Another eBookWholesaler Publication

Simple PHP with Robert Plank

Copyright 2003 by Robert Plank All rights reserved.

Proudly brought to you by

Lewis Philips signature books

Email

Recommended Resources

Web Site Hosting Service

Internet Marketing

Affiliate Program

Contents

Copyright © 2003 Robert Plank

Copyright © 2003 Robert Plank

Introduction

Welcome to Simple PHP. Originally written as a 17 week article series, Simple PHP is the most comprehensive tutorial available on the Internet.
Each chapter uses interactive instruction, closing with an assignment to complete before progressing to the next chapter.
The most effective way to learn the material provided here is to read an article, perform the assignments for that week, then take the quiz to see how well you understood that article's content.
Robert Plank is the coder of Lightning Track, RedirectPro, TurboThanks and CodeWarden, Ezine Rocket, Hypersplitter, and countless others.

If you'd like to give feedback, please visit RobertPlank.com.

Enjoy!

Copyright © 2003 Robert Plank

Chapter 1 Site Personalization With PHP

Your HTML files can work as PHP scripts.

Take any HTML file you have and rename it to a PHP extension. (So for example, if your HTML file is named sales.html, rename it to sales.php).

Put sales.php on your web server and run it in your browser. You get the exact same result as you did with the HTML page. Now, for the good part.

Say you wanted to personalize one of your sales pages. Let's call that sales.php. Now, if you had a newsletter or wanted to give this "personalized link" to someone, your visitor's name could be added on-the-fly to your sales page.

Your link should look something like this:

http://www.your.host/sales.php?firstname=Big&lastname=Bird

In this example, your.host represents your domain name and sales.php your HTML file turned PHP script. This would be the link you could give Big Bird when you ask him to visit your sales page.

Now, edit sales.php and find where you want Big Bird's name to appear. The first and last names are given to the script separately so you could have anything from "Hi Big Bird!" to "Dear Mr. Bird..." For simplicity let's just stick with showing both the first and last names for now.

Insert this anywhere into your "script":

```
<?php echo "$firstname $lastname"; ?>
```

Copyright © 2003 Robert Plank

Now try that sample link I gave you earlier on that sales page of yours. You should see the phrase "Big Bird" anywhere on your page.

Remember that you can stick this in anywhere on the page. So for example if you wanted it to say "Dear Big Bird," you would just do this (with the comma at the end):

```
Dear <?php echo "$firstname $lastname"; ?>,
```

If you wanted to show only the first name, use
`<?php echo "$firstname"; ?>` instead. The same applies to the last name.

If you want to go ahead and shorten the URL even further, you can even use the letters "F" and "L" instead of firstname and lastname.

For example, try this in your page:

```
<?php echo "$f $l"; ?>
```

And then use this URL (substituting with your correct URL, of course):

http://www.your.host/sales.php?f=Oscar&l=Grouch

The result is the same. I think we want our URLs to look a little better, though.

The format I used in that above example was:

http://www.your.host/sales.php?f=Big&l=Bird

Now, the problem with this is that this link looks really ugly. Another setback with this format is that anything to the right of the "?" won't be indexed by some of the smaller search engines.

Here's a way to change this:

http://www.your.host/sales.php?f=Firstname&l=Lastname

Copyright © 2003 Robert Plank

To this:

http://www.your.host/sales.php/f=Firstname/l=Lastname

It's really easy. Just change your script to this:

```
<?php

$myvars = explode("/",$REQUEST_URI);
for ($i=0;$i<count($myvars);$i++) {
  $holder = explode("=",$myvars[$i]);
${$holder[0]} = $holder[1];
}

echo "$f $l";

?>
```

And in just a few lines of code, your personalized PHP script's URL just looked a whole lot better.

I won't bore you with the details of what that script does, but it basically chops up the URL you gave it and picks out the pieces it wants using array exploding and variable-variables.

This snippet can also be used on almost any PHP script as well. (I used this method when I coded Brian Garvin's Lightning Track ad tracker... if you ever see a URL anywhere with "go.php/etc" in it, that's my script.)

Hopefully we all learned something today. This mini-tutorial was brought to you by the letters "P-H-P."

Copyright © 2003 Robert Plank

Assignment:

Try this script again, this time trying to insert a plus sign (+) for the value of "f", for example: example.php?f=first+name

QUIZ

1. A variable called "onion" would be referenced like this:
 A: %onion
 B: $onion
 C: ***onion***
 D: #onion

2. True or false: A file ending in .php containing only HTML (no <? or ?> tags) will behave exactly like an HTML document.
 A: True.
 B: False.

3. If you wanted to output the variables $one, $two, and $three, with a space in between each, how would you do it?
 A: echo "$one two $three";
 B: echo $one $two $three;
 C: echo onetwo$three;
 D: output "$one $two $three";

4. Your script is located at "http://www.your.host/sales.php". You want to pass a variable called "monkeys" with the value "20" and a variable "fork" with the value "tasty" into this script. How would you do it?
 A: http://www.your.host/sales.php?monkeys?20?fork?tasty
 B: http://www.your.host/sales.php?monkeys=20?fork=tasty
 C: http://www.your.host/sales.php,monkeys=20,fork=tasty
 D: http://www.your.host/sales.php?monkeys=20&fork=tasty

Copyright © 2003 Robert Plank

5. Your script, located at "http://www.your.host/t.php" contains the following code:
<?php echo $YELLOW; ?>

If you go to "http://www.your.host/t.php?yellow=yes", what will you see?
A: I won't see anything.
B: It will output "yes".
C: The script won't even work.
D: It will output "yellow".

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 2 Password Protection With PHP

Read the previous article/chapter, because I have one addition to make your life a little easier.

Now, remember how we personalized a page for your visitor? This works fine, but what do we do if they didn't use that special link, and just went to the page?

What I'm saying is, if your special personalized page was at http://www.your.host/sales.php/f=Oscar/l=Grouch but your visitor only went to http://www.your.host/sales.php. Instead of the name there would just be a blank spot! Last chapter I forgot to cover this.

All we have to do to fix it is to tell PHP that if they didn't leave a name, to substitute one in for them. So let's say that if they left their first name blank to make their first name "Friend". This way instead of saying "Dear Oscar:" it would say "Dear Friend:".

Put the following line of code JUST ABOVE THE LINE that says something similar to: echo "$f $l" :

```
if ($f == "") { $f = "Friend"; }
```

That way, you can use your special personalized page as a normal page and no one will be the wiser.

Password protection is something you need every once in a while. Whether it's a secret site you're running or just the control panel of your favorite script.

Sometimes you don't need a fancy solution like .htaccess if you're only worrying about a single user (you). But JavaScript passwords can be worked around, and

Copyright © 2003 Robert Plank

HTML-based passwords based on cookies, written in PHP are complicated and take time to write. Htaccess is nice but it's a pain if you just want to use it for one person.

Here is a simple way to use HTTP authentication (the same you see used by htaccess) with just a few lines of code. Below are the sample contents of a file you can use.

```
<?php

$myusername = "myusername";
$mypassword = "mypassword";
$areaname = "My Protected Area";

if ($PHP_AUTH_USER == "" || $PHP_AUTH_PW == "" ||
$PHP_AUTH_USER != $myusername || $PHP_AUTH_PW != $mypassword) {
  header("HTTP/1.0 401 Unauthorized");
  header("WWW-Authenticate: Basic realm=\"$areaname\"");
  echo "<h1>Authorization Required.</h1>";
  die();
}

?>

my main text.
```

* The function die(); I've placed below it tells PHP to stop everything immediately. I do this out of habit (and you should too) because it makes sure that once you've done the redirect, nothing else gets sent. Which is what we call "a good thing".

Last chapter we learned that PHP code can be integrated into your HTML. All you have to do is make sure the file ends in .php (for example, "firehydrant.php") and it will work. Everything that comes in between this:

```
<?php
```

Copyright © 2003 Robert Plank

```
/* And this: */

?>
```

Is treated as PHP code. Everything outside of those tags is treated as plain HTML.

When copying this code over be SURE to include that last line where it says "my main text." Note that "my main text" is located outside of the PHP code brackets. This means that where you see "my main text" can be your normal HTML file!

Take all of this code and Upload the script onto your web server and run it in the browser. You should be greeted by a password popup box similar to those you see with htaccess. Enter "myusername" as the username and "mypassword" as the password. You should be given a page that says "my main text" and nothing else.

Close your browser window (this is very important) and go back to that page. Try entering the wrong info. The box will come up again. You have three tries and then are given that dreadful "Authorization Required" message.

If you want to take the next step, go back to your code and change "myusername" and "mypassword" to a username and password of your choice. Upload it back to your web server and try again. Now go to that page again and you'll see that you can only be let in using the username and password you chose for yourself.

Now change the part that says "My Protected Area" to something else, say "John Calder's Bar and Grill." Upload and try it. You'll see when that password box comes up under "Realm" it'll say "John Calder's Bar and Grill." You can change this to whatever you like.

But what if you want to password protect just a handful of files? Do you have to copy and paste this code onto PHP script after PHP script?
Hell no!

Copyright © 2003 Robert Plank

Take the code you just modified and take the last line out of it. You know, the one that said "my main text." All you should have in there now is everything in between the PHP brackets (<?php and ?>).

Save this file as "auth.php". You can rename this later, on your own time.

Make a new file called "test.php" or just rename a normal HTML to this name. It doesn't matter. At the very top of test.php (the VERY top, meaning the first line) copy and paste this line of code:

```
<?php include("auth.php"); ?>
```

Upload auth.php and test.php to your web server and run test.php. Make sure both files are placed in the same folder. Now, try to go to test.php in your web browser. You'll see that you can't get to test.php without the right username and password. You can do this to any file with a ".php" extension just by adding that one line of code.

The catch to it is that this line of code has to be at the very top of the file. On the very first line. The reason for this is that when the script asks for a person's username and password, these are sent using HTTP headers and *must* come before anything else.

Of course, this doesn't take care of your secret sites or private members' areas, where you have to deal with several logins, but that's what htaccess is for.

While we're on the subject of includes, one last thing before we finish up.

Includes are basically a way of absorbing other files into your script. As you saw when we included auth.php, the script read everything that was in auth.php and used it as if the contents of that file were actually there. This works with not only PHP scripts but also with other files as well.

Make a new file called "header.html". Put anything you want in it, but I just put "This is my header
" when I did it.

Copyright © 2003 Robert Plank

Make a second file called "footer.html". Again, go again and put anything you want in it, but I just put "This is my footer
" in.

Make a third file called "main.php". Copy the following into it.

```
<?php include("header.html"); ?>

This is my main page<br>

<?php include("footer.html"); ?>
```

Upload all three into the same folder and run main.php. You should see the following:

This is my header
This is my main page
This is my footer

This is just a basic example of how includes can be used. But if you have a web site with several pages and the same layout... wouldn't it be easier just to put everything above your main text in header.html and everything below that main text in footer.html? That way if you change your design you only have to edit 2 files instead of 100 or 200?

You'd think.

Assignment:

Look up the include() function at PHP.net. Find out the difference between include(), include_once(), require(), and require_once().

QUIZ

1. You have a file called "test.html". Your file contains this code:
<?php echo "hey friend"; ?>

Copyright © 2003 Robert Plank

What will be the resulting HTML? Meaning, what will you see in the browser (not in the source code):
A: The phrase "hey friend" in bold
B: The phrase "hey friend"
C: The phrase "hey friend" and then a line break
D: I won't see anything because the filename doesn't end in ".php"

2. Let's say you wanted to include the file "hello.txt" in your script. How would you do this?
A: include("hello.txt");
B: include=hello.txt
C: #include <hello.txt>
D: It's impossible.

3. In one of the code snippets in this article, I made use of the die() function. What do you think this means? (Feel free to look it up at php.net if you don't know.)
A: Causes the entire server to shut down.
B: Reduces the heartbeat of the user to 0 beats per minute.
C: Causes the script to give up right there and then.
D: No one really knows what it does for sure.

4. True or false: For HTTP authentication to work, the "realm" *must* be set to "John Calder's Bar and Grill".
A: True.
B: False.

5. True or false: HTACCESS uses HTTP authentication.
A: True.
B: False.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 3 Autoresponders With PHP

First off, you're going to learn how to make some simple text-only autoresponders.

To do that, we need to know three easy things: redirection, mail sending, and form submission.

When we finish with that, you will know how to put those components together and create an autoresponder. Because if you think about it, that's all an autoresponder does. Somebody enters in their e-mail address, is sent an e-mail message, and is then redirected to a new page.

Of course there are more complex autoresponders, like Gary Ambrose's Opt-In Lightning, or Wes Baylock's Mail Master Pro which handle multiple follow-ups and record the e-mail addresses of those who have signed up for the responder. But today we're just going to focus on how to make a very basic, very simple autoresponder.

Hopefully, you've seen what form fields in HTML look like. Here's some code you can use for an example:

```
<form action="some-script.php" method="post">
Enter Your E-Mail Address: <input type="text" name="email" size="30"><br>
<input type="submit" value="Submit">
```

Copy and paste this code into a file called "signup.html" and upload it to your web server. You'll see a text box waiting for your visitor to enter his or her e-mail address so they can be sent that autoresponse message.

Of course, the form won't work just yet because, if you look at the first line of that HTML code I gave you, you'll see that the form submits to a script called

Copyright © 2003 Robert Plank

"some-script.php". And we haven't made that just yet.

Look on the second line of "signup.html", at the last half of the line. You should be familiar with HTML tags, but if you're not, an HTML tag consists of two parts: the parent tag and the attributes.

The parent tag is simply the tag's designation. For example, if you had a slice of HTML code that looked like this:

<font face="Verdana" size="1">

Then the parent tag would be "font". The rest of what's enclosed in the tag tells the browser what to do with it. For example, in this tag the attributes are that the font should be Verdana with size 1.

Why am I telling you all this? Because it relates to the HTML code you see in signup.html.

Now, when you look at this:
<input type="text" name="email" size="30">

The code tells the receiving browser that this is an "input" tag, meaning that it's a form field. The name of this item is "email" and its size is 30, meaning this text box should be 30 characters in width.

When the form is submitted, it takes all the values of all the fields inside that form and throws it at its destination. In this case, our destination is "some-script.php".

If you're lost, this will all make a whole lot more sense once you try this next step.

Make a file called "some-script.php" and paste this line of code into it:

```
<?php echo $email; ?>
```

Upload the script in the same folder as signup.html, and go to "signup.html".

Copyright © 2003 Robert Plank

Type your e-mail address in and click the submit button.

You should see a new page containing just your e-mail address and nothing else.

Is this starting to make sense? You told the PHP script to dump the contents of the variable called "email" to the screen, and you just submitted a form with a text box called "email".

If you want to try one more exercise like this, change the name of the text box to, say, "goober" in signup.html and change the $email in some-script.php to $goober. Upload both, go to signup.html, and type anything into the text box. You'll get the same result.

This is how you'll pass data from forms (like text fields, drop down menus, radio buttons and the like) along into the PHP scripts you create.

We've just covered how to submit form elements into PHP. Now let's focus on sending mail.

PHP has a really simple function that uses whatever mail sending program is installed on your server to send messages to the outside world. If you have a crappy web server, this step might not work and you'll have to use a different web host if you want to try this.

But if you're on a good web host that has PHP installed *correctly*, this shouldn't be a problem.

Up until now we haven't used functions in PHP too much, aside from simple things like include() and header(). Today's your lucky day, because functions work in a very similar way to HTML tags. You have the parent tag, and the attributes (or parameters).

The mail() function basically works like this:

```
mail("recipient","subject","body","headers");
```

Let's start off by sending a simple e-mail message to yourself. We won't need

Copyright © 2003 Robert Plank

any special headers this time around, so this will be quick and painless. Copy this one line of code into "mailtest.php":

```
<?php mail("billg@microsoft.com","Hello","Hi. This is the body of my message."); ?>
```

Replace "billg@microsoft.com" with your actual e-mail address, but be *sure* to keep quotes around it. Save it, and upload mailtest.php to your web server and run it in the browser. You should see a blank page. Wait a few minutes and check your mail. You should see a mysterious mail message in your box with the subject "Hello" and the message "Hi. This is the body of my message."

If you're using a free e-mail service or a weird ISP, the message won't come through because a lot of mail servers these days require that certain headers are present in the message.

Let's do that now.

What's below isn't important enough to explain thoroughly, but it's just header information that is interpreted by the mail server. This data tells us that we're sending a plain text e-mail, that the message came from your e-mail address (and gives your name), and tells us that the e-mail "client" we used was PHP.

```
$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname <$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer: PHP";
```

This is the code you should have by this point, complete with the header information and the variables which tell the script what your name and e-mail address are:

```
<?php

$email = "billg@microsoft.com";

$myname = "Your Name Here";
```

Copyright © 2003 Robert Plank

```
$mymail = "your@email.here";

$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

mail($email,"Hello","Hi. This is the body of my message.",$headers);

?>
```

Notice how we've simplified things a bit by using variables in the mail() function. That way we don't have to retype things. This method also looks better (in my opinion anyway) and is easier to tweak once you're ready to actually customize it for yourself.

Try this out again. Believe it or not, but you just made your first autoresponder! Before we move on let's make this look even cleaner:

```
<?php

$myname = "Your Name Here";
$mymail = "your@email.here";

$subject = "Hello";
$body = "Hi. This is the body of my message.
Notice how I can continue typing right on the next line!";

$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

if ($email != "") { mail($email,$subject,$body,$headers); }

?>
```

All I did here was just make things look nicer, but notice how I removed the line that set $email to "billg@microsoft.com." This is because the value of $email

Copyright © 2003 Robert Plank

will be passed to the script from that form we made earlier.

