CHAPTER 17

PHPスクリプトとHTML

第16章では、PHPの基本的な使い方を学びました。ここでは、PHPによるMySQL操作の基礎となるHTMLの知識と、Webページからのデータ送信と受信の処理を学びます。

MySQL+PHPのようなWebアプリケーションを使う上で、欠かすことのできない知識です。

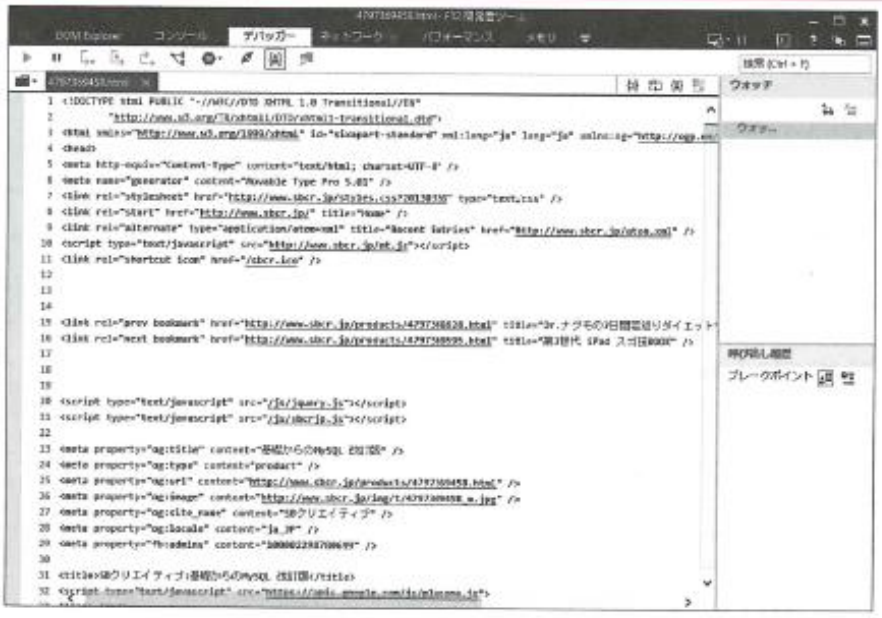
HTMLソース

日々何気なくブラウザ越しに見ているWebページ。その中身がどうなっているのか覗いて見たことがありますか？

以降の章から最後までを通して作っていくのは、Webページです。まずは、Webサーバーから送られてくるWebページの中身を確認してみることにしましょう。

ブラウザで、ソースを表示します。たとえばChromeやInternet Explorer、FireFoxなどは、画面上で右クリックして表示されるコンテキストメニューから「ソースの表示」を選択します。

FIG 17-01 Webページのソース



表示したソースは、「<html>」や「<a ～ /a>」など、「< >」で囲まれた文字で構成されている「HTML」ファイルです。「< >」のような記述をタグといい、「ここにこんな文章を表示しろ」「ここに○○の画像ファイルを表示しろ」というような命令を意味します。

HTMLファイルは、「HTML」という規則に従って作られた単純なテキストファイルです。そしてブラウザは、Webサーバーから送られてきたHTMLファイルの命令に従って、いっしょに送られてきた文字や画像を配置しているだけなのです。

Webページを生成する2つの方法

P.337では、「静的なWebページ」と「動的なWebページ」の話をしました。この「静的」「動的」について、もう少し掘り下げてお話しします。

Webページを作成する方法には、次の2つがあります。

①エディタでタグを入力する、または「Webページ作成ソフト」でタグを作成する
②PHPスクリプトなどのプログラムによって、タグを書き出す

①静的なWebページ作成

まずは、①のWebページ作成方法です。

HTMLファイルはテキストでできているので、文法規則を理解していれば、メモ帳などのエディタで簡単に作ることができます。かつては、Webページを作成するには、エディタに直接キーボードから入力していくしか方法がありませんでした。しかし今では、タグのことを何も知らなくても、たとえば「Adobe Dreamweaver」や「ホームページ・ビルダー」のような「Webページ作成ソフト」を使えば、ワープロ感覚で作ったページから勝手にタグを作り出してくれます。

「エディタで直接入力する」「ソフトウェアで書き出す」という違いがあるにせよ、この方法で作成したWebページの内容は変化せず、タグで指定した内容のまま表示されます。たとえば「こんばんは」という内容のWebページは、誰がいつアクセスしても「こんばんは」としか表示されません。このようなものを「静的なWebページ」といいます。

②動的なWebページ作成

そして②の動的なWebページを作成する方法です。

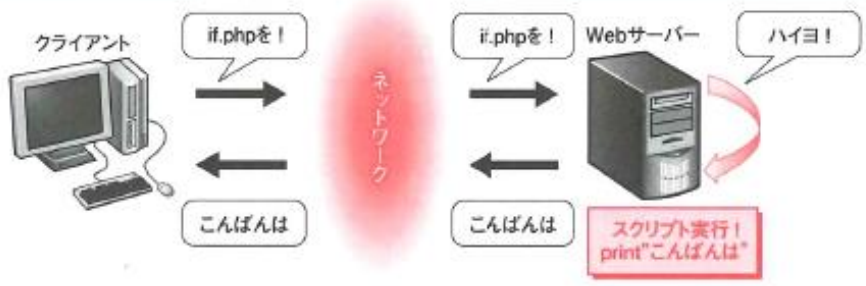
たとえば次は、P.377で作成したPHPスクリプトです。

```
<?php
if (date("G")>=18){
    print "こんばんは";
}elseif(date("G")>=9){
    print "こんにちは";
}elseif(date("G")>=6){
    print "おはようございます";
}else{
    print "眠くないですか";
}
?>
```

実行すると、ブラウザには0時以降6時までは「眠くないですか」、6時以降9時までは「おはようございます」、9時以降18時までは「こんにちは」、18時以降0時までは「こんばんは」と表示されます。これは、PHPが「時刻」という条件によって、異なる内容を書き出しているためです。

