データベース 2009 #4

久野　靖*

2009.10.29

はじめに

今回は前回説明できなかったPHPの機能で、Webアプリケーションで必須の機能と言える「セッション機能」を扱います。が、その前に前回の検索機能に注文の追加/削除をつけるというのもやっておきましょう。

そのあと前回残った話題であるPHPのセッション処理をやって、それでようやく材料が揃ってWeb書店作りができるので、宿題でできるだけ作ってみて頂こうと思っています。

1　前回の復習と整理

1.1　PHPの復習:基本

前回やったことは次のようなことでした。

- PHPはサーバ上で動くプログラミング言語。
- PHPプログラムはHTMLの中に「`<?php ... ?>`」という形で埋め込むことができる。
- PHPプログラムの中から`echo`でHTMLを出力できる。
- `header()`を使う場合はHTMLの一番先頭の「`<?php ... ?>`」の中で使う必要がある(日本語の文字コード指定等)。

PHPの文法と機能のまとめを別資料で配布しておきます。

1.2　PHPの復習:フォームデータの受信

PHPでフォームのデータを受け取るには次のようにすればよいのでした。

- `<form methpd="post" action="#">...</form>`を使用。
- PHP側では普通にHTML部品を書いておいてよい。
- 最初の表示時もフォーム送信時も同じPHPプログラムが動く。
- なので、プログラムの中でデータが来ているかどうかに基づき処理を枝分かれするのが普通。
- フォーム部品の情報は「名前:値」の対で受け取られる。
- ある名前のデータの有無は「`isset($_POST['名前'])`」で調べられる。
- そのデータ自体は「`$_POST['名前']`」で参照できる。

このように、簡単にWebページのデータを受け取れることはPHPの便利なところです。

ところで、昔のPHPでは配列`$_POST`を経由しなくても、「いきなりnameで指定した変数が用意されそこに値が入っている」ようになっていました。便利そうでいいと思いますか? それならなぜなくなったのでしょう?

*経営システム科学専攻

1.3 PHP の復習:PostgreSQL との接続

PHP から PostgreSQL を呼び出すには次の機能を使えばよいのでした。

- `pg_connect(`オプション指定文字列`)` — PostgreSQL サーバと接続し、接続オブジェクト (以下単に「接続」と記す) を返す。接続が失敗した場合は FALSE を返す。オプション指定文字列では色々な指定ができるが、とりあえず「`"dbname=`ユーザ名`"`」という指定のみで十分。なお、我々の環境では PHP は nobody というユーザ ID で動作しているので `pgsql` で「`grant all on` 表名 `to nobody;`」を実行して PHP が扱う表のアクセス許可を与えておくこと。
- `pg_query(`接続，実行文字列`)` — 任意の SQL 文を文字列として受け取って実行し、結果オブジェクト (以下単に「結果」と記す) を返す。実行が失敗した場合は FALSE を返す。
- `pg_num_rows(`結果`)` — 結果の中に何個の組が含まれているかを返す。
- `pg_fetch_row(`結果`)` — 結果から 1 つ組を取り出して配列として返す。繰り返し呼ぶと 1 つずつ組を取り出すことができ、終わりになったら FALSE を返す。
- `pg_escape_string(`文字列`)` — 文字列を受け取り、その中の特殊文字をエスケープする。なぜこれが必要かというと、入力文字列の中に「SQL コマンドをいったん終了させ、別のコマンドを実行する」ような仕掛けを組み込むことでデータベースの内容を任意に操作されることを防ぐため。つまり、フォームなどでユーザから入力されてきた情報は「すべて」この関数で変換してからでなければ `pg_query()` に渡してはならない。この関数はコマンドの区切りとなる「;」など「危険な」文字にエスケープ文字を付加して普通の文字として扱わせるように加工し、上記のような危険を抑止する。

基本さえ分かってしまえば、あとはどちらかというと SQL を書く話なのでデータベース屋さんには楽勝なわけです。

2 フォームを使った書籍検索に注文機能を入れる

では、前回やったフォームによる書籍検索に注文の追加/削除機能を入れてみます。よくあるように、各書籍の横に「数量欄」と「注文」「削除」ボタンをつけますが、そのデータをどうやって PHP 側に送信したらよいでしょう。次の 2 つの方針があり得ます。

