A

Answers

Most of the chapters in this book provide an exercise (or more) to help you reinforce your new knowledge and/or hone your new skills. This appendix provides the solutions for the exercises by chapter. Here you'll find the full text of each exercise and its solution. Code solutions may be provided here or as a download from `www.wrox.com`. Of course, the code provided here also is available at that site.

Many of the exercises are flexible; meaning that there may be several workable solutions, so you may arrive at a different solution than the one shown here. The solution provided is designed to be generic, and in some cases the code has been written for readability, not necessarily to show the fastest or most abstracted method for providing the result. And some of the exercises, especially in the earlier chapters, may not involve code at all, or may include parts that require writing pseudo code or planning-type documents. The exercises are important because they encourage familiarity with the process by which programming projects are actually done, something you'll definitely want to know.

Chapter 1

Exercise

You'll find this exercise helpful whenever you have to install with or work with PHP on a new platform. To complete this exercise, do the following:

Create a document that summarizes all of the following:

- What are the hardware capabilities of the computer on which PHP is running? Describe the CPU, hard drive, RAM, and so on, and any particular limitations you perceive.
- What operating system is running on the hardware? List the version and any current patches and known bugs.
- What Web server software is running on the machine? List the version, patches, and known bugs. Also, list how the Web server is configured, the root folder, how PHP is set up to work with the server, and any file permissions you've set.
- What version of PHP was installed? List the version, the files installed, the folders in which they were installed, and any Registry settings that were set or created to support the PHP installation.

- What configuration settings were set or changed (from the default) to install PHP? List them.
- What extensions were enabled? List them all, and why you enabled them.

Having a document such as this makes it much easier for the next programmer to understand why PHP programs behave in a particular way, as well as make it easier for you to find all the information you need in one place later, rather than having to examine everything again when you get stuck.

Solution

Create a document like the following for your own system:

Documentation—Server 01	
CPU – Hard drive – RAM	Pentium 4, 1.7 Ghz – 120GB – 512MB
Operating System	Red Hat Linux Fedora
Web Server Software	Apache 2.0
Scripting Software	PHP 5.0
Php configuration settings	register_globals = off, default development settings

Chapter 2

Exercise 1

Create a PHP program that rearranges sentence one into sentence two, and outputs what it is doing (and the result) to the user. The two sentences are:

"now is the time for all good men to come to the aid of their country"
"the time is now to come to the aid of good men in the country"

Solution

Write a PHP script such as this:

```
<?php
$my_array = explode(" ","now is the time for all good men to come to the aid of
their country");
echo $my_array[2] . $my_array[3] . $my_array[1] . $my_array[0] . $my_array[8] .
$my_array[9] . $my_array[8] . $my_array[2] . $my_array[12] . $my_array[13] .
$my_array[6] . $my_array[7] . "in" . $my_array[2] . $my_array[15];
?>
```

Exercise 2

Create a PHP program that creates two arrays of numbers and adds the values in each array to their corresponding values by index number. The two arrays should have the following values:

2,4,6,8,10
3,5,7,9,11

Solution

Write a PHP script such as this:

```
<?
$arr01 = array(2,4,6,8,10);
$arr02 = array(3,5,7,9,11);

for ($counter = 0; $counter <= 4; $counter ++){
   $arr03[] = $arr01[$counter] + $arr02[$counter];
   echo "The answer is now " . "$arr03[$counter]" . "<br>";
}
?>
```

Chapter 3

Exercise

PHP contains a very useful function named `isset()`, which tells your PHP program whether a particular variable has been set. For example, suppose you have a page with a form containing a submit button named `login` and another one named `logout`. When your user submits the form, you can tell which button he clicked by using the `isset()` function, like this:

```
if (isset($login)) {
//do this
} elseif (isset($logout)) {
//do this
}
```

For this exercise, create a Web page containing a form that submits to itself, and make your PHP program using the `isset()` function to tell when a form submission has occurred. If a form submission has not occurred, make your program display a form asking for the user's first and last name (but not the answer), and if a form submission has occurred, make your program not display the form but instead display the answer (a short statement such as: "Your first name is XX and your last name is XX.").

Hints: Use a hidden form field to determine if the form has been submitted, and use the $PHP_SELF variable to make the form submit back to itself.

Solution

```
<html><head><title>My Title</title></head>
<body>
<?php
if (isset($_POST['posted'])) {
   echo "Thanks. Your first name is $_POST[first_name] and your last name is
$_POST[last_name]";
} else {
?>
<h2>Please fill out this form</h2>
<form method="POST" action="<?php echo "$_SERVER[PHP_SELF]"; ?>">
<input type="hidden" name="posted" value="true">
```

```
First Name<input type="text" name="first_name"><br>
Last Name<input type="text" name="last_name">
<input type="submit" value="Send Info">
</form>
<?php
}
?>
</body>
</html>
```

The full code for the solution is a file named `ch03ex01.php`.

Chapter 4

Exercise

"We have a Web site, and it's outdated. It doesn't look very nice, but we're having a graphic artist redo the logo, so we'll probably be getting a lot more traffic once it's done. We'll be hiring more people to keep up with demand, and we want to have a way to gather resumes online.

