特集1 Part2

Webアプリ開発特有の流儀や約束事を知ろう

ここではPHPのコードを用いて、Webアプリケーション開発に必要な技術について説明していきます。見えないフォーム、cookie、セッションを使ったデータのやり取りなどをマスターしてください。

園田 誠

インターネットを介してクライアントとサーバーがやり取りするWebアプリケーションの開発では、デスクトップアプリケーションの開発とは大きく異なる部分があります。Part2では、Webアプリケーション開発で押さえておきたい特有の流儀や約束事について説明します。

サーバー側のプログラムはPHPで記述します。PHPは中・小規模なWebアプリケーション開発でよく使われているスクリプト言語です。掲載するPHPプログラムのコードについては、本文の説明やリスト中のコメントをご覧いただけば、事前にPHPの知識がなくても理解していただけるかと思います。22～23ページの別掲記事「XAMPP環境を用意してPHPプログラムを実行する」を参照して、動作を確認しながら読み進めてください。

Webアプリケーションの基本形を押さえる

最初にWebアプリケーションの基本形として、入力フォームを持つHTMLドキュメントと、そのフォームが送ったデータを受け取って応答を返すPHPプログラムについて勉強します。ブラウザに表示される画面は**図1**、送信ボタンを押した後のWebサーバーからの応答画面は**図2**です。入力側のHTMLであるfrom.htmlと、フォームが送信したデータを処理するプログラムであるto.phpを使って、コードの書き方の“流儀”を学びましょう。

どの要素への入力なのかをname属性で伝える

まずはfrom.htmlです。テキストエディタなどで**リスト1**のコードを入力したら、文字コードをUTF-8にしてファイルに保存してください（**別掲記事**の「文字コードはUFT-8が基本」を参照）。

<form>タグ内でaction属性に指定しているのは、送信先のPHPファイルの名前です。続くmethod属性では、データを送信する際のメソッドを指定します。リスト1ではGETにしていますが、後述するようにPOSTにすることもできます。

フォームには二つのテキストボックスと、二つで1組のラジオボタンの合計四つの入力用の要素（エレメント）、およ

図1● 入力フォームを備えたHTMLドキュメント（from.html）の実行画面

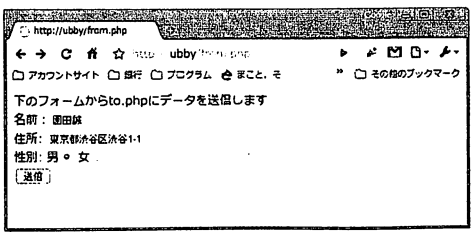
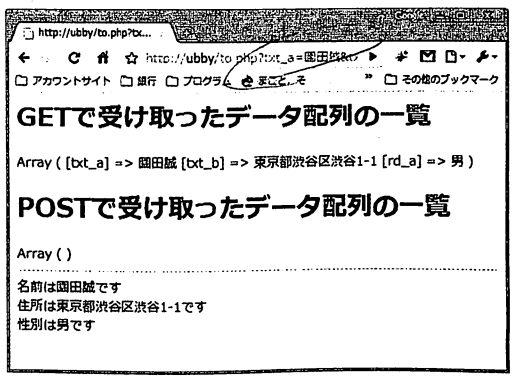


図2● from.htmlからデータを受け取ったPHPプログラム（to.php）の実行画面



リスト1●入力フォームを備えたHTMLドキュメント（from.html）のコード

```
<html><head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
</head><body>
下のフォームからto.phpにデータを送信します
<form action="to.php" method="GET">
名前：<input type="text" name="txt_a"><br />
住所：<input type="text" name="txt_b"><br />
性別：
男<input type="radio" name="rd_a" value="男">
女<input type="radio" name="rd_a" value="女"><br />
<input type="submit">
</form>
</body></html>
```

び送信ボタンがあります。HTMLではそれぞれのエレメントを<input>タグで記述します。<input>タグのtype属性にtextを指定するとテキストボックス、radioを指定するとラジオボタン、submitを指定すると送信ボタンになります。

Webサーバーにデータを送信するフォームで重要になるのは、各エレメントのname属性です。この属性の値が、受信側のPHPプログラムで「フォーム上のどのエレメントに入力されたデータなのか」を判別する際のカギになります。上の二つのテキストボックスには、name属性の値としてそれぞれtxt_aとtxt_bを指定しています。

ラジオボタンのエレメントは、どちらもname属性がrd_aになっています。同じname属性を持つラジオボタンは、どちらか一方しか選択できない択一型になります。送信ボタンを押すとブラウザは、選択した方のラジオボタンのエレメントに指定したvalue属性の値を送信します。ラジオボタンのvalue属性には、わかりやすくするために見出し語（ここでは「男」と「女」）を設定することが多いようですが、実際には、「man」と「woman」、あるいは「1」と「2」のように任意の値を指定できます。

データを入力して送信ボタンを押すと、<form>タグのaction属性で指定したファイルあてに、method属性で指定したメソッドにのっとって、各エレメントのデータを送信します。例えば、図1のように入力して送信ボタンを押すと、Webサーバーからの応答画面は図2のようになります。図2でブラウザのURL欄を見てください。図2では一部しか見えませんが、ファイル名（to.php）に続いて