(1) これまで通り、全体を 1 つのフォームとする→注文に関係ない書籍の情報もすべて毎回送られて来ることになる。ボタンや数量フィールドごとに ISBN 番号を値の一部に入れるなどして、どの本の注文/削除なのかを識別する必要がある。

(2) 書籍 1 冊ごとに別のフォームとする→書籍 ISBN は hidden フィールドなどで保持すればよい。その書籍のみのデータだけを送れば済むので通信量が少なくて済む。

これらとは別に、「提出」1 回で何冊でも本が注文できるようにすることもできそうですが、ちょっと処理が複雑で間違いが起きやすそうなので、それはやめておきます。

次に、カート情報をどうやって表現したらいいでしょうか。それは「カート」というリレーションを作り「顧客番号」「ISBN」「数量」を持たせるようにします。

```
create table カート (id varchar(20), isbn varchar(20), 数量 int);
grant all on カート to nobody;
```

さて、書籍を表示するときに「現在見ている顧客の」カートに入っている書籍については、図 1 のようにその数量を表示するべきですよね? これを行うために、左外ジョイン (left outer join) を使います。

左外ジョインとは? 通常のジョインでは、「2 つの表を結合して、両方の表に一致するデータがある列だけが組み合わさった表」ができますが、今はそうではなく「基本的には書籍データ、ただしその書籍データにカートデータがジョインできるばあいはそれも取る」わけです。このように、両方にデータが揃っていな

図 1: カートへの注文記録

いものも併せて取る join を outer join と言います。この場合、「書籍にだけあるものは取る、カートにだけあるものは (他人のカートは見せてはいけないので) 取らない」なので left outer join なわけです。

SQL で外ジョインを指定するには join 構文を使うのですが、その勉強をゆっくりやっている暇はないので具体例だけ示します。

```
select 書籍.isbn, 題名, 著者, 数量 from 書籍
  left join カート
    on ( カート.顧客番号 = xxx and カート.isbn = 書籍.isbn )
      where 題名 like '%〜%';
```

ここで顧客名は今のところ (まだ顧客管理ができないので)PHP プログラム中に固定で書き込んで起きます。

ではコードを示してみます。

```
<?php
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { padding: 4px; background: rgb(255,215,200) }
td { margin: 2px 4px; padding: 4px; background: rgb(200,215,255) }
</style></head><body>
<form method="post" action="#"><div class="test">
<p>検索語:<input type='text' name='word'>
<button name="cmd" value="lookup">検索</button></div></form>
<?php
  $userid = 'kuno';  // ここは適当に直す
  $cmd = $_POST['cmd'];
  if($_POST['word'] == '') {
    echo "<p>検索キーワードを入れてください。</p>";
  } else if(!($conn = pg_connect("dbname=kuno"))) {
    echo "<p>データベース接続できません。管理者に連絡を。</p>";
  } else if($cmd == 'lookup' || $cmd == 'order' || $cmd == 'remove') {
    if($cmd == 'order') {
```

```
      $isbn = $_POST['isbn'];
      if(is_numeric($_POST['num'])) $num = $_POST['num']; else $num = 1;
      $res = pg_query($conn, "delete from カート"
       ." where id = '{$userid}' and isbn = '{$isbn}'");
      $res = pg_query($conn, "insert into カート"
       ." values('{$userid}', '{$isbn}', {$num})");
    } else if($cmd == 'remove') {
      $isbn = $_POST['isbn'];
      $res = pg_query($conn, "delete from カート"
       ." where id = '{$userid}' and isbn = '{$isbn}'");
    }
    $word = pg_escape_string($_POST['word']);
    $res = pg_query($conn, "select 書籍.isbn, 題名, 著者, 数量 from 書籍"
      ." left join カート on ( カート.id = '{$userid}' and"
      ." カート.isbn = 書籍.isbn ) where 題名 like '%{$word}%'"
      ." order by 書籍.isbn");
    if(pg_num_rows($res) > 0) {
      echo "<table><tbody><tr><th>ISBN</th><th>題名</th><th>著者</th>"
          ."<th>数量</th></tr>\n";
      while($a = pg_fetch_row($res)) {
        echo "<form method='post' action='#'>";
        echo "<input type='hidden' name='isbn' value='{$a[0]}'>";
        echo "<input type='hidden' name='word' value='{$word}'>";
        echo "<tr><td>{$a[0]}</td><td>{$a[1]}</td><td>{$a[2]}</td>";
        echo "<td><input type='text' size='3' name='num' value='{$a[3]}'></td>"
            ."<td><button name='cmd' value='order'>注文</button>"
            ."<button name='cmd' value='remove'>削除</button>"
            ."</td></tr></form>\n";
      }
      echo "</tbody></table>";
    } else {
      echo "<p>検索結果が空でした。</p>";
    }
  }
?>
</body></html>
```