"It should be easy for people to find the job listings, but the link shouldn't be the biggest thing on the page. We'll want to have their contact information, of course, and a search for jobs. If they don't have a college degree, they might as well not apply unless they're looking for entry-level jobs in the sales department or the shipping department. None of the jobs pays more than $20,000 to start, but management positions do offer incentive bonuses. They should send their salary requirements and at least 2 years of work experience, except for management positions (management positions must have at least 5 years experience, unless they have a Ph.D.). We'd like people to be able to search for jobs and apply for the ones they think would be a good fit for them, and also to be able to submit their resume and find out what jobs they qualify for."

Whew! Well, that's often exactly how the requirements for an application are first presented: someone who probably doesn't know much about what programmers do asks you to devise a program to do a particular function.

This exercise will definitely give you some practice in how to go from the statement of the problem all the way to the finished product. Your job is to decipher what's been said, use the capabilities of PHP to perform the processing, and collect the data and respond to the user with Web pages. To complete this exercise, you should include the following in addition to your finished program:

- A list of all required information, in addition to what has been presented.
- A description of the screens that you'll make, why you'll make them (why they are needed), and how the user will interact with them.
- A short description of how you'll integrate the screens with the existing Web site.

There's no one single PHP programming solution for this exercise, although all solutions will perform similar processing and use similar steps. Because databases or file system haven't been discussed yet, there's no way for you to store the resumes long-term, and the resume information will be lost once processing is complete.

You've got your assignment. Go on, get started!

Hint: To build a solution, try to lay out the problem as a series of screens first, and ask yourself what you'd expect to see. The, review the PHP capabilities that have been discussed so far to see which ones might be able to provide the data, and exactly what data must be present to complete the next step. It's much easier to arrive at a workable solution by breaking down the execution of the solution into small steps.

Solution

First, write a list of the information required. Essentially, the client wants to collect resumes. He doesn't seem to be concerned about making users log in, so the following basic categories of information are required:

1. Contact information, such as name, address, phone numbers, and so on. For many jobs, social security number will also be required.
2. Work experience
3. Education
4. Special qualifying information, such as degrees, certificates, and so on
5. Miscellaneous information

The requirements also describe the capability for people to search for jobs, and that implies that there will be a list of jobs internal to the system, maintained by a person within the client's company, and that person will enter search terms (keywords) that apply to each job. You should also ask whether the client wants jobs to be searchable by salary range.

There are a few conditions that may (or may not) apply to the job search and resume submission. One is that people without a degree should not even be able to apply for jobs except for entry-level jobs in sales or shipping. This prompts the question of whether they should even be able to begin their job search before entering some of their qualifications (why display jobs for which they can't qualify?).

Another condition is that they must have at least two years experience for all jobs except management positions, for which they may have 5 years' experience or a Ph.D.

The mention of salary does not seem to have a programming implication, other than to notify prospective employees that this company does not pay very well, although managers could make bonuses.

The next step is to begin devising the process by which users will interact with the system. Altogether, it would seem that a good process would be to first display a request for some basic information to prequalify job hunters, and then allow them to search for jobs or simply submit their resume and then view a list of any jobs for which they qualify.

You could do this by displaying a welcome screen asking for basic prequalification information, and then a screen providing a choice of searching for jobs or submitting a resume. When users reach the point of applying online, the resume screen could be broken into sections so they don't end up submitting all their data in one giant form on one screen.

As each form is filled out and submitted, the application should have a means of storing the information across page requests, and filling in the fields again in case the user wants to go back and edit.

So the pages you might have would be:

1. Welcome screen, with a form for entering pre-qualifying information, and a link to the Search/Resume screen.
2. The Search/Resume screen, with a form for submitting resume information (broken into logical chunks), and a search text field with search button.

The programming logic in PHP would then:

- Display the welcome screen.
- Request the user enter some prequalifying information, such as years of experience, level of education, and anything else that might be pertinent (most jobs have age and residency requirements, and that sort of thing, and you may have to ask a few questions to get these from the client).
- Store the prequalifying information, unless they don't qualify for any jobs at all (in which case this would be a good point to tell them that). Because you're handicapped by lack of any long-term storage device (which would not be the case in a real application), you'll have to store the data in hidden form fields.
- Depending upon which qualification requirements the user meets, display the appropriate resume submission form and the search form.
- As each part of the resume is submitted, store the data in hidden form fields, and provide the user a navigation link so previously entered data could be called up and revised as desired.
- Provide a search function that can display a list of jobs, with links, based on search terms entered by the user. This function would depend on pulling up the list of jobs from a database or file of some kind, but you can assume that the information is available in an array, so all you'd need to do is search the array using a `for` or `foreach` loop, and identify a keyword in the array by using a PHP function such as `substr()` to match up search terms entered with keywords found.
- Finally, if the user successfully enters a resume and applies for a particular job, the application would display a thank-you screen.

A relatively simple application with the process described here could be made all as one file using plain HTML forms and Web pages, a `switch` statement for each case when a form is submitted, and hidden form fields to store data across page requests.

A full solution, `ch04ex01.php`, is available at this book's Web site.