このように、PHPスクリプトなどのプログラムによって間接的に書き出され、条件によって異なる内容を表示するものを「**動的なWebページ**」といいます。

FIG 17-02 動的なWebページ



ブラウザは静的なページと動的なページを区別しない

①の方法、つまりエディタやWebページ作成ソフトで、直接タグを書き出すことで作ったHTMLファイル。そして②の方法、つまりPHPなどのプログラムを使って間接的に、動的に書き出したHTMLファイル。この2つを、ブラウザは区別することはありません。クライアント側では、まったく同じように表示されるのです。

私たちの目標は、MySQLとやり取りした結果をWebページとして書き出し、クライアントに送り返すことです。やり取りした結果は当然、いつも同じにはなりません。PHPスクリプトを使ってMySQLを操作し、MySQLが処理して送ってきた結果をまたPHPスクリプトで受け取り、Webページを動的に書き出していくことになります。

HTMLの規則

PHPスクリプトを使ってMySQLを操作するには、HTMLの知識が必要です。ここではMySQLを操作する上で、必要と思われる範囲の知識に絞って解説します。HTMLに一通りの知識のある方は、P.401に進んでもかまいません。

なお、HTMLにはさまざまなバージョンがありますが、本書では2017年7月現在最新であり、一般的に使用されているHTML5に関して解説しています。

HTMLファイルの拡張子は「.html」、PHPスクリプトがあれば「.php」

HTMLファイルは、<html>などのようタグからできているテキストファイルです。拡張子は「.html」または「.htm」です（本書では「.html」に統一しています）。ただし、この中にPHPスクリプトを含める場合は、「.php」の拡張子を付ける必要があります（→P.348）。

タグの書き方

HTMLは<html>のようなタグを記述することでWebの画面を作成していきます。タグの多くは次のように「開始タグ」と「終了タグ」がセットになっていて、その間にはさまれている文字列を含めて「要素」といいます。次は<a>タグの例ですが、これをa要素と呼ぶ場合もあります。

上記の「href」のように、「開始タグ」には、その詳細の情報を入れることができます。これを「属性」といいます。

HTMLではタグを、属性を含め半角文字で記述します。基本的に大文字、小文字は区別されません。ただし最近では小文字で書かれることが一般的なので、本書でも小文字で表記しています。

ただし、後で紹介する
タグのように終了タグの必要ないものも複数ありますので、注意してください。

HTMLファイルの構成

一般的なHTMLファイルの構成は次のようになっています。

書式 HTMLファイルの構成

```
<!DOCTYPE html>
<html>
<head>
    ページタイトルやリンクの関係などを記述
</head>

<body>
    Webページとして表示される本体を記述
</body>
</html>
```

先頭は文書型宣言というもので、HTML5の場合は「<!DOCTYPE html>」と記述する決まりです。

以降は全体を<html> ～ </html>で囲み、その中にhead要素とbody要素を記述します。これらの中にもたくさんのタグを記述しますが、それらについては後述します。ここで覚えておいていただきたいのは、「画面表示されるのはbody要素」であるということです。

HTMLファイルのサンプル

次はHTMLファイルの例です。

LIST 17-01 tag.html

```
<!DOCTYPE html>                    1
<html>                             2
<head>                             3
<title>SQLカフェのページ</title>
</head>

<body>                             4
SQLカフェにようこそ！
</body>
</html>                            5
```

HTML5では先頭に文書型宣言「<!DOCTYPE html>」を記述します（1）。

HTMLのメイン部分は<html>（2）で始まり、</html>（5）で終わります。

<head> ～ </head>にはタイトルやリンクの記述を行います（3）。<title> ～ </title>内には、そのページのタイトルを記述します。このタイトルは、ブラウザのツールバーに表示されたり、ブックマークや検索結果などにも表示されます。<head> ～ </head>内に記述してください。

<body> ～ </body>が本体になります（4）。ここでは「SQLカフェにようこそ！」という文字を表示しているだけです。

プログラムを記述したら、公開するフォルダにtag.htmlとして保存しますが、このとき文字エンコーディングを「UTF-8」にするのを忘れないようにしましょう。そして、以下のURLを開いてください。

```
http://localhost/tag.html
```

FIG 17-03 実行結果

属性は「" "」か「' '」で囲う

たとえばソフトバンククリエイティブ（株）「http://www.sbcr.jp/」へのリンクを設定する場合、<a>タグを使って次のように記述します。

```
<a href="http://www.sbcr.jp/">ソフトバンククリエイティブへ</a>
```

<a> ～ の間に、リンクとして表示される文字列を記述します。リンク先のURLは、<a>タグの属性であるhrefを使って設定します。「a」と「href」の間は、必ず半角のスペースにしてください。全角スペースを入れると機能しなくなってしまいます。

この例の「href=」で設定するURLのように、属性として設定する値を「属性値」といいます。属性値は「" "」（ダブルクォーテーション）か、または「' '」（シングルクォーテーション）で囲みます。

ここで注意点があります。PHPで動的にHTMLタグを書き出すときはprintやechoを使いますが、この時、次のように出力するタグ自体も「" "」や「' '」で囲う必要があります。

```
print "<a href=~>ソフトバンククリエイティブへ</a>";
```

このように「" "」や「' '」を二重に使用しなければならない場合、次のように両方とも同じ記号で囲むとエラーになります。

```
print "<a href=" http://www.sbcr.jp/ ">ソフトバンククリエイティブへ</a>";
```

エラーを回避するには、「内側の"に¥を付けてエスケープ処理する」か、「内側の' 'を外側の" "で囲む」か「内側の" "を外側の' 'で囲む」(→P.360)などの対策が必要となります。

本書では次のように、PHPで書き出す文字列は「" "」で囲い、内側のHTMLタグの属性値は「' '」で囲うようにしています。

```
print "<a href=' http://www.sbcr.jp/ '>ソフトバンククリエイティブへ</a>";
```

htmlファイルをPHPスクリプトで書き出す

それでは、ソフトバンククリエイティブへのリンクを設定したWebページを、PHPスクリプトで書き出してみることにしましょう。ここでは<html>や<head>、<title>、<body>などの要素も、すべて省略せずに記述してみます。