This also sends the e-mail message ONLY if the value of $email is not blank. So if someone just hit the submit button without entering an address, the script won't try to send the e-mail message.

Everything should be ready for you to try out now. Re-upload "some-script.php" and go to signup.html. Enter your e-mail address in the field, hit submit and wait for that mail message to arrive.

There's only one step left to making this autoresponder complete. And that's sending the user somewhere so they aren't given a blank page.

Find this line in your script:
```
if ($email != "") { mail($email,$subject,$body,$headers); }
```

And paste this directly underneath it:
```
header("Location:http://www.my.host"); die();
```

Put this in a text file, upload this script and try it out. You'll see that once the autoresponse message is sent, you're directed to www.my.host (or, whatever you end up changing my.host to). Now, go ahead and change it to whatever URL you want to use. Or, use it with a variable so the end result is like this:

```
<?php

$myredirect = "http://www.my.host/thankyou.html";

$myname = "Your Name Here";
$mymail = "your@email.here";

$subject = "Hello";
$body = "Hi. This is the body of my message.
Notice how I can continue typing right on the next line!";

$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
```

Copyright © 2003 Robert Plank

```
PHP";

if ($email != "") { mail($email,$subject,$body,$headers); }
header("Location:$myredirect"); die();

?>
```

Don't forget to change the values above. "http://www.my.host/thankyou.html" needs to point to the URL where thankyou.html is stored.

You're done.

Assignment

Try this code:

```
header("Location:mailto:email@my.host?subject=hello");
```

QUIZ

1. Given this text box in HTML:

<input type="text" name="email" size="30">
If this was submitted to a script using the <form> tags, what would be passed to the script?
A: The variable $email, containing whatever is written in the text box.
B: The variable $text, containing whatever is written in the text box.
C: The variable $email, containing the value "text".
D: The variable $text, containing the value "email".

2. Given this HTML code:

Copyright © 2003 Robert Plank

<a href="link.html">
What is the parent tag within this HTML tag?
A: a
B: href
C: link
D: html

3. What does this code do?
header("Location:http://www.simplephp.com");

A: It doesn't do anything, it's a fraud.
B: It outputs the text "Location:http://www.simplephp.com" to the browser.
C: It redirects to http://www.simplephp.com
D: It makes a phone call to http://www.simplephp.com and tells him/her to expect three dinner guests.

4. True or false. This code will execute properly:
<?php mail("shitcan@shit.can", "ergonomics" ,"jell-o pudding."); ?>
A: True.
B: False.

5. If this was somewhere in the *header* of an e-mail you were sending:
Content-Type: text/plain; charset=us-ascii\n

What would it do?
A: Not allow the message to ever leave U.S. soil.
B: Tell the recipient's e-mail client that the message is in plain text.
C: It keeps the message from being delivered for 48 hours.
D: It's kind of like Rush Delivery, but for e-mail.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 4 More Autoresponders With PHP

Last chapter, we learned how to make simple text-only, one-shot autoresponders.

After the article was published, I was asked to teach a method of creating follow-up responders. You know the type... someone subscribes and receives their initial message. After a few days, or a few weeks, or even months, another message in the autoresponder series is mailed out. This is useful if you're presenting a course that is split it up into many different lessons, which you want to give someone at a certain pace.

Well, I thought about showing you how to do that this week, but we're really not at that point yet. If I showed you how to do that, I'd lose most of you entirely. That's way ahead of us for now.

I was also asked, however, if it were possible to create HTML autoresponders. And those are quite easy to make.

If you haven't read the last chapter's tutorial, I recommend you do.

Now, this should be the code you have for your autoresponder (your personalized values will vary). Remember that our script was called "some-script.php" in our example.

```
<?php
$myredirect = "http://www.my-domain.name/thankyou.html";

$myname = "Your Name Here";
$mymail = "your@email.here";

$subject = "Hello";
```

Copyright © 2003 Robert Plank

```
$body = "Hi. This is the body of my message.
Notice how I can continue typing right on the next line!";

$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

if ($email != "") { mail($email,$subject,$body,$headers); }
header("Location:$myredirect"); die();
?>
```

Replace this line:

```
$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";
```

With this one:

```
$headers = "Content-Type: text/html; charset=iso-8859-1\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";
```

If you look closely, all that we've done is basically changed the file format of the text we're sending. Upload it to your server and run the script like you did last time, entering your e-mail address in and submitting the form.

You should notice that the text looks slightly different. Let's make it completely different.

Replace this code:

```
$body = "Hi. This is the body of my message.
Notice how I can continue typing right on the next line!";
```

With this:

Copyright © 2003 Robert Plank

```
$body = "<font face=\"Arial\" size=\"6\" color=\"#FF0000\"><b>Look, it's an
HTML message!</b></font><br><br><img
src=\"http://www.my.host/gumby.jpg\">";
```

Re-upload and try now. Look at what you've just done! An HTML autoresponder, by changing just a few lines of code.

It might also interest you that an e-mail message can be sent using a variety of file types... not only in text and HTML but also in Rich Text Format.

Now that we've gotten that out of the way, let's move on to putting attachments in your e-mail autoresponder. Compared to a lot of the stuff we've gone over so far, this is more complicated.

You're going to learn how to send attachments in e-mail using PHP. Just as we sent text e-mails before, you'll learn how to attach an image to your message.

This is how you can make those autoresponders that mail the customer the product or e-book once they've bought. Or it would work as a really creative way to distribute an article or publication to someone. They might even think you personally sent the e-mail and the attachment.

Making it isn't going to be easy, though.

Let's start off by creating a whole new script. Let's call it "myscript.php". When I made my example I just copied the first few lines from some-script.php because we're going to use them again. So, let's do that:

```
$myredirect = "http://www.my.host/thankyou.html";
$myname = "Gumby";
$mymail = "null@jumpx.com";
$subject = "Attachment sample";
```

Of course you can change all of this... but this just sets the variables I'll use later, my thank you page, from name, from address, and subject.

Next, I'll add my message.

Copyright © 2003 Robert Plank

```
$message = "Hey!  Boy, have I got a file for you.

-Gumby";
```

This is going to be the plain text message that accompanies the attachment. We could have gone without one, but since this is an autoresponder after all, I think we'd better keep a message in with the attachment.

Notice how I called this variable $message and not $body. This will come up again in a bit.

Next, we'll have to read the file we plan on sending. I haven't showed you file reading before, but hopefully it won't be too big of a deal. But I'll go over it line by line.

It's best to think of file reading in this way. Let's say that you are the computer and all your papers are stored away in a filing cabinet. You want to know what's on a certain piece of paper, but first you have to take that piece of paper out and read all the way through it to know what's on that particular piece of paper. First:

```
$myfile = "gumby.jpg";
```

This just declares a variable that gives us the file we want to open. You don't *have* to do this, but I normally do just because it looks nicer.

```
$fp = fopen($myfile,"r");
```

We just opened up the file "gumby.jpg" (see how we called $myfile?) and assigned it to the file pointer called $fp. This file pointer is just a way of representing "gumby.jpg". I don't know how else to explain it in words.

What we want to do next is read all the data in "gumby.jpg" and stash it in a variable we can use later. So, we do this:

```
$contents = fread($fp,filesize($myfile));
```

Copyright © 2003 Robert Plank

What this does is look at our file pointer, $fp. The second parameter in this function (where it says "filesize($myfile)") tells PHP how far in the file to read. The filesize() function calculates the exact size of a file. So, let's say our file is 12 kilobytes in size. We want to sweep through all 12 KB of the file.

The contents of the file will be stored in a variable called $contents.

And lastly, we have to put that piece of paper away. So:

```
fclose($fp);
```

Closes that file.

So, now we've read an entire file into a variable. Just for fun, open up notepad or some other text editor you have handy, and open up an image file. Any image file.

Blechh!

Look at all those weird and funky characters. We need to make that look nicer if we plan on sending it over e-mail.

E-mail attachments usually use base 64 encoding. I won't go into details about it right now, but base 64 encoding is just a way of substituting all those weird characters for nice readable letters.

There's a write-up on base 64 here if you're REALLY curious. But it isn't at all necessary reading.

http://www.freesoft.org/CIE/RFC/2065/56.htm

Conveniently, though, PHP has a base64 encoding function. So, what we'll do is take $contents, which contains everything in that image file, and encode it into base64, storing the new data into a variable called $myimage.

```
$myimage = base64_encode($contents);
```

Copyright © 2003 Robert Plank

Now we have the file we want to send in the proper "encoded" format. And now for the fun part, the headers.

```
$headers = "From: $myname <$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nBcc: admin@jumpx.com\nX-Mailer: PHP\n";
```

This line should look familiar to you. It's similar to the header line we had in some-script.php but without the Content-Type stuff. (You know, the chunk that told us if we wanted the message in plaintext or HTML.) We'll add the Content-Type information later.

One thing to note, is that since this is a multi-part MIME message, our entire message (even the "body") will be contained in the headers. So things might look a bit messy. You've been warned.

```
$headers .= "MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=\"myboundary\";
Content-Transfer-Encoding: 7bit\n";
```

I've split up the contents in $header for easier reading, though. If you see a ".=" and not just a "=", it means that instead of completely filling the variable with what we're adding, we're ADDING-ON to that variable.

In other words, I could have shown you $headers as one big long ugly string of text but I split it up so that I could explain each piece of it.

Look at that piece of code above. There isn't much you need to know, just that these are needed if you plan on doing a multi-part MIME message (which this is). One thing to notice is the part where it says boundary=\"myboundary\";

You might be wondering what the heck those \" are for. They're just a way of showing a quotation mark when you're already inside a set of quotation marks.

Example. We've used the echo function before. You can easily have a line of code that says:

```
echo "John said hi. Hello there, John said.";
```

Copyright © 2003 Robert Plank

And it would output:
John said hi. Hello there, John said.

But what if you wanted to show quotes in there? You'd just do this:

```
echo "John said hi.  \"Hello there,\" John said.";
```

And it would output:
John said hi. "Hello there," John said.

Easy? I hope so.

The purpose of the boundary is to separate each "chunk" of the e-mail message. Because when this gets to its destination, your recipient's mail client needs a way of knowing which part is the attachment and which part is your message, right?

Remember for now that our boundary line will be defined by the word "myboundary."

```
$headers .= "--myboundary
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 7bit\n\n";
```

Aha! And there it is, myboundary in all its glory. Where we say --myboundary is where we define each part of the message. You can call your boundary anything from bacon12345 to EatAtJoes. But for this example I just decided on myboundary.

At the start of each "chunk" of the message, you must set the content-type. You've seen this part before.

And now, for the actual message.

```
$headers .= $message."\n";
```

That just plopped the message we wrote earlier (you know the one... "Hey! Boy, have I got a file for you.") into the message. Nothing really to see there, but note

Copyright © 2003 Robert Plank

the extra line I added in afterwards. It's important to keep that there.

And now, to lay down our other chunk which contains the file:

```
$headers .= "--myboundary
Content-Type: image/jpeg; name=$myfile;
Content-Transfer-Encoding: base64
Content-Disposition: attachment

$myimage
--myboundary--";
```

Pretty easy to understand there. The type is a JPEG, and we're just going to give it the same name as the actual file (gumby.jpg), it's base64 encoded, and it's an attachment. Then we go 2 lines down and give it our base64 encoded image. After that, on the next line, we give out our boundary again. Notice though, this time, the extra dashes on the end. It's --myboundary-- now, not just --myboundary. That means we're at the end of the message and there are no more chunks to deliver.

And the rest is simple!

```
if ($email != "") { mail($email,$subject,$body,$headers); }
header("Location:$myredirect"); die();
```

If you decided not to go step by step with me here, this is the source:

```
<?php

$myredirect = "http://www.jumpx.com/tutorials/3/thankyou.html";

$myname = "Gumby";
$mymail = "null@jumpx.com";

$subject = "Attachment sample";

$message = "Hey!  Boy, have I got a file for you.
```

Copyright © 2003 Robert Plank

```
-Gumby";

$myfile = "gumby.jpg";
$fp = fopen($myfile,"r");
$contents = fread($fp,filesize($myfile));
fclose($fp);

$myimage = base64_encode($contents);

$headers = "From: $myname <$mymail>\nReply-To: <$mymail>\nReturn-Path:
<$mymail>\nBcc: admin@jumpx.com\nX-Mailer: PHP\n";

$headers .= "MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=\"myboundary\";
Content-Transfer-Encoding: 7bit\n";

$headers .= "--myboundary
Content-Type: text/plain; charset=\"iso-8859-1\"
Content-Transfer-Encoding: 7bit\n\n";

$headers .= $message."\n";

$headers .= "--myboundary
Content-Type: image/jpeg; name=$myfile;
Content-Transfer-Encoding: base64
Content-Disposition: attachment

$myimage
--myboundary--";

if ($email != "") { mail($email,$subject,$body,$headers); }
header("Location:$myredirect"); die();

?>
```

I've taught you how to send an image as an attachment. But, one last thing

Copyright © 2003 Robert Plank

before we're finished.

I know... that was a lot to take in one chapter, but there is just one last thing.

Chances are if you plan on using an autoresponder like this you'd use it to send an e-book, an application, whatever... and you probably wouldn't want it in the form of a JPEG file, am I right?

So we'll just make a simple modification here and set it up to send a zip file instead.

That is, if you need it.

I'm going to be sending tutorial4.zip as an attachment, instead of using gumby.jpg. Well, first of all, we have to change the file we plan on using. Find this line:

```
$myfile = "gumby.jpg";
```

And replace it with:

```
$myfile = "tutorial4.zip";
```

Next, we have to change the actual type of the file we're sending. So, find this:

Content-Type: image/jpeg; name=$myfile;

And replace it with:

Content-Type: application/zip; name=$myfile;

Now try it. Remember that "image/jpeg", "application/zip" and all these are MIME types. If you go to the end of this page you'll see a short list of some: http://www.infomotions.com/waves/mimetypes.html

I don't know where you can find a longer list of MIME types, but you really have the potential to go anywhere with this. Attach HTML files on-the-fly, text files,

Copyright © 2003 Robert Plank

word documents, MP3s, PDFs, SWFs... just go crazy.

Hopefully the next tutorial won't be as brutal. :-)

Assignment

Choose any file. Upload it up to your web server and create a new script. Make a new script, called checksize.php, and put this code into it:

```
<?php
$myfile = "your_chosen_files_name";
echo filesize($myfile));
?>
```

Change the value of $myfile to your file's name. Upload the script and run it. Then check the actual size of the file. What does the number returned by filesize() represent?

QUIZ

1. True or false: Flash can be embedded into e-mail messages.
A: True.
B: False.

2. If we have an attachment to put into an e-mail, what format must it be in?
A: Base 10
B: Base 64
C: Base 16
D: All Your Base Are Belong to Us

Copyright © 2003 Robert Plank

3. What functions are needed if we want to read the contents of a file into a variable? (Hint: We did it when we attached the file "gumby.jpg".)
A: open() and read()
B: fopen() and fattach()
C: fopen() and fread()
D: open() and attach()

4. What does ".=" do as opposed to "="?
A: ".=" adds onto the end of a variable's value while "=" just sets a new value.
B: There is no difference.
C: There is no such thing as ".="
D: None of the above.

5. In e-mail messages, it's possible to mix-in several types of messages in the same e-mail (e.g. text, HTML, and an attachment all in the same e-mail document).
A: True.
B: False.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Charpter 5 JavaScript Fun With PHP

Hello! Today I want to have some fun using some JavaScript and PHP tricks.

PHP and JavaScript are a really powerful combination because it's possible to combine the server-side advantages of PHP with the client-side advantages of JavaScript. I've used this dual approach to make some really unique yet simple scripts... a couple of HTML generators, a web-based support chat tool, even a very basic Opt-In Lightning clone. (You know, those one-click JavaScript autoresponders.) In this article I'm going to sort of dip into JavaScript a little bit and show you how to do a few useful things, using PHP as well, of course.

We all want to be available through our web sites. Back in the old days, before spiders and spam-bots ran back and forth around the net, we'd throw our e-mail address on our front page so we'd be readily available. Of course that isn't something you can really do these days, without getting 20 pieces of spam in your mailbox every day. Sure, there are filters, but with your address getting out to more and more people, it just becomes a chore.

I think I first saw a solution to this around 3 or 4 years ago. Basically each letter (and by letter, I mean "character") in the e-mail address is converted into an ASCII value. If you don't know what this is, don't worry about it. Basically, computers store letters as numbers. For example, an upper-case "A" is stored as the number 65, an upper-case "B" as the number 66, and so on.

So, if your e-mail address was, say "some@email.host", if you encoded in it this way, a sample e-mail link using this address would look like this:

<a href="mailto:some@e
mail.host">
some@emai
l.host">E-Mail Me

Copyright © 2003 Robert Plank

Now, this works even without JavaScript. Which would be nice, but the spam bots are even smarter these days, and can just as easily harvest the real address. I wouldn't even bother with it.

As you might expect, a JavaScript solution has arrived. Pretend again that your address is "some@email.host". If you're at all familiar with e-mail links then something like this:

```
<a href="#"
onclick="location.href='mail'+'to'+':'+'some'+unescape('%40')+'email.host';return
false;">E-Mail Me</a>
```

When a normal user clicks on this, the browser will try to go to the link, the hash or pound sign (#) you see there, but that will really just make the user stay on the same page. But then, JavaScript will say, "Ah-ha! They've clicked on that link!" and do its thing.

