブジェクトを使ってスクリプトを作ってみましょう。

●WP_Queryクラスの概要

この章の冒頭でも述べたように、WordPressではところどころでオブジェクト指向を取り入れていて、いくつかのクラスが定義されています。その中から、「WP_Query」というクラスを取り上げ、オブジェクトを利用する場合のプログラミングの方法を紹介します。

WP_Queryクラスは、WordPressループをつかさどるクラスです。WP_Queryクラスを使うと、WordPressのデータベースから、さまざま条件で投稿を読み込むことができます。そして、読み込んだ投稿を元にWordPressループを実行することができます。

ここでは例として、サイト内の投稿をすべて読み込んで、タイトルと日付を一覧表示するスクリプトを作ってみます。また、タイトルは個々の投稿のページにリンクするようにします（画面7.1）。

画面7.1　サイト内の投稿を一覧表示する

●オブジェクトの新規作成

PHPのオブジェクト指向プログラミングでは、オブジェクトを新規作成する際には「new」という演算子を使います。new演算子の書き方は以下の通りです。

```
変数 = new クラス名(引数);
```

「変数」には、作成したオブジェクトを割り当てるための変数を指定します。といっても、オブジェクト用に特殊な変数があるわけではなく、通常の変数を指定します。

また、「引数」には、オブジェクトの初期化の際に必要な値を渡します。その値はクラスによって異なります。

WP_Queryクラスの場合だと、引数として、読み込む投稿を指定するための条件を書きます。主な条件の指定方法は表7.1の通りです。また、「orderby=キー」の条件で使う並べ替えのキーは、表7.2のように指定します。

```
変数 = new WP_Query('条件名=値&条件名=値・・・');
```

例えば、以下のように書くとします。すると、日付（orderby=date）の新しい順

★基本的なデータの取り出し方

データベースに対する操作はいろいろありますが、その中で最も基本的な操作は、データを取り出すことです。この節では、SQLでデータを取り出す方法のうち、基本的な部分を解説します。

■すべてのレコードを取り出す

RDBMSでは、射影／選択／結合（339ページ参照）を組み合わせて、データベースからさまざまな形でデータを取り出すことができます。SQLでは、データの取り出しは、すべて「SELECT」という命令で表します。

SELECT文のもっとも基本的な操作は、射影／選択／結合を行わずに、テーブルからすべてのレコードを取り出すことです。この場合の書き方は、以下のようになります。

```
SELECT * FROM テーブル名
```

例えば、投稿のテーブル（wp_posts）からすべてのレコードを取り出すには、以下のSQLを実行します。

```
SELECT * FROM wp_posts
```

●大文字／小文字について

MySQLでは、SQLの命令や、テーブル名／フィールド名は、大文字／小文字のどちらで書いてもかまいません。例えば、今あげた例だと、以下のようにすべて小文字で書いても動作します。

```
select * from wp_posts
```

また、以下のようにすべて大文字で書いても動作します。

```
SELECT * FROM WP_POSTS
```

ただ、WordPressでは、SQL文は以下の慣習に従って書くようになっています。

①MySQLの予約語（命令や関数など）は、大文字で書きます。
②テーブル名やフィールド名は、定義されている通り（大文字なら大文字、小文字なら小文字）に書きます。

そこで、本書のこれ以後の解説でも、MySQLの予約語は大文字で書くことにします。

■フィールドを限定する（射影）

テーブルからデータを取り出す際に、一部のフィールドに限定することができます。これを「射影」と呼びます（339ページ参照）。射影を行う場合、以下のようにして、SELECT文にフィールド名を指定します。

```
SELECT フィールド名, フィールド名, ・・・, フィールド名 FROM テーブル名
```

例えば、ユーザーのテーブル（wp_users）には、表8.1のようなフィールドがあります。このテーブルから、ユーザーのIDとユーザー名を取り出すには、以下のようなSELECT文を書きます（画面8.7）。

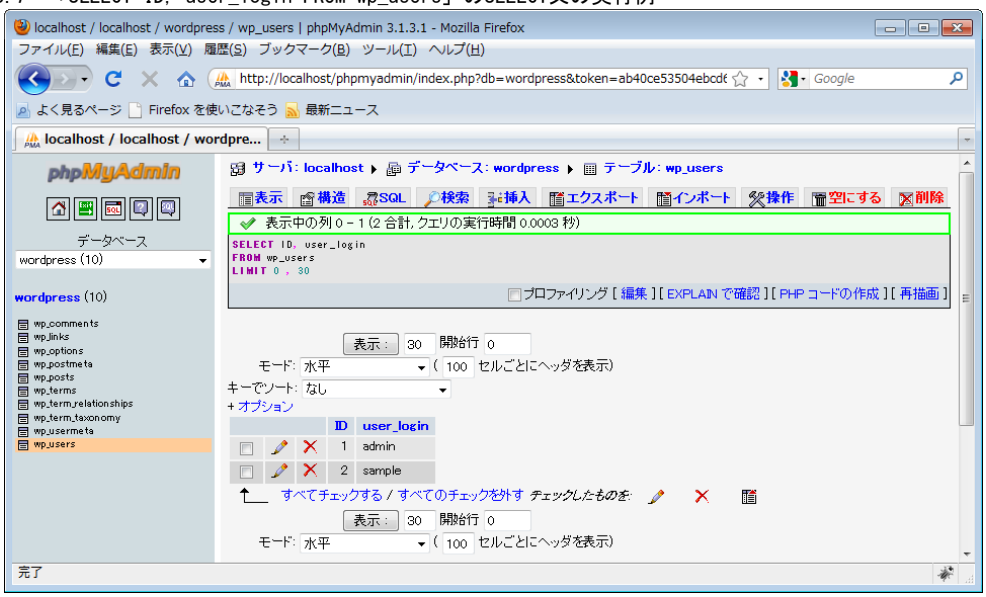
```
SELECT ID, user_login FROM wp_users
```