LIST 17-02 link_page.php

```
<?php
print "<!DOCTYPE html>";
print "<html>";
print "<head>";
print "<title>";
print "ソフトバンククリエイティブへのリンク";
print "</title>";
print "</head>";
print "<body>";
print "<a href='http://www.sbcr.jp/'>ソフトバンククリエイティブへ</a>";
print "</body>";
print "</html>";
?>
```

単純にタグを「print」(またはecho)で書き出すだけです。実際に実行してみてください。表示されたWebページのリンクをクリックすると、該当のWeb ページが表示

されますか？

FIG 17-04 実行結果

本書のサンプルについて

さてここまでの解説からわかるように、PHPでWebページを動的に出力する場合も、本来は文書型宣言やタグをHTMLの作法に則って記述する必要があります。しかし、本書のサンプルでは、次のようにほとんどタグを書いていません。

```
<?php
print "ようこそ！";
?>
```

これはサンプルの1つ1つに正規のHTMLタグを記述するのは手間ですし、サーバー側の設定が適切なら、現在のブラウザではちゃんと結果を表示してくれるからです。仕事でWebアプリケーションを作成する場合はこのような省略はできませんが、学習のためのコードということでご了承ください。

なお、以降ではHTMLのフォームを使ったPHPサンプルが登場しますが、そこでもHTMLの記述は必要最低限で記述しています。

さらに覚えておきたいタグ

ここでは、本書のサンプルで登場するタグだけを紹介しておきます。実際にprintを使ったPHPスクリプトを実行して、１つ１つその働きを確認してください。

タグ

改行を表すタグです。HTMLファイル中の文字列をいくらEnterで改行しても、ブラウザの表示が改行されることはありません。改行させるときは、必ず
タグを使います。

ただしHTML5では、要素と要素の間を改行するときなど、レイアウトを調整する目的で
を使ってはいけないことになっています。

LIST 17-03 kaigyo.php

```
<?php
print "ようこそ<br>ＳＱＬカフェへ！";
?>
```

FIG 17-05 実行結果

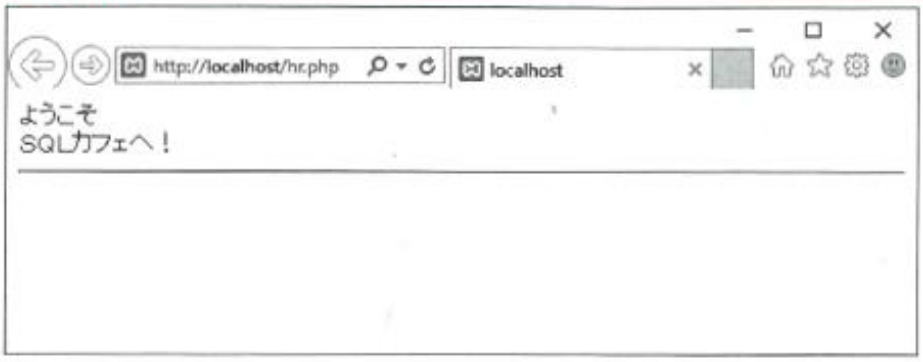
<hr>タグ

<hr>タグは、以前は水平線を入れるためのタグでしたが、HTML5では段落区切りを表します。ただし現在のところ、一般的なブラウザでは水平線が入ります。

LIST 17-04 hr.php

```
<?php
print "ようこそ<br>ＳＱＬカフェへ！";
print "<hr>";
?>
```

FIG 17-06 実行結果



<img>タグ

画像を表示するときは、<img>タグを使います。次のような形式で記述します。

書式 <img>タグの使い方

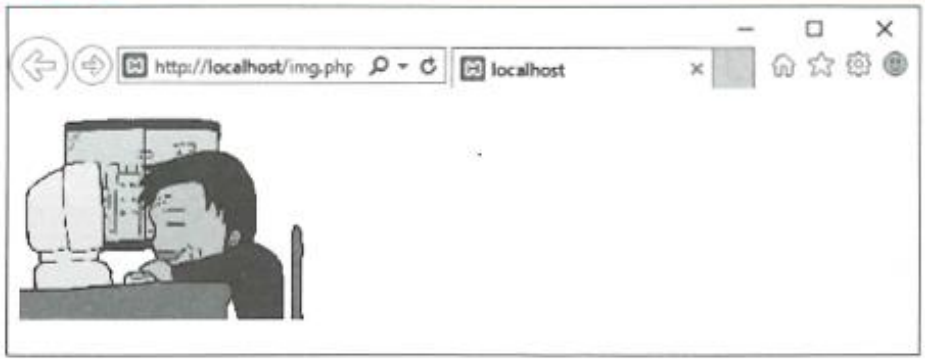
```
<img src="画像ファイルのURL" alt="代替テキスト">
```

src属性には、表示する画像ファイルの名前や保管されている場所を指定します。たとえば次は「pic」フォルダにある「oya.gif」という画像を表示するものです。

LIST 17-05 img.php

```
print "<img src='pic/oya.gif' alt='お父さんのイラスト'>";
```

FIG 17-07 実行結果



これで「picフォルダにあるoya.gifファイル」が表示されます。

「alt属性」は、画像が表示できない場合に表示されるテキストを指定しますが、HTML5では必ず指定する必要があります。上記の例では、画像表示がなんらかの理由でできない場合、「お父さんのイラスト」と表示されます。

<meta>タグ

<meta>タグは、ページの情報を設定するタグです。たくさんの機能があります。たとえば「検索ロボットの検索に引っかかってほしいキーワードを設定」、「検索結果に表示される文章を設定」、「自動的に指定したページに移動させるように設定」「文字エンコーディングを設定」・・・と、いろいろな情報が設定できます。

ここでは、「ページの文字エンコーディングを設定する方法」を説明します。

一般的にブラウザは、送られてきたWebページの文字エンコーディングを自動的に判定します。しかし、まれに間違う場合もあり、誤った文字エンコーディングが選択された場合、文字化けが起こります。きっとみなさんも、文字化けしたWebページを見たことがあるでしょう。こんなときには、<meta>タグで文字エンコーディングをブラウザに伝えてやれば、正しく表示されるようになります。