Chapter 5

Exercise

Create at least three regular expressions and insert them into the appropriate spots in the most recent form validation example. Suggested regular expressions are:

- 7– and 10–digit U.S. phone numbers
- Social Security numbers
- Gender using M and F for male and female

Solution

There are a number of ways to match telephone numbers, and the Regular Expression Library provides several examples that match U.S. phone numbers. The simplest is to look only for 7 or 10 digits in a row, excluding alphabetic characters. So you start by matching the beginning of the line with the carat, followed by one character from 2 to 9, two characters from 0 to 9, a dash, then one character from 2 to 9 and two characters from 0 to 9, a dash, and finally four characters from 0 to 9. That matches a 10-character phone number with dashes in between the segments, so you put in the "or" symbol (|) and repeat the second two portions of the first pattern so that you can also match a 7-character phone number. Then end the whole thing with the dollar sign ($) to match the end of the line.

```
^([2-9][0-9]{2}-[2-9][0-9]{2}-[0-9]{4}|[2-9][0-9]{2}-[0-9]{4})$
```

For Social Security numbers, you specify exactly three characters from 0 to 9, a dash, exactly two characters from 0 to 9, a dash, and exactly four characters from 0 to 9. The simplest expression is:

```
^[0-9]{3}-[0-9]{2}-[0-9]{4}$
```

Although this expression correctly evaluates the digit format for SSNs, it also accepts all zeros or 666, neither of which is found in a valid SSN. For extra credit, find a way to exclude character combinations with all zeros or 666.

For male and female gender, character classes may be used:

```
^([M]|[F])$
```

This `regexp` uses the carat and dollar sign to match the beginning and end of the line, and character cases for uppercase M or (using the "or" symbol |) uppercase F.

The full code for this solution is a file named `ch05ex01.php`.

Chapter 6

Exercise

Write a function that enables the user to construct his own Web form, with the capability to choose field names and field types, and then allow the users to submit the form to the page and see what they submitted echoed back to them. Use a `switch..case` statement to select various functions within your main function.

Solution

This solution involves writing HTML tags with PHP. You might start by constructing a Web form that lets users enter a predetermined number of field names and field types, and then define a number of variables that contain the HTML for those field names and types. When the user submits the form, another form will be generated based on the data entered by the user. The `switch..case` statement could be placed inside the function that generates the completed new form, selecting the HTML to produce the field name and type within a predefined HTML form.

There should also be a means of asking the user to provide extra information when a drop-down list (the `<select>` tag) has been chosen because these tags operate by means of multiple `<option>` tags inside them.

Here's an example of creating a function to write HTML tags:

```
//create the function to write the HTML tags
function createTags($field_name,$field_type)
{
   global $option_text01,$option_text02,$option_text03;
   global $option_value01,$option_value02,$option_value03;
   switch ($field_type) {
      case "text";
         $next_field = "<tr><td>$field_name:</td><td><input type='$field_type'
name='$field_name'></td></tr>";
         break;
      case "radio";
         $next_field = "<tr><td>$field_name:</td><td><input type='$field_type'
name='$field_name'></td></tr>";
         break;
      case "checkbox";
         $next_field = "<tr><td>$field_name:</td><td><input type='$field_type'
name='$field_name'></td></tr>";
         break;
      case "hidden";
         $next_field = "<tr><td>$field_name:</td><td><input type='$field_type'
name='$field_name'></td></tr>";
         break;
      case "textarea";
         $next_field = "<tr><td>$field_name:</td><td><textarea cols='40'
name='$field_name'></textarea></td></tr>";
         break;
      case "select";
         $next_field = "<tr><td>$field_name:</td><td>
<select name='$field_name'>"
            ."<option value='$option_value01'>$option_text01</option>"
            ."<option value='$option_value02'>$option_text02</option>"
            ."<option value='$option_value03'>$option_text03</option>"
            ."</select></td></tr>";
         break;
      default;
         break;
   }
   return $next_field;
}
```

For the full solution, see the code file `ch06ex01.php`.

Chapter 7

Exercise

Create a PHP application that can be used to find a particular directory when given a valid parent directory to search. Make the application look through the given directory and all directories under the given directory.

Solution

This application can make use of the PHP file system functions to search for the directory and to simply iterate through all directories and subdirectories given a valid parent directory. The first step might be to list all the directories in the parent directory, go through all of those directories identifying any subdirectories that have the name you're looking for, and then to go through all of them, and so on. Here's some code that finds a folder by name within the default folder.

```
//get the folder to search for
$folder_to_find = "$_POST[folder]";
//set the default directory
$default_dir = "C:";
//create the function to find the folder if it exists
function find_folder($default_dir,$folder_to_find) {
   if (!($dp = opendir($default_dir))) {
      die("Cannot open $default_dir.");
   } else {
      while ($file = readdir($dp)) {
         if ($file == $folder_to_find){
            return ($file);
         }
      }
   closedir($dp);
   }
}
  $folder = find_folder($default_dir,$folder_to_find);
if ($folder != "") {
   echo "Found folder named " . $folder;
} else {
   echo "Folder not found.";
}
```

This code finds a folder in the default directory by name, but does not search subfolders. For the full solution, see the code file `ch07ex01.php`.

Chapter 8

Exercise

Create an application that reads an XML file with multiple elements and attributes into a simpleXML object, changes the values of some or all of its elements or attributes, and then returns the modified document back into the file.

