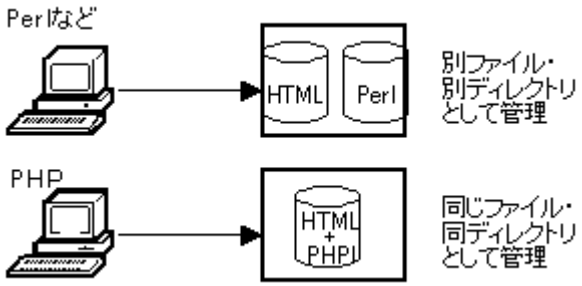
第４章　言語 PHP

言語 PHP とは

言語 PHP（正式には PHP HyperText Processor）は、CGI を実装する言語としては比較的単純で、かつCや Java 言語を学んだものにとってあまり違和感なく使える言語である。 もちろん、CGI には他の言語、たとえば Perl や Java が使われているケースも多い。言語の間の違いを理解すれば、容易に別の言語に移れるだろう。



PHP の特徴は

1. HTML 文書のテキストと、CGI プログラムとが、同じファイルに書かれる。
   Perl などの他の言語では、CGI プログラムは HTML テキストとは別ファイルに置かれる。<FORM>タグの action 引数によって、起動したい CGI プログラムのファイルを指定する。しかも通常は HTML 文書の置かれる場所とは異なるディレクトリにおく。PHP は、同じ場所に、つまり HTML と同じファイルの中に置く。PHP でも<FORM>タグに書かれた action 引数で起動することには変わりないが、その起動される PHP ファイルの中に、HTML のテキスト部分を含んで構わない。PHP のインタープリタ（解釈実行するソフト）は、PHP の部分は実行プログラムとみなして実行し、HTML タグの部分は HTML として表示する。
2. （Apache サーバーでは）、サーバープログラム内で実行されるので起動が早い。Perl の CGI プログラムでは、格段の仕込みをしない限りは、CGI プログラムは Web サーバーとは別のプロセスとして立ち上がる。そのときにプロセスの生成と初期化に関わるオーバーヘッドが生じ、アクセス負荷の高いサーバーではこのオーバーヘッドが性能低下をもたらす。それに比べて PHP は、Apache サーバープロセスの一部として実行されるため、このオーバーヘッドがない。

と言われている。
上記は利点として述べているが、その裏には欠点もあるわけで、1については、

①HTML 文書と PHP のプログラムが混在して区別しづらい点や、
②PHP のプログラムをページの一部として置いてあるためにセキュリティをかいくぐってソースを見ることが出来る、見られてしまう可能性がある、

という問題がある。
2については、Apache サーバーではあらかじめ CGI アプリケーションプログラムを走らせるためのプロセスをいくつか立ち上げておき、プログラムが起動されるときにそれを使うことによって回避できる。また Java の場合には、Apache ではなく、たとえば TomCat などのように全体を Java で書いたコンテナ型のサーバーを使い、CGI プログラムはその中へ取り込んで実行するものもある。
とにかく、一般には PHP は CGI プログラムを作成するのに簡便な言語であると評価されている。

PHP のオリジナルホームページは、 http://www.php.net にある。ダウンロードキットやマニュアルなどが整備されている。マニュアルは日本語版も用意されている。また、参考書の類もかなりの数が出版されてきており、書店でもよく見かける。

PHP にはバージョン4から5に移行しており、オブジェクト指向化の動きが強くなっているが、このプロジェクトでは大きなサーバープログラムを構築することは考えていないので、オブジェクト指向にはこだわらないことにする。流通しているパッケージはオブジェクト指向を使っているものが多い。ここでは多少古いが PHP4 を使っている。

PHP プログラムの基本

最初に、とにかく自分で、サンプルプログラムを動かしてみよう。まずは「フォーム」とは関係の無い、単純な例を試す。

［例題演習４－１］ PHP プログラムの実行を試してみる

次の PHP のプログラムを、ファイル（自ホーム）/public_html/mysample1.php として作り、ブラウザからアクセスしてみよ。このとき、必ずブラウザの URL 指定の欄に

http://venus.is.sci.toho-u.ac.jp/~(自分の ID)/mysample1.php

のように指定し、venus のファイルとして（Web サーバー経由で）アクセスすること。

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      echo "<p>Hello World</p>";
    ?>
  </body>
</html>
```

右図のような画面が得られるはずである。

（注意） Windows 上ファイルを作って、mysample1.php と名づけてセーブしたとき、Windows 上で（ファイル一覧から）mysample1.php ファイルをダブルクリック すると、ブラウザが起動してファイルが見えるが、このような木道の仕方をすると Web サーバーを経由しないので PHP が解釈実行されず、見え方はおかしいはずである。

そのときのブラウザ内の「アドレス」の欄にどう表示されているか、確かめてほしい。 ファイルをダブルクリックして表示すると、 Z:¥public_html¥mysample1.html となり、http://venus.is.sci.toho-u.ac.jp/~(自分の ID)/mysample1.php でない。

なお、ダブルクリックした場合の見え方は、ブラウザによって異なるようだ。

この PHP プログラム（PHP ファイル）を読んで解釈してみよう。先頭から5行目までは、ごく普通の HTML のファイルである。まず<html>タグがあって、次に<head>のセクションがあって、その中に<title>として「PHP Test」と書いてある。その次に<body>が始まっている。

その次からが新しい構文になる。「 **<?php** 」 から始まって 「 **?>** 」 で終わる部分が、PHP のプログラムとなっている。言ってしまえば全体で <?php … ?> という形なので、 <?php ?> タグの引数が長いプログラムになっている、という形である。

この例では、PHP プログラムは 「 echo "．．．．"; 」 という1行だけである。「echo」は画面にそれ以下セミコロン（;）までの文字列を表示するという構文である。セミコロン（;）は、Java などと同様に、文の終りを示す。 ここでは「 Hello World 」という文字列を表示せよと言っていることになる。

端末は受け取った文字列を HTML だと思って解釈表示するので、<P>と</P>（パラグラフの開始と終了を示すタグ）は画面上には表示されない。

PHP のブロックが終わったあとは、いつもどおりの HTML のタグで、</body></html> でボディと HTML 環境を閉じている。

このように、PHP のファイルは、HTML の（タグを含む）文の中に、 <?php から ?> までの PHP の分を埋め込んだような形になっている。このファイルは上から順序に解釈され、PHP 部分は PHP プログラムとして実行される。

PHP の開始のタグは、正式には **<?php** であるが、（サーバーの環境設定をすることによって） 省略形として **<?** を使うことが出来るが、ここではフルスペル **<?php** を使うことにしておく。

いろいろな PHP 構文とプログラム

最低限のプログラム例と、PHP プログラムの実行の仕方を理解できたので、次にもう少し複雑な構文を見てみることにしよう。 構文の公式な解説は PHP 言語のマニュアル http://www.php.net/manual/ja/ を見て欲しい（日本語版）。

≪変数と代入≫

次の例を解釈してみよ。