If you're familiar at all with JavaScript, you know that the plus sign (+) joins strings together. Let's take everything inside of the "onclick" parameter in that link:

```
location.href='mail'+'to'+':'+'some'+unescape('%40')+'email.host';return false;
```

And join it together:

```
location.href='mailto:some'+unescape('%40')+'email.host';return false;
```

Now, let's see... 40 is the ASCII value of the "at" symbol (@) and the unescape function in JavaScript just converts ASCII values to understandable characters. So let's take that part that says "unescape('%40')" and change it to the at sign.

```
location.href='mailto:some'+'@'+'email.host';return false;
```

And of course join it all together:

```
location.href='mailto:some@email.host';return false;
```

Copyright © 2003 Robert Plank

Now I'll separate this onto multiple lines so it's easier to read:

```
location.href='mailto:some@email.host';
return false;
```

This says, take the user to this place (the mailto link) and then stop completely. If you're familiar with using e-mail addresses in HTML links, putting "mailto:" at the beginning tells the user's web browser that, oh, it's an e-mail address, and causes a send window in the user's mail client to open immediately. Which is exactly what this does.

My line of thinking, however, is... how long will it be until the spam bots figure even this out? If your web browser can decode that into a clickable link, chances are that at least a few harvesters already can.

Which brings us to the script you and I are going to make. This is going to be a JavaScript-based tiny little script that will let people send quick e-mails to you. "JavaScript?!?", you say? "What in the hell is this? I thought I was learning PHP?" And you are. But this will be fun.

Of course there are ways to fudge it and send HTML using forms. (What you do is, have the form submit to an e-mail address instead of a location on the web.) Furthermore, there are ways to create the HTML invisibly and then send e-mail using your visitor's HTML client. Personally I don't like this, mainly because different e-mail clients will do unexpected things when this happens, like sending the message inside an attachment like "wf7xrb.tmp".

Plus, as should be obvious, your e-mail address is plainly visible like that. What we want is a way for people to just send us a quick note without them knowing our address. If you've read the previous chapters you should already know how to do that, since in the article about autoresponders we covered not only how to interpret the data someone puts into your script using a form, but also how to use the mail() function in PHP to actually send mail.

First, you want to know how to make a link to a JavaScript object.

Copyright © 2003 Robert Plank

There's another way to do this, though this method is preferred.

To start off we need a function name. Though we could have something like "activate()" or "do_now()", I'd like to call it NoteWire.

You can name your function whatever you want to, but I'm calling mine NoteWire. :-)

Oh, and let's not forget that these names are case sensitive. If you call your function "NoteWire" and you try to call "notewire", you'll get a JavaScript error.

Now, for the link:

```
<a href="#" onclick="javascript:NoteWire()">Click on me!</a>
```

This goes to the function called NoteWire() in your JavaScript code. We don't have a function called NoteWire() yet, so let's make one:

```
<script language="JavaScript">
<!--

function NoteWire() {
alert("Hello");
}
// -->
</script>
```

This JavaScript code can go in between the <HEAD> tags in your HTML document, inside the <BODY> tags, wherever. Just as long as it's there.

Save or refresh the page if you need to and open it in your browser. If you want to upload it to your web host, go for it though that's not necessary yet. Click on that link, and if you've done this right, you should be seeing a popup box that says "Hello" to you.

Let's think about what information we want from our visitor. Their name, their e-mail address (so we can reply to them if need be), and of course their message.

Copyright © 2003 Robert Plank

It'll be a short message because they'll only have one line to type on.

Change your JavaScript code to this:

```
<script language="JavaScript">
<!--

function NoteWire() {

var name = "";
var mail = "";
var message = "";

alert("Hello");
}
// -->
</script>
```

What we've done here is added three variables, one called "name", one called "mail", and one called "message". I've filled each of these variables with empty (blank) values.

Now, to get information from the user. So we'd have something like this:

```
<script language="JavaScript">
<!--

function NoteWire() {

var name = "";
var mail = "";
var message = "";

alert("Hello");

name = prompt("Your Name:","");
mail = prompt("Your E-Mail Address:","");
```

Copyright © 2003 Robert Plank

```
message = prompt("Type your message:","");
}
// -->
</script>
```

Now, run this code in your browser. When you click on the link, first it will give you a "Hello", then ask for your name, e-mail address, and message. That's all nice, but afterwards... it doesn't do anything.

Now it's time to start getting ready to add the PHP part of this script. Since this is just going to be a really short message someone is sending (under 1000 characters) we could probably get away with just using a query string on it. By a query string I mean when you have a link such as "http://host/script.php?firstthing=1&secondthing=2" ... where you give the script the variables you want based on what you say you want to load.

If you're unfamiliar with using query strings, re-read my first PHP article.

All we have to do for this is take the inputs, and then piece them together in a string to get the URL we want to load. I'd start by specifying the URL where our PHP script is *going* to be. We haven't put it up yet, but we're going to remember where we said it is. So add a line like this:

```
var myurl = "http://www.my.host/notewire.php";
```

This is of course assuming that your host is "my.host" and that the script will be called notewire.php and is on the highest level. You can make this anything you want as long as you can have a script there.

The next step is figuring out what exactly our completed request will look like, complete with the script to be requested and the query string. I always use the alert() function for this, as it shows me almost exactly what we're really going to see.

Remember that the piece of JavaScript code we looked at earlier used the plus sign to glue strings of text together. That's what we'll be doing again, like so:

Copyright © 2003 Robert Plank

```
alert(myurl+"?name="+name+"&mail="+mail+"&message="+message);
```

Here's what we've built so far.

```
<script language="JavaScript">
<!--

function NoteWire() {

var name = "";
var mail = "";
var message = "";

alert("Hello");

name = prompt("Your Name:","");
mail = prompt("Your E-Mail Address:","");
message = prompt("Type your message:","");

var myurl = "http://www.my.host/notewire.php";
alert(myurl+"?name="+name+"&mail="+mail+"&message="+message);
}
// -->
</script>
```

Almost immediately I see a problem with this. If the message contains spaces or even the "&" symbol, our query string will be all messed up. Again we pull something out of the JavaScript snippet at the top of the article. Remember unescape('%40')? Well, we want our data to look something like that, so it'll be nice and tidy in the query and nothing will go wrong.

So, where you see, say, the variable "name" being outputted, we want escape(name) so that each character is turned into a number, boxing it up in a sense. It's kind of like mailing a parcel: you want to be *certain* everything is boxed up nicely.

You only need to change that last line, so it becomes:

Copyright © 2003 Robert Plank

```
alert(myurl+"?name="+escape(name)+"&mail="+escape(mail)+"&message="+e
scape(message));
```

And again, our code so far:

```
<script language="JavaScript">
<!--

function NoteWire() {

var name = "";
var mail = "";
var message = "";

alert("Hello");

name = prompt("Your Name:","");
mail = prompt("Your E-Mail Address:","");
message = prompt("Type your message:","");

var myurl = "http://www.my.host/notewire.php";
alert(myurl+"?name="+name+"&mail="+mail+"&message="+message);
}
// -->
</script>
```

Copy this code and save it, then try it. Once you've filled everything out, a new long query string will pop up. This will be our URL. We will have JavaScript open this URL for us, giving the data to the PHP script without ever seeing the target's e-mail address.

There's only one way I know of having JavaScript open a URL *without* a redirect. It uses the Image object.

If you don't know what an object is, don't worry about it. I like to think of a specialist, who knows how to do a set of things, but it's lost without your

Copyright © 2003 Robert Plank

guidance. If you need work done on your toilet, you call a plumber. A skilled plumber comes over, who can do a ton of toilet-related things, but he'll only do what you tell him to.

I've mostly seen the Image object used to preload images for rollovers. You know on sites where you put your cursor over an image and it changes to something else? Well, some people have the image load while your cursor is over it, while others load the image as the rest of your page is loading, so when the mouse cursor points to it, the image changes instantly.

First, we need to make a new instance of Image. Let's call it trigger. So:

```
trigger = new Image();
```

Then we define a variable in the Image object called "src". "src" is, as you might have guessed, the source or location of this image we want to open. We already know exactly what we want to load based on that alert window we have popped up. So I want you to get rid of that line that has the popup box (the line that uses the alert function) and change it to this:

```
trigger.src = myurl+"?name="+name+"&mail="+mail+"&message="+message;
```

Finally, it's always a good idea to tell an Image object what to do once it's loaded. Just for fun let's make a new function called "done", and all it does is popup a box that says "We are done". Add something like:

```
function done() {
alert("We are done");
}
```

Remember to create this OUTSIDE the brackets of the NoteWire() function, because this is a totally new function. Go back up to that line that begins with "trigger.src" and add this on the line after trigger.src:

```
trigger.onload = done;
```

This tells our Image object called "trigger" that, when the image is loaded, to go

Copyright © 2003 Robert Plank

to the function called done(). I know that I left out the parentheses at the end of that line, that's just fine in this case.

WE'RE NOT FINISHED YET! Unfortunately that's all the time we have for now. I'll see you next chapter when we create the PHP side of this project, which does the real work of sending the e-mail we've given to it.

Assignment

No assignment for this chapter. Go back and re-read to make sure all the JavaScript stuff we've covered makes sense.

QUIZ

1. True or false: JavaScript code needs to be enclosed within the <script> and </script> tags.
A: True.
B: False.

2. Which of the following is true?
A: PHP scripts run only on the server, while JavaScript is run on the visitor's machine.
B: JavaScript runs only on the server, while PHP is run on the visitor's machine.
C: Both PHP and JavaScript can only be run on the server.
D: Both PHP and JavaScript can only be run on the visitor's machine.

3. What built-in JavaScript function can be used for a simple "popup message"?
A: popup()
B: MsgBox()
C: alert()
D: No such function exists.

4. NoteWire is...

Copyright © 2003 Robert Plank

A: One of JavaScript's many built-in functions.
B: Something specific to PHP only.
C: A brand of maple syrup.
D: Something the author made up.

5. The "glue" for joining strings in JavaScript (*NOT* PHP) is:
A: The percent sign.
B: The comma.
C: The plus sign.
D: The semicolon.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 6 More JavaScript Fun With PHP

If you didn't read the first half of this book, you really need to. Go back and read it.

Here's what we should have up to this point:

```
<script language="JavaScript">
<!--

function NoteWire() {

var name = "";
var mail = "";
var message = "";

alert("Hello");

name = prompt("Your Name:","");
mail = prompt("Your E-Mail Address:","");
message = prompt("Type your message:","");

var myurl = "http://www.my.host/notewire.php";
trigger = new Image();
trigger.src = myurl+"?name="+name+"&mail="+mail+"&message="+message;
trigger.onload = done;
}

function done() {
alert("We are done");
```

Copyright © 2003 Robert Plank

```
}
// -->
</script>
```

Of course if you try this code, you *shouldn't* get that popup that says "We are done" because there really will not be an image there. We have to put a PHP script there, to process the query string, and then display an image so JavaScript can put that alert message there as confirmation.

Open a NEW FILE, and call it "notewire.php" if that's what you decided your script name should be. The first step is to define who our message will be sent to. This would usually be your e-mail address:

```
<?php

$tomail = "your@email.host";

?>
```

Then, take our data out of the query string, and of course convert it from the ASCII values into readable text:

```
$mymail = urldecode($mail);
$fullname = urldecode($name]);
$message = urldecode($message]);
```

Oh, but it's always a good idea to strip any "friendly" characters out of the body of your message, so change that last line to:

```
$message = stripslashes(urldecode($message));
```

We have the send to address, the send from address, the sender's name, and of course the body of the message. Sending the e-mail should be small beans if you're familiar with the mail function from previous articles:

```
mail($tomail, $subject, $message,"From: $fullname <$mymail>Reply-To:
<$mymail>\nReturn-Path: <$mymail>\nX-Priority: 3\nMIME-Version:
```

Copyright © 2003 Robert Plank

```
1.0\nContent-Transfer-Encoding: 8bit\nContent-Type: text/plain; charset = \"iso-8859-1\"\nX-Mailer: NoteWire");
```

Then, lastly, to display the image:

```
header("Content-type: image/gif");
$contents =
"R0lGODlhAQABAIAAAAAAAAAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw==";
echo base64_decode($contents);
die();
```

This first tells the browser that we're giving it an image (a GIF). What I did here was, I saved a 1 pixel by 1 pixel transparent GIF image and converted the raw image data into something that I could store in a script without any weird characters (base64). If you don't understand anything I've just said in this paragraph, just know that this is our way of making JavaScript happy. JavaScript came here looking for an image and it's getting one.

What we have at this point is:

```
<?php

$tomail = "your@email.host";

$mymail = urldecode($mail);
$fullname = urldecode($name);
$message = stripslashes(urldecode($message]));

mail($tomail, $subject, $message,"From: $fullname <$mymail>Reply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Priority: 3\nMIME-Version: 1.0\nContent-Transfer-Encoding: 8bit\nContent-Type: text/plain; charset = \"iso-8859-1\"\nX-Mailer: NoteWire");

header("Content-type: image/gif");
$contents =
"R0lGODlhAQABAIAAAAAAAAAACH5BAEAAAAALAAAAAABAAEA
```

Copyright © 2003 Robert Plank

```
AAICRAEAOw==";
echo base64_decode($contents);
die();

?>
```

Looks good, so what are we missing? Well, I always like to have a subject. It seems to me that it would be at least a little bit handy to snag the sender's IP address, so if anyone decides to send you 100 "quick notes", you at least have their IP address. Put this under the line that begins with "tomail":

```
$subject = "NoteWire ($REMOTE_ADDR)";
```

The final change I'd like to make to this script, is that if someone leaves out their e-mail address, name, or message, to not display the image. JavaScript won't see the image and that confirmation popup box won't appear.

So, I want to have an if-statement that makes sure $mymail, $fullname, and $message are all not empty.

```
<?php

$tomail = "your@email.host";
$subject = "NoteWire ($REMOTE_ADDR)";

$mymail = urldecode($mail);
$fullname = urldecode($name);
$message = stripslashes(urldecode($message));

if ($mymail != "" && $fullname != "" && $message != "") {

mail($tomail, $subject, $message,"From: $fullname <$mymail>Reply-To:
<$mymail>\nReturn-Path: <$mymail>\nX-Priority: 3\nMIME-Version:
1.0\nContent-Transfer-Encoding: 8bit\nContent-Type: text/plain; charset = \"iso-
8859-1\"\nX-Mailer: NoteWire");

header("Content-type: image/gif");
```

Copyright © 2003 Robert Plank

```
$contents =
"R0lGODlhAQABAIAAAAAAAAAAACH5BAEAAAAALAAAAAABAAEA
AAICRAEAOw==";
echo base64_decode($contents);
die();
}

?>
```

If you've followed these directions well enough, you should be able to click on the JavaScript link, answer each question, and the message will be delivered to your inbox. Obviously this isn't an attempt to replace web-based forms by any means, but to me it is kind of handy if someone needs to send a quick comment, or some sort of reminder.

If you feel like going back to that JavaScript code of yours and making it just that much better, close the PHP script and open your HTML document that contains the JavaScript.

Find this line:
name = prompt("Your Name:","");

And try adding something like this on the line below:
if (!name || name == "" || name == null) { alert("No name given, action cancelled."); return false; }

Let's do the same for the lines where we gather data for the mail and message variables as well. Our final JavaScript code here should be:

```
<script language="JavaScript">
<!--

function NoteWire() {

var name = "";
var mail = "";
var message = "";
```

Copyright © 2003 Robert Plank

```
name = prompt("Your Name:","");
if (!name || name == "" || name == null) { alert("No name given, action
cancelled."); return false; }

mail = prompt("Your E-Mail Address:","");
if (!mail || mail == "" || mail == null) { alert("No e-mail address given, action
cancelled."); return false; }

message = prompt("Type your message:","");
if (!message || message == "" || message == null) { alert("No message given,
action cancelled."); return false; }

var myurl = "http://www.my.host/notewire.php";
trigger = new Image();
trigger.src =
myurl+"?name="+escape(name)+"&mail="+escape(mail)+"&message="+escape
(message);
trigger.onload = done;
}

function done() {
alert("We are done");
}
// -->
</script>
```

That done() function looks like a waste of a function. It would be nice if we could just have a line like:

```
trigger.onload = alert("We are done");
```

But we can't. It EXPECTS a function to be there. So we'd have to put that inside a Function object, like so:

```
trigger.onload = new Function(alert("We are done"));
```

Copyright © 2003 Robert Plank

This might be something you want to use on multiple pages. If that's the case, take all that JavaScript code and move everything INSIDE the <script> and // --> </script> tags into a new file. Call it something like "functions.js". So functions.js should look like this:

```
function NoteWire() {

var name = "";
var mail = "";
var message = "";

name = prompt("Your Name:","");
if (!name || name == "" || name == null) { alert("No name given, action
cancelled."); return false; }

mail = prompt("Your E-Mail Address:","");
if (!mail || mail == "" || mail == null) { alert("No e-mail address given, action
cancelled."); return false; }

message = prompt("Type your message:","");
if (!message || message == "" || message == null) { alert("No message given,
action cancelled."); return false; }

var myurl = "http://www.my.host/notewire.php";
trigger = new Image();
trigger.src =
myurl+"?name="+escape(name)+"&mail="+escape(mail)+"&message="+escape
(message);
trigger.onload = new Function(alert("We are done"));
}
```

Then, if we want to use this code on any HTML page, have this there:

```
<script language="JavaScript" src="functions.js"></script>
```

That's it. You've just made a sort-of useful script that let's your visitors send you quick messages. The user stays on that page, there's no redirect, your real e-mail

Copyright © 2003 Robert Plank

address isn't exposed, they don't have to open their e-mail client for the message to leave their Outbox and actually be sent... it mostly happens on your server.