Solution

You've already seen how to change values in XML files, and this solution is an extension of that. Here's an example of the code that can change element values based on their array index number:

```
//capture which element to change
$element_to_change = $_POST[element_to_change];
$change_value = $_POST[change_value];

if ($element_to_change == 0) {
```

```
    $first_xml_string->program[0]->price = $change_value;
} elseif ($element_to_change == 1) {
    $first_xml_string->program[1]->price = $change_value;
}
```

Naturally, the drawback to this code is that the element index numbers must be hard-coded in, and it doesn't show how to save the resulting XML document back into a file. For the full solution, including the ability to make changes to attribute values, see the code file `ch08ex01.php`.

Chapter 11

Exercise 1

Make a list of suitable field descriptors for each of the columns in this table. *Hint: Remember that restaurants generally have many individual orders in any given day.*

First name	Order _ID	User _ID	Resturaunt	Password	Last name	Total Price ($)	Email
David	2	1	Nandos	12345	Mercer	21.45	davidm@contechst .com
David	4	1	Mimos	12345	Mercer	20.95	davidm@contechst .com
Nic	3	2	St Elmos	23212	Malan	15.45	therot@doggies touch.co.za
Brian	5	4	Spur	32123	Reid	22.00	pads@doggiestouch .co.za
Darren	1	3	Home	43212	Ebbs	11.85	Bacardi@doggies touch.co.za

Solution

Here's a list of suitable field descriptors:

```
VARCHAR(30)
MEDIUMINT(7)
MEDIUMINT(7)
VARCHAR(30)
VARCHAR(20)
VARCHAR(50)
FLOAT(7)
VARCHAR(60)
```

Exercise 2

The site manager decides to have a page for users to view their past orders and their totals. Using principles of normalization, create new tables to represent one entity each. Give each table a suitable name, and a primary key. (Check that your changes would reduce redundancy in large amounts of data.)

Solution

Here's the Customer table:

User_ID	Firstname	Lastname	Password	Email
1	David	Mercer	12345	davidm@contechst.com
2	Nic	Malan	23212	therot@doggiestouch.co.za
3	Darren	Ebbs	43212	Bacardi@doggiestouch.co.za
4	Brian	Reid	32123	pads@doggiestouch.co.za

Here's the Orders table:

Order_ID	User_ID	Resturaunt	TotalPrice
1	3	Home	11.85
2	1	Nandos	21.45
3	2	St Elmos	15.45
4	1	Mimos	20.95
5	4	Spur	22.00

Exercise 3

Create these tables in your database using SQL statements.

Solution

The following SQL statements will create the Customer table in your database:

```
CREATE TABLE Customer (
User_ID MEDIUMINT(7) NOT NULL AUTO_INCREMENT,
Firstname VARCHAR(30) BINARY NOT NULL,
Lastname VARCHAR(50) BINARY NOT NULL,
Password VARCHAR(20) BINARY NOT NULL,
Email VARCHAR(60),
PRIMARY KEY (User_ID),
UNIQUE (Email)
);
```

These SQL statements will create the Orders table in your database:

```
CREATE TABLE Orders (
Order_ID MEDIUMINT (7) NOT NULL AUTO_INCREMENT,
User_ID MEDIUMINT (7) NOT NULL,
Resturaunt VARCHAR (30) NOT NULL,
Total_Price FLOAT (7) NOT NULL,
PRIMARY KEY (Order_ID)
);
```

Exercise 4

After the Web site's been in business for six months, customers begin to complain that the site is going slowly when they want to view their previous orders. The site administrator looks at the Orders table and finds that there are now thousands of records. Why is the site slow, and how can the administrator speed things up? Update the table accordingly.

Solution

The site is slow because there are now a large number of records in the Orders table. The lookup is being performed using the `User_ID` field because that is how you identify which records to return to a given user. To prevent full table accesses while performing these look ups, add an index to the `User_ID` field. (Remember, that while this will speed up `SELECT` statements, it slows down `INSERT` and `UPDATE` statements. In this case, it's probably worth your while to make the site faster for customers.) The administrator can speed things up by issuing the following query:

```
ALTER TABLE Orders ADD INDEX (User_ID);
```

Exercise 5

The boss decides he wants to see how popular Nandos is with his customers. What query could the administrator build to obtain the totals for all the orders (a) made by one customer and (b) made to Nandos. *Hint: Try `Select SUM(column) from table`.*

Solution

(a):

```
SELECT SUM(Total_price) FROM Orders WHERE User_ID = 1;
```

(b):

```
SELECT SUM(Total_price) FROM Orders WHERE Resturaunt = 'Nandos';
```

Exercise 6

Create a script that enables a user to enter his details into the database by registering at the site. The database should be set up so that the user ID is assigned automatically. Once the user is registered he should be able to place an order with any of the five restaurants. (Just complete one to begin with. Don't worry about creating a full interface—being able to enter a price is the important thing.) Clicking an order button should create a new entry in the Orders table, with an incremented `Order_ID`, and the correct `User_ID` inserted for the customer. *Hint: You may want to use a session to store the users `User_ID` upon registration.*