?txt_a=園田誠&txt_b=東京都渋谷区渋谷1-1&rd_a=男

と表示されています。Part1で学習したように、GETメソッドのリクエストでは、URLと一緒に入力データを送信していることが確認できます。

受け取ったデータは連想配列に格納する

リスト2はfrom.htmlが送信したデータを受け取るサーバー側のPHPプログラム（to.php）のコードです。先ほどまでサーバーから応答画面として見てきた図2は、このプログラムを実行して作成されたものです。

リスト2の<?php ～ ?>の中に書かれているのがPHP

文字コードはUFT-8が基本

Windowsベースで開発を行うと文字コードとしてシフトJISを使うことが多くなります。しかし、Webアプリケーションの場合、実践で使うサーバーの多くはLinuxベースです。最近の主要なLinuxディストリビューションでは、ほぼすべてでUTF-8が採用されています。Linuxサーバー上でコードを修正するなど保守の観点から、PHPプログラムのソースコードはUTF-8の文字コードで記述するクセをつけておいた方がよいでしょう。

同一サイト内で使用するHTMLも同様にUTF-8に統一しておくと安全です。文字コードを統一するのは、ブラウザ表示の際の文字化けの発生を避けるためにも有効です。HTMLドキュメント内にJavaScriptを含むようなケースで、JavaScriptのエラーや誤動作が発生することもあります。

PHPプログラムやHTMLドキュメントの文字コードは、エディタでファイルに保存する際に指定します。Webブラウザはプログラムなどの文字コードを自動判別して表示しますが、文字コードの自動判別に失敗して文字化けが起こってしまう場合もあります。そうしたことが起こらないように、PHPプログラムやHTMLドキュメントには、<head>～</head>内に次のように書いて文字コードを指定しておきます。

```
<meta http-equiv="Content-Type"
 content="text/html;charset=UTF-8">
```

HTMLドキュメントとPHPプログラムをUTF-8で作成する場合、MySQLなどのデータベースにもUTF-8でデータを納めるのがよいでしょう。ただし諸般の事情で（実際の開発現場では多々あります）データベースの初期設定の文字エンコードを変更できないこともあります。こうした場合は、データベースとのやり取りの際に、PHPプログラム内で文字コードを変更することになります。フォームからUTF-8で届いたデータをEUCに変換してデータベースに格納し、データベースから出すときにはEUCからUTF-8に変換してHTML上に書き出すといった具合です。

プログラムです。PHPでは、行頭に「//」を付けると、その行はコメントになります。echo "～"は、～の部分をそのまま出力する命令です。命令の最後には「;」を付けるのを忘れないようにしてください。

プログラムでは、画面の上半分で受け取ったデータをデバッグ用に表示し、下側で各エレメントから受け取ったデータをそれぞれ表示しています。ここに登場している$_GETという変数がプログラムのキモです。

実は、ブラウザがGETメソッドで送信したデータは、サーバー側のPHPプログラムでは$_GETという名前の連想配列の変数に格納されます。連想配列とは、配列の添え字（インデックス）として数値以外にテキストなどを利用できる配列です。連想配列におけるインデックスを「キー」と呼びます。

連想配列$_GETのキーとしてフォームの各エレメントのname属性の値を指定すると、そこに入力されたデータを参照できる仕組みです。すなわちエレメントへの入力データは

```
$_GET[エレメントのname属性の値];
```

で参照できます。

from.htmlにおいて名前欄のエレメントのname属性はtxt_aでした。従って

```
$_GET[txt_a];
```

と書けば、名前欄に入力したデータを参照できます。住所欄に入力したデータも同様です。ラジオボタンの入力は$_GET[rd_a];で参照できます。このときの値は、選ばれた方のエレメントのvalue属性の値です。リスト2の(1)では、echo命令内で「.（ピリオド）」を使うことによって、前後に文字列をつなげてわかりやすく表示しています。

リスト2●from.htmlが送信したデータを受け取るPHPプログラム（to.php）のコード

```
<html><head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
</head><body>
<?php
//--------デバッグ用表示--------------------
echo "<h1>GETで受け取ったデータ配列の一覧</h1>";
//フォームからGETで来たデータの配列一覧
print_r($_GET);                                          (2)

echo "<h1>POSTで受け取ったデータ配列の一覧</h1>";
//フォームからPOSTで来たデータの配列一覧
print_r($_POST);
echo "<hr />";
//--------デバッグ用表示ここまで-----------

//POSTで送信された場合は$_GETを$_POSTに書き換える
//name=txt_aの欄から受け取ったデータ
echo "名前は".$_GET[txt_a]."です<br />";
//name=txt_bの欄から受け取ったデータ
echo "住所は".$_GET[txt_b]."です<br />";              (1)
//ラジオボタンから受け取ったデータ
echo "性別は".$_GET[rd_a]."です<br />";
?>
</body></html>
```

リスト2の(2)のprint_rは、図2の上半分のように、配列の中身を一覧表示してくれるデバッグ向けの関数です。

```
print_r(配列名);
```

という書式を覚えておくとよいでしょう。

from.htmlで<form>のmethod属性として、GETの代わりにPOSTを指定することもできます。この場合は、POSTメソッドでデータを送信します。

ブラウザがPOSTメソッドで送信したデータは、サーバー側のPHPプログラムでは、$_POSTという名前の連想配列に格納されます。各エレメントのデータを参照するには

```
$_POST[エレメントのname属性の値];
```

のように書きます。PHPプログラム上は、$_GETを$_POSTに書き換えただけです。POSTメソッドでデータを送信すると、GETメソッドのときと異なり、応答画面のブラウザのURL欄にはファイル名（to.php）だけが表示されることを確認してください。