<meta>タグは<head> ～ </head>の領域(→P.392)に、次のような書式で記述します。

書式 <meta>タグで文字エンコーディングを設定

```
<meta charset="文字エンコーディング">
```

「文字エンコーディング」の部分には、たとえば次を指定します。

TABLE 17-01 文字エンコーディングの指定方法

文字エンコーディング	指定する文字
UTF-8	UTF-8
シフトJIS	Shift_JIS
日本語EUC	EUC-JP
JIS	ISO-2022-JP

「Shift_JIS」の「_」はアンダーバー、「UTF-8」の「-」はハイフンなので注意してください。

「UTF-8」を指定する場合、次のようになります。

```
<!DOCTYPE html>
<html>
<head>
<title>meta要素のサンプル</title>
<meta charset="UTF-8">
</head>
…
```

本書では、第21章で紹介する「ちょっと実用掲示板」で、この<meta>タグを使っています。

CSSによる色やフォントサイズの指定

HTML5では、背景や文字の色、サイズなど「見栄え」に関する設定はCSS（カスケーディングスタイルシート：Cascading Style Sheet）を使って行います。本書ではCSSに関する詳細は省略し、使用しているプロパティの意味だけを簡単に解説します。

背景の色の指定

CSSの設定方法にはいろいろなパターンがありますが、本書ではタグのstyle属性に直接CSSを記述する方法を採用しています。CSSは「プロパティ：値」という構文に則って記述します。

たとえば次は画面の背景を水色（aqua）に設定する例です。これにはbackground-colorプロパティを使用します

```
<body style = "background-color:aqua">
```

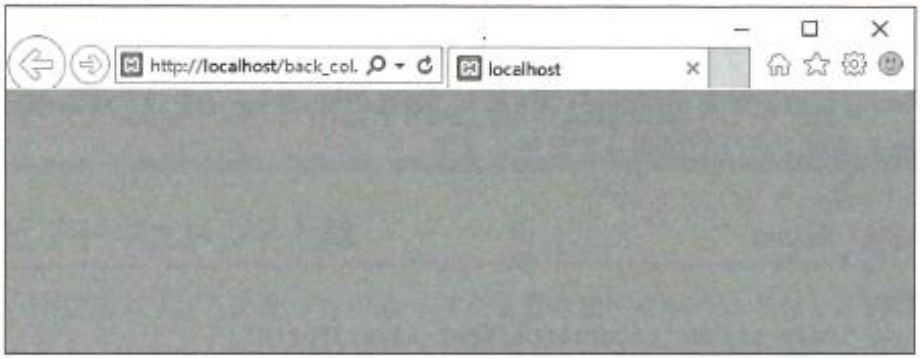
style属性　プロパティ　設定する値

次は上記を実行するPHPスクリプトです。

LIST 17-06 back_col.php

```
<?php
print "<body style = ' background-color:aqua '>";
?>
```

FIG 17-08 実行結果



プロパティの値はコロン（:）で区切って記述します。background-colorプロパティに

限らず、CSSにおける色の指定は、色の名前（カラーネーム）やRGBの各値を示す16進数で行います。

TABLE 17-02 styleで指定する色

色	カラーネーム	RGBの値
黒色	black	#000000
濃紺	navy	#000080
青色	blue	#0000ff
緑色	green	#008000
青緑色	teal	#008080
ライム色	lime	#00ff00
水色	aqua	#00ffff
くり色	maroon	#800000
紫色	purple	#800080
オリーブ色	olive	#808000
灰色	gray	#808080
銀色	silver	#c0c0c0
赤色	red	#ff0000
赤紫色	fuchsia	#ff00ff
黄色	yellow	#ffff00
白色	white	#ffffff

文字のサイズと色の指定

段落全体の文字色やフォントサイズを設定する場合、 や<p>タグでstyle属性を指定します。文字の色はcolorプロパティ、フォントサイズはfont-sizeプロパティで設定します。 と<p>は、ともに囲まれた範囲をブロックとして定義するために使いますが、<p>の場合はブロックの前後に1行の空きができるのに対して、 では空きができないという違いがあります。

次は「ＳＱＬカフェ掲示板だよ」の段落の文字列を、 を使って「青色」で「35ポイント」に設定する例です。このように複数のプロパティを設定する場合、それぞれをセミコロン（;）で区切って記述します。

LIST 17-07 div.php

```
<?php
print "<div style='color:blue;font-size:35pt'>";
print "ＳＱＬカフェ掲示板だよ";
print "</div>";
?>
```

FIG 17-09 実行結果

同じ行にある文字列の一部分だけを設定するときは、次のようにタグのstyle属性で設定します。次は「掲示板」の文字列だけを50ポイントにする例です。

LIST 17-08 span.php

```
<?php
print "ＳＱＬカフェ<span style='font-size:50pt'>掲示板</span>だよ";
?>
```

FIG 17-10 実行結果

ヒアドキュメントとnl2br関数

ヒアドキュメントとは

PHPスクリプトを使ってWebページを書き出すということは、HTMLタグをテキストとして「print」や「echo」で書き出すことになります。「print」などで書き込むタグは文字列なので、「" "」で囲むことになります。しかし、「属性値を' 'で囲え」とか「¥を付けろ」とか、面倒な指定が多いですね。

こんなときに便利なのが、大量の文字列を変数に格納するヒアドキュメント(here document)です。ヒアドキュメントは、「<<<終了を表す文字」の次から「終了を表す文字;」までの範囲を「一連の文字列」として扱います。

書式 ヒアドキュメント

```
<<<終了を表す文字

    文字列

終了を表す文字;
```

たとえばP.394の、Webページを書き出すスクリプトを思い出してください。HTMLを出力するたびに「print "××"」を何度も何度も入力して、毎回こんなふうに打ち込むのは面倒だ…と感じた方は多いでしょう。

これを、ヒアドキュメントを使って記述すると、List17-09のようになります。「終了を表す文字」は「eot」(end of text)としてみました。

LIST 17-09 here.php

```
<?php
$mozi=<<<eot
<!DOCTYPE html>                                          1
<html>
<head>
<title>ソフトバンククリエイティブへのリンク</title>
</head>
<body>
<a href="http://www.sbcr.jp/">ソフトバンククリエイティブへ</a>
</body>
</html>
eot;
print $mozi;                                             2
?>
```