なお、350ページの「すべてのレコードを取り出す」のところで、SELECTの後に「*」を書きました。この「*」は「すべてのフィールド」を表す記号です。

表8.1 ユーザーのテーブル（wp_users）の主なフィールド

フィールド	内容
ID	ID
user_login	ユーザー名
user_pass	暗号化されたパスワード
user_email	メールアドレス
user_url	サイトのアドレス

画面8.7　「SELECT ID, user_login FROM wp_users」のSELECT文の実行例



■レコードを限定する（選択）

テーブルからデータを取り出す際に、レコードを限定することもできます。これを「選択」と呼びます（340ページ参照）。

選択を行うには、SELECT文に「WHERE」という句を追加し、レコードを選択するための条件を指定します。「SELECT」と「FROM」の間にフィールド名のリストを書けば、フィールドを限定することもできます（射影）。

```
SELECT フィールド名, フィールド名, ・・・, フィールド名 FROM テーブル名 WHERE 条件
```

また、フィールド名に「*」を指定すれば、テーブル内のすべてのフィールドを取り出すことができます。

●フィールドを一定の値と比較する

条件の書き方は多数ありますが、フィールドを一定の値と比較する書き方をよく使います。その場合の条件の書き方は、表8.2のようになります。「等しい」の条件は「=」で表し、PHPの「==」とは異なります。

比較対象の値が文字列である場合は、その前後を「'」で囲みます。また、比較対象の値が日付や時刻である場合、「'2009-01-01 12:34:56'」のように、年月日時分秒を書いて前後を「'」で囲みます。

表8.2　条件の書き方

条件	条件の表し方
フィールドの値が○○に等しい	フィールド名 = ○○
フィールドの値が○○ではない	フィールド名 <> ○○
フィールドの値が○○より大きい	フィールド名 > ○○
フィールドの値が○○より小さい	フィールド名 < ○○
フィールドの値が○○以上	フィールド名 >= ○○
フィールドの値が○○以下	フィールド名 <= ○○

例えば、投稿のテーブル（wp_posts）には、表8.3のようなフィールドがあります。このテーブルから、post_date（日付）が2009年8月1日以降の投稿を取り出し、その中からID／post_title（タイトル）／post_date（日付）のフィールドを取り出すとします。この場合、SELECT文はリスト8.1のようにします（画面8.8）。

なお、リスト8.1のサンプルは、「part08」→「simple」フォルダの「compare.sql」ファイルです。

表8.3　wp_postsテーブルの主なフィールド

フィールド	内容
ID	投稿のID
post_author	投稿したユーザーのID
post_content	本文
post_title	タイトル
post_status	公開状況（公開、下書きなど）
post_date	公開日時
post_modified	最終更新日時
post_type	タイプ（記事／WordPressページなど）
post_mime_type	MIMEタイプ
comment_count	その投稿についているコメントの数

リスト8.1　日付で条件を指定して投稿を取り出す

```
01 SELECT ID, post_title, post_date
02 FROM wp_posts
03 WHERE post_date >= '2009-08-01'
```

★特定カテゴリだけのアーカイブを作る

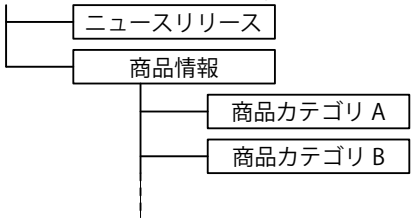
データベースに直接にアクセスする1つの例として、「特定カテゴリだけのアーカイブを作る」という事例を紹介します。

■この節で作る例

WordPressで小規模な企業サイトを作る場合、カテゴリの構造を図9.1のようにして、ニュースリリースと商品情報を分ける、という手法を取ることが考えられます。

この状態で、wp_get_archivesタグを使って、サイドバーに月別アーカイブリストを出力すると、ニュースリリースと商品情報が混在した月別アーカイブリストができます。しかし、サイトの構造からすると、ニュースリリースだけの月別アーカイブを出力する方が望ましいと思われます。