```
$xyz = 3;                  /* 変数$xyz に値 3 を代入する */
$enshuuritu = 3.1416;      /* 変数$enshuuritu に値 3.1416 を代入する */
$toiawase = "What is this?";
                           /* 変数$toiawase に値"What is this?"を代入する */
```

いずれも、変数に値を代入する代入文である。左辺は変数の名前だが、ルールとして **$から始まる名前** を使う。右辺は代入される値（定数）で、整数だったり小数だったり文字列だったりしている。

大事なこと: **変数は宣言しない**。 さらに型も気にしない。（注: 型がないわけではない。正確に言えば、PHP が実行中に自動的に型を決めて処理してくれる。詳しくはマニュアルの「型」の項を参照すること。）

それぞれの行の右半分の /*…*/ はコメントである。 PHP では C/C++言語と同じ形（/*…*/で囲まれたコメントと、//で始まる1行コメント）のコメントと、UNIX シェルと同じ形のコメント（#で始まる1行コメント）を書ける。

≪制御構造≫

Java や C 言語と似ていて、if 文、while 文、for 文、switch 文などが使える。詳しくは PHP マニュアルの「制御構造」の項を参照のこと。

```
if ($x < 0) {
  $abs = -$x;
} else {
  $abs = $x;
}
```



```
$sum = 0;
for ($i = 1; $i <= 100; $i++) {
  $sum = $sum + $i;
}

$i = 1; $sum = 0;
while ($i <= 100) {
  $sum = $sum + $i;
  $i++;
}
```

各々の文の終りにセミコロン「;」を置くのも、Java や C 言語と同じ。

［例題演習４－２］ PHP プログラムでループを試してみる

次の PHP のプログラムを、ファイル Z:¥public_html¥mysample2.php として作り、ブラウザからアクセスしてみよ。このとき、必ず http://venus.is.sci.toho-u.ac.jp/~(自分のID)/mysample2.php のように、venus のファイルとしてアクセスすること。

```
<html>
  <head>
    <title>PHP Loop Test</title>
  </head>
  <body>
    <?php
      $sum = 0;
      for ($i = 1; $i <= 100; $i++) {
        $sum = $sum + $i;
      }
      print "<p>Sum is "  . $sum . "</p>";
    ?>
  </body>
</html>
```

（注） 出力文として、echo ではなく print を使ってみた。正確に言うと微妙に違うのだが、まあほとんど同じに使える。

≪関数呼出しと関数宣言、return 文≫

PHP の関数の呼び出し方は、Java や C 言語同様に、関数名の後に引数を書く。たとえば関数 sin に引数 π/8 を与えて呼び出す時は、sin(3.14/8) などとすればよい。
PHP では、関数という形でさまざまな機能を提供しており、かなり多くの関数があらかじめ用意されている。あとで使う SQL データベース の呼び出し機能も、関数の形で提供されている。このあたりも Java や C 言語の感覚と似ているかもしれない。
あらかじめ組み込まれている個々の関数の機能と呼び出し方は、マニュアルの「関数リファレンス」の項を参照すること。たとえば、正弦関数 sin の詳細はマニュアルの「数学関数(Math)」の項に書かれている。
非常に多岐にわたる関数が提供されているので、一目では理解しづらいかもしれない。サンプルプログラムを見ながら、便利そうな関数から試してみると良いだろう。また「こういうことができないかな」と思ったら、まず組み込み関数をチェックしてみるというのもよい。
プログラム中で、自分で関数を定義したい場合(ユーザ定義関数と呼ぶ)にどうするか、について触れておく。たとえば、1からNまでの和を求めるユーザ定義関数 sum(N) は次のような形になる。

```
function sum ($upperlimit) {
  $retval = 0;
  for ($i = 1; $i <= $upperlimit; $i++) {
    $retval = $retval + $i;
  }
  return $retval;
}
```

キーワード function の次に、定義したい関数名 sum と引数のリストを書く。関数の本体の書き方は Java や C 言語とよく似ている。関数の戻り値は、上記の例ではプログラム中で変数$retval に既に計算されているので、それを return $retval によって戻している。
この関数を呼び出す側のプログラムは、たとえば

```
print "<p>Sum of 1..100 will be " . sum(100) . "</p>";
```

などとすればよい。print 文の引数の中で sum を呼び出した形である。

[例題演習4－3] PHP プログラムで関数を試してみる

次の PHP のプログラムを、ファイル Z:¥public_html¥mysample3.php として作り、ブラウザからアクセスしてみよ。このとき、必ず http://venus.is.sci.toho-u.ac.jp/~(自分のID)/mysample3.php のように、venus のファイルとしてアクセスすること。

```
<html>
  <head>
    <title>PHP Loop Test</title>
  </head>
  <body>
    <?php
      function sum ($upperlimit) {
        $retval = 0;
        for ($i = 1; $i <= $upperlimit; $i++) {
          $retval = $retval + $i;
        }
        return $retval;
      }
      print "<p>Sum is " . sum(100) . "</p>";
    ?>
  </body>
</html>
```

≪演算子≫
マニュアルの「演算子」の項を見て欲しい。代数演算、論理演算、比較演算は、Java や C 言語と似たようなものである。演算子の優先順位は、気にしておいた方がよい。
文字列に対しては、代入のほか、結合演算子「.」(ピリオド)が使える。2つの文字列を結合した結果を返す。