1の範囲が「$moziに」なります。この範囲では「" "」を気にする必要はありません。2で「$mozi」を書き出しています。つまり「$mozi=<<<eot」と「eot;」の間にある純粋なHTMLの部分は、そのまま記述できます。

printで書き出す文字列は「" "」や「' '」で囲まなくてはいけません。「" "」や「' '」で囲まれた文字列の中に「"」や「'」を文字として書き出そうとする場合、「¥」を付けるなどの処理が必要となってきます。これに対して、ヒアドキュメントの中に「" "」や「' '」が入るのは問題ありません。「" "」や「' '」がよく入るSQL文やHTMLタグの記述では

ヒアドキュメントが重宝します。

nl2br関数とは

さて、List17-10のPHPスクリプトは、どのように表示されるでしょうか？

LIST 17-10 nl2br_nasi.php

```
<?php
$str=<<<eot
こんにちは                                    1
こんばんは
eot;
print $str;
?>
```

「こんにちは」「こんばんは」の2つの文字列をヒアドキュメントとして変数「str」に代入し、「print」でこれを書き出しています。

1では、「こんにちは」と「こんばんは」の間で改行しています。しかし、これを実行すると単に1行で「こんにちはこんばんは」と表示されてしまいます。

FIG 17-11 実行結果

文字列が改行されるということは、「文字列の中に改行コードが入る」ということを意味します。しかし、
（→P.396）などのような改行を命令するタグがなければ、Webページの表示が改行されることはありません。

たとえば、こんな問題があります。掲示板の入力ボックスに、改行したメッセージが入力されたとします。この入力された文字をそのまま「print」で書き出した場合、改行のないつながった文字列になってしまうのです。

このように、改行が無視されると困る場合もあります。こんなときは、「改行コードがある場合、改行タグを挿入する」機能を持つnl2br（「エヌエルツービーアール」と読む）という関数を使いましょう。

List17-10の例から、「$str」の値をnl2br関数で処理してprintで書き出します。「print $str;」の部分を「print nl2br ($str) ;」とするわけです。

LIST 17-11 nl2br.php

```
<?php
$str=<<<eot
こんちは
こんばんは
eot;
print nl2br($str);                                    1
?>
```

FIG 17-12 実行結果

1のようにnl2br関数を使うことで、改行の機能が再現されました。

COLUMN

<textarea>タグ

たくさんの文字列が入力できるテキストエリアを設定するタグに、「<textarea> ～ </textarea>」があります（→P.489）。ここに入力された文字列の改行を処理するときに、nl2br関数が活躍します。

ブラウザからPHPでデータを送受信する

さて、ひととおり基礎知識が身に付いたところで、PHPを使ってブラウザからデータをやり取りするする手順を学んでいきましょう。

ブラウザとPHPファイルとのデータのやり取り

まず、ブラウザからMySQLを操作するときの、データのやりとりを考えてみましょう。次のような手順になります。

①ブラウザに表示されたWebページから、ユーザーがデータを送信

▼

②Webサーバーに置かれているPHPファイルが、データを受け取る

▼

③PHPスクリプトが、MySQLデータベースとやり取りを行う

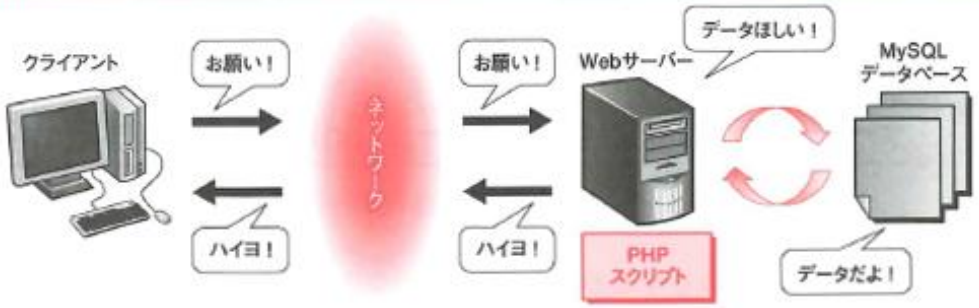
▼

④PHPスクリプトが、結果を載せたWebページを書き出しブラウザに送り返す

▼

⑤ブラウザが結果を受け取る

FIG 17-13 ブラウザとPHPとMySQL



MySQLを使う前に、ここではHTMLファイルとPHPスクリプトの間でデータをやり取りする手順を学びましょう。次のような仕組みを作ってみます。

データの送り側

- HTMLファイル「okuri.html」
- テキストボックスに入力した文字列を、送信ボタンをクリックすることで送信する

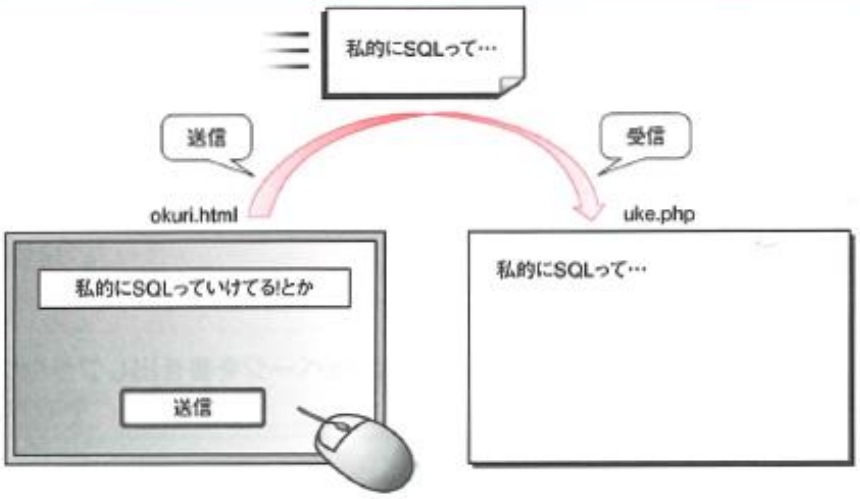
データの受け側

- PHPファイル「uke.php」
- 受信したデータをそのまま表示する

「okuri.html」は送り側のWebページ（HTMLファイル）、「uke.php」は受け側のPHPスクリプトです。「okuri.html」が送ったデータを、「uke.php」が表示する、というシ