Now we can think about how a HumanClick-type script could be made... just have a loop that reloads an Image object in JavaScript every few seconds. If there's a new message for the visitor, the PHP script on the receiving end shows the image, and if not, it doesn't. When there IS an image, have JavaScript refresh to a pop up window where the messages will be contained.

I'm sure if you implemented something like this on a site you could think up some other uses. For example, let's say you provided some sort of service. Any service, it doesn't matter what. And only took you a minute or so to provide a quote for that service.

If you have a way of having it known on your site that you were online, you could say something like "Your request for a quote has been sent. I will contact you via e-mail. If you really want to be notified IMMEDIATELY, leave your browser open and a popup box will appear with your quote." Some people might leave the browser open and would stay at your site longer. If I were to do this I'd definitely assign a cookie to the user and then maybe every 10-15 seconds or so have that image try to reload to see if the quote request for that particular user has been fulfilled.

Of course that's just one idea. Be sure to send feedback on the chapter, thanks!

Assignment:

We used the variable $REMOTE_ADDR this week to figure out a user's IP address. Try playing with the variable $HTTP_USER_AGENT and see what this variable represents.

Copyright © 2003 Robert Plank

QUIZ

1. True or false: The header() function can be used to tell the browser to expect different types of files (HTML, images, sound, etc.)
A: True.
B: False.

2. True or false: PHP scripts can act as images.
A: True.
B: False.

3. True or false: It's possible to capture a visitor's IP address using PHP.
A: True.
B: False.

4. We saw last week what the "glue" is that can connect strings in JAVASCRIPT, so what is it in PHP?
A: The slash.
B: The comma.
C: The ampersand.
D: The period.

5. True or false: JavaScript's escape() and unescape() functions, used for making text "URL-safe", works the same as which two functions in PHP:
A: escape() and unescape()
B: urlescape() and urlunescape()
C: urlencode() and urldecode()
D: encode() and decode()

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 7 Basic Arrays and PHP

If you've read any of the previous chapters of the book (which you should have), you'd know already know that variables are ways of storing information. The simplest way to describe a variable is just that it's a piece of paper. You have a piece of paper, with a name, say, "brown", and you write the number "5" on it.

```
<?php

$brown = "5";

?>
```

Obviously you write the "5" down with a pencil so you can always erase it and change it to, say, "6" ... or even made it a text value like "bear":

```
<?php

$brown = "bear";

?>
```

But of course, you should know this already. You can also do things like set the value of a variable to an integer instead of a string, but since PHP is a really forgiving language it's not too important that you know about that yet. What you will be learning about, are arrays.

So what's an array? If you take a lot of those pieces of paper, and you stuff them all into a manila folder, that folder is your array. An array is just an organized set of variables.

Copyright © 2003 Robert Plank

Arrays make it really easy and structured when it comes to more complicated tasks. For example, back in high school I had a C++ class where one of the assignments was a Battleship game. If you've ever played the game in real life, you'd know it involves two boards made up of 9x9 squares.

This gives us 81 squares per board, and since there's two boards that's 162 spaces to deal with! Could you imagine trying to juggle 162 free-floating variables? I sure couldn't. But arrays make it easy. Here's a simple array:

```
$bear[0] = "grizzly";
$bear[1] = "panda";
$bear[2] = "polar";
$bear[3] = "sloth";
$bear[4] = "pooh";
$bear[5] = "paddington";
```

This is what we call an indexed array. Note that we start at ZERO, not one. This may seem confusing, but it's a lot easier doing it that way in the long-run.

What we have is the equivalent to a manila folder called "bear" with six pieces of paper inside. The paper labeled "1" has the word "panda" written on it. The next piece of paper, "2", has "polar" written on it, and so on.

What's the use? Well, if I have something like:

```
echo "$bear[4] bear";
```

It will give me:

pooh bear

That's because we first asked for element number 4 in the array called "bear", and then added a space followed by "bear". We could do the same for any of the other elements in this array and say "paddington bear", "polar bear", etc.

So what, then? Well, it's time for a fun little link rotator.

Copyright © 2003 Robert Plank

Start over in a new text file. I want to start off by making an array, of say, five elements (meaning we go from 0-4) of an array called "links". Each element in the array will contain one of the possible URLs to send our visitor.

The ultimate goal of this script is to choose a random number, choose that element in the array, and redirect our visitor there.

Here's where we start our script:

```
<?php

$links[0] = "http://www.google.com";
$links[1] = "http://www.amazon.com";
$links[2] = "http://www.ebay.com";
$links[3] = "http://www.php.net";
$links[4] = "http://www.theezine.net";

?>
```

What we have now are the five links we're going to use as random links. It would be nice to be able to just get a random number between 0 and 4, and plug it into the array like we did with our "pooh bear" example. Oh, but wait, there is!

Add this to your script (still within the "?>", of course)...

```
echo rand(1,10);
```

Upload this to your web host and run it in your browser. (I don't care what you call it; something like rand.php is fine.)

You should see a random number between 1 and 10. Try refreshing the page. If the number doesn't change, we're screwed because you're using a slightly out-of-date version of PHP!

Not to worry, however... there's always a solution. And that's this line:

Copyright © 2003 Robert Plank

```
srand((double)microtime()*1000000);
```

Add the above line near the top of your script. You don't need to know what this is about... this is just important so that it gives a different random number each time.

All I ask of you now... please please PLEASE don't use srand() more than once. You only need to seed ONCE in your script, and if you do it more than once you could really slow things down.

Our script so far:

```
<?php

srand((double)microtime()*1000000);

$links[0] = "http://www.google.com";
$links[1] = "http://www.amazon.com";
$links[2] = "http://www.ebay.com";
$links[3] = "http://www.php.net";
$links[4] = "http://www.theezine.net";

echo rand(1,10);

?>
```

So all we have to do is take what rand(1,10) gives us and put it into the array, right? No, we can't do that because there are 5 elements in the array, not 10... so what, do we change the 10 to a 5? Well, that's a quick fix but what if we had 20 or 50 or 100 links here?

That's where the count() function comes in. Delete that line that has the rand(1,10) stuff in there and throw this line in there instead:

```
echo count($links);
```

Our modified script:

Copyright © 2003 Robert Plank

```
<?php

srand((double)microtime()*1000000);

$links[0] = "http://www.google.com";
$links[1] = "http://www.amazon.com";
$links[2] = "http://www.ebay.com";
$links[3] = "http://www.php.net";
$links[4] = "http://www.theezine.net";

echo count($links);

?>
```

This gives us the number 5, each time. Because there are one, two, three, four, five elements in that array. If you add more lines into that array (eg, $links[5], $links[6] and so on...) this number will increase. So, that's how we can figure out the size of an array.

So, just combine the rand(1,10) and count($links)... we have:

```
echo rand(1,count($links));
```

This gives us a random number between 1 and 5... wait, didn't we want it from 0 to 4? Well, no problem then... we just take that random number and subtract it by one:

```
echo rand(1,count($links)) - 1;
```

At this point what I'd do is put this random value into a variable, let's call it $result. This line changes to:

```
$result = rand(1,count($links)) - 1;
```

Then just plug this into $links:

Copyright © 2003 Robert Plank

```
echo $links[$result];
```

Our efforts so far:

```
<?php

srand((double)microtime()*1000000);

$links[0] = "http://www.google.com";
$links[1] = "http://www.amazon.com";
$links[2] = "http://www.ebay.com";
$links[3] = "http://www.php.net";
$links[4] = "http://www.theezine.net";

$result = rand(1,count($links)) - 1;
echo $links[$result];

?>
```

Upload this and try it. Oops, this just gives us the TEXT of our random URL. We want to redirect, don't we? Instead of echoing this result, we'll put it into an HTTP Location header.

```
<?php

srand((double)microtime()*1000000);

$links[0] = "http://www.google.com";
$links[1] = "http://www.amazon.com";
$links[2] = "http://www.ebay.com";
$links[3] = "http://www.php.net";
$links[4] = "http://www.theezine.net";

$result = rand(1,count($links)) - 1;
header("Location:".$links[$result]);
die();
```

Copyright © 2003 Robert Plank

```
?>
```

Do you see what we did? We added "Location:" to the front, so for example for "http://www.google.com" we'd get "Location:http://www.google.com". Putting this inside the function called header() makes sure this is stuffed into the HTTP headers that are sent.

Bye.

Assignment:

Look up the documentation on arrays at PHP.net and figure out one other way of putting text into an array. (Hint: If you're still stuck, take a peek at chapter 11.)

QUIZ

1. In PHP, an item stored in an array (eg, $myarray[2]) is usually called a(n):

A: element
B: hash
C: sign post
D: thingy

2. When generating random variables, srand() should be used...

A: Never, in the script.
B: Only once in the script.
C: Each time you need a new random number.
D: As many times as possible.

3. True or false: It's possible to take a variable, stick a value into it, and then plug that variable into an array (like $myarray[$myvariable]).

Copyright © 2003 Robert Plank

A: True.
B: False.

4. Indexed arrays start counting at...

A: NULL.
B: Zero.
C: One.
D: Ten.

5. True or false: Elements in an array act the same as variables.
A: True.
B: False.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 8 File Handling With PHP

Read the previous chapter. You have to. Now I continue.

We've already played around with a link rotator. That's nice if you have a few links... but what if you have a huge text file full of links? Or a random tip to rotate? Or random article or quote?

You can make a simple rotator, of any kind, that reads off a text file, in just a few lines of code. That's what I really like about PHP; it has a ton of functions (with names that are actually useful for a change) that do the work for you that you don't find built-in to most popular languages.

Here's what we made last week:

```
<?php

srand((double)microtime()*1000000);

$links[0] = "http://www.google.com";
$links[1] = "http://www.amazon.com";
$links[2] = "http://www.ebay.com";
$links[3] = "http://www.php.net";
$links[4] = "http://www.theezine.net";

$result = rand(1,count($links)) - 1;
header("Location:".$links[$result]);
die();

?>
```

As you recall, it looks at the array, counts it, chooses a random element in it, then

Copyright © 2003 Robert Plank

plugs it back into the array and pukes out that element. Simple.

Even simpler. There's a function built into PHP that reads a text file and puts the entire file into an array. Each line represents one element of the array.

Take a text file and do this:

hey i am line one
and i am line two
who am I?
you are line three, silly

This will be your database. Save it as, I don't know... "array.txt" and then open a new file. This will be your script.

```
header("Content-type:text/plain");
$myarray = file("array.txt");
print_r($myarray);
```

Save this as array.php, for example. Upload both and run array.php. If you've done this right, you'll see this:

```
Array
(
   [0] => hey i am line one

   [1] => and i am line two

   [2] => who am I?

   [3] => you are line three, silly
)
```

And what's this? All we're doing is reading that text file into an array using the file() function and storing it in $myarray. Then we display $myarray using the function print_r();

Copyright © 2003 Robert Plank

print_r() means "print readable" and it takes things that we normally wouldn't be able to see regularly, like arrays, and make them nice and pretty.

That header("Content-type:text/plain"); I threw in there just makes it so that we can actually see what's going on.

All that's left now is to do a little borrowing from last week's script to add:

```
$result = rand(1,count($myarray)) - 1;
echo $myarray[$result];
```

And that's it. Here's everything:

```
$myarray = file("array.txt");
$result = rand(1,count($myarray)) - 1;
echo $myarray[$result];
```

We did all that in just three lines of code!

Well, if you don't mind, I want to add a fourth. One thing I don't like too much about the file() function, is that it leaves the line feeds in. If you noticed the pretty picture print_r() drew for us, those lines were double spaced. Well, they aren't supposed to be double-spaced, but because there's a line feed at the end of each element, it looks that way.

Change to this:

```
$myarray = file("array.txt");
for ($i=0;$i<count($myarray);$i++) { $myarray[$i] = trim($myarray[$i]); }
$result = rand(1,count($myarray)) - 1;
echo $myarray[$result];
```

You'll see I've added in that second line. What's happening there, is the stuff at both ends are being chopped off. Theoretically if you ran print_r again on that array you'd see this:

Copyright © 2003 Robert Plank

```
Array
(
   [0] => hey i am line one
   [1] => and i am line two
   [2] => who am I?
   [3] => you are line three, silly
)
```

Which is what we want.

Hmm... now what if the quotes we want to rotate are more than one line? That's a little trickier, but not too hard.

Start over in a new text file. This will be your script. I'm assuming you already have a text file filled with your favorite multi-line quotes. If you don't, here's a sample list of quotes I stole:

```
"Computers in the future may weigh no more than 1.5 tons."
- Popular Mechanics, 1949
%%
"I think there is a world market for maybe five computers."
- Thomas Watson (chairman of IBM), 1943
%%
"640K ought to be enough for anybody."
- Bill Gates, 1981
%%
"There is no reason anyone would want a computer in their home."
- Ken Olson, president, chairman and founder of Digital Equipment Corp., 1977
```

Paste that stuff into a file called, quotes.txt. "Now wait just a second there, partner", you say. "What are those percent signs for?" Those double percent signs are what we use to separate each quote with. Before this, we separated each quote with a new line, but now PHP needs to know how to distinguish each quote.

First I would start off by stating what our separator is, and defining what our

Copyright © 2003 Robert Plank

quote file name is.

```
<?php

$separator = "%%";
$quotefile = "quotes.txt";

?>
```

Now, in order to get our random quote, I see us taking four steps:

1. Read the contents of the entire file
2. Split up the file (according to our separators) into an array
3. Choosing a random number in the array
4. Plug that random number back into the array

Once you get the hang of this, you'll tend to combine steps three and four. Let's stick with four steps for now, though.

At this point you really don't know how to read whole files into a string. First, we have to open the file for reading using the fopen() function.

```
$fp = fopen($quotefile,"r");
```

$fp is what we call our file pointer. Once we've opened a file, when we want to do something to it, we'll do the things we want to do to this variable instead of to the file directly. Remember though that $fp could have just as easily been $handle or $garbage or whatever else we wanted to call it.

The second parameter for fopen(), "r", tells PHP that we want to open the file for reading only. If we want to write or append to the file we'd have to change this. But luckily, today we're only reading this file.

We defined the value of $quotefile above. It's the file "quotes.txt" if you've forgotten.

```
$mytext = fread($fp,filesize($quotefile));
```

Copyright © 2003 Robert Plank

fread(), as you might have guessed, reads a file. The first parameter is our file pointer, $fp, while the second parameter is the length we want to read.

Since we want to read the whole, we use the filesize() function to get the length, in bytes, of the file we want to read. And then just toss that into the second parameter.

We now have everything in quotes.txt stored in the variable called $mytext. We won't be needing to play with quotes.txt anymore, so we can just close it:

```
fclose($fp);
```

If you're following along, we just finished step one. That was probably the hardest one.

Step two is really easy. There's another friendly function that will take a string and chop it up into pieces of an array based on a separator we give it.

```
$mytext = explode($separator,$mytext);
```

What I've just done is taken the string $mytext, split its pieces into an array, and put that array back into $mytext. So now, $mytext has sort of been promoted from a plain old string into an array.

That was step two.

And we know how to choose a random element in an array, don't we?

```
$result = rand(1,count($mytext)) - 1;
```

That was step three.

Then, plug that back into an array so that we really have a random element from the array.

```
echo $mytext[$result];
```

Copyright © 2003 Robert Plank

That was step four. Done.

One thing to note, is that if you're taking your quotes out of a text file and you're displaying them on an HTML page, things like line feeds won't show up as the HTML markup you're looking for.

So we just make use of yet another handy PHP function, nl2br(), which takes those line feeds and puts
 tags in their place.

Here's what we've got:

```
<?php

$separator = "%%";
$quotefile = "quotes.txt";

// Read the file into the variable $mytext
$fp = fopen($quotefile,"r");
$mytext = fread($fp,filesize($quotefile));
fclose($fp);

// Display a random quote
$mytext = explode($separator,$mytext);
$result = rand(1,count($mytext)) - 1;
echo nl2br($mytext[$result]);

?>
```

There's your simple random quote script. Want some more useful stuff? No? Ok, I'll go ahead anyway... ;-)

Let's pretend that you had a file full of quotes, or tips, or recipes, or something, and you wanted them sorted. It's a good thing that PHP has a sort function.

In this chapter, I can show you how to take that "one line per quote" file and sort it, then write it back into a file.

Copyright © 2003 Robert Plank

This is what we had...

```
<?php

$myarray = file("array.txt");
$result = rand(1,count($myarray)) - 1;
echo $myarray[$result];

?>
```

We'll have to take out those last two lines. We are no longer interested in any one element of our array. Our task now, instead of showing one output to the user, is to write this all back into a new, sorted file.

```
<?php

$myarray = file("array.txt");
sort($myarray);
$myarray = implode("",$myarray);

?>
```

WHOA! What just happened there? Well, first we took the array and sorted it alphabetically using the sort() function. Then we used implode(), which does the exact opposite of explode.