```
$mojiretu1 = "hallo";
$mojiretu2 = "world";
$mojiretu3 = $mojiretu1 . $mojiretu2
```

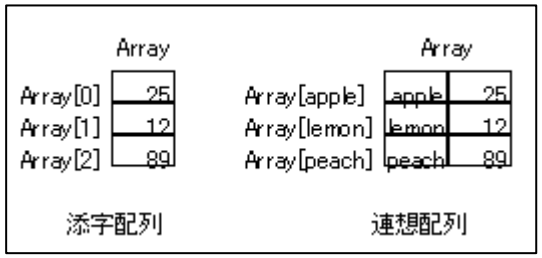
とすると、結合結果の $mojiretu3 は、「halloworld」となる。この時あいだに空白などが入らないことに注意。
なお、演算子「.=」は、C 言語でも見られた「+=」と同様に、

```
$s .= "ABC";     は
$s = $s . "ABC";     と同じ
```

という規則である。

≪配列≫
PHP の配列は C 言語の配列とは考え方が違い、Perl 等でも用いられている「連想配列」の考え方に基づく。

配列は、(同じ型の)要素が並んでいるものだが、どの要素かを識別する方法が、Java や C 言語(や従来の Algol 系と言われる言語全般)では(0 から始まる)番号が付いていた(添字配列)のに対して、PHP では名前をつけて識別する(連想配列)。
図で分かるように、配列の各要素には名前 (例では apple, lemon, peach) が付けられ、配列のアクセスは Array[apple]のようにして要素を指定する。アクセスの結果は、Array[apple]の値は 25 だということになる。
(注: 名前として先頭から 0 から始まる連番をふれば、C 言語の配列と同様に Array[0]が 25 という使い方もできるので、 その意味からは連想配列のほうが広い概念といえるかも知れない。)



PHP の配列は連想配列なので、宣言の仕方もアクセスの仕方も、JavaやC言語のそれとは異なっている。
まず宣言は、C 言語では静的に「宣言」し実行開始前に作られるし、Java では型の宣言部分と実体の(個数を含めた)生成の2つのステップが必要だが、PHP の場合は実行時に(array 関数を呼び出すことによって)生成する。(型宣言は必要ない。なぜなら、PHP では型は自動的に決められるからである。)

図のような配列を作るためには

```
$Array  = array( “apple”  => 25,  “lemon” =>12,  “peach” =>89);
```

とすることになる。また、

```
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
```



```
echo $arr[12];    // 1
```

この例では、配列$arr は foo という名前の要素と 12 という名前の要素の2つを持ち、foo 要素には初期値 bar を、12 要素には初期値 true をもつものとして作られる。従って、echo 文で$arr["foo"]を読み出せば値 bar が、$arr[12]を読み出せば値 true (=1) が読み出せることになる。
なお、配列の要素は任意の型であって構わない。従って、配列の配列を作ることが出来る。

初期値の書き方として次のようなものを許される。

```
$arr = array(3 => 15, 17, 19, 21)
```

これは、 $arr = array(3 => 15, 4 => 17, 5 => 19, 6 => 21) と同じと見なす。更に開始添字(3=>)をも省略すると 0 からを仮定される。つまり

```
$arr = array(15, 17, 19, 21)
```

は、 $arr = array(0 => 15, 1 => 17, 2 => 19, 3 => 21) と同じである。

配列要素へのアクセス(読み出し、書込み)は、例にあるように、角括弧を用いた表記で書く。

配列を扱える関数がある。マニュアルの「配列関数」の項を参照せよ。

[例題演習4－4]
下記のプログラムが何をするものか、説明せよ。

```
<html>
<title>Sample Array</title>
<body>

<?php
$N = 20;
$arr = array();
for ($i=2; $i<=$N; $i++) {
  $arr[$i] = 1;          // すべて 1 にする
}
```

```
for ($i=2; $i<=$N; $i++) {
  for ($j=2; $j<$i; $j++) {
    if (($i % $j) == 0) {     // %演算子は、$i を$j で割った余り
      $arr[$i] = 0; // 余りが 0 なら(割切れた)配列に 0 をマーク
    }
  }
}