演習 各自の書籍検索画面も、注文数をカートに入れられるように改造してみよ。

3 セッション管理

Web アプリケーションを開発するときに面倒で分かりにくいことの 1 つに、「個々のページはあくまでも独立にサーバから取り寄せて来る」ということがあります。つまり、ユーザにとっては「あのページのあのリンクを選択したらこのページに来た」という「つながり」に見えていても、Web サーバ側ではそれぞれのページはあくまでも別のものなのです。

このことは通常は問題になりませんが (そしてそのおかげで WWW は効率よく多数のユーザをサポートできるのですが)、ショッピングサイトのようなものでは「あのページで選んだ商品がこのページでもカー

トに入っている」のでないと役に立ちません。つまり、1 回ログインしたらその後次々に取り寄せるページは「つながった単位」として扱いたいわけです。そのような単位を一般に**セッション**と呼びます。

セッションを作る一般的な方法は HTTP の**クッキー機能**を使うことです。クッキーとはサーバからブラウザに渡す小さな (せいぜい数百バイトの) 情報で、その同じ情報は、ブラウザで同じサーバのページを見た時に、ブラウザからサーバに提出されてきます。たとえば、ログインしたユーザのブラウザに「0123WXYZ」という固有のクッキーを送っておけば、次にブラウザから「0123WXYZ」が送られて来たら、ああさっきのユーザだな、と分かるわけです。

なお、ここでクッキーとしてユーザ ID のようなものを直接使えばいいのではと思ったかも知れませんが、それは**絶対にいけません**。というのは、クッキー情報はブラウザ上で自由に加工可能なので、誰かが私の ID のクッキーを勝手に焼いて使い、私になりすましてしまうかも知れません。上記のような乱数に基づくクッキー文字列では、誰がどのクッキーかを知ることは簡単ではなく (パケットの盗み見という方法はありますが、これは通信の暗号化で防げます)、また知られてもログアウトしたら無効になる情報なので危険が小さいのです。

なんと面倒な、と思ったかも知れませんが、PHP では**セッション管理**と呼ばれる機能が用意されていて、`session_start()` という関数を呼ぶことで、以後クッキーを使ったセッションの維持を自動的に行ってくれます。なお、この関数はヘッダにクッキー情報を投入するため、`header()` 関数と同様、ページ内容が出力されるより前に実行する必要があります。

セッション管理が開始されると、以後は「`$_SESSION`」という配列がセッションごとにに用意されるので、セッション内でページをまたがって保持したいデータはここに適当な添字 (名前文字列でもよい) をつけて格納しておけばよいのです。

ではこれを利用して、パスワードを用いたログイン処理のある例題を作って見ましょう。今度はセッションがあるので、ページを次のように 3 つ用意します。

- ページ A — ログインページ。ID を持っている人は ID とパスワードを打ち込んでログインするとページ C へ行く。ID を持っていない人は新しいログイン ID を選び、ページ B へ行く。
- ページ B — ユーザ情報変更ページ。新規ユーザも既存ユーザも、このページでユーザ情報 (パスワード、名前、電話番号) を変更することができる。変更は何回でも繰り返し行え、変更完了したらページ C へ行く。
- ページ C — 「本体」ページ。これは先のカートへの記録を行うページを多少改良して用意した。