Instead of splitting a string up into an array, we took each piece of an array and glued it together into one big long string. Now, to write everything we've got. First I'll define my output file:

```
$outfile = "output.txt";
```

And then, open that file for writing, take our finished string and write to it, then close the file.

```
$fp = fopen($outfile,"w");
```

Copyright © 2003 Robert Plank

```
fwrite($fp,$myarray);
fclose($fp);
```

If you're doing any of this on a Unix machine be advised that it won't work unless the proper permissions are set. Meaning you need to do one of two things:

Either, the folder you plan to write output.txt needs to be chmoded to 0777, or... the file output.txt needs to be created already and be chmoded to 0777.

Don't know what chmoding is? Chmod lets you change the permissions of a file. 0777 is fully writeable, which we want because PHP has to be able to write to this file. Setting permissions changes depending on your FTP client.

I use FlashFXP, and in that you browse to the file you want to set permissions to, right click and choose "Attributes (CHMOD)" check all the boxes until the numbered total at the bottom reads "0777", and click OK.

What's the last thing we need to do. Oh. Sorting multi-line quote files. Well, you take the same approach, and I won't bore you by going through the whole process again, but here is the completed code:

```
<?php

$separator = "%%";
$quotefile = "quotes.txt";

// Read the file into the variable $mytext
$fp = fopen($quotefile,"r");
$mytext = fread($fp,filesize($quotefile));
fclose($fp);

// Explode, sort, and implode
$mytext = explode($separator,$mytext);
sort($mytext);
$mytext = implode($separator,$mytext);
```

Copyright © 2003 Robert Plank

```
// Write to the file
$fp = fopen($quotefile,"w");
fwrite($fp,$mytext);
fclose($fp);

?>
```

Assignment

Use the rand() function to simulate the rolling of two dice. (Hint: It's *NOT* a random number from 1 to 12! It's the sum of two different random numbers each from 1 to 6...)

QUIZ

1.The file() function...
A: Tastes like brownies.
B: Writes whatever you want, into a file.
C: Reads a file into an array separated by each line.
D: Reads a file into a variable.

2. print_r() means...
A: print readable
B: print recursive
C: print reverse
D: print red

3. In article 8, I used this code:
for ($i=0;$i<count($myarray);$i++)

What does it do?

Copyright © 2003 Robert Plank

A: It looks kind of neat but doesn't have a purpose.
B: It loops through each element in the array $myarray
C: It breaks apart the array $myarray, one element at a time.
D: Makes 12 ounces of black cherry flavored Jell-O gelatin.

4. What function did we use to break apart an array?
A: break()
B: separate()
C: chunk()
D: explode()

5. What does nl2br() do?
A: Puts a
 tag at the end of each line so a file is readable in HTML.
B: Removes a
 tag at the end of each line so a file isn't readable in HTML.
C: That isn't a function in PHP.
D: Takes all citizens of the Netherlands and also makes them citizens of Brazil.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 9 Anything of the Day With PHP

For the last few chapters we've been covering arrays. I bet you're sick of them by now. I know I'm sick of writing about them.

Actually, that's a lie. Arrays are so cool to talk about, because from that topic we can dip into a ton of different topics. Like the "Anything of the Day" script you and I will be building, together. Right now.

By "Anything of the Day" I mean quote of the day, tip of the day, this day in history... those sorts of things people like to put on their web site that changes daily, without them having to do a thing. *However* the thing about these was that during the course of that day, our item can't change.

This will be slightly more involved than our random quote scripts. But that's okay.

A good place as any to start is with a sample quotes file. I've provided one below:

"Illegitimacy is something we should talk about in terms of not having it."
- Al Gore
%%
"I bet a fun thing would be to go way back in time to where there was going to be an eclipse and tell the cave men, 'If I have come to destroy you, may the sun be blotted out from the sky.' Just then the eclipse would start, and they'd probably try to kill you or something, but then you could explain about the rotation of the moon and all, and everyone would get a good laugh."
- Jack Handey
%%
"The internet is a great way to get on the net."

Copyright © 2003 Robert Plank

- Bob Dole
%%
"Rarely is the question asked: Is our children learning?"
- George W. Bush
%%
"It isn't pollution that's harming the environment. It's the impurities in our air and water that are doing it."
- Al Gore

Save this text into a file called "quotes.txt". We start again by reusing old code:

```
<?php

$separator = "%%";
$quotefile = "quotes.txt";

// Read the file into the variable $mytext
$fp = fopen($quotefile,"r");
$mytext = fread($fp,filesize($quotefile));
fclose($fp);

?>
```

The first step of course is to split the string into an array using our variable called $separator:

```
$mytext = explode($separator);
```

If you've been paying attention to previous weeks' chapters you'd know that we've exploded the string $mytext and placed it back into $mytext, "promoting" $mytext into an array containing each quote.

But now the problem remains, how to have a quote stay constant throughout the day. Without the luxury of storing any sort of information.

We'll start by experimenting with the mktime() function. This function gives us the date in a large number... well over one billion. It's actually the number of

Copyright © 2003 Robert Plank

seconds that have elapsed since January 1, 1970 at midnight of your server's time zone.

This is important to us because it gives us a simple way of manipulating the date. Let's try experimenting with by adding this to your script:

```
$date = mktime();
echo $date;
```

Upload it and run it. You'll see that it gives you that large number. Refresh and you'll see that the number increases slightly. That's because the seconds are ticking away.

We need a way of getting the number of *days* that have elapsed since this date. So, change the above to:

```
$date = mktime() / 86400;
echo $date;
```

What have we done, exactly? Well, there's 86,400 seconds in one day. 60 seconds times 60 minutes times 24 hours. That result is the number of days since 1/1/1970. Oh, but wait! It's a decimal number now!

And then we use another handy PHP function. floor() is a math function that rounds a number down to the nearest integer. We place that around that other stuff.

```
$date = floor(mktime() / 86400);
echo $date;
```

Try this one. You get a number. Refresh and it stays the same. This represents the number of *whole* days that have elapsed. Which gives us a number that does not change throughout the day. It only increases when the next day arrives.

The question is, how do we apply this still slightly monstrous number, to our tiny quotes file of only five entries? Modulus.

Copyright © 2003 Robert Plank

If you don't know what modulus is, it's the fancy way of saying the remainder of two numbers. For example, 7 modulo 3 equals 1.

For us, modulus is a way of guaranteeing that no matter what the value of $date is, we'll have a way of representing that number evenly inside any size array. Our array has five elements so the numbers we want to land inside are 0 through 4. If we progress through each day and apply it to modulo 5...

Day 12098 % 5 = 3
Day 12099 % 5 = 4
Day 12100 % 5 = 0
Day 12101 % 5 = 1
Day 12102 % 5 = 2
Day 12103 % 5 = 3
Day 12104 % 5 = 4
Day 12105 % 5 = 0
Day 12106 % 5 = 1

Get it? No matter what day it is, as day increases by one each time, we will still see the sequence 0, 1, 2, 3, 4. If we take the modulus of 2 we'd see "0, 1, 0, 1, etc.". If we take the modulus of 12 we'd see our number go from 0 to 11, start over, go from 0 to 11, ad nauseum.

Remove the line that gives us echo $date; and stick this in:

```
$result = ($date % count($mytext));
```

At this point we have:

```
<?php

$separator = "%%";
$quotefile = "quotes.txt";

$fp = fopen($quotefile,"r");
$mytext = fread($fp,filesize($quotefile));
fclose($fp);
```

Copyright © 2003 Robert Plank

```
$mytext = explode($separator);

$date = floor(mktime() / 86400);
$result = ($date % count($mytext));

?>
```

Do you see what the next step is? I sure hope so.

```
echo nl2br($mytext[$result]);
```

Make sense? We split the string up into an array, get the number of seconds since 1970, break that down into days, then divide that into the size of our array so all we're left with remainders ... numbers ALWAYS within the range of the array.

Final code:

```
<?php

$separator = "%%";
$quotefile = "quotes.txt";

$fp = fopen($quotefile,"r");
$mytext = fread($fp,filesize($quotefile));
fclose($fp);

$mytext = explode($separator);

$date = floor(mktime() / 86400);
$result = ($date % count($mytext));

echo nl2br($mytext[$result]);

?>
```

Copyright © 2003 Robert Plank

One drawback is that it's very unlikely you'd start at 0. If you don't care, fine. But the point of most quotes/sites/tips of the day is to load a bunch of quotes in and just forget about it.

What we need is a way of telling PHP when Day One was. Then, before we calculated the number of days since 1970, we would subtract the current date with the start date. That way, we'd be figuring out the number of days since we first started.

A local solution for this would be the filemtime() function. This function figures out the time a file was last saved (by saved, I mean the last time you hit the "Save" button, *not* the time and date it was uploaded to your host).

So near the top (just before all the date stuff) let's have this:
```
$startdate = filemtime($quotefile);
```

Then modify this line:
```
$date = floor(mktime() / 86400);
```

And make it do this:
```
$date = floor((mktime() - $startdate) / 86400);
```

Of course, this doesn't always work... because what if you make a slight change or add in more quotes? It'll start right back at zero. So, what I suggest is, where you have $startdate, actually set the date at which the quotes begin.

This is what we change it to:

```
$startdate = strtotime("February 15, 2003");
```

The function strtotime() takes a date in human readable format, anything like "2-15-2003" or "February 15, 2003" and converts it into the Unix time we can do math with.

All that's holding you back now from adding a quote, or site, or tip, or whatever of-the-day to your site is a way to include it in...

Copyright © 2003 Robert Plank

Using PHP: <?php include("path/to/your-script.php"); ?>
Using SSI: <!--#include virtual="path/to/your-script.php?$QUERY_STRING_UNESCAPED" -->

And lastly, there's one more thing I want to do with arrays. That's a recommend type of script. You've seen these before. There's a form with a place to enter in your name, your e-mail address, and then several boxes to add in "friends" with which to share the site with. Here's some example HTML for you:

```
<form action="recommend.php" method="post">
<input type="hidden" name="redirect" value="thankyou.html">

Your Name: <input type="text" name="myname" size="30"><br>
Your E-Mail: <input type="text" name="mymail" size="30"><br>

Friend 1 E-Mail: <input type="text" name="friend[]" size="30"><br>
Friend 2 E-Mail: <input type="text" name="friend[]" size="30"><br>
Friend 3 E-Mail: <input type="text" name="friend[]" size="30"><br>
Friend 4 E-Mail: <input type="text" name="friend[]" size="30"><br>
Friend 5 E-Mail: <input type="text" name="friend[]" size="30">
</form>
```

When this data gets passed into our script, everything will go as smoothly as you're used to, except for the variable $friend. You've never seen form fields used like this before... not to worry though. The first address someone enters will be placed into $friend[0], the next into $friend[1], etc.

I'm looking back onto Week 3 to steal an snippet of code:

```
<?php

$myname = "Your Name Here";
$mymail = "your@email.here";

$subject = "Hello";
$body = "Hi. This is the body of my message.
```

Copyright © 2003 Robert Plank

```
Notice how I can continue typing right on the next line!";

$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

if ($email != "") { mail($email, $subject, $body, $headers); }

?>
```

I've named our variables so that most of the work is already done. Remember this line from last week?

```
for ($i=0;$i<count($myarray);$i++) { $myarray[$i] = trim($myarray[$i]); }
```

This loops through the entire array called $myarray. That's what we'll do here. But this time, we'll be looping through the array called $friend instead, and we'll be sending an e-mail each time.

Take out this line:

```
if ($email != "") { mail($email, $subject, $body, $headers); }
```

And put this in:

```
for ($i=0;$i<$email;$i++) { mail($email[$i], $subject, $body, $headers); }
```

Obviously the values of $myname and $mymail will be given to us by the form, so we can remove the lines that give them values.

Change the value of $subject and $body of course, and you are finished.

```
<?php

$subject = "Hey I Like This Site";
```

Copyright © 2003 Robert Plank

```
$body = "Hey I like this site, it's at www.crap.junk, hurry go there now";

$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

for ($i=0;$i<$email;$i++) { mail($email[$i], $subject, $body, $headers); }

?>
```

Have fun playing with those scripts.

Assignment

Look up the strtotime() function on PHP.net and check out the other time formats you can put into the function (ie, "Now" or "+1 week")

QUIZ

1. What does mktime() do, anyway?
 A: Returns the number of seconds since January 1, 1970.
 B: Returns the number of minutes since January 1, 1970.
 C: Returns the number of microseconds since January 1, 1970.
 D: Returns the number of microminutes since January 1, 1970.

2. What does floor() do?
 A: Just rounds, either up or down.
 B: Always rounds up.
 C: Always rounds down.
 D: Make sure a number is never rounded.

3. What does a hidden input type do?

Copyright © 2003 Robert Plank

A: Allows us to pass data from an HTML document into a PHP script, without that data being displayed to the user.
B: Gives everyone a chance to hide away secret wallets, keys, messages, documents, etc.
C: Makes the data passed into a PHP script invisible to that script.
D: It doesn't do anything, it's useless.

4. What is modulus (%)?
A: A trigonometric function similar to sine or cosine.
B: The remainder of some number.
C: A Roman emperor.
D: There's no such thing.

5. What does strtotime() do?
A: Calculates how long a particular string is.
B: Figures out how much CPU time it takes to process that particular string.
C: Takes any kind of string, even "ABCDE", and gives it a timestamp.
D: Takes a date or length of time written in plain English and converts it to Unix time.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 10 Affiliate Script With PHP

Affiliate scripts are surprisingly simple. Some of those with a ton of features like link tracking, payout management and all that stuff, sell for as much as $100. Here's a start on your own simple affiliate script.

If you haven't used an affiliate script before, basically what you do is you have a template page that contains variables such as the affiliate's name, affiliate's e-mail address, stuff like that. If you use something like PayPal then you might want to have something like their PayPal address, or if you're using Clickbank you might want their Clickbank member name.

On the other hand, maybe you're using a different payment processor or handling payments on your own and you just want to keep track of who made the sale for you.

Start with an HTML file like this:

Hello, I'm a sample HTML page.

Our affiliate is: <%firstname%> <%lastname%>

My e-mail address is: <%email%>

Here is where I talk about other stuff.
You can call me at <%phone%> during the day.

Maybe we'll have a PayPal button here with credit to <%payment%>...

Save this as sample.html.

Remember that this is just a *sample* page we're using to show how we can implement fields off a database. Each of our <%things%> are sort of like

Copyright © 2003 Robert Plank

variables, but they're variables that only apply to the current affiliate.

For example, let's say we had a list of 10 different affiliates. We had one, whose affiliate id was "mickey". His first name was "Mickey", his last name "Mouse", his e-mail address "mickey@my.host". His phone number is 817-555-1111. His payment id is 12345.

If someone came to our site and went to page.php?mickey, we'd want Mickey's info to appear, and not someone else's. Conversely, when a visitor went to page.php?donald, we would want Donald Duck's info to show and not Mickey's.

Since I want to really make this script reusable, I want to define each field in the first line of the text file. So we'll have something like this:

```
id|firstname|lastname|email|phone|payment
donald|Donald|Duck|donald@crappy.host|817-555-1023|dduck
someguy|Bugs|Bunny|aperson@another.host|817-555-6665|wellthen
mickey|Mickey|Mouse|mickey@my.host|817-555-1111|12345
hexagon|Goofy|Man|goof@my.host|817-555-1424|goofie
racecar|Minnie|Mouse|evergreens@crappy.host|817-555-5454
```

This is our database. Take this text and save it into a file called database.txt. All our data is separated by pipes (|). On the first line, I define which field is which.

So, looking at the user donald's entry, id equals "donald", firstname equals "Donald", lastname equals "Duck", email equals "donald@crappy.host", phone equals "817-555-1023", and payment equals "dduck". Does this make sense? It does to me.

In previous articles you've seen how to read a file into a string (the hard way) and also how to read files into an array (the easy way). This time we'll get to do things the easy way.

To start, define the filename of our database and read that file into an array:

```
<?php
```

Copyright © 2003 Robert Plank

```
$dbfile = "database.txt";
$contents = file($dbfile);

?>
```

Now, we loop through the array and take out those stupid line feeds at the ends:

```
for ($i=0;$i<count($contents);$i++) { $contents[$i] = trim($contents[$i]); }
```

At this point, $contents looks like this:

```
Array
(
   [0] => id|firstname|lastname|email|phone|payment
   [1] => donald|Donald|Duck|donald@crappy.host|817-555-1023|dduck
   [2] => someguy|Bugs|Bunny|aperson@another.host|817-555-6665|wellthen
   [3] => mickey|Mickey|Mouse|mickey@my.host|817-555-1111|12345
   [4] => hexagon|Goofy|Man|goof@my.host|817-555-1424|goofie
   [5] => racecar|Minnie|Mouse|evergreens@crappy.host|817-555-5454
)
```

Just to make things easier, let's assume that the affiliate we want is "donald". Of course we'll take this part out later but it really helps testing-wise.

Add this to the top of your script:

```
$id = "donald";
```

Oh, and one other thing I'd like to do is tell PHP what we're using as a separator (in this case, a pipe):

```
$sep = "|";
```

So far all we have is this:

```
<?php
```

Copyright © 2003 Robert Plank

```
$id = "donald";
$sep = "|";

$dbfile = "database.txt";
$contents = file($dbfile);

for ($i=0;$i<count($contents);$i++) { $contents[$i] = trim($contents[$i]); }

?>
```

And now, we mix in a little regular expressions. Nothing really complicated, just this:

```
for ($i=1;$i<count($contents);$i++) {
  if (eregi("^".$id."\\".$sep,$contents[$i])) { $loc = $i; break; }
}
```

What this does is, first of all, loop through the array. This time it starts at element *one*, not zero, because the top line is where we defined our field names.