for ($i=2; $i<=$N; $i++) {
  if ($arr[$i] == 1) {
    print($i); print("<br>");
          //配列で 1 が残った所だけ$i の値を表示
  }
}
?>
</body>
</html>
```

[例題演習４－５] 上記プログラムを試してみよう
例題４－４として読み解いた PHP のプログラムを、Z:¥public_html¥mysample4.php として作り、ブラウザからアクセスしてみよ。このとき、必ず
http://venus.is.sci.toho-u.ac.jp/~(自分の ID)/mysample4.php のように、venus のファイルとしてアクセスすること。

≪おまけ≫
この章では何度も、ファイルを作るときには Z:¥public_html¥mysample.php のように Z ディスクのファイルとみなしてエディタ(秀丸など)で編集するが、実際にブラウザでアクセスするときには、サーバーvenus を介して
http://venus.is.sci.toho-u.ac.jp/~(自分の ID)/mysample4.php
としてアクセスせよと言ってきた。 これは、我々の使う実習室の環境が、サーバー venus のユーザファイル(それぞれのユーザのホームディレクトリ)が、端末 PC の Windows 上で Z ディスクとして見える(リモートマウントされる)からである。このリモートマウントのサービスは、実習室固有のものであって、普通にホームページサーバー

を使う(レンタルするなど)ときはこのリモートマウントのサービスは使えない。

そういう場合には、通常は、「ファイル転送アプリケーション」で転送する(アップロードと呼ぶ)。 実習室サーバーの場合でも、自宅PCでファイルを編集しようとすれば、(リモートマウントされないので) ファイル転送をしなければならない。

実習室サーバーは、遠隔ログインもファイル転送も SSH と呼ばれる仕組(プロトコル)を使う。(注: FTP と呼ばれるファイル転送の仕組は使えない。) SSH に基くファイル転送を、SCP (Secure CoPy) と呼び、それのための Windows 上の転送アプリケーションとして、「 WinSCP 」(最新バージョンを入手すること) というフリーのプログラム(http://winscp.net/eng/docs/lang:jp からダウンロード)がある。インストール法などについてはインターネットで検索するとあちこちの大学の計算センターなどで解説しているものが見つかるので、それを見て欲しい。

また、一種の離れ技ではあるが、WinSCP をインストールした Windows PC 上で、[手元へダウンロード転送⇒手元で編集⇒サーバーへアップロード転送] という手順をしなくても、WinSCP 上でリモート(サーバー)側のファイル名を右クリック⇒「編集」をクリックすると、手元にコピーを作らなくても(＝ファイル本体はサーバー側にある状態で)編集作業が出来る。編集の終了(セーブする)時には、リモートのサーバー上の元ファイルへセーブしてくれるので、あたかもファイルが手元にあるかのように扱える。 但し実態は、手元の Windows PC 上の一時ファイルにダウンロード転送して編集させ、セーブ時にリモートへアップロードしているだけであるが。

終了の確認

≪ [例題演習４－４]下記のプログラムが何をするものか、説明せよ。 ≫ の結果をTAに確認してもらってください。 これによって、第4章を修了したこととします。

終了したら、次の章へ進んでください。

第5章　フォームと PHP プログラム

PHP プログラムのデバッグ

プログラムがうまく動作しない場合、原因を見つけて修正するデバッグが必要となる。PHP に限らず、一般に CGI のプログラムのデバッグは結構面倒になる。 というのは、いつもやるように、入力がキーボード、出力が画面、とならないため、簡単に試せない。 CGI 環境で実行される時の環境（入力を含めた環境）に依存するため、それを実際に CGI として実行される時と同じに設定しなければ、本当の動きは分からないわけだが、これが結構面倒になる。
プロジェクトで扱うような小さなプログラムの場合、私はホームページの環境で実行しながらデバッグしてしまう。

私流のデバッグの場合、

（A）実行している場所

プログラム上のどの文を実行しているのか、正確には「今」の状態は取り出せないので、「どの文を経由して実行して異常終了したのか」もしくは正常終了した場合でも「どの文を経由して実行したのか」を、画面に表示する

（B）変数や入力の値

問題のありそうな部分が分かっているとき、それを実行する前の変数の値と実行した後の変数の値を、画面に表示する

の2つの情報を元にしてデバッグする。
前者については、プログラムはどの言語であっても、変数や入力の値によって実行の経路が変わる。 実行の経路とは、if による分岐がどちらを選ぶか、while などのループの終了条件がいつ成り立つか、などのことを想定している。for 文のループ構造も、突きつめれば if と同じ分岐によってループを脱出する仕組なので、ループ脱出の分岐点でどちらの経路が選ばれるかに注意すればよい。
分岐の経路選択は変数の値によって決まる。たとえ if 文が関数値によって分岐していたにしても、その関数の値は結局、変数の値から計算される。その変数の値は、プログラム内で他の変数から計算されるか、入力値から計算されるか、どちらかの

はずである。だから、変数値と入力値を見張っていれば、if やループの分岐の経路も予想できる。
プログラムの、≪期待される動き≫と、≪上記で得た実行の経路＋変数の値≫とを比較し、問題箇所の範囲を狭めて原因を特定する、というプロセスでデバッグを進めることになる。

この（A）と（B）を実現する方法はいろいろ考えられるが、Java や C 言語と同様、PHP でも print 文を各所に挿入して追跡することが出来る。PHP の print 文は、それが置かれた場所を実行すれば、変数値を画面に表示することができる。つまり、表示されればその場所が実行されたことと変数の値が分かることになる。ただし、

> CGI プログラムの場合は、print 文で書かれる画面とは、端末側から見るホームページの画面になる。また、print 文で書き出した文字列は、端末のブラウザで HTML として解釈されるので、HTML の文として成り立つ形にして出力しなければならない。

たとえば、次のようになる。

```
<html>
 <head>
    <title>PHP Debug Sample</title>
  </head>
  <body>
  <?php
    $now = gettimeofday(); $time = $now["sec"]; $datestring = date("r", $time);
    print("<P>Debug Test.    time: $time, datestring: $datestring. </P>¥n");
  ?>
  </body>
</html>
```

この例では、$time と$datestring の値を、print 文によって表示させている。前後に<P></P>を書いているが、これによって HTML の形式に合うようににしている。（実際は、<P>を省略してもうまく表示される。）
この例で使った関数 gettimeofday、関数 date については、PHP のマニュアル（http://www.php.net/manual/jp/）から「日付／時刻」の項を参照せよ。 更に、2番目の代入文の$now["sec"]は、変数$now が配列で、その["sec"]要素を取り出す構文である。配列は連想配列になっていることは、前章で説明した。なお、"sec"は引用符「"」が必要である。

［例題演習５－１］ 実行途中の変数の値を print してみる。
上記のプログラムを実際に実行してみよ。ここにあるだけでなく、実効途中の変数の値をいろいろと表示してみよ。

フォーム入力を使う PHP プログラム

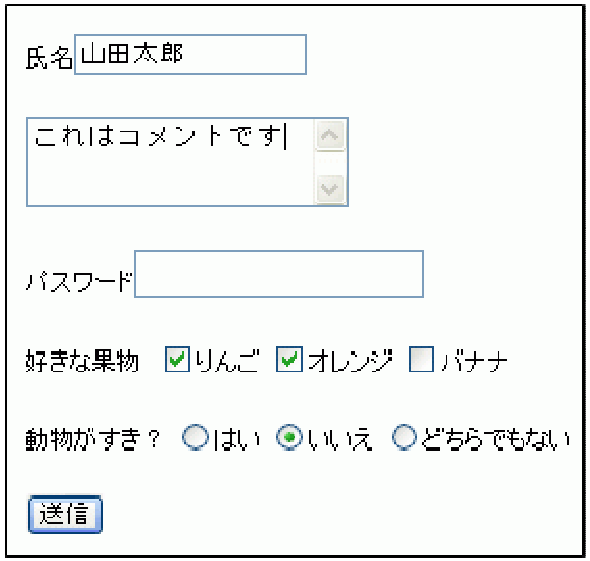
次に、フォーム（入力画面）を受けるプログラムを作ってみよう。
前に説明したとおり、CGI による動的なページは、
（1)フォーム（<FORM>）による入力画面と、
（2) action として指定されたプログラム、
とが対になっていなければならない。フォームの HTML の書き方は3章で説明したので、ここでは入力を受けて処理をするプログラムの書き方を見てみよう。

［例題演習５－２］ フォーム入力を使う CGI プログラムを PHP 言語で書いてみる。