Solution

`common_db.inc`:

```
<?php
$dbhost = 'localhost';
$dbusername = 'phpuser';
$dbuserpassword = 'phppass';
```

```
$default_dbname = 'sample_db';
$max_menu_items = 10;
$MYSQL_ERRNO = '';
$MYSQL_ERROR = '';

function db_connect($dbname='') {
   global $dbhost, $dbusername, $dbuserpassword, $default_dbname;
   global $MYSQL_ERRNO, $MYSQL_ERROR;

   $link_id = mysql_connect($dbhost, $dbusername, $dbuserpassword);
   if(!$link_id) {
      $MYSQL_ERRNO = 0;
      $MYSQL_ERROR = "Connection failed to the host $dbhost.";
      return 0;
   }
   else if(empty($dbname) && !mysql_select_db($default_dbname)) {
      $MYSQL_ERRNO = mysql_errno();
      $MYSQL_ERROR = mysql_error();
      return 0;
   }
   else if(!empty($dbname) && !mysql_select_db($dbname)) {
      $MYSQL_ERRNO = mysql_errno();
      $MYSQL_ERROR = mysql_error();
      return 0;
   }
   else return $link_id;
}

function sql_error() {
   global $MYSQL_ERRNO, $MYSQL_ERROR;
   if(empty($MYSQL_ERROR)) {
      $MYSQL_ERRNO = mysql_errno();
      $MYSQL_ERROR = mysql_error();
   }
   return "$MYSQL_ERRNO: $MYSQL_ERROR";
}
?>
```

`Takeaway.php`:

```
<?php
include_once "./common_db.inc";

function login_form() {
global $PHP_SELF;
?>
<HTML>
<HEAD>
<TITLE>Login</TITLE>
</HEAD>
<BODY>
<INPUT TYPE="HIDDEN" NAME="action" VALUE="register">
```

```
    <DIV ALIGN="CENTER"><CENTER>
       <H2>Welcome to Speedy Deliveries!</H2>
       <H3>Please take the time to register your details.</H3>
    <TABLE BORDER="1" WIDTH="400" CELLPADDING="2">
       <TR>
          <TH WIDTH="25%" ALIGN="RIGHT" NOWRAP>Firstname</TH>
          <TD WIDTH="82%" NOWRAP>
             <INPUT TYPE="Input" NAME="userfirstname" SIZE="30">
          </TD>
       </TR>
            <TR>
          <TH WIDTH="25%" ALIGN="RIGHT" NOWRAP>Lastname</TH>
          <TD WIDTH="82%" NOWRAP>
             <INPUT TYPE="Input" NAME="userlastname" SIZE="30">
          </TD>
       </TR>
       <TR>
          <TH WIDTH="25%" ALIGN="RIGHT" NOWRAP>Password</TH>
          <TD WIDTH="82%" NOWRAP>
             <INPUT TYPE="PASSWORD" NAME="userpassword" SIZE="30">
          </TD>
       </TR>
       <TR>
          <TH WIDTH="25%" ALIGN="RIGHT" NOWRAP>Email</TH>
          <TD WIDTH="82%" NOWRAP>
             <INPUT TYPE="Input" NAME="useremail" SIZE="30">
          </TD>
       </TR>
       <TR>
          <TD WIDTH="100%" COLSPAN="2" ALIGN="CENTER" NOWRAP>
             <INPUT TYPE="SUBMIT" NAME="Submit">
          </TD>
       </TR>
    </TABLE>
    </CENTER></DIV>
</FORM>
</BODY>
</HTML>
<?
}

function register_user() {
   global $default_dbname;
   $PHP_SELF = $_SERVER['PHP_SELF'];

   $link_id = db_connect($default_dbname);
    $query = "INSERT INTO Customer
       VALUES('','$_GET[userfirstname]','$_GET[userlastname]',
       $_GET[userpassword],
       '$_GET[useremail]')";
     $result = mysql_query($query);
   if(!$result) {
      Echo "Please go to the Home page and register.";
```

```
      ?>
   <FORM method="GET" action="<?php echo $PHP_SELF ?>">
     <INPUT type="submit" value="Home">
   </FORM>
<?php
  exit;
     }
  return $result;
}

function get_userid(){
     global $default_dbname;
   $link_id = db_connect($default_dbname);

   $query = "SELECT User_ID from Customer WHERE Password =
   '$_GET[userpassword]'";

   $result = mysql_query($query);
   $result_array = mysql_fetch_row($result);
   $userid = $result_array[0];
   if(!$userid) $userid = "";
   return $userid;
}
function order_menu(){
   $PHP_SELF = $_SERVER['PHP_SELF'];

?>
<HTML>
  <HEAD><TITLE>Resturaunts</TITLE></HEAD>
    <BODY>
      <DIV ALIGN="CENTER"><CENTER>
      <H2>Welcome to Nandos, <?php echo $_GET['userfirstname']; ?>!</H2>
       </CENTER></DIV>
    <FORM method="GET" action="<?php echo $PHP_SELF ?>">
        <INPUT TYPE="HIDDEN" NAME="action" VALUE="order">
Winglets Starter: $1.50
        <INPUT name="Choice1" type="checkbox" value="1.50">
          <BR>
Chicken Burger:   $3.00
        <INPUT name="Choice2" type="checkbox" value="3.00">
          <BR>
Softdrink:        $0.75
        <INPUT name="Choice3" type="checkbox" value="0.75">
          <BR>
          <BR>
        <INPUT type="submit" value="Order My Food">
    </FORM>
  </BODY>
</HTML>

<?php
}

function place_order(){
```

```
    global $default_dbname, $max_menu_items;
    $PHP_SELF = $_SERVER['PHP_SELF'];
    $link_id = db_connect($default_dbname);
    $total = 0;
    for ($counter = 1; $counter <= 10; $counter ++)
    {
        $a = "Choice" . "$counter";
        if (isset($_GET[$a])){
        $total += $_GET[$a];
    }

}

$query = "INSERT INTO Orders VALUES ('', '$_SESSION[userid]', 'Nandos',
$total)";

$result = mysql_query($query);
if(!$result) die ("Could not place your order, please try again!");
  else {
     //On a live system, this should take you to a confirmation page, as
        simply refreshing the page as it is will cause the order to be
        duplicated.
     echo "Your order has been taken! You can expect delivery within the hour.
           The total came to \$$total";
?>
   <FORM method="GET" action="<?php echo $PHP_SELF ?>">
     <INPUT type="submit" value="Home">
   </FORM>
 <?php
  }
}

function customer_session(){
     $_SESSION['useremail'] = $_GET['useremail'];
     register_user();
     $userid = get_userid();
     $_SESSION['userid'] = $userid;
}

session_start();
session_register("userid", "useremail");
if (empty($_GET['action'])){
      $_GET['action'] = "";
   }

switch($_GET['action']) {
   case "order":
      place_order();
      break;
   case "register":
        customer_session();
      order_menu();
      break;
   default:
```

```
        login_form();
        break;
}

?>
```