When we loop through the array, we use the eregi() function, which we use to find things in a string. The carat (^) we see tells us that what we want has to be at the beginning. Not only that, but we need to our separator just after the ID we're looking for.

Note: the dot joins strings together. The dot in front of our separator "\\" is our way of telling PHP that we want a backslash put in JUST BEFORE the separator. This is important because we don't want our separator becoming confused with regular expression notation.

Once we've found what we're looking for, the entry which contains our ID, we tell PHP to set the value of $loc to the element number where we found what we're looking for. The break; command tells us to stop looping. Since we've already found what we're looking for, there's no use in continuing to search.

Now that we've figured out where the entry is, we need to do something with it.

Copyright © 2003 Robert Plank

Here's the next step:

```
$tmp = explode($sep,$contents[$loc]);
```

Understand this? We know where the item we want is, so we are taking that and then splitting it up into an array, our separator being the split point. This array is put into a new array called $tmp.

The story so far...

```
<?php

$id = "mickey";
$sep = "|";

// Read the database file into an array and trim

$dbfile = "database.txt";
$contents = file($dbfile);

for ($i=0;$i<count($contents);$i++) { $contents[$i] = trim($contents[$i]); }

// Search the database for the item we want and extract it

for ($i=1;$i<count($contents);$i++) {
  if (eregi("^".$id."\\".$sep,$contents[$i])) { $loc = $i; break; }
}

$tmp = explode($sep,$contents[$loc]);

?>
```

So, if we ran this, the array called $tmp would contain:

```
Array
(
   [0] => mickey
```

```
  [1] => Mickey
  [2] => Mouse
  [3] => mickey@my.host
  [4] => 817-555-1111
  [5] => 12345
)
```

Ouch, but we're working in field *names*, not numbers. Well, it's a good thing that the first line of the database file contained field names.

What we will do then is the same explode() trick with the separators we did on this line, but on the top line, like this:

```
$fields = explode($sep,$contents[0]);
```

Then simply loop through the array called $fields, look at each value, and then place each value into a new *associative* array called $data.

I don't think I've explained associative arrays yet. But they're really very simple. Instead of arranging elements by numbers, they're arranged by names. For example:

```
$crap["junk"] = "blah";
```

By the way, *don't* add the above line to your script. ;-)

Now, we go ahead and do what I just explained:

```
for ($i=0;$i<count($fields);$i++) {
  $data[($fields[$i])] = $tmp[$i];
}
```

Now, you're looking at that little line and thinking to yourself, "Eww, icky". Yes I know. Look at it a couple more times and you'll see that we're just plugging into the array.

If you could look at the array called $data, you'd see that it's completely worth it,

Copyright © 2003 Robert Plank

because:

```
Array
(
   [id] => mickey
   [firstname] => Mickey
   [lastname] => Mouse
   [email] => mickey@my.host
   [phone] => 817-555-1111
   [payment] => 12345
)
```

Look at that beautiful array. It's so easy to read, no crappy numbers. Now we can start worrying about replacing into our template. (You almost forgot about that thing, didn't you?)

So, how would we go about doing that? Remember the eregi() function? Lucky for us, there's a similar function called eregi_replace().

Start off by telling the script what the name of our template file is:

```
$template = "sample.html";
```

Then open and read the file (yes, the icky hard way). Don't forget to close the file since we're done with it after this.

```
$fp = fopen($template, "r");
$text = fread($fp,filesize($template));
fclose($fp);
```

Now all that's left is to loop through our shiny new array and replace each element. Easy, except... oh no, we're using associative arrays now. Gone are the days of those cozy for() loops. Well don't worry. I've got something even simpler for you to use now.

That's the foreach() function. Use it like this:

Copyright © 2003 Robert Plank

```
foreach($data as $key => $value) {
  $text = eregi_replace("<%".$key."%>",$value,$text);
}
```

What we're doing here is looping through the array, in each step resetting the value of the variable $key to the current element's name. And of course setting the value of the variable called $value to the value of the element we're currently at.

Also, since our variables in the template are in this format:

<%shit%>

We surround each element name with a <% and %>, as you see. Each replacement is applied back into the variable $text.

Now all that's left is to:

```
echo $text;
```

Our script:

```
<?php

$id = "mickey";
$sep = "|";

// Read the database file into an array and trim

$dbfile = "database.txt";
$template = "sample.html";

$contents = file($dbfile);
for ($i=0;$i<count($contents);$i++) { $contents[$i] = trim($contents[$i]); }

// Search the database for the item we want and extract it
```

Copyright © 2003 Robert Plank

```
for ($i=1;$i<count($contents);$i++) {
  if (eregi("^".$id."\\".$sep,$contents[$i])) { $loc = $i; break; }
}

$tmp = explode($sep,$contents[$loc]);
$fields = explode($sep,$contents[0]);

for ($i=0;$i<count($fields);$i++) {
   $data[($fields[$i])] = $tmp[$i];
}

// Apply our results to the template

$fp = fopen($template, "r");
$text = fread($fp,filesize($template));
fclose($fp);

foreach($data as $key => $value) {
   $text = eregi_replace("<%".$key."%>",$value,$text);
}

echo $text;

?>
```

Not entirely done yet. We need to be able to get the ID from the outside. So, we remove the line that says:

```
$id = "mickey";
```

This way, all we have to do is call the script as "page.php?id=mickey", for example, if we want "mickey's" affiliate page. But, maybe you want this script to be more in style of most affiliate scripts, which is in the form of page.php?mickey. That's easy.

You know up near the top where you deleted that line? Throw this on that spot:

Copyright © 2003 Robert Plank

```
$id = $QUERY_STRING;
```

And now for our finished code:

```
<?php

$id = $QUERY_STRING;
$sep = "|";

// Read the database file into an array and trim

$dbfile = "database.txt";
$template = "sample.html";

$contents = file($dbfile);
for ($i=0;$i<count($contents);$i++) { $contents[$i] = trim($contents[$i]); }

// Search the database for the item we want and extract it

for ($i=1;$i<count($contents);$i++) {
  if (eregi("^".$id."\\".$sep,$contents[$i])) { $loc = $i; break; }
}

$tmp = explode($sep,$contents[$loc]);
$fields = explode($sep,$contents[0]);

for ($i=0;$i<count($fields);$i++) {
   $data[($fields[$i])] = $tmp[$i];
}

// Apply our results to the template

$fp = fopen($template, "r");
$text = fread($fp,filesize($template));
fclose($fp);

foreach($data as $key => $value) {
```

Copyright © 2003 Robert Plank

```
  $text = eregi_replace("<%".$key."%>",$value,$text);
}

echo $text;

?>
```

Assignment:

If you're feeling really ambitious, look up the documentation on fwrite() and fread()... keep in mind the "a+" mode, which appends to a file.

Now, create an HTML signup form where an affiliate might signup. Take these results and write them to a text file in the same format your data file for this week is currently in.

You've just made an affiliate signup script!

QUIZ

1. The difference between indexed arrays and associative arrays is that...
A: Indexed arrays can contain special characters.
B: Indexed arrays don't always work properly in PHP.
C: Indexed arrays contain numbered elements (0, 1, 2, etc.) while associative arrays contain non-numbered elements (like "toast", "bacon", etc.)
D: There is no difference between the two.

2. The difference between ereg() and ereg_replace() is...
A: ereg() only finds a match, ereg_replace() makes a replacement at that match.
B: ereg() finds matches more frequently than ereg_replace().
C: ereg() is case sensitive.

Copyright © 2003 Robert Plank

D: There is no difference between ereg() and ereg() replace.

3. What is the difference between ereg() and eregi()? (Look it up on php.net if you're not sure.)
A: eregi() doesn't care if a matching pattern is uppercase or lowercase.
B: ereg() stops at the first match, but eregi() continues on to the rest.
C: eregi() does not work with strings.
D: There is no difference between ereg() and eregi().

4. foreach() as we used it in article 10 is best for looping through...
A: Indexed arrays.
B: Associative arrays.
C: All of the above.
D: None of the above.

5. The variable $QUERY_STRING gives us...
A: Everything to the right of the question mark in the URL (eg, script.php?junk)
B: Everything to the left of the question mark.
C: Nothing; it's just some other variable.
D: A list of every string used in our script.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 11 Our Knowledge With PHP

If you've been paying attention to the things we've learned so far, you'd know that there are a lot of new cool things you can make just by applying the information in these first ten chapters.

Start off with one of those dropdown box URL dropdown boxes...tasty. These are where you have one of those dropdown menus where a visitor chooses a location and is promptly redirected to a new URL based on their selection.

We've already seen how you can post HTML form fields into a script. We have also seen from the article with the link rotator example, how to redirect someone to a particular URL. All we need to know is this, along with some simple HTML.

Start by creating the form:

```
<form action="jump.php" method="post">
<select name="url">
<option value="http://www.google.com">Google</option>
<option value="http://www.yahoo.com">Another link</option>
<option value="http://www.altavista.com">Yet another link</option>
</select>
<input type="submit" value="Go"></input>
</form>
```

Save this as form.html.

This HTML submits our choice to the script jump.php as a variable called $url. The value of this variable is the same as that within the value parameter. For example, the value of "Google" (the first choice) is http://www.google.com.

Copyright © 2003 Robert Plank

This will be your jump.php:

```
<?php

header("Location:$url"); die();

?>
```

As you've seen from our earlier examples, this redirects our visitor to the value of $url. Since the value of $url *is* the URL we want to redirect to, that's all we need in the entire script.

That's all you need in your script. Now, we need to make one last change to form.html.

See this line?

<select name="url">

Change to this:

<select name="url" onchange="location.href=this.value">

This way, if they choose a new item on the dropdown menu, if JavaScript is enabled, they'll be redirected. This way, however, you can redirect even your non-JavaScript users.

Next...

A text counter is also something we know how to make. If you recall two weeks ago, we played around reading and writing to text files. In the same article, we learned how to call PHP scripts even from our SHTML pages. That will be useful now.

Start by opening up your text editor. I want you to type in "0" (without the quotes) and nothing else. Save this as text, as count.txt, and upload to your web

Copyright © 2003 Robert Plank

host. Chmod to 0777 if necessary.

The file count.txt will be used to record how many visitors have been to our site so far. At this point we just have 0.

The next step is to create our script, counter.php. If you remember from our random quotes example, here's how to dump the contents of a file into a string:

```
<?php

$fp = fopen($quotefile, "r");
$mytext = fread($fp,filesize($quotefile));
fclose($fp);

?>
```

Only, in our case, I'd like us to change the variable that holds our filename to something that suits our script better, like $countfile.

Also, the second parameter in fopen(), "r", needs to be changed to "w+". Why? Because this time, we want to open the file for reading *and* writing, not just reading. Also, we need to take out that line with fclose() because we're not finished with this file yet.

So we make the changes:

```
<?php

$countfile = "count.txt";

$fp = fopen($countfile, "w+");
$mytext = fread($fp,filesize($countfile));

?>
```

Next, it's always a good idea to trim() the variable $mytext just to get rid of any crap that shouldn't be there. The next step...

Copyright © 2003 Robert Plank

```
$mytext = $mytext + 1;
```

This increases the value of the variable $mytext by one.

We write this newfangled value of $mytext back into our file:

```
fwrite($fp,$mytext);
```

And finally, close the file...

```
fclose($fp);
```

Then output our new counter value:

```
echo $mytext;
```

Oh, and if you want to output commas in our counter value (for example, if our counter is at "2500" and we want to show "2,500"), change that echo line to:

```
echo number_format($mytext);
```

That's our text counter! Remember from article 9, the ways to include PHP scripts using PHP or SSI...

Using PHP: `<?php include("path/to/your-script.php"); ?>`
Using SSI: `<!--#include virtual="path/to/your-script.php?$QUERY_STRING_UNESCAPED" -->`

DONE.

Ooh, the next one is kind of a fun one. A slot machine.

So how do we make a slot machine script? First, we'll have an array. Then, we'll pick a random value out of that array. We'll do this three times. Then, we compare our results of the three random tests and see if they match.

Copyright © 2003 Robert Plank

If they do match, we have a winner.

Start by creating a new file. slots.php maybe? Start with an indexed array called $slots and fill it with some slot machine values:

```
<?php

$slots[0] = "banana";
$slots[1] = "cherry";
$slots[2] = "grape";
$slots[3] = "lemon";
$slots[4] = "orange";
$slots[5] = "pear";
$slots[6] = "tomato";

?>
```

Oh, but guess what? When you're just filling indexed arrays like that, there's an easier way to do this:

```
$slots = array("banana", "cherry", "grape", "lemon", "orange", "pear", "tomato");
```

Don't forget to first seed the random number generator (article 7):

```
srand((double)microtime()*1000000);
```

Remember, only use srand() ONCE in your script!

Find a random value in the array and assign this to $firstslot.

```
$firstslot = rand(1,count($slots)) - 1;
```

Do the same with $secondslot and $thirdslot.

```
$secondslot = rand(1,count($slots)) - 1;
$thirdslot = rand(1,count($slots)) - 1;
```

Copyright © 2003 Robert Plank

The next logical step is to show each result:

```
echo $slots[$firstslot]." ".$slots[$secondslot]." ".$slots[$thirdslot]." ";
```

And then finally compare each to tell us if the player is a winner or loser:

```
if ($firstslot == $secondslot && $secondslot == $thirdslot) {
  echo "WINNER!";
}

else {
  echo "LOSER!";
}
```

Our script:

```
<?php

$slots = array("banana", "cherry", "grape", "lemon", "orange", "pear", "tomato");

srand((double)microtime()*1000000);

$firstslot = rand(1,count($slots)) - 1;
$secondslot = rand(1,count($slots)) - 1;
$thirdslot = rand(1,count($slots)) - 1;

echo $slots[$firstslot]." ".$slots[$secondslot]." ".$slots[$thirdslot]." ";

if ($firstslot == $secondslot && $secondslot == $thirdslot) {
  echo "WINNER!";
}

else {
  echo "LOSER!";
}

?>
```

Copyright © 2003 Robert Plank

Upload the script to your web host and run it. Refresh the page to play multiple times.

How is something like a slot machine useful? The most ingenious implementation of a script like this was used to collect leads. I don't know who started this trend, but the idea has been duplicated in autoresponders and refer-a-friend scripts.

Let's say you decide to give away a prize if they give you their e-mail address. However, there are two little problems: first, you want them to give you a REAL e-mail address and not just some random ones, and second, you don't want to devalue your $50 or $100 product by giving it away any visitor who simply asks.

The great thing about slot machines is that intuitively, people don't realize how low their odds are. In the example above we have seven symbols in one "reel". We have three reels, so we have 7 times 7 times 7 possibilities. Totaling 343.

Only 7 of those possibilities will yield wins, leading to 1 win in 49 tries. That means, on average, for every 49 fresh leads you get, you have to give away one of those prizes of yours.

Of course if you're sneaky you could throw in some extra random variables. For example, inside that if-statement:

```
if ($firstslot == $secondslot && $secondslot == $thirdslot) {
  echo "WINNER!";
}
```

You could have ANOTHER random value inside of that... say, a random number between 1 and 2. If the result of THAT random choice was 1, they really win, but otherwise, have the slot machine go again. This decreases your visitors' odds to 1 in 98. And of course you can try a random number between 1 and 5, or 1 and 10 or whatever you like. Or even add extra symbols in each reel.

So, how can you be sure that these people give you a valid e-mail address? You can do all kinds of junk like regular expression filters, or POP3 user

Copyright © 2003 Robert Plank

verification... but the easy way to do it is just to tell your gamblers that their prize will be sent to them by e-mail.

That way, if they give you a bogus address and they win, they'll sure regret it because they won't be able to collect their prize!

So, first you need an HTML page to start off on. Call this form.html and have something like this in it:

```
<form action="slots.php" method="post">
Enter Your E-mail Address to Play: <input type="text" name="email"
size="30"><br>
<input type="submit" value="Play Slots">
</form>
```

This will put the player's e-mail address, in the variable called $email, into our script.

Looking back on article 3, we have this code:

```
$myname = "Your Name Here";
$mymail = "your@email.here";

$subject = "Hello";
$body = "Hi. This is the body of my message.
Notice how I can continue typing right on the next line!";

$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

if ($email != "") { mail($email,$subject,$body,$headers); }
```

Well hey, we can reuse this, can't we? If they win, send them their congratulations message complete with a download link:

```
<?php
```

Copyright © 2003 Robert Plank

```
// Run the slot machine
$slots = array("banana", "cherry", "grape", "lemon", "orange", "pear", "tomato");

srand((double)microtime()*1000000);

$firstslot = rand(1,count($slots)) - 1;
$secondslot = rand(1,count($slots)) - 1;
$thirdslot = rand(1,count($slots)) - 1;

echo $slots[$firstslot]." ".$slots[$secondslot]." ".$slots[$thirdslot]." ";

// If they win, send the e-mail with download link.
if ($firstslot == $secondslot && $secondslot == $thirdslot) {

  $myname = "Your Name Here";
  $mymail = "your@email.here";

  $subject = "Hello, you played slot machines and won my prize";
  $body = "Hey you won, great... download link here.";

  $headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

  if ($email != "") { mail($email,$subject,$body,$headers); }

  echo "WINNER!  Your prize is in your e-mail.";
}

// If not, thank them for playing.
else {
  echo "LOSER!";
}

?>
```

Copyright © 2003 Robert Plank

Is that it? Of course not. We need to make sure they don't play over and over again, so we record the e-mail address each time. So create a text file, call it emails.txt, upload, and chmod to 0777 so we can write to it.