これらの動作には当然データベースが必要ですが、これには次のような表を使うようにしました。

```
create table ユーザ (id varchar(20), pass varchar(20),
                     name varchar(20), tel varchar(20));
grant all on ユーザ to nobody;
```

ではさっそく、ページ A(ログインページ、図 2) から見てみましょう。

```
<?php
session_start();
if(!($conn = pg_connect("dbname=kuno"))) {
  $mesg = "データベース接続不能; 管理者に連絡を。";
} else if($_POST['cmd'] == 'newuser' && $_POST['newid'] != '') {
  $newid = pg_escape_string($_POST['newid']);
  $res = pg_query($conn, "begin");
  $res = pg_query($conn, "select id from ユーザ where id = '{$newid}'");
  if(pg_num_rows($res) == 0) {
    $res = pg_query($conn, "insert into ユーザ values('{$newid}',NULL,NULL)");
    if(pg_query($conn, "commit")) $_SESSION['userid'] = $newid;
```

図 2: セッション管理 — ログインページ

```
    header("Location: sam05b.php"); exit;
  } else {
    $res = pg_query($conn, "commit");
    $mesg = "その ID は取得されています; 別の ID を選んでください。";
  }
} else if($_POST['cmd'] == 'login' && $_POST['userid'] != '') {
  $userid = pg_escape_string($_POST['userid']);
  $res = pg_query($conn, "select id,pass from ユーザ where id='{$userid}'");
  $a = pg_fetch_row($res);
  if(pg_num_rows($res) == 0) {
    $mesg = "ユーザ ID かパスワードが違います。";
  } else if($a[0] == $userid && $a[1] == $_POST['pass']) {
    session_destroy(); session_start();
    $_SESSION['userid'] = $userid;
    header("Location: sam05c.php"); exit;
  } else {
    $mesg = "ユーザ ID かパスワードが違います。";
  }
} else {
  $mesg = "ユーザ ID とパスワードを入れてください。";
}
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { margin: 2px 20px; padding: 4px; background: rgb(200,215,255) }
</style></head><body>
<form method="post" action="#"><div class="test">
<p>ユーザ ID とパスワードを入力してログインしてください。</p>
 ユーザ ID:<input type="text" name="userid">
 パスワード:<input type="password" name="pass">
```

```
 <button name="cmd" value="login">ログイン</button><br>
<p>はじめての方は使用されるユーザ ID を選んでください。</p>
 新規ユーザ ID:<input type="text" name="newid">
 <button name="cmd" value="newuser">新規 ID 登録</button><br>
<?php if($mesg) echo "<p>{$mesg}</p>"; ?>
</div></form></body></html>
```

概要は次の通りです。

- 最初に session_start() する。また、この例題では「よそへ飛ぶ」という処理が何回もあるが、よそへ飛ぶためには「Locaiton:　行き先」というヘッダを出力する必要があるため、ほとんどの処理を冒頭 (HTML を出力するより前) で行う。もちろん、HTML を出力しない場合は Content-type: も出力してはいけない。
- 次の枝分かれは、(1) データベース接続不調、(2) 新規ユーザボタン、(3) ログインボタン、(4) それ以外 (初回) の処理、の 4 つの枝分かれになっている。
- メッセージはページ本体の中で出す必要があるので、変数 mesg に入れておいて後でこれを参照する。たとえばデータベース接続不調の場合は「接続できない」というメッセージを入れるだけ。
- 新規 ID の場合は、まずそのユーザ ID が表に登録されているか検索し、結果数が 0 なら新たなデータを挿入し (このときユーザ ID 以外の欄はとりあえず NULL にしてある)、ページ B へ移動する。0 でなければ、「もうある」というメッセージを出力する。なお、チェックしてから挿入までの間に誰かが同じ ID を挿入すると困るので、これらの処理は begin〜commit で囲んである。
- 通常ログインの場合は、そのユーザ ID のデータを検索して、データベース内のパスワードと入力されたパスワードの一致を確認する。一致していれば「セッションを新しくして」からページ C へ移動する (セキュリティ上必要)。一致していなければ違うというメッセージを出す。
- いずれでもない場合は最初にこのページへ来たので、単に「ID とパスワードを入れてください」というメッセージを出す。
- 以上で PHP 部分はほとんど終わりで、その先は HTML によりユーザ ID 欄、パスワード欄、新規 ID 欄、そしてログインボタン、新規登録ボタンを用意している。ただし、一番最後に変数 mesg の内容を表示しておく。