```
<HTML><BODY>
<FORM action="myaction.php" method="post">
  <P>氏名<INPUT size="20" type="text" name="yourname" value="山田太郎"
maxlength="30"></P>
  <P><TEXTAREA rows="3" cols="20" name="freecomment">自由にコメントを書き込ん
    でください</TEXTAREA></P>
  <P>パスワード<INPUT size="20" type="password" name="yourpassword"
maxlength="15"></P>
  <P>好きな果物 <INPUT type="checkbox" name="fruit1" value="apple">りんご
    <INPUT type="checkbox" name="fruit2" value="orange">オレンジ
    <INPUT type="checkbox" name="fruit3" value="banana">バナナ</P>
  <P>動物がすき？
    <INPUT type="radio" name="animal" value="yes">はい
    <INPUT type="radio" name="animal" value="no">いいえ
    <INPUT type="radio" name="animal" value="neither">どちらでもない
  </P>
  <P><INPUT type="submit" name="send" value="送信"></P>
</FORM>
</BODY></HTML>
```

フォームの入力画面を表示する HTML ファイルを準備する。 第3章の例題3－1で作った HTML ファイルを使うことにする。public_html フォルダの中に、ファイル名 testform.html として置く。

これは、右図のような入力画面を表示させる。 第3章ではここまで実験した。



次に、このフォームで「送信」ボタンをクリックした時に起動されるプログラムファイル myaction.php を作り、上と同じ public_html フォルダ内に作る。 ファイル名は、<FORM>タグの中に“action=”のところで指定されたファイル名を付ける必要がある。

フォームに対する入力を読み取るには、ここでは<FORM>タグ内の method に post を指定しているので、POST メソッドで取り込まれる。
POST メソッドで取り込んだ入力は、(連想)配列 $_POST[入力欄の名前] に入って来る。 この“$_POST”という名前は、PHP で決められた特別な名前である。 それを PHP プログラムで読み出すことが出来る。上記の例だと、入力欄「氏名」では、欄の名前(識別名)を yourname としている (INPUT 文で name="yourname") ので、入力の取り出しは

```
$inname = $_POST["yourname"];
```

のようにする。
つまり、$_POST["yourname"]の中に、入力した文字列が入ってきているので、それを PHP の変数$inname に代入しているのである。
同じ様にして、入力欄 freecomment, fruit1, fruit2, fruit3, animal を読み出す。

チェックボックスの場合は、HTML ファイルでは


```
<P>好きな果物 <INPUT type="checkbox" name="fruit1" value="apple">りんご
  <INPUT type="checkbox" name="fruit2" value="orange">オレンジ
  <INPUT type="checkbox" name="fruit3" value="banana">バナナ
</P>
```

のように設定してあり、各チェックボックスに異なる名前 fruit1・2・3 をつけているので

```
$fruit1result = $_POST["fruit1"];
$fruit2result = $_POST["fruit2"];
$fruit3result = $_POST["fruit3"];
```

のように読み出す。入力画面でチェックマークをつけると、value で指定した値(apple, orange, banana のどれか)が戻ってくる。上記の入力例の画面ではりんごとオレンジにチェックをしているので、fruit1 には apple が入り、fruit2 には orange が入ってくる。

次の行のラジオボタンの場合は、HTML ファイルでは

```
<P>動物がすき？
  <INPUT type="radio" name="animal" value="yes">はい
  <INPUT type="radio" name="animal" value="no">いいえ
  <INPUT type="radio" name="animal" value="neither">どちらでもない
</P>
```

としてあるので、名前 animal で読み出してみると、この3種の選択のいずれか(ラジオボタンは択一なので)の value の値(yes か no か neither か)が入ってくる。

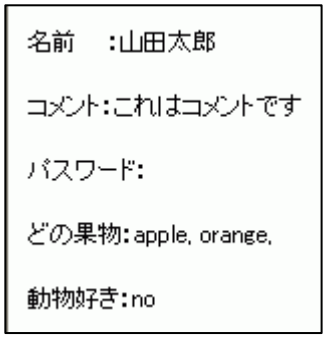
```
$animal = $_POST["animal"];
```

上記の入力例画面ではユーザは「いいえ」にチェックしているので、animal の値は no が入ってくるはずである。

これらを併せた全体のプログラムは、たとえば次のようになる。このプログラムは FORM 入力から読み取ったデータを、後ろの5行の print 文で画面に書き出すようになっている。パスワードは入力していないので、値がない(空)。

```php
<?php
    $name = $_POST["yourname"];
    $comment = $_POST["freecomment"];
    $pass = $_POST["yourpassword"];
    $fruit1result = $_POST["fruit1"];
    $fruit2result = $_POST["fruit2"];
    $fruit3result = $_POST["fruit3"];
    $animal = $_POST["animal"];
    print("<p>名前　　:" . $name . "</p>¥n");
    print("<p>コメント:" . $comment . "</p>¥n");
    print("<p>パスワード:" . $pass . "</p>¥n");
    print("<p>どの果物:" . $fruit1result . ", " . $fruit2result . ", " . $fruit3result . "</p>¥n");
    print("<p>動物好き:" . $animal . "</p>¥n");
?>
```

この PHP プログラムを、ファイル　myaction.php　として、フォームを作る testform.html ファイルと同じ public_html フォルダ内におく。そうしておいて、testform.html をブラウザから（http://venus.is.sci.toho-u.ac.jp/~(自分の ID)/testform.html という名前で）　開くと、まず入力の画面が現れる。入力した後に「実行」ボタンをクリックすると、myaction.php が実行されて、右上のような結果が表示されるだろう。　これを試してみて欲しい。



ところで、チェックボックスでの結果の戻り値（上記 $_POST["fruit1"] など）を読み出して変数 $friut1result に代入しようとする時、実行時に下記のエラーが起きることがある。

Notice: Undefined index: fruit1 in **sample.php** on line **10**

これは、チェックボックスの値を読み出す連想配列の、fruit1 に対応する要素が未定義なことによるエラーである。 このブラウザにおいてこの要素を初めて使用し、かつチェックしていない場合に生じる。 プログラムのロジッ

クとしてはあまり気にしなくてもよいが、見栄えが悪いので回避したいという人は、プログラム上で次のように回避できる。