ンプルな例です。2つのファイルは、Webサーバーの公開するフォルダに保存します。

FIG 17-14 ファイル間のデータ送受信



次項から、「Webページからのデータ送信」を行うHTMLファイル（okuri.html）を作成します。さらにP.411から、受け取る側のPHPスクリプト（uke.php）を作成します。

データを送信するWebページ「okuri.html」を作る

まずは、「Webページからのデータ送信」を行うHTMLファイルです。ここでは、

- ボタンをクリックすると、テキストボックスに入力されたデータが送信される

という仕組みを設定します。

使う「部品」は2つ。「ボタン」と「テキストボックス」です。この2つの部品は、後述する「<input>」タグで設定します。送信する仕組みを作る場合、一般的にはフォームを作成し、その中にテキストボックスやボタンなどの部品を配置します。

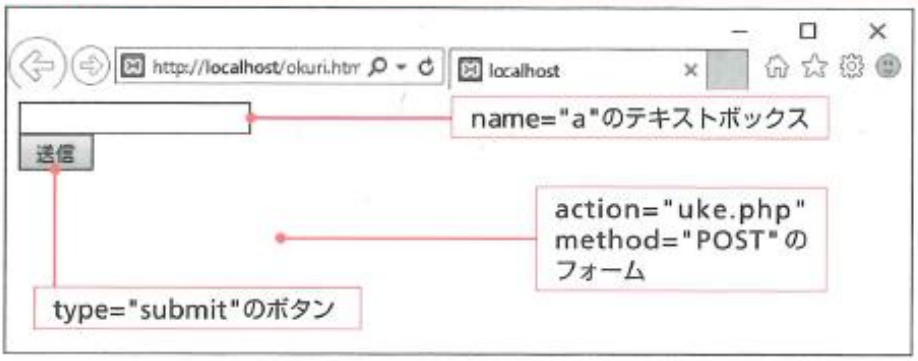
List17-12のようにテキストファイルを作成し、公開されるフォルダに「okuri.html」のファイル名で保存してください。なお、ここでは<html>、<head>、<body>などのタグは省略しています。

LIST 17-12 okuri.html

```
<form method="POST" action="uke.php">
<input type="text" name="a">
<div>
<input type="submit" value="送信">
</div>
</form>
```

間違いなく、「okuri.html」のファイル名で、公開されるフォルダに保存しましたか？　Apacheが動作し、スクリプトの記述に誤りがないことを確認したら、ブラウザのアドレスバーに「http://localhost/okuri.html」と入力してください。次のように表示されるはずです。

FIG 17-15 「okuri.html」



フォームの表示に成功したら、入力したタグの機能を見てみましょう。

<form>タグ

送信用のフォームを定義するのが「<form>」タグです。<form>タグで送信用フォームを作成し、この中に<input>タグなどでテキストボックスやボタンを設定します。<form>タグは、次のような形式で記述します。

書式 <form>タグの形式

```
<form method="送信の方法" action="データの送信先">
    ここに<input>タグ等を記述することで、ボタンやボックスを設定
</form>
```

<form>タグには、次のようにactionやmethodの属性を記述します。

TABLE 17-03 <form>タグに設定する主な属性

属性	内容
action	データを送信する先、つまり受け取って処理してくれるプログラムを指定する
method	データを送信する方法（→P.412）を指定する。POSTかGETの2種類のどちらかを指定

actionやmethod以外にも、送信方法（MIMEタイプ）を指定する「enctype」等が設定できます。

<input>タグ

「<input>」タグを「<form ××> ～ </form>」の間に記述することで、送信用のデータを入力するテキストボックスや、送信用のボタンなどの部品が設定できます。

書式 <input>タグの形式

```
<input type="ボタンの種類" name="データを識別する名前" size="サイズ"
       value="表示する文字"や"送信する値" "デフォルト値">
```

TABLE 17-04 <input>タグに設定できる主な属性

属性	内容
type	部品の種類を指定する文字列を記述する
name	その部品に付ける「データを識別する名前」を設定する
size	テキストボックスの幅を設定する
value	ボタンなどで文字列が表示される場合、その文字列を設定する

type属性には、次のような部品を設定できます。

TABLE 17-05 type属性の設定

記述	設定される部品の種類
type="submit"	データを送信する機能を持つボタン
type="button"	ボタン
type="text"	テキストボックス
type="checkbox"	チェックボックス
type="radio"	ラジオボタン
type="hidden"	表示なし（データ送信だけ）

<input>タグを使った例

List17-13は<input>タグを使った例です。「データを識別するための名前」を「a1」としたサイズ50のテキストボックスです。

LIST 17-13 input_text.html

```
<input type="text" name="a1" size="50">
```

FIG 17-16 実行結果

List17-14は、「クリック」と表示されている送信用ボタンです。

LIST 17-14 input_submit.html

```
<input type="submit" value="クリック">
```

FIG 17-17 実行結果

List17-15は、「データを識別するための名前」を「r1」として、「bad」という値を送信するラジオボタンです。

LIST 17-15 input_radic.html

```
<input type="radio" name="r1" value="bad">
```

FIG 17-18 実行結果

List17-16は、ブラウザには何も表示されませんが、「隠れデータ」という文字を送信します。

LIST 17-16 hidden.html

```
<input type="hidden" value="隠れデータ">
```

「type="hidden"」とした<input>タグは、ブラウザには何も表示されません。これは、ユーザーには見せる必要のないデータを強制的に送信するときに使います（→P.489）。

作成したフォームの確認

P.406で作成したList17-12のフォームを確認してみましょう。

```
<form method="POST" action="uke.php">
<input type="text" name="a">
<div>
<input type="submit" value="送信">
</div>
<form>
```

<form method="POST" action="uke.php">

<form>タグには「action="uke.php"」が付いています。このため、［送信］ボタンをクリックすると、「uke.php」というPHPスクリプトにデータを送信します。

method属性は「POST」を指定しているので、POST送信を行うことになります。POST送信についてはP.412から解説します。

<input type="text" name="a">

type属性が「text」になっているので、テキストボックスを作ります（P.408の表を参照）。「データを識別する名前」であるname属性には「a」が指定されています。このため、データを受け取るPHPスクリプト「uke.php」では「a」という名前で、データを判断します。