Chapter 12

Exercise 1

Define the difference between a class and an object.

Solution

A class is like a blueprint. It only provides the instructions for creating an object. An object is an instance of a class. Classes are manipulated at design time when you alter the code in a PHP file. Objects are manipulated at runtime when properties are accessed or changed and methods are invoked. A class is represented by text in a text file. An object exists as instructions in memory.

Exercise 2

Explain inheritance and give an example of when it should be used. Don't repeat any of the examples already provided in this chapter!

Solution

Inheritance is the capability of an object to acquire properties or methods of a parent object. Inheritance implies a related hierarchy and reuse of code. The child class is a more specialized version of the parent class, having additional methods and properties and/or different implementations of the same methods and properties. A child cannot have fewer methods or properties than its parent. An example of inheritance could be the relationship between Vertebrates and Mammals. All vertebrates have a spinal column composed of some number of segmented bony vertebrae surrounding a spinal cord, and a large brain encased in a skull. Mammals are a more specialized type of vertebrate. Although they have all of the properties of vertebrates, they also have hair or fur, are warm blooded, and lactate. A class `Vertebrate` might have properties such as `numberOfVertebrae` and `skeletonType` (values for the latter being `cartilaginous` as in sharks, or `bony` as with mammals and reptiles). Mammals would inherit all those properties and add `furColor` and `normalBodyTemperature`.

Exercise 3

Describe the utility of an interface and its practical application in a software architecture. How is an interface different from an inherited class? Can you come up with additional examples of when this might be useful?

Solution

An interface is an abstract construct that defines a "contract"—a mandate requiring implementing classes to have certain methods. Although an inherited class implies that there's a direct relationship between the parent class and the inheriting subclasses, an interface implies no such relationship, but is used to identify classes that are capable of performing the same functions. Because all classes that implement the interface must, by definition, have at least the set of methods defined in the interface (they may have additional methods, but not fewer) you can use an interface to pass otherwise unrelated objects to the

same functions. For example, PHP comes with two interfaces—`Iterator` and `IteratorAggregate`—that enable you to pass objects into a `foreach` construct if and only if those objects implement one of these two interfaces. These interfaces allow objects that contain other objects to be looped through using `foreach`. You can have any class implement these interfaces, so you can use totally unrelated objects in the same `foreach` construct.

Chapter 15

Exercise

The application you created in this chapter works well for composing and sending individual e-mails. For this exercise, modify the application so that it sends the e-mail to a list of e-mail addresses retrieved from a database.

Solution

The first step in creating an application that sends e-mails to a list of e-mail addresses is to create a list of e-mail addresses in an array. Typically, when records are retrieved from a database such as MySQL or Postgres, the retrieval process begins by executing some type of SQL `SELECT` statement using the `pg_exec()` function or the `mysql_query()` function. The retrieved records are stored in a PHP variable (often named `$res` for result). The number of records in such an array can then be determined using the `pg_numrows()` or the `mysql_numrows()` function, and a `for` loop provides the means to retrieve individual rows (records) from the `$res` variable using the `pg_fetch_array()` or the `mysql_fetch_array()` function.