図9.1　企業サイトのカテゴリの構造の例



そこで、この節では「ニュースリリース」カテゴリだけの月別アーカイブリストを、サイドバーに出力できるようにします（画面9.2）。

また、サイドバーでニュースリリースの月別アーカイブのリンクをクリックしたときに、月別のニュースリリースの一覧（タイトル／日付／抜粋）が表示されるようにします（画面9.3）。

月別ニュースリリースページのアドレスは、「http://WordPressのインストール先のアドレス/newsrelease.php?y=年&m=月」のようになるようにします。

なお、「newsrelease.php」は、テンプレートではなく、WordPressの機能を使う独立したスクリプト（105ページ参照）として作ります。そして、WordPressのインストール先ディレクトリに入れるようにします。こうすることで、「http://WordPressのイン

ストール先のアドレス/newsrelease.php?・・・」のアドレスでアクセスできるようになります。

画面9.2　サイドバーに「ニュースリリース」カテゴリだけの月別アーカイブを出力する

画面9.3　月別のニュースリリースの一覧

■「ニュースリリース」カテゴリだけの月別アーカイブリストを出力する

最初に、「ニュースリリース」カテゴリだけの月別アーカイブリストを、サイドバーに出力できるようにします。

●月別アーカイブリストの考え方

まず、カテゴリを限定せずに、すべての投稿から月別アーカイブリストを出力する方法を考えます。

月別アーカイブリストは、過去に投稿があった月と、その月に書いた投稿の数を、一覧で表示したものです。SQLのSELECT文を使って、以下のような集計を行えば、投稿があった月と、その月に書いた投稿の数を得ることができます。

①投稿のテーブルから、YEAR／MONTH関数（363ページ参照）を使って、投稿の年／月の値を取り出します。
②投稿を月単位でグループ化し、COUNT関数（371ページ参照）を使って、各月の投稿数をカウントします。
③公開されている投稿だけを、集計の対象にします。
④集計結果は年／月の降順に並べ替えます

投稿が公開されているかどうかは、wp_postsテーブルのpost_statusプロパティの値が「publish」であるかどうかで判断できます。また、wp_postsテーブルには投稿だけでなくWordPressページも含まれます。投稿だけを取り出すには、post_typeプロパティの値が「post」であるという条件も必要です。

ここまでの話に沿って、月別アーカイブリストの情報を得るためのSELECT文を作ると、リスト9.3のようになります。

1行目で投稿の年／月と、投稿のカウントを求めます。WHERE句（3行目と4行目）で、公開されている投稿だけを集計の対象にします。そして、GROUP BY句（5行目）で年／月でグループ化し、ORDER BY句（6行目）で年／月の降順で並べ替えています。

リスト9.3　月別アーカイブリストの情報を得るSELECT文

```
01 SELECT YEAR(post_date) AS post_year, MONTH(post_date) AS post_month, COUNT(*) AS m
   onthly_count
02 FROM wp_posts
03 WHERE post_status = 'publish'
```

```
04 AND post_type = 'post'
05 GROUP BY post_year, post_month
06 ORDER BY post_year DESC, post_month DESC
```

●「ニュースリリース」カテゴリの投稿だけを集計対象にする

次に、ここまでで作ったSELECT文を元にして、「ニュースリリース」カテゴリの投稿だけを集計の対象にします。

第8章の384ページで、カテゴリ毎の投稿を数えるSELECT文を紹介しました。このSELECT文は、投稿とカテゴリのテーブルを結合した上で、カテゴリのIDでグループ化して集計を行っていました。

その時のSELECT文に、カテゴリーの名前を限定する条件を追加すれば、そのカテゴリだけの集計を行うことができます。さらに、グループ化の方法を変えて、年月毎にグループ化するようにすれば、そのカテゴリだけの年月毎の投稿数をカウントすることができます。

実際にSELECT文を作ると、リスト9.4のようになります。このSELECT文の内容は、以下の通りです。

リスト9.4　「ニュースリリース」カテゴリだけの月別投稿数をカウントするSELECT文

```
01 SELECT YEAR(post_date) AS post_year, MONTH(post_date) AS post_month, COUNT(*) AS m
   onthly_count
02 FROM wp_posts p, wp_term_relationships tr, wp_term_taxonomy tt, wp_terms t
03 WHERE p.ID = tr.object_id
04 AND tr.term_taxonomy_id = tt.term_taxonomy_id
05 AND tt.term_id = t.term_id
06 AND p.post_status = 'publish'
07 AND p.post_type = 'post'
08 AND tt.taxonomy = 'category'
09 AND t.name = 'ニュースリリース'
10 GROUP BY post_year , post_month
11 ORDER BY post_year DESC, post_month DESC
```

①1行目

424ページのリスト9.3のSELECT文と同じで、投稿の年／月と、投稿の数を得ます。

また、これらの値のフィールドには、それぞれpost_year／post_month／monthly_countの名前を付けます。後でWordPressからデータにアクセスしますが、その際に各フィールドに名前が必要になりますので、名前をつけています。

②2行目