<input type="submit" value="送信">

type属性が「submit」になっているので、データを送信する働きのあるボタンを設定します。また、value属性が「送信」となっているので、ボタンに「送信」という文字が表示されます。

データを受信して表示する「uke.php」を作る

今度は、受け取り側の「uke.php」についてです。「uke.php」は、P.406の「okuri.html」のテキストボックスで入力されたデータを受信します。

「okuri.html」の<form>タグでは、method属性を"POST"としたので、POST送信が

行われます。POST送信で送られたデータは、次のように$_POSTという変数で受け取ることができます。

書式 method="POST"で送信されてきたデータが入る変数

```
$_POST["データを識別する名前"]
```

「$_POST」は、PHPであらかじめ決められている変数名です。ここにPOSTで送信されたデータが配列として保管されます。

「$_POST」は、連想配列(→P.384)として扱うことができます。配列の添え字は「データを識別する名前」を指定します。

今回送信されるデータが入力されるテキストボックスには、「a」という「データを識別する名前」が付いていました(name="a")。このため「uke.php」側では、連想配列の添え字に"a"を指定して、「$_POST["a"]」と記述すれば、送信されたデータを取り出すことができます。

書式 「name="a"」のテキストボックスの値を受け取る変数

```
$_POST["a"]
```

List17-17は、「okuri.html」で送られたデータを、単純に書き出す「uke.php」です。作成して、「okuri.html」と同様に、公開されるフォルダに保存してください。

LIST 17-17 uke.php

```
<?php
print $_POST["a"];                                    1
?>
```

1でデータを認識する名前によって受け取った値を書き出します。

COLUMN スーパーグローバル変数

「$_POST」のように、何も宣言せずに、プログラムのどこからでも使える変数を「スーパーグローバル変数」といいます。他に、GET送信(→P.413)で使う「$_GET」、Cookieの利用で使う「$_COOKIE」などがあります。

データを送信し受信する

実際に送受信をしてみます。あらかじめ、P.406の「okuri.html」と前ページの「uke.php」が、公開されるフォルダに保存されていることを確認してください。また、Apacheが動作している（→P.18）ことを確認してください。

ブラウザのアドレスバーに「http://localhost/okuri.html」のURLを入力すると、P.407のように「okuri.html」が表示されます。テキストボックスに、送信する文字列を（ここでは「送信文字列」と入力）入力し［送信］ボタンをクリックします。これで「uke.php」にデータが渡され、入力した文字列が表示されるはずです。

FIG | 17-19 実行結果

同じパソコン内で「送った」「受け取った」といわれても、実感がわかないかもしれませんが、Web上での送受信でも基本的な方法は同様です。

POSTとGETによるデータ送信

データを送信し受信する

WebページからPHPスクリプトにデータを渡すとき、P.406で<form>タグの中に「method="POST"」と記述しました。この形式でデータを渡すことを、**POST送信**といいます。POST送信については、ここまでに何度か出てきましたね。

データを渡す方法には、POSTの他に**GET送信**があります。POSTとGETそれぞれの特徴は次のようになります。

POST送信

- データはURLには付けない
- 送信できるデータはテキストとバイナリの両方が可能

GET送信

- URLに付けてデータを送る（データが見られてしまう。不正なデータを送信される恐れがある）
- 送信できるデータはテキストのみ
- 何も宣言しなければGETメソッドが使われる（デフォルト）

GET送信とそれぞれの違いについて、詳しく見ていきましょう。

GETメソッドでGET送信を行う

「GET送信」を行う場合は、たとえば前項で作成した「okuri.html」と「uke.php」なら、リスト中に記述されている「POST」の文字をそのまま「GET」に変えれば、送信のメソッドが「GET」になります。

実際に「okuri.html」と「uke.php」の送信方法を「GET」に変更してみることにしましょう。List17-18は、受け側の「uke.php」の「$_POST ["a"]」の部分を単純に「$_GET ["a"]」に変更したものです（1）。これで、GET送信で送られきた「name="a"」のデータが受信できます。PHPファイルの名前は「get_uke.php」としました。

LIST 17-18 get_uke.php

```
<?php
print $_GET["a"];                                    1
?>
```

List17-19のスクリプト「get_okuri.html」は、送り側の「okuri.html」の「method="POST"」の部分を、「method="GET"」に変えたものです（2）。また、受け側のPHPファイルの名前を「get_uke.php」としたため、「action="get_uke.php"」に変更しています。

LIST 17-19 get_okuri.html

```
<form method="GET" action="get_uke.php">                    2
<input type="text" name="a">
<div>
<input type="submit" value="送信">
</div>
</form>
```

FIG 17-20 実行結果

GETとPOSTの違い

GETで送信するとき、そしてその結果を表示するときも、POSTとほとんど同じです。しかし、1か所だけPOST送信とは違うところがあります。どこが違うか注意深く観察してください。

FIG 17-21 POST送信の場合

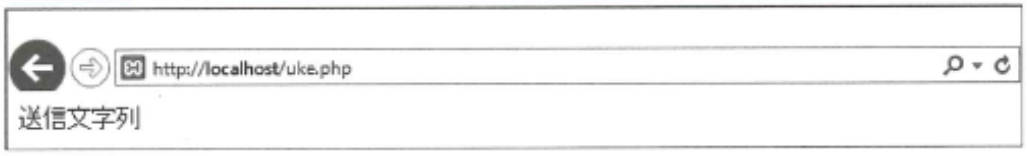
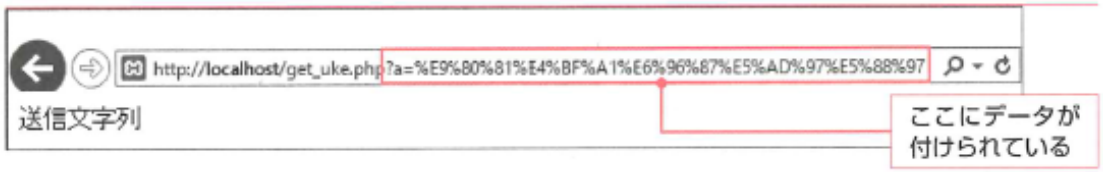