A first good step is to see if the e-mail address they gave us is even there, or how about if it contains an "at" (@) sign, using those handy regular expressions:

```
if (!eregi("\@",$email)) { echo "Sorry, you didn't give me a valid e-mail address."; }
```

The cool thing about this is that if someone puts in a blank address, there still won't be an @ sign so blank e-mail addresses won't work either!

Trim the value of $email just in case...

```
$email = trim($email);
```

Record what our file is going to be...

```
$ourfile = "emails.txt";
```

We need to check and see if this person has played using this e-mail address before. I know, I know... the most "efficient" way would be to use fopen() because we'll be writing back into the file in a minute, but using file() is a lot easier.

```
$check = file($ourfile);
```

Then loop through and take out all the line feeds:

```
for ($i=0;$i<count($check);$i++) {
  $check[$i] = trim($check[$i]);
}
```

Then just see if this address has been used before:

```
if (in_array($email,$check)) {
```

Copyright © 2003 Robert Plank

```
  echo "Sorry, this address has been used before."; die();
}
```

Notice that since die() is in there, if they'd played before, the script will refuse to continue on. Which is what we want. Now we write this address to our text file so that we know this address has been used.

Open our file, emails.txt for writing (remember, this is w).

```
$fp = fopen($ourfile, "w");
```

Then write this address, followed by a new line (which is represented in PHP by \n).

```
fwrite($fp,$email."\n");
```

And close.

```
fclose($fp);
```

Here's what we've got now:

```
<?php

// Make sure this is a valid e-mail address
if (!eregi("\@",$email)) { echo "Sorry, you didn't give me a valid e-mail
address."; }

$email = trim($email);
$ourfile = "emails.txt";

// Check to see if this address has been used before

$check = file($ourfile);

for ($i=0;$i<count($check);$i++) {
  $check[$i] = trim($check[$i]);
```

Copyright © 2003 Robert Plank

```
}

if (in_array($email,$check)) {
  echo "Sorry, this address has been used before."; die();
}

// Write this new address into a text file
$fp = fopen($ourfile, "w");
fwrite($fp,$email."\n");
fclose($fp);

// Run the slot machine
$slots = array("banana", "cherry", "grape", "lemon", "orange", "pear", "tomato");

srand((double)microtime()*1000000);

$firstslot = rand(1,count($slots)) - 1;
$secondslot = rand(1,count($slots)) - 1;
$thirdslot = rand(1,count($slots)) - 1;

echo $slots[$firstslot]." ".$slots[$secondslot]." ".$slots[$thirdslot]." ";

// If they win, send the e-mail with download link.
if ($firstslot == $secondslot && $secondslot == $thirdslot) {

  $myname = "Your Name Here";
  $mymail = "your@email.here";

  $subject = "Hello, you played slot machines and won my prize";
  $body = "Hey you won, great... download link here.";

  $headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

  if ($email != "") { mail($email,$subject,$body,$headers); }
```

Copyright © 2003 Robert Plank

```
  echo "WINNER!  Your prize is in your e-mail.";
}

// If not, thank them for playing.
else {
  echo "LOSER!";
}

?>
```

Now, let's say when they played this game, you wanted to add that address into your newsletter. After all, that's why you're capturing the leads, isn't it? So why not save yourself the chore...

I actually got this idea partly from Bill Humes who wanted me to make a custom script for him and I couldn't believe I hadn't thought of it sooner. But... most autoresponders just wait for an e-mail to be sent to them, and adds the address of the sender to the list.

Ever since chapter 3 we've known how to send e-mail, right? Not only that, but we've known how to specify the sender and the receiver for any e-mail we send. So why not take the e-mail we've just been given, and send it to our autoresponder address?

It'll treat the mail the same way as a normal subscription request, and this way, you can put the address right into your autoresponder, using the autoresponse software you already use. Look here:

```
$sysmail = "my@auto.responder";
```

Let's set $sysmail to whatever your autoresponder address is. We have to define what our headers are, of course, so just steal this from our script example above:

```
$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";
```

Copyright © 2003 Robert Plank

Oops, but we'll have to change this so that it comes from $email and not from [$mymail], won't we? So that becomes this:

```
$headers = "Content-Type: text/plain; charset=us-ascii\nFrom:
<$email>\nReply-To: <$email>\nReturn-Path: <$email>\nX-Mailer: PHP";
```

Then just send our e-mail. It's always important to have *something* in the body, so I'll just throw in a couple of line feeds and newline characters (represented by PHP as \r\n:

```
mail($sysmail, "", "\r\n\r\n",$headers);
```

There! Now, our visitor plays slots, and if he or she was won, they'll be e-mailed the URL to pick up their prize. But they were also subscribed to your autoresponder or newsletter. Our completed code:

```
<?php

// Make sure this is a valid e-mail address
if (!eregi("\@",$email)) { echo "Sorry, you didn't give me a valid e-mail
address."; }

$email = trim($email);
$ourfile = "emails.txt";

// Check to see if this address has been used before

$check = file($ourfile);

for ($i=0;$i<count($check);$i++) {
  $check[$i] = trim($check[$i]);
}

if (in_array($email,$check)) {
  echo "Sorry, this address has been used before."; die();
}
```

Copyright © 2003 Robert Plank

```
// Write this new address into a text file
$fp = fopen($ourfile, "w");
fwrite($fp,$email."\n");
fclose($fp);

// Run the slot machine
$slots = array("banana", "cherry", "grape", "lemon", "orange", "pear", "tomato");

srand((double)microtime()*1000000);

$firstslot = rand(1,count($slots)) - 1;
$secondslot = rand(1,count($slots)) - 1;
$thirdslot = rand(1,count($slots)) - 1;

echo $slots[$firstslot]." ".$slots[$secondslot]." ".$slots[$thirdslot]." ";

// If they win, send the e-mail with download link.
if ($firstslot == $secondslot && $secondslot == $thirdslot) {

  $myname = "Your Name Here";
  $mymail = "your@email.here";

  $subject = "Hello, you played slot machines and won my prize";
  $body = "Hey you won, great... download link here.";

  $headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";

  if ($email != "") { mail($email,$subject,$body,$headers); }

  echo "WINNER!  Your prize is in your e-mail.";
}

// If not, thank them for playing.
else {
  echo "LOSER!";
```

Copyright © 2003 Robert Plank

```
}

$sysmail = "my@auto.responder";
$headers = "Content-Type: text/plain; charset=us-ascii\nFrom: $myname
<$mymail>\nReply-To: <$mymail>\nReturn-Path: <$mymail>\nX-Mailer:
PHP";
mail($sysmail, "", "\r\n\r\n",$headers);

?>
```

So, you think, couldn't someone abuse this thing by just putting a ton of other people's addresses into the slot machine form? Well of course... but that's why you have double opt-in... :-)

Assignment:

We've discussed ways of "rigging" the slot machines to make it harder to win... think up some ways of alterng your code to make it EASIER to win instead.

QUIZ

1. This line of code:
$fp = fopen($ourfile,"w");
Opens the file called $ourfile for:
A: Reading only.
B: Writing only.
C: Reading and writing.
D: ... all to see.

2. What is the difference between "\r" and "\n"?

Copyright © 2003 Robert Plank

A: "\r" means carriage return, "\n" means newline.
B: "\r" means newline, "\n" means carriage return.
C: "\r" is "\n" plus one.
D: No difference, they are exactly the same.

3. What must we do if we want a file to be writeable by PHP?
A: Make sure the file contains the number "3".
B: Chmod the file to 0777.
C: Chmod the file to 0600.
D: Upload in ASCII instead of BINARY.

4. What function did we use in article 11 to see if a particular value existed within an array?
A: in_array()
B: check_array()
C: inarray()
D: array()

5. What does the trim() function do?
A: Deletes the entire string.
B: Changes the name of the variable containing the string.
C: Eliminates whitespace (extra spaces, line feeds, carriage returns, etc.) from the ends of a string.
D: Keeps the size of a string down.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 12 Cookie Fun With PHP

The last chapter left off with me briefly discussing about how one might abuse that precious lead-gathering slot machine script of ours... and how to prevent things like that.

It got me thinking to list different ways of preventing people from abusing the slot machine, such as recording each IP address to make sure the same one couldn't play twice, or to set a cookie...

A cookie...

A cookie.

What a great chapter idea. Cookies.

And I'm not talking about chocolate chip. A common analogy to cookies are dry cleaning tickets. When you bring clothes into the dry cleaning place, they give you a ticket stub with a number on it. You come at a later time with that stub and give it to the man behind the counter to get those same clothes back. This is sort of how a cookie works.

We've set, unset, and changed variables countless times in PHP. Cookies are variables that are stored on a visitor's computer. This is handy for us because we will be able to identify them when they return later (even if "they" are, say, on a dial-up connection where their IP might change).

So, you want to get started already. That's good. This is an example of setting a cookie:

```
header("Set-Cookie:brown=bag;expires=Tuesday, 1-Jan-1980 00:00:00
```

Copyright © 2003 Robert Plank

```
GMT;domain=.my.domain;\n\n");
```

If you've ever looked at the PHP Manual online, you'd know that PHP does indeed have a setcookie() function. I don't use it, because they're a headache to get working and the way setcookie() does things have been known to cause problems with IE 5.5 and Netscape 4.x.

Now, to break down that HTTP header I sent:

```
Set-Cookie:
brown=bag;
expires=Thursday, 1-Jan-2004 00:00:00 GMT;
domain=.my.domain;
```

It already makes a lot more sense now that I've split it up a little, doesn't it?

First, we need to specify our cookie name and its value. In this example, the cookie's name is brown and its value is set to bag.

The next piece is pretty explanatory. All cookies need to expire eventually. If you want to be a smartass and have your cookie last (almost) forever, go ahead and set it to January 1, 2037.

But most of the time you'll have a reason for letting a cookie expire. For example, if you have some sort of script where a user has to login, and you use cookies for this, it's a good idea to ask them to login again a couple of hours later.

The expiration date on a cookie has to be set in the way I've defined above.

And lastly, domain tells the user's browser what domain this cookie will be defined to. If you leave this out, the browser will just assume the current domain. This is what we call a "Bad Thing".

Why? Well, say your site is at http://your.host and you set a cookie, without specifying a domain. Then, your visitor goes to http://www.your.host and you try to access the cookie. It might look the same to you, but the browser says

Copyright © 2003 Robert Plank

"Oh, that isn't the right domain for that cookie."

The solution, is to put a dot in front of the domain. So in the case of your.host, set the cookie to .your.host. That way, the cookie will be applied not only to your domain, but to all its subdomains (yes, kids, the "www" before your domain name does indeed count as a sub).

So, let's say we want to set a cookie to expire 20 minutes from now. How will we do it? Well, for starters, let's declare what our domain (e.g. microsoft.com) is. DON'T FORGET THE DARN DOT AT THE BEGINNING!

```
<?php

$mydomain = ".some.domain";

?>
```

Next we need the timestamp of 20 minutes from now, in Unix epoch time (remember that big long number from chapter 9 generated from the date?), so we just use our buddy strtotime():

```
$datevalue = strtotime("+20 minutes");
```

Let's look back on that piece of the cookie now:

expires=Thursday, 1-Jan-2004 00:00:00 GMT;

That number needs to be formatted like that. Which we'll format using gmdate().

```
$expire = gmdate("D\, d M Y H\:i\:s",$datevalue);
```

gmdate() gives us the value of the date in Greenwich Mean Time (GMT), taking the numeric value from $datevalue and making it into a string the cookie can understand.

Then put our domain and this expiration date into that first cookie example I gave you:

Copyright © 2003 Robert Plank

```
header("Set-Cookie:brown=bag;expires=$expire
GMT;domain=$mydomain;\n\n");
```

All our code here for setting a cookie:

```
<?php

$mydomain = ".some.domain";
$datevalue = strtotime("+20 minutes");
$expire = gmdate("D\, d M Y H\:i\:s",$datevalue);

header("Set-Cookie:brown=bag;expires=$expire
GMT;domain=$mydomain;\n\n");

?>
```

Done. We've now set a cookie called "brown" and set its value to "bag". It's valid for our domain name only and it will expire 20 minutes from when it was set.

Now, let's modify the same script to erase the cookie. How we erase? Simple... just set the expiration date to a date before today's date. The "proper" date to set is of course, January 1st 1970, which is "0" in Unix time.

January 1 1970 12:00 AM and 1 second, is 1 in Unix time.
January 1 1970 12:00 AM and 2 seconds, is 2 in Unix time.

Get it?

```
<?php

$mydomain = ".some.domain";
$datevalue = 0;
$expire = gmdate("D\, d M Y H\:i\:s",$datevalue);

header("Set-Cookie:brown=bag;expires=$expire
```

Copyright © 2003 Robert Plank

```
GMT;domain=$mydomain;\n\n");

?>
```

I've changed $datevalue to 0 and our cookie is now gone. Yes, sometimes we want to delete cookies.

BUT REMEMBER THIS: Whenever you change a cookie, be *sure* to clear it first. Just because.

What's so good about this? Well, from any script you run on your domain, you can take the variable $brown and it will be set to "bag" until you change it or it expires. So, you could keep a user's login "sticky" so to speak, for a certain amount of time... or have your page personalized for that person forever.

In fact, let's take cookies all the way back to chapter 1. You remember chapter one, don't you? Personalizing your pages?

We called our script like this:

http://your.host/script.php?f=Elmo

And then, once we displayed our page, we had:

```
Dear <?php echo $f; ?> ,

Hey, this is a sample page.  Fun, isn't it?  Do do do do do do do do do do...
```

start off by telling PHP our domain, and how long we want the cookie to last. Let's just say one year.

```
<?php

$mydomain = ".some.domain";
$last = "1 year";

?>
```

Copyright © 2003 Robert Plank

Start off by setting our expiration date to 0 because we want to clear the cookie. Then actually clear the cookie.

```
$expire = gmdate("D\, d M Y H\:i\:s",0);
header("Set-Cookie:f=null;expires=$expire GMT;domain=$mydomain;\n\n");
```

Now that the cookie is gone, we can feel to *set* our cookie now.

```
$datevalue = strtotime("+".$last);
$expire = gmdate("D\, d M Y H\:i\:s",$datevalue);

header("Set-Cookie:f=$f;expires=$expire GMT;domain=$mydomain;\n\n");
```

NOT FINISHED YET! But here's what there is so far...

```
<?php

$mydomain = ".some.domain";
$last = "1 year";

// Erase the cookie that's there
$expire = gmdate("D\, d M Y H\:i\:s",0);
header("Set-Cookie:f=null;expires=$expire GMT;domain=$mydomain;\n\n");

// Set the new cookie
$datevalue = strtotime("+".$last);
$expire = gmdate("D\, d M Y H\:i\:s",$datevalue);

header("Set-Cookie:f=$f;expires=$expire GMT;domain=$mydomain;\n\n");

?>
```

The last thing we need to do is... make sure that $f actually contains something! The point of this is to make the cookie when $f is actually there, you see?

When $f has a value, we make the cookie, so that when it's accessed later by the

Copyright © 2003 Robert Plank

same person when $f has no value, our script will just grab that from the cookie.

So, we need to surround all of this with an if-statement, checking to make sure that $f actually contains something.

Also, don't forget to take all this and stick it at the top of script.php.

```
<?php

if ($f != "") {

  $mydomain = ".some.domain";
  $last = "1 year";

  // Erase the cookie that's there
  $expire = gmdate("D\, d M Y H\:i\:s",0);
  header("Set-Cookie:f=null;expires=$expire GMT;domain=$mydomain;\n\n");

  // Set the new cookie
  $datevalue = strtotime("+".$last);
  $expire = gmdate("D\, d M Y H\:i\:s",$datevalue);

  header("Set-Cookie:f=$f;expires=$expire GMT;domain=$mydomain;\n\n");

}

?>
```

Dear <?php echo $f; ?> ,

Hey, this is a sample page. Fun, isn't it? Do do do do do do do do do do do...

Now upload and load the script like this:

http://your.host/script.php?f=Elmo

Then, take out the personalized part...

Copyright © 2003 Robert Plank

http://your.host/script.php

If you've done everything right, the name "Elmo" should remain. You've just made your first script using cookies!

There are a few things to remember about cookies before I go...

First, as I explained earlier, you can't read or write cookies from other domains (so if a user goes to apple.com and that site sets a cookie, you won't even be able to see it.

Cookies are useless your cookies are disabled (or blocked) on your visitor's browser.

Cookies will reset if your visitor decides to clear his or her browser cache.

Cookies don't carry over onto other browsers or to other computers.

And, most importantly, cookies won't show up until a new page is loaded! So, if you set a cookie called "eskimo" to the value "pie", you won't be able to read it until your visitor clicks through to another page (or reloads that same page).

The end.

Assignment

What happens to your cookie if the format of the expiration date is changed around?

Copyright © 2003 Robert Plank

QUIZ

1. How can a cookie be deleted?
A: Using a special function to delete cookies.
B: Set the cookie's date to a point more than a year into the future.
C: Set the cookie's date to a point in the past.
D: Cookies can't be deleted, once a cookie is made it's there to stay.

2. What does date() do?
A: Gives us the date, in Unix time.
B: Formats a Unix timestamp into a readable date.
C: Nothing.
D: Everything.

3. What's the difference between date() and gmdate()?
A: date() gives us the time elapsed since 1/1/1970 in *seconds*, while gmdate() gives us milliseconds.
B: date() is crispier than gmdate().
C: date() gives us a formatted date in Greenwich Mean Time, gmdate() is only the local time.
D: gmdate() gives us a formatted date in Greenwich Mean Time, date() is only the local time.