```
$fruit1result = isset($_POST["fruit1"]) ? ($_POST["fruit1"]) : null;
$fruit2result = isset($_POST["fruit2"]) ? ($_POST["fruit2"]) : null;
$fruit3result = isset($_POST["fruit3"]) ? ($_POST["fruit3"]) : null;
```

まず全体の構文は、代入文なのだが、右辺は「条件式」になっている。条件式は、その形が

（条件～if 文の条件と同じ形）？（True の時返したい値）：（False の時返したい値）

であり、たとえば

```
y = ( x>=0 ) ? 1 : -1
```

のようにすれば、x が正であれば値１が返り、負であれば-1 が返る式となる。これを左辺の y に代入すれば、この例のようになる。

関数 isset(…) は、要素 $_POST[…] の値がセットされている（書き込まれている）か否か（全く初めて、つまり初期化されていない）を判定する関数で、何かセットされていれば True を返す。

従って、上の式の意味は、「もし $_POST["fruit1"] がセットされていれば、その値 $_POST["fruit1"] を左辺に代入するが、もしセットされていなければ、値 null を左辺に代入する」 となる。 従って、$_POST[] の中で "fruit1" に相当する要素が未定義である、というエラーは発生しなくなる。

好きな動物（ラジオボタン）の書込みが空欄（デフォルト値を設定しない）であったり、コメントが空欄（デフォルト値を設定しない）であったりした場合にもこのエラーが起こり得るが、同じような方法で回避できる。

```
$animal = isset($_POST["animal"]) ? ($_POST["animal"]) : null;
```

更に、名前欄のように自由に記述できる欄に、HTML での特殊文字（ "<", ">" などの、HTML 上で役割もっている文字）を書き込まれた場合を用心したければ、

```
$animal = isset($_POST["name"]) ? htmlspecialchars($_POST["name"]) : null;
```

とするとよい。この関数 htmlspecialchars は、引数（文字列）に含まれる特殊文字を検出し、「正しい」（代わりの）文字列に変換する。 たとえば、"<" は "<" に変換される。マニュアルは http://php.net/manual/ja/function.htmlspecialchars.php 参照

もう1つ簡単な CGI プログラムのサンプル

もう1つ、簡単なサンプルを作って試してみよう。

［例題演習５－３］

入力画面から数値Nを受取って、1からNまでの和を求めて表示する。

前の例題５－２と同じように、

①最初にフォームによる入力ページを表示してその入力枠に数値 N を入力して、送信ボタンを押す、

②それによって action で指定される PHP プログラムが実行され、その中で入力値 N を受け取って、1からNまでの和を求めて、最後に表示する、という手順である。

ファイルは、フォームによる入力ページを表示する HTML ファイル（samplesum.html）と、送信ボタンを受けて action で起動される PHP プログラムのファイル（samplesum.php）の2つのファイルからなる。

まずは自分で、それらを作ってみよ。

以下に作成した例を示す。 フォームを表示する HTML ファイルは、

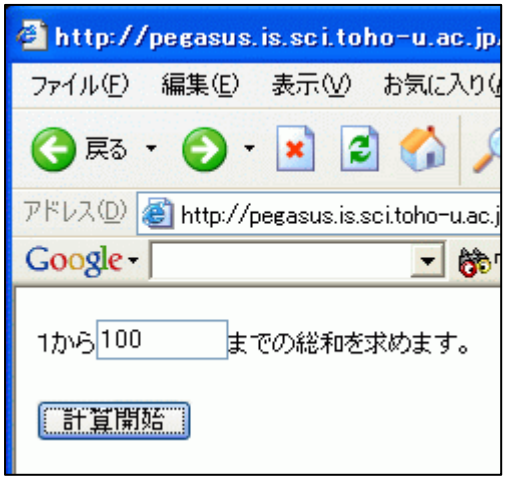
```
<HTML>
<BODY>
<FORM action="samplesum.php" method="post">
  <P>1から<INPUT size="10" type="text" name="max">までの総和を求めます。</P>
  <P><INPUT type="submit" name="send" value="計算開始"></P>
</FORM>
</BODY>
</HTML>
```

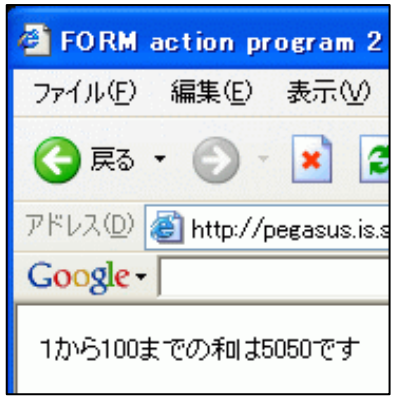
のように書くことが出来るだろう。その結果表示される画面は、下のようになる。

また、このとき、action で呼び出される PHP プログラムの形は、