なお、このページからページ B/C に移動するときは、その前に$_SESSION['userid'] にこのユーザのユーザ ID を格納します (これはクッキーに直接入るわけではなく、PHP のセッション管理機構がサーバ内のみに保持します)。そして、他のページでこれを参照することで、「どのユーザの」処理かを識別できるわけです。

では次に、ページ B(図 3) のコードを示しましょう。

```
<?php
session_start();
$userid = $_SESSION['userid'];
if(!($conn = pg_connect("dbname=kuno"))) {
  $mesg = "データベース接続不能; 管理者に連絡を。";
} else {
  $res = pg_query($conn, "select * from ユーザ where id = '{$userid}'");
  if(pg_num_rows($res) == 0) { header("Location: sam05a.php"); exit; }
  $a = pg_fetch_row($res);
  $name = $a[2];
  $tel = $a[3];
  if($_POST['cmd'] == 'ok') {
```

図 3: ページ B — ユーザ情報更新ページ

```
  $mesg = "";
  $f = array('ユーザ ID','パスワード','名前','電話番号');
  foreach($a as $field => $value) {
    if($value == NULL) $mesg = $mesg . "「{$f[$field]}」が空です。<br>";
  }
  if($mesg == "") { header("Location: sam05c.php"); exit; }
} else if($_POST['cmd'] == 'update') {
  $res = pg_query($conn, "begin");
  $mesg = "";
  if($_POST['pass1'] != '') {
    if($_POST['pass1'] != $_POST['pass2']) {
      $mesg = $mesg."2 つのパスワード入力が一致しません。<br>";
    } else {
      $pass = pg_escape_string($_POST['pass1']);
      $res = pg_query($conn, "update ユーザ set pass='{$pass}' ".
        "where id='{$userid}'");
    }
  }
  if($_POST['name'] != '') {
    $name = pg_escape_string($_POST['name']);
    $res = pg_query($conn, "update ユーザ set name='{$name}' ".
      "where id='{$userid}'");
  }
  if($_POST['tel'] != '') {
    $tel = pg_escape_string($_POST['tel']);
    $res = pg_query($conn, "update ユーザ set tel='{$tel}' ".
      "where id='{$userid}'");
  }
  if(pg_query($conn, "commit")) {
    $mesg = $mesg."更新完了。さらに変更して「更新」;変更不要なら「完了」";
  } else {
    $mesg = $mesg."更新エラー;「更新」を再試行してください。";
```

```
    }
  } else { // 最初に入力ページを表示した時はここに来る
    $mesg = "フィールドに記入して「更新」; 変更不要なら「完了」";
  }
}
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { margin: 2px 20px; padding: 4px; background: rgb(200,215,255) }
</style></head><body>
<form method="post" action="#"><div class="test">
<?php
echo "<p>ユーザ「{$userid}」のデータ登録/更新</p>";
echo "パスワード:<input type='password' name='pass1'><br>";
echo "確認用入力:<input type='password' name='pass2'><br>";
echo "名前:<input type='text' name='name' value='{$name}'><br>";
echo "電話:<input type='text' name='tel' value='{$tel}'><br>";
echo "<p>{$mesg}</p>";
?>
<button name="cmd" value="update">更新</button>
<button name="cmd" value="ok">完了</button>
</div></form></body></html>
```