So you might run a `SELECT` query against a database holding customer information (including e-mail address), and then if there are multiple records in the result, use a `for` loop to fetch each e-mail address and insert it into a function such as the following to send it out (with a very little bit of e-mail address validation thrown in for good measure):

```
$query = "SELECT email FROM customers;";
$res = pg_exec($conn,$query);
if(pg_numrows($res) >= 1) {
   for ($i=0; $i<=pg_numrows($res)-1; $i++) {
      $rec = pg_fetch_array($res,$i);
      if ($rec[email] !== "") {
         if (strpos($rec[email],"@") >= 0) {
            $body = "This is the body"\n\n";
            //Set the From:
            $headers .= "From: Example <info@example.com>\n";
            //Send the mail
            $to = "$rec[email]";
            $subject = "Example Email";
            mail($to, $subject, $body, $headers);
         } else {
            echo = "The $i email address is no good.<br>";
         }
      }
   }
}
```

The full code for the solution is `ch15ex01.php`.

Chapter 16

Exercise 1

Create a PHP script that opens an image file and adds a one-pixel black border to the image.

Solution

The only trick to successfully complete this exercise is that the top-left corner of an image is x=0, y=0. This means that you must remember to subtract 1 from the width and height of the image to get the rightmost position and bottommost positions of the image, respectively. Here's the code (`exercise1.php`) to achieve Exercise 1:

```
<?php
$myImage = imagecreatefromjpeg('cape_point.jpg');
$black = imagecolorallocate($myImage,0,0,0);
$width = imagesx($myImage);
$height = imagesy($myImage);
imagerectangle($myImage, 0, 0, $width-1, $height-1, $black);
header("Content-type: image/jpeg");
imagejpeg($myImage);
imagedestroy($myImage);
?>
```

If your script was similar to this one, your output should include a nice clean border around the image, like the one shown in Figure A-1.

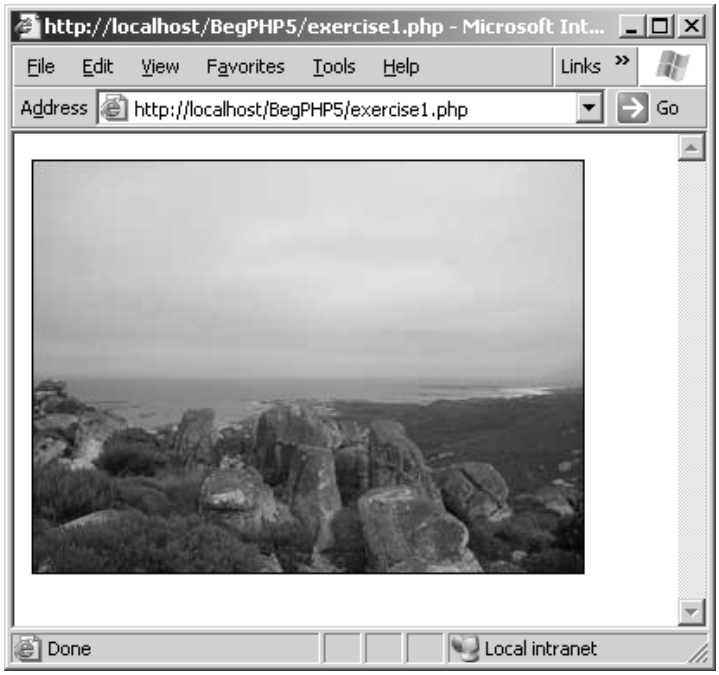

Figure A-1

Exercise 2

Using the `disk_total_space()` and `disk_free_space()` functions show how much disk space you have used on your hard drive in a graphical way.

Solution

There are two obvious ways that you can graphically display the results for of this exercise. The quickest and easiest way is to create a simple sliding scale: a rectangular shape that is made up of two blocks of color, one color representing the used space on the disk, and the other representing the free space. This gives you a nice quick overview of your disk usage. Following is the code (`exercise2a.php`) for doing this.

It's always good practice to store the width and height of your images in variables within the script. Then if you want to change the size of the image you can do so easily at the beginning of the script and any calculations you do with the image width and height are automatically updated.

```
<?php
$iWidth = 500;
$iHeight = 50;
```

Create the image and allocate black and white. Because white is the first color allocated, it will be the background color of the image. Black will be used to add a border to the image.

```
$myImage = imagecreate($iWidth, $iHeight);
$white = imagecolorallocate($myImage,255,255,255);
$black = imagecolorallocate($myImage,0,0,0);
```

In this example solution, red and green are used to represent used disk space and free disk space respectively. You can use any colors you like.

```
$red = imagecolorallocate($myImage, 255,0,0);
$green = imagecolorallocate($myImage,0,255,0);
```

Get the total amount of space on the disk and the amount of space that you have free. Both of these values are returned in bytes. The actual values are irrelevant—all you need them for is to get a proportion so that you can draw the bar.

```
$diskTotal = disk_total_space('/');
$diskFree = disk_free_space('/');
```

Then draw a one-pixel black border around the outside of the image.

```
imagerectangle($myImage, 0, 0, $iWidth - 1, $iHeight - 1, $black);
```

The `$threshold` variable will be used to mark the position along the x axis where you move from the used disk space to the free disk space in your diagram. Because used disk space is usually represented on the left side of such diagrams, you first need to calculate that space—it is the total disk space minus the free disk space. Divide that by the total amount of disk space to get a number between 0 and 1, which you then multiply by the width of the image. Remove 2 from the width of the image because you already used 2 pixels drawing the image border. And then add 1 because pixels start at 0.

```
$threshold = intval((($diskTotal - $diskFree) / $diskTotal) * ($iWidth-2))
+ 1;
```

For example, if your disk was 400 bytes in size and you had used 200 bytes, 200 divided by 400 equals 0.5. If your image width were 500 pixels, then you'd multiply the 0.5 and 498 (width minus 2 for the border) to get 249. Then you'd add 1 to get 250 the x pixel position that is halfway across the image.