```
<html>
    <head>
      <title>FORM action program 2</title>
    </head>
    <body>
      <?php
        $maxval = $_POST["max"];
        $sum = 0;
        for ($i = 1; $i<=$maxval; $i++) {
          $sum = $sum + $i;
        }
        print("<p>1から" . $maxval . "までの和は" . $sum . "です</p>¥n");
      ?>
</body>
</html>
```

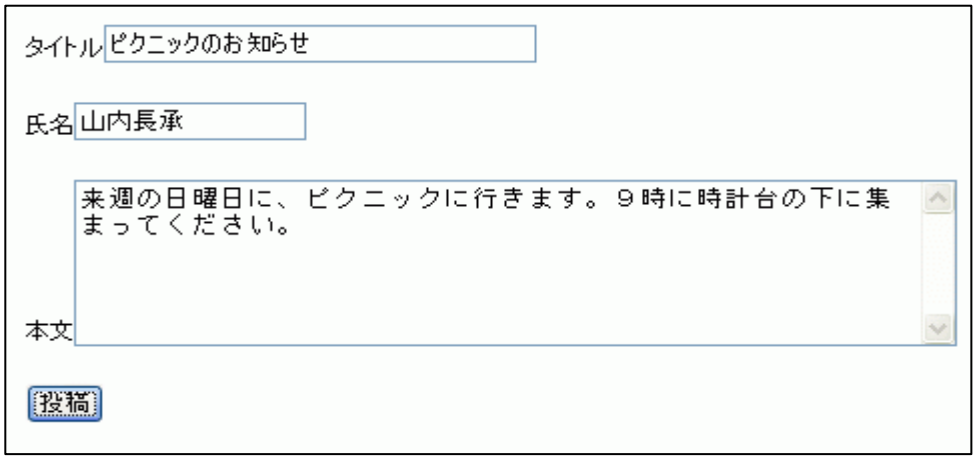
のように作ればよい。 この処理の結果は、下の画面コピーのようになる。



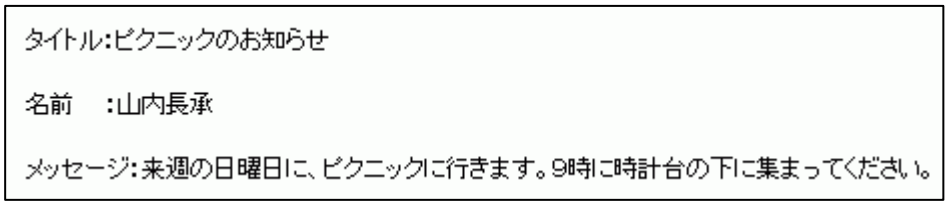


終了の確認

［演習５－４］ 次のようなフォーム入力を使った PHP プログラムを書け。
入力画面は、タイトル（１行テキスト）、名前（１行テキスト）、本文 （テキストエリア）、送信ボタンとする。



action で起動される PHP プログラムは、タイトル、名前、作成日時、本文を表示する。
将来は （データベースが使えるようになったら）、このメッセージをデータベースに格納するようにしたいのだが、今は入力画面から読み取った情報を表示するだけの形で作る。
また、（今回の）出力画面のイメージは、下図のようにする



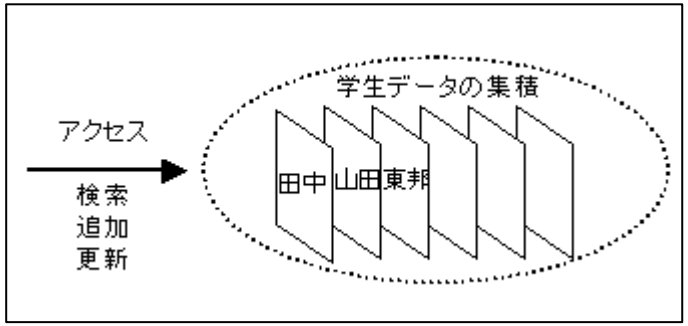
このようになるように、HTML と PHP でプログラムを自分で作って、実際に試してみよ。
ここではサンプルは示さないので自分で考えて作ってみよ。

この結果をTAに確認してもらってください。 これによって、第5章を修了したこととします。
終了したら、次の章へ進んでください。

第６章　　データベースの基礎知識

さて、ここからはデータベースを使ったホームページ処理を学ぶ。 まず最初に、データベースが何であるか、データベースをどう使うのか、考えて見ることにする。

データベースとは



データベースとは、同じ形（イメージとしてJava言語でのクラス、C言語の構造体）のデータ（Javaでのインスタンス）がたくさんあるときに、それを効率よく格納し、検索する仕組である。

例としては、学生１人のデータとして名前、生年月日、学籍番号、住所、入学年度があるとしよう。形は、たとえば名前は最大１６文字の文字列、生年月日は年・月・日の3つの整数、学籍番号は7桁の整数、住所は５０文字の文字列、入学年度は整数、を含むクラス（または構造体）として考えることが出来る。

このような形を持ったデータが 5000 人分あるとしよう。Java やC言語のプログラミングで出てくるようなクラスや構造体の配列を定義するのは１つの方法だが、格納も検索も効率が悪いことが多い。データの個数が非常に大きくなると（例えば１万件とか１０万件とか）、検索するのに効率の悪いやり方をしていては、とても実用に耐えなくなるだろう。 かと言って、効率のよい優れたプログラムを毎回作成するのも大変である。

そこで、大量の（同じ形の）データを効率よく扱える仕組として、データベースが用いられている。データベースは、大量のデータを扱うことを目的にして専用に設計された、共通で使える仕組（ミドルウェア）といってもよいだろう。プログラマは、このような仕組を使うことによって、アプリケーション毎に効率向上の工夫をしなくても、効率よいアクセスが出来るようになる。

正確には、「データベース」と「データベース管理システム（DBMS）」は区別して扱わなければならない。「データベース」はデータの集まりそのものを指す。 「データベース管理システム」（DBMS）はデータベースを入れる入れ物（上に説明した効率よいアクセスを実現するためのソフトウェア）を指す。つまりデータベースはDBMS上にデータを集めたものです。たとえば、過去の気象のデータベース、農作物の生産量のデータベース、人口のデータベースなどと呼ぶことが出来る。これらのデータ集合を、DBMSで管理すると効率がよい、ということになる。

データベースでデータ間の関連を表す必要がある。そのための表し方にはいくつかのモデル（データモデル）がある。現在広く使われているのは「関係モデル」で、データ同士の関連を「関係」（relation）として表す。具体的には表の形式で表現される。別のモデルとしては「階層モデル」や「ネットワークモデル」が有名である。この詳細はデータベース論の講義に譲ることにし、ここでは深く追求しないことにする。

名前	生年月日	学籍番号	住所	入学年度
田中太郎	1992/9/9	5510950	船橋市三山. . .	2010
山田一郎	1992/5/5	5510951	習志野市藤崎. .	2010
東邦花子	1993/3/3	5510952	千葉市美浜区. .	2010
佐倉純一	1991/7/7	5510953	柏市. . .	2010

ここでは、広く使われている「関係モデル」によるデータベースを使うことにする。 関係モデルでは、データを表の形式で記述する。 右図は、上記の学生のデータベースを表形式で表した例で、学生１人が１行のデータになっており、その中に名前や生年月日などの情報が横に並んでいる。

データベースをどう使うのか？

データベースに対する操作は、データの挿入・削除や検索などが中心になる。関係モデルのデータベース（関係データベース）では、データ操作のための SQL と呼ばれる言語が標準化されており、広く使われている。SQL に従ったデータベースを SQL データベースと呼ぶことがある。このプロジェクトでは、広く使われている SQL データベースを使って、ホームページから登録されたデータを格納したり、ホームページから与えた条件で検索したりする。

SQL データベースにはいくつも製品がある。商用で有料のソフトでは Oracle 社や IBM 社のデータベース管理システムが有名なほか、 Microsoft 社の Windows サーバー上で動く製品もある。 またフリーのソフトもいくつか出回っており、PostgreSQL や MySQL などはかなり広く使われている。いずれも、標準化された SQL 言語を介してデータにアクセスするので、どの製品でも同じようなものだが、それぞれに拡張が成されていたり、運用管理上の手段がいろいろと提供されていたり （これは SQL 標準には含まれない）、性能や安定性にも差がある （これは同じソフトでもバージョンによって異なる）。 プロジェクトでは、データへのごく簡単なアクセスを、標準的な SQL の文で実現するので、どの種類の SQL データベースで利用できるが、製品固有の拡張を使うと、他の製品では使えないということも起こる。

実際に SQL データベース（データベース管理システム DBMS）を使うには、

（1） 表を作る（定義する）
（2） 表にデータを挿入する
（3） 表を検索し欲しいデータを抽出する
（4） 必要なら表の中のデータを更新する（書き換える）

といった操作をしなければならない。

たとえば、上に示した学生データを管理する場合、まず最初に表を作らなければならない。表を定義するためには、各々の「列」（正式にはフィールドと呼ぶ）のデータの形の定義を DBMS に伝える必要がある。たとえば、「名前を書く欄は name という名前

で呼ばれ、最大 16 文字の文字列データが入る」 といったことである。すべての列について定義する必要がある。この表を作るための SQL の構文として、CREATE TABLE がある。これによって1つ表を作ることが出来る。表には名前をつけておく。また引数として、表の列（フィールド）の名前や型を指定する。なお、表の操作にはこの他に列（フィールド）の変更（ALTER）や表自体の削除（DROP TABLE）などがある。

次に、今作成した表にデータを挿入する。ここで言うデータの挿入は、表の上では行を1つ追加することに当たる。 表の行のこと（たとえば学生データベースにおける山田太郎のデータ）を「レコード」と呼ぶ。 データ（行、レコード）の挿入は INSERT INTO 構文で行う。引数に、対象とする表の名前と挿入する値を与える。1行追加する時に、すべての列の値を与える必要はない。値を与えていない列は、値が決まっていない状態のままとなる。

表にたくさんのデータ（行、レコード）が入ると、その表を検索して特定の行（レコード）を取り出すことが必要になる。

（注意： データベースでは、行（レコード）がどういう順番で入っているか、つまり何行目に入っているかは問題にしない。その代わり、特定の条件に合う行（レコード）を検索して取り出そうとする。）

学生データの例で言えば、

名前が山田一郎の学生のレコードを取り出せ　とか
学籍番号が 5510234 の学生のレコードを取り出せ　とか
生年月日が XX から YY までの学生のレコードを取り出せ

などのように条件を指定して取り出す。 この操作を、SQL では SELECT という構文を使って行う。

たとえば、