FIG 17-22 GET送信の場合



違いがわかりましたか？　実は、GET送信のときはURLの後に、「?a=%E9%80&81…」のような表示が付くのです。これに対してPOST送信のときは、アドレスバーには単純に「http://localhost/uke.php」と表示されるだけです。つまり、GETメソッドを使った場合「データはURLにくっ付けて送っている」のです。

形式は、次のようになります。

書式 GETメソッドでのアドレスバーの表示

```
URL?データ名=データ
```

「データ名」とは、「name="a"」などデータを入力したボックスに付けた「データを識別する名前」です。送信されるデータが日本語など2バイトの場合、別のコードに変換されて送られます。

そのため、GET送信では「送信するデータ」と「データを識別する名前」が見えてしまう、ということになります。悪意を持った人が「不正なデータをURLに付けて送る」ことや、データを盗み見られてしまう可能性も考えられます。GET送信でシステムを構築するときは、こうしたセキュリティ上の問題を十分考慮する必要があります。

GETメソッドでURLに値を付けて送信してみる

「GET送信では、データをURLにくっ付けて送る」ということを説明しました。ということは、実はHTMLファイルを使わなくても、URLにデータをくっ付ければGET通信ができてしまう、ということなのです。

では、GET送信でデータを送ってみましょう。P.413の「get_okuri.html」を使って、「12345」のデータを送ってみます。次の文字列をアドレスバーに入力しEnterキーを押してください。

```
http://localhost/get_uke.php?a=12345
```

FIG 17-23 実行結果

ちゃんと「12345」と表示されますね。

日本語を送ることもできます。今度は「練習」の文字列をmethod="GET"で送信してみることにしましょう。UTF-8では「練」の文字コードが「E7 B7 B4」、また「習」が「E7 BF 92」です。そこで、「name="a"」が設定されているテキストボックスに「練習」を入力して「submit」する、ということは、次のURLを送信するのと同じことになります。

```
http://localhost/get_uke.php?a=%E7%B7%B4%E7%BF%92
```

FIG 17-24 実行結果

何も宣言せずにデータを送信する

なお、何も宣言しなければ自動的に「GET送信」になります。

たとえば、特に<form>タグを設定することなく、いきなり次のハイパーリンクだけのファイルを作ってみてください。

LIST 17-20 get_okuri2.html

```
<a href="get_uke2.php?a=777">777を送るからね</a>;
```

FIG 17-25 実行結果

そして、List17-21のファイルでこれを受けます。

LIST 17-21 get_uke2.php

```
<?php
print $_GET["a"];
?>
```

FIG 17-26 実行結果

「get_okuri2.html」のハイパーリンクをクリックするだけで「777」が送受信され、「777」と表示されるはずです。method="GET"などの、面倒な宣言不要でデータが送れてしまうのです。

GETでは、このように手軽に、ハイパーリンクにデータをくっ付けた送信ができるのです。

COLUMN

Googleでもデータ URLにくっ付いている？

YahooやGoogleなど、検索エンジンを使って検索を実行したとき、アドレスバーのURLの後ろにごちゃごちゃと怪しげな文字列が付いているのに気がついている方も多いでしょう。これも、URLに検索の文字列をくっ付けて送信しているんですね。

2017年6月現在、アドレスバーに次のように入力してEnterを押せば、Googleの検索窓に「abcde」と入力して検索するのと同じになります。

次はGoogleで「abcde」の文字列を検索する記述です。

```
http://www.google.co.jp/search?q=abcde
```

FIG 17-27 実行結果

半角アルファベットなら、「?q=××」の××に好きな文字を入れれば簡単にできます。ぜひいろいろ試してみてください。

ちなみに、Yahooでは次のようにすると検索できます。こちらは「p」の文字を使います。

```
http://search.yahoo.co.jp/search?p=abcde
```

FIG 17-28 実行結果

なお、日本語の文字の検索は、上記の方法ではできないので注意してください。

まとめ

この章では、以下のような事柄を解説しました。

- 動的なWebページと静的なWebページの違い
- 基本的なHTMLタグの記述方法
- Webページからのデータ送信とPHPスクリプトでの処理、その流れ
- データを送信するフォームの作成方法
- 送信方法の種類

PHPに限らず、Webアプリケーションを作成する上でPOSTとGETはその基本となる知識です。POSTとGETの違いも、ここでしっかりと覚えておきましょう。

チェックしてみよう

この章で学んだ以下の事柄を理解しているかチェックしてみましょう。

- ☐ HTMLの基本的な記述方法がわかる
- ☐
、<hr>、<a>、<img>、<meta>、 、<p>、の各タグの意味を理解している

- ☐ <form>タグを使ってフォームが作成できる
- ☐ <input>タグの使い方を理解している
- ☐ テキストボックスの値をPOST送信することができる
- ☐ GET送信ができる
- ☐ POSTとGETの違いを理解している

練習問題

問題1

次のような「100個のラジオボタンから選択して年齢データを送信するフォーム」(radio.php)と、それを「受信して、年齢を含むメッセージを表示するPHPスクリプト」(radio_uke.php)を作成してください。

FIG 17-29 radio.php

FIG 17-30 radio_uke.php

HINT ラジオボタンを100個作るところがポイント。10個ずつに改行する方法はいろいろある。whileとifを使うのがわかりやすい

解答例

問題 1

次のPHPスクリプトを作成します。

LIST 17-22 サンプル「radio.php」(送信用フォームのPHPスクリプト)

```
<form method="POST" action="radio_uke.php">
あなたの年齢を選択して、送信ボタンをクリックしてください。<br>

<?php

$i=1;
$c=1;
print "<div>";
while($i<=100){
print "<input type='radio' name='r' value='$i'>$i  ";
   if($c==10){
      print "</div><div>";
      $c=0;
   }
$i++;
$c++;
}

?>

<input type="submit" value="送信">
</div>
</form>
```

LIST 17-23 サンプル「radio_uke.php」(受信用PHPスクリプト)

```
<?php
print "あなたの年齢は".$_POST["r"]."歳なのですね";
?>
```