解説は次の通りです。

- データベースに接続できない場合は先の例と同様。
- 接続できる場合は、セッションで記録しているユーザ ID に対応する組を取得し、そこから name と tel を変数に取り出しておく (パスワードは画面表示しないので取り出さないでよい)。
- 続いてコマンドの種類で (1) 更新完了、(2) 更新、(3) 最初にこのページへ来た状態、のどれかに分岐する。
- (1) の場合は、まだ NULL の項目があったら完了ではないので、mesg に空文字列を入れ、ループで全フィールドについて NULL かどうかチェックし NULL なら警告を追加する。ループが終って mesg が空だったら NULL はなかったのでページ C へ飛ぶ。
- (2) の場合は、各フィールドごとに SQL の update を使って値を設定してゆく。一連の動作がまとめて完了したら OK なので、全体は begin 〜 commit で囲んでいる。
- (3) の場合は、単に説明メッセージを設定する。
- あとはほぼ通常の HTML だが、名前と電話の入力欄の初期値はデータベースから取得した値を入れておく (確認用)。

最後に、本体ページ (ページ C — 図 4) ですが、これは今回最初の例題の冒頭部分だけ違うので、そこだけ示します。を示します。

```
<?php
  session_start();
  if(!isset($_SESSION['userid'])) { header("Location: sam05a.php"); exit; }
  $userid = $_SESSION['userid'];
```

図 4: ページ C — 本体ページ

```
  $cmd = $_POST['cmd'];
  if($cmd == 'logout') {
    session_destroy(); header("Location: sam05a.php"); exit;
  } else if($cmd == 'update') {
    header("Location: sam05b.php"); exit;
  }
  header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { padding: 4px; background: rgb(255,215,200) }
td { margin: 2px 4px; padding: 4px; background: rgb(200,215,255) }
</style></head><body>
<form method="post" action="#"><div class="test">
<p>検索語:<input type='text' name='word'>
<button name="cmd" value="lookup">検索</button>
<button name="cmd" value="update">ユーザ情報更新</button>
<button name="cmd" value="logout">ログアウト</button>
</div></form>
<?php
  echo "<p>こんにちは、${userid}さん。</p>";
  if($_POST['word'] == '') {
    echo "<p>検索キーワードを入れてください。</p>";
(以下同じにつき略)
```

解説は次の通りです。

- ユーザ ID がなければページ A へ移動。
- コマンドが logout なら、セッションを破棄してページ A へ移動。
- コマンドが update なら、ページ B へ移動。
- 以下は先の例題と同じ…

演習 自分の設計した顧客情報データの登録/更新/ログインが行えるように、ページ群を PHP で実装せよ。

演習 現在はカート内容を一括チェックする手段がない。そのような機能を提供してみよ。

演習 注文処理 (カート内容を 1 つの注文としてデータベースに書き込みカートはクリア) を実現せよ。それらしく「注文画面」を表示するとかっこいい。

4 PHP のまとめ

PHP を使えば、ブラウザとデータベースの間に立ってデータをやりとりする仲介プログラム、つまりデータベースを用いた Web アプリケーションが比較的楽に書けることはお分かり頂けたかと思います。ただ、Web の場合は各ページが単独で動作するため、セッション管理その他の機能を駆使して「これまで何をやってきて今はどう動作するべきか」をうまくプログラムとして作り上げる必要があるので、そこにコツが必要だと言うことになるでしょうか。

5 宿題/レポート課題

さて、次回でいよいよ最終回となります。そのとき、各自が「自分のシステム」をできているところまで互いに紹介する時間を持ちたいと思います。紹介のための資料を適宜作成し、各自 6 部コピーしてきてください。動くものがあれば動くもののデモをしてもらうと楽しいと思いますが、なければこれまでのように紙を示して説明でよいです。

最終レポートはこの資料をより完成に近付けて、次の内容を含めてください。

- 作成した分析図 (IDEF0、DFD、IDEF1X など — 既に作ったはず)
- 作成した PHP コード (適宜)
- 作成した PHP コードが動いているときのブラウザ画面 (適宜)
- 上記内容に関する説明 (適宜)
- 全体的な考察と感想

紙のレポートとして、綴じて表紙をつけて、「4F 階段室の久野のボックスに」提出してください。期限は「12 月 6 日一杯まで」とさせて頂きます。

注文主である私が検収テストをするわけではないので、「できたところ」までで出して頂いて構いませんが、何も出ないと困るので各自の実力に応じてそれなりに頑張ってください。ではよろしく