```
SELECT * FROM students WHERE name='山田一郎'
```

とすると、表 students の中から、列名（フィールド名） name が値 '山田太郎' である行（レコード）を検索・抽出し、マッチする行のすべての列（フィールド）を表示せよ、という意味になる。

表の中のデータを更新したい場合は、UPDATE 構文を使う。引数で、どの表の中の、どの列（フィールド）の値を書き換えたいかを指定した後、対象となる行（レコード）をWHERE 節で指定する。 WHERE 節の書き方は検索と同じで、条件を満たす行（レコード）を検索し、その検索で当てはまった行に対して、指定された列（フィールド）を指定された値に書き換える、という操作をします。たとえば、山田太郎の生年月日を修正するときには、SQL の上では name が山田太郎である行を探して、その行に対して生年月日をZZに置き換える、というような操作の仕方をする。

この他に重要な操作として、表の結合がある。
例として、上記の学生の表 students の他に、もう1つ点数の表 score があるとしよう。点数の表 score は、学生番号の列（フィールド）id と、点数の列（フィールド）ten を持っているとする。そこで、表 students と表 score を同時に見ることを考える。たとえば「'山田一郎の点数は？」という問い（つまり名前が山田一朗である学生の点数は？という問い）に対して、まず表 students を引いて山田一郎の学生番号が 5510951 であることを求め、更に表 score を引いて学生番号が 5510951 である学生の点数を求めることになる。これを1回で自動的にやってしまおうというのが「結合」である。この例では、

```
SELECT score.ten FROM students, score
    WHERE students.name='山田一郎' AND students.id=score.id
```

のようにする。SELECT の後ろの score.ten はこの検索の出力を「表 score 内の列 ten」にせよ、と指示している。表の名前. 列の名前で書かれている。また最後の students.id=score.id は結合の条件で、表 students の id と表 score の id は同じものとみなせということを指示している。

SQL データベースを使えるようになるためには、SQL の文と機能をひととおり学ぶ必要がある。ここではすべての機能を詳細に説明する余裕がないので、簡単な例を試すだけにし、細かい点はデータベースの講義と自習に譲ることにする。特に、値に関わる機能（この列の値は正でなければならない、とか、この列の値は指定していなければデフォルト値として Z をとるとか）や、行（レコード）をユニークに識別できる「キー」としてどの列を使う（たとえば、学生番号は重複が無いのでキーに出来る）とかの機能は、細かくなるので省略するが、重要である。自分で勉強して欲しい。

参考書として、山本森樹 「体系的に学ぶデータベースのしくみ第2版」 日経BPソフトプレス ISBN978-4-89100-66505 1900 円 を挙げておく。 筆者がデータベースの講義を担当したときの教科書だが、これにこだわる必要もない。

さて、このような SQL の「文」が使えるとして、実際にどのように使うのだろうか？
MySQL の場合、データベースのアクセスは、
(1)プログラムからのアクセス、と
(2)端末から手でコマンドを打って（インタラクティブに）アクセスする、
の2つが提供されている。 前者はこれから作るサーバーのデータの保管所として action の PHP プログラムからアクセスする場合に当たる。 後者は、1つにはデータベースを理解するためにいろいろと試してみるのに使ってみるのと、サーバーの開発中にデータベースの状態を確認するために使うことができる。

終了の確認

この章は、終了の確認はありません。
第2回目の授業で、最低限ここまで終了していることが望まれます。
時間がある人は、どんどん次の章に進んでください。