Fill the image with a red rectangle that goes to the point of your threshold to represent the used disk space:

```
imagefilledrectangle($myImage, 1, 1, $threshold, ($iHeight-2), $red);
```

And a green rectangle from the threshold onward for the free disk space:

```
imagefilledrectangle($myImage, ($threshold + 1), 1, ($iWidth - 2),
$iHeight-2, $green);
```

Close it out in the usual manner.

```
header("Content-type: image/png");
imagepng($myImage);
imagedestroy($myImage);
?>
```

Figure A-2 shows the result of running this example on my machine.

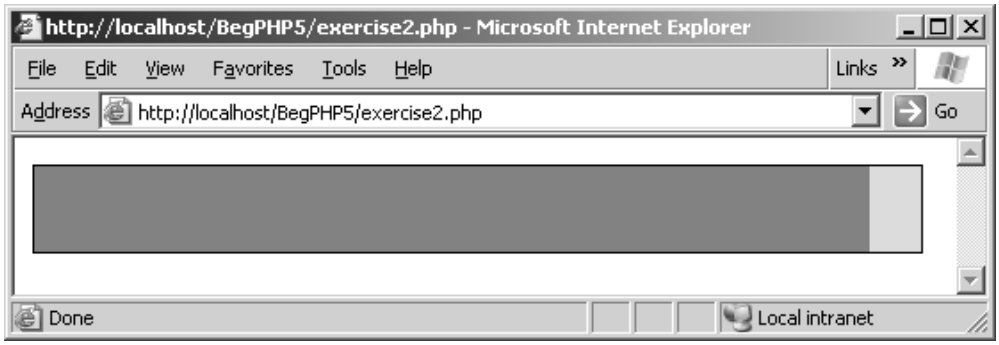

Figure A-2

The alternative solution to this exercise is to draw the space as a pie chart. The file `exercise2b.php` contains the code for this solution. Here's how it's done.

Instead of working out a threshold, you work out a number of degrees. The calculation works in the same way except you don't multiply by the width of the image, you multiply by 360—the number of degrees in a circle.

```
<?php
$iWidth = 200;
$iHeight = 200;
$myImage = imagecreate($iWidth, $iHeight);
```

```
$white = imagecolorallocate($myImage,255,255,255);
$red = imagecolorallocate($myImage, 255,0,0);
$green = imagecolorallocate($myImage,0,255,0);
$diskTotal = disk_total_space('/');
$diskFree = disk_free_space('/');
$usedDegrees = intval((($diskTotal - $diskFree) / $diskTotal) * 360);
```

The `imagefilledarc()` function works in the same way as the `imagearc()` function, except that it takes an additional argument specifying how that arc should be filled. The PHP constant `IMG_ARC_EDGED` causes PHP to connect the two end points of the arc to the center point of the arc and fill it with the color specified. You start at 0 degrees and draw the arc through to the degree that you worked out for `$usedDegrees`.

```
imagefilledarc($myImage, ($iWidth/2), ($iHeight/2), $iWidth - 2,
$iHeight - 2, 0, $usedDegrees, $red, IMG_ARC_EDGED);
```

To draw the arc that represents the free space, you simply start where the used space left off at `$usedDegree` and draw the arc through to 360 degrees, the end of the circle:

```
imagefilledarc($myImage, ($iWidth/2), ($iHeight/2), $iWidth - 2, $iHeight - 2,
$usedDegrees, 360, $green, IMG_ARC_EDGED);
```

And then finish up as usual:

```
header("Content-type: image/png");
imagepng($myImage);
imagedestroy($myImage);
?>
```

Figure A-3 shows the result on my machine.

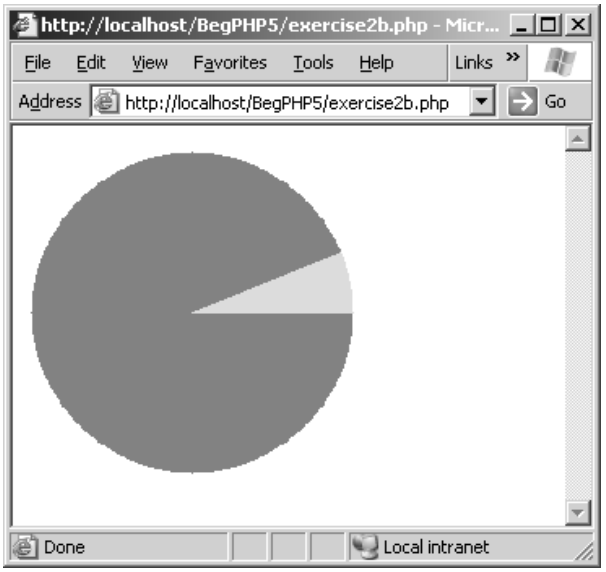

Figure A-3