4. True or false: It's recommended to delete a cookie before giving a new value to it.
A: True.
B: False.

5. True or false: Cookies don't take effect until a new page is loaded (or the current page is reloaded).
A: True.
B: False.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 13 Comparison With PHP

So... we've played with cookies, we've played with flat-text databases, arrays, file handling, and lots of other fun crud. What's next? Well, we've totally skipped over if-statements. Which I think are important, but boring.

But I think this quiz example might be kind of fun.

All right... if-statements let you do some basic logic. If you're familiar with just about any programming language this is known to you already.

Otherwise, well, you learned about inequalities at school, right? For example:

5 > 4

This says "5 is greater than 4". When I learned this in the fourth grade, the teacher drew little teeth on the angle because "the mouth is pointing at the 5, which is bigger than 4, and the alligator is hungry."

This also works the other way, like:

3 < 4

We can also have "less than or equal to" operators, such as:

6 <= 6

Is 6 less than or equal to 6? Well, it's equal to 6, so yes.

And of course there's an equality operator, which is this:

Copyright © 2003 Robert Plank

4 == 4

Notice how in PHP we use two equals signs (==).

This is useful in a situation like this:

```
<?php

$x = 5;

if ($x > 6) { echo "YES IT IS BIGGER THAN 6"; }
else { echo "Darn... it is not bigger than 6."; }

?>
```

If $x is greater than 6, we say "YES IT IS BIGGER THAN 6". Otherwise, we show the other disappointing message.

You won't have to worry about greater than or less than for this article, but they are good to know.

In this article we'll be making a fun little quiz! And, we will be relying on the equality operator (the two equals signs) to see if the answers a person gives us match the real answers.

Since we've gotten as far as we have, I plan on using not only comparisons, but also file handling, string/array manipulation and a little basic math. Maybe some really simple regular expressions too. Just putting together things you already should know.

Here's a sample quiz file:

1: According to the five kingdom view of life, how many kingdoms are there?
A: 3
B: 5
C: 7
D: 9

Copyright © 2003 Robert Plank

2: Is "none of the above" a good answer?
A: Sometimes
B: Always
C: All of the above
D: None of the above

3: Two plus three is...
A: Five
B: Seven
C: I don't count that high
D: Let me get my calculator

4: Excrement is...
A: Blue
B: Brown
C: Orange
D: Depends on the time of day

5: True or false. A three hour tour.
A: True
B: False

Save this as questions.txt. This will be our data file of questions. Only this time, instead of using pipes (|) as was our previous example, in article 10, our separator is "a colon followed by a space" or ": ".

This is to show that you can really use any separator you like. Just make SURE that any separators you use aren't re-used in places they aren't supposed to be (like as parts of actual questions or answers). If that happens... the whole thing will be messed up. So don't.

We need to start over with a new file... called quiz.php. Start off by reading the contents of questions.txt into a variable called $data, then closing the file.

This should be a pretty easy task for you at this point.

Copyright © 2003 Robert Plank

```
<?php

$qfile = "questions.txt";

$fp = fopen($qfile,"r");
$data = fread($fp,filesize($qfile));
fclose($fp);

?>
```

Now we need to use explode() to split up the contents of this file. What will our separator be? Two newline characters (\n).

Depending on how you upload your text file to your FTP server, a line feed could either be represented by \n or \r\n. So first we need to take out all the \r's. That is, if they exist anyway...

```
$data = eregi_replace("\r","",$data);
```

Then explode this string into an array based on what separates each number from the other...

```
$data = explode("\n\n",$data);
```

Right now, $data would look like this:

```
Array
(
   [0] => 1: According to the five kingdom view of life, how many kingdoms are
there?
A: 3
B: 5
C: 7
D: 9
   [1] => 2: Is "none of the above" a good answer?
A: Sometimes
B: Always
```

Copyright © 2003 Robert Plank

C: All of the above
D: None of the above
  [2] => 3: Two plus three is...
A: Five
B: Seven
C: I don't count that high
D: Let me get my calculator
  [3] => 4: Excrement is...
A: Blue
B: Brown
C: Orange
D: Depends on the time of day
  [4] => 5: True or false. A three hour tour.
A: True
B: False
)

For now let's just play with $data[0]...

```
$tmp = $data[0]
```

The variable $tmp looks like this:

1: According to the five kingdom view of life, how many kingdoms are there?
A: 3
B: 5
C: 7
D: 9

What we've got to do now is use explode() on this string and split it up by each line. So let's say:

```
$tmp = explode("\n",$tmp);
```

Now $tmp looks like:

Array

Copyright © 2003 Robert Plank

```
(
   [0] => 1: According to the five kingdom view of life, how many kingdoms are
there?
   [1] => A: 3
   [2] => B: 5
   [3] => C: 7
   [4] => D: 9
)
```

We know that the first line, that is, $tmp[0], is the question we want to ask. So we say:

```
echo $tmp[0];
```

Which gives us this:

1: According to the five kingdom view of life, how many kingdoms are there?

This takes care of asking the question, but what about our visitor choosing an answer? This is where our little bit of HTML knowledge with radio buttons come in. If you don't know HTML that well, I'll take you through an example real quick.

Pretend that you're submitting to a form, and the variable you want to be called $greek with the options "alpha", "beta", "gamma", and "delta". You would have something like this:

```
<input type="radio" name="greek" value="alpha"> my first option<br>
<input type="radio" name="greek" value="beta"> my second option<br>
<input type="radio" name="greek" value="gamma"> my third option<br>
<input type="radio" name="greek" value="delta"> my fourth option<br>
```

Understand how this works? The first radio button says "greek equals alpha", the second says "greek equals beta", etc. The stuff written after the radio button doesn't matter, it's just there to tell our visitors what exactly they're clicking on.

Now, on to creating our HTML form.

Copyright © 2003 Robert Plank

Instead of echoing $tmp[0], let's try looping through it, then at each step, exploding THAT and outputting each question:

```
echo "<br>";

for ($j=1;$j<count($tmp);$j++) {
  echo $tmp[$j]."<br>";
}
```

We get:

1: According to the five kingdom view of life, how many kingdoms are there?
A: 3
B: 5
C: 7
D: 9

So, how do we take those options and turn them into radio buttons? Easy... change that loop to this:

```
for ($j=1;$j<count($tmp);$j++) {
  $button = explode(": ",$tmp[$j]);
  echo "<input type=\"radio\" name=\"question\" value=\"$button[0]\">
$button[1]<br>";
}
```

See, it rips apart *that* piece by our separator and then takes the first half, and uses it as our HTML value, then shows the whole thing to our visitor.

You know, just for the sake of writing clean code, change all the
 in our script so far to
\n.

Here's the whole thing so far in case you missed it:

```
<?php
```

Copyright © 2003 Robert Plank

```
$qfile = "questions.txt";

$fp = fopen($qfile,"r");
$data = fread($fp,filesize($qfile));
fclose($fp);

$data = eregi_replace("\r","",$data);

$data = explode("\n\n",$data);

$tmp = $data[0];

$tmp = explode("\n",$tmp);

echo $tmp[0];

echo "<br>";
for ($j=1;$j<count($tmp);$j++) {
  $button = explode(": ",$tmp[$j]);
  echo "<input type=\"radio\" name=\"question\" value=\"$button[0]\">
$button[1]<br>\n";
}

?>
```

Try it, upload to your web server and go.

NOTE NOTE NOTE! That certain browsers (i.e., Netscape) are picky about showing form elements where an actual <form> tag is not defined. So be warned, that as this stands now it may not preview well in Netscape.

Now, view the source of that HTML and you'll see this:

```
1: According to the five kingdom view of life, how many kingdoms are
there?<br>
<input type="radio" name="question" value="A"> A: 3<br>
<input type="radio" name="question" value="B"> B: 5<br>
```

Copyright © 2003 Robert Plank

```
<input type="radio" name="question" value="C"> C: 7<br>
<input type="radio" name="question" value="D"> D: 9<br>
```

Hey! That worked out pretty well now, didn't it? Only one problem, though... it only does the first question in questions.txt, now, doesn't it? That's because of this line:

```
$tmp = $data[0];
```

If we looped through the array $data instead of just giving its *first* value to $tmp, we could take care of all the questions. This part can get kind of tricky. We want to loop through the array, and each time, give the current sequence to the variable $tmp, and then do everything else.

```
<?php

$qfile = "questions.txt";

$fp = fopen($qfile,"r");
$data = fread($fp,filesize($qfile));
fclose($fp);

$data = eregi_replace("\r","",$data);

$data = explode("\n\n",$data);

for ($i=0;$i<count($data);$i++) {

  $tmp = $data[$i];
  $tmp = explode("\n",$tmp);

  echo $tmp[0];

  echo "<br>";
  for ($j=1;$j<count($tmp);$j++) {
    $button = explode(": ",$tmp[$j]);
    echo "<input type=\"radio\" name=\"question\" value=\"$button[0]\">
```

```
$button[1]<br>\n";
  }
}

?>
```

Get what we did here? We put a loop around another loop, which I know can seem confusing at first... but, just think we did everything the same as with the first question, only we took that same process and went through each and every question.

The only problem is that all the options will be named "question". Which is a bad thing, because our script won't be able to tell the difference between, say question #1 and question #2. So on this line:

```
echo "<input type=\"radio\" name=\"question\" value=\"$button[0]\">
$tmp[$j]<br>\n";
```

Instead of naming our radio button "question", we name it "question[$i]". This way all the radio buttons on the first question will be called question[0], the buttons on the next will be question[1], all through our questions.

While we're at it, add another "
\n" at the end of each "$i" loop, so it looks nicer.

Our code:

```
<?php

$qfile = "questions.txt";

$fp = fopen($qfile,"r");
$data = fread($fp,filesize($qfile));
fclose($fp);

$data = eregi_replace("\r","",$data);
```

Copyright © 2003 Robert Plank

```
$data = explode("\n\n",$data);

for ($i=0;$i<count($data);$i++) {

  $tmp = $data[$i];
  $tmp = explode("\n",$tmp);

  echo $tmp[0];

  echo "<br>";
  for ($j=1;$j<count($tmp);$j++) {
    $button = explode(": ",$tmp[$j]);
    echo "<input type=\"radio\" name=\"question[$i]\" value=\"$button[0]\">
$button[1]<br>\n";
  }
  echo "<br>\n";
}

?>
```

Ok, now we've got to actually tell our form to submit somewhere. So up at the top, we put this in:

```
echo "<form action=\"finish.php\" method=\"post\">";
```

And then, at the end, we put in our submit button and close the form:

```
echo "<input type=\"submit\" name=\"submit\" value=\"Done\"></form>";
```

Now we're all good as far as being able to read the questions and options. Now, we take care of the answers.

Now create an answer file. Let's call this answers.txt.

What delimiter do you want to use for this? Pipes? Nah, let's just have a separate answer on each line. Put the answers in the file like this:

Copyright © 2003 Robert Plank

B
D
A
B
A

Save as answers.txt. Now, on to the final step of this. Start off by setting our separator and our answer file.

```
<?php

$sep = ": ";
$answer = "answers.txt";

?>
```

Next, we can use the file() function since each entry is on its own line, which saves headaches...

```
$data = file($answer);
```

Then of course loop through and strip out the excess junk...

```
for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
}
```

Now to compare the data. If you could see what was being passed into finish.php you'd see this:

```
[question] => Array (
  [0] => B
  [1] => A
  [2] => A
  [3] => A
  [4] => B
)
```

Copyright © 2003 Robert Plank

Hey, an array. We know how to handle loops with those thingies, now don't we? So, what I would do is just loop through the array containing the file and compare those to the answers.

Like this:

```
for ($i=0;$i<count($data);$i++) {
  if ($data[$i] == $question[$i]) { echo "You got question $i right.<br>\n"; }
  else { echo "You got question $i wrong.<br>\n"; }
}
```

What we've got:

```
<?php

$sep = ": ";
$answer = "answers.txt";

$data = file($answer);

for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
}

for ($i=0;$i<count($data);$i++) {
  if ($data[$i] == $question[$i]) { echo "You got question $i right.<br>\n"; }
  else { echo "You got question $i wrong.<br>\n"; }
}

?>
```

I tried it out, answering the questions randomly and got this as a result:

You got question 0 right.
You got question 1 wrong.
You got question 2 right.

Copyright © 2003 Robert Plank

You got question 3 wrong.
You got question 4 wrong.

Question ZERO?! Uh oh. Got to fix that. So we go to these two lines:

```
if ($data[$i] == $question[$i]) { echo "You got question $i right.<br>\n"; }
else { echo "You got question $i wrong.<br>\n"; }
```

Now, we want to say $i plus one, that way we can say question 1, question 2, and so on. It changes to this:

```
if ($data[$i] == $question[$i]) { echo "You got question ".($i+1)." right.<br>\n";
}
else { echo "You got question ".($i+1)." wrong.<br>\n"; }
```

Make sense? We had to take the $i stuff out of the quotes, then surround with parentheses and add a "+1" to the end.

Now heed your attention to this part of the code:

```
for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
}

for ($i=0;$i<count($data);$i++) {
  if ($data[$i] == $question[$i]) { echo "You got question ".($i+1)."
right.<br>\n"; }
  else { echo "You got question ".($i+1)." wrong.<br>\n"; }
}
```

Hey... seems like a little bit of a waste doesn't it? Looping through that array twice. So we can compress it down to this:

```
for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
  if ($data[$i] == $question[$i]) {
     echo "You got question ".($i+1)." right.<br>\n";
```

Copyright © 2003 Robert Plank

```
  }
  else {
    echo "You got question ".($i+1)." wrong.<br>\n";
  }
}
```

Woo hoo!

```
<?php

$sep = ": ";
$answer = "answers.txt";

$data = file($answer);

for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
  if ($data[$i] == $question[$i]) {
    echo "You got question ".($i+1)." right.<br>\n";
  }
  else {
    echo "You got question ".($i+1)." wrong.<br>\n";
  }
}

?>
```

Now I'm betting you want to be able to tell the person, "You got 2 out of 5 correct" or, "Hey there, you got 40% right."

Assignment

If you're feeling ambitious again, try and figure out a way to give our visitor his or her percentage *before* looking at the next chapter.

Copyright © 2003 Robert Plank

QUIZ

1. The statement 5 > 7 will return...
 A: True.
 B: False.

2. True or false: When using flattext files, we can only use one type of delimiter (eg, pipes or spaces).
 A: True.
 B: False.

3. filesize($somefile) gives us...
 A: The size of the current file, in bytes.
 B: The size of the file $somefile, in bytes.
 C: The size of all files in the directory called $somefile.
 D: Nothing.

4. Why do we always have to loop through the array we generated as a result of file() and trim() each element?
 A: file() is just a crappy function.
 B: file() was designed to work with trim().
 C: file() separates each line of a file but doesn't get rid of newline characters.
 D: An array made from file() isn't really an array until we've trimmed() each element.

5. What does count($myarray) give us?
 A: A total of the number of elements contained within $myarray.
 B: A list of all elements within $myarray.
 C: The array $myarray, only it's sorted properly.
 D: The exact same array we began with.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 14 Loops With PHP

Hello. Hmm. So what are we going to do today? Last chapter we did that nifty quiz that we made together. What good's a quiz on your site? Well, maybe if you gave some sort of prize or other incentive for answering all the questions correctly it would be good.

For example, your questions could be things like, "How many bonuses come with my product?" or, "How many testimonials are on the front page?" Some people will just hunt around for the answers, yes, but there's a chance you might make a sale along the way. :-)

And... you could even borrow code from that slot machine script from article 11 to require their e-mail address for a chance to win...

Anyway, last week we had this script:

```
<?php

$sep = ": ";
$answer = "answers.txt";

$data = file($answer);

for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
  if ($data[$i] == $question[$i]) { echo "You got question ".($i+1)."
right.<br>\n"; }
  else { echo "You got question ".($i+1)." wrong.<br>\n"; }
}

?>
```

Copyright © 2003 Robert Plank

What we had was... one script that read from a question file and generated an HTML form, which then submitted that information to a second script which read from an answer file and compared the answers.

We wanted the ability to be able to tell our test taker "you got X number of questions wrong" or "you got X% wrong". Something like that.

So, what we'd have to do is keep a counter, and up that counter by one each time they get a question right.

Start off by making your counter variable. We start at 0. Let's just call it $count.

```
$count = 0;
```

Then, find this line:

```
if ($data[$i] == $question[$i]) { echo "You got question ".($i+1)." right.<br>\n";
}
```

And put $count++; inside of it. This tells $count to increase by 1.

```
if ($data[$i] == $question[$i]) {
  echo "You got question ".($i+1)." right.<br>\n";
  $count++;
}
```

Then, outside the loop, output the result of $count:

```
echo $count;
```

This is what we have:

```
<?php

$sep = ": ";
$answer = "answers.txt";
```

Copyright © 2003 Robert Plank

```
$data = file($answer);
$count = 0;

for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
  if ($data[$i] == $question[$i]) {
     echo "You got question ".($i+1)." right.<br>\n";
     $count++;
  }
  else { echo "You got question ".($i+1)." wrong.<br>\n"; }
}

echo $count;

?>
```

So, I took the quiz and picked a few answers, some right, some wrong, and it gave me this:

```
You got question 1 right.
You got question 2 wrong.
You got question 3 right.
You got question 4 wrong.
You got question 5 wrong.
2
```

That "2" tells us that we've got 2 correct. But maybe we want to say, "You got 2 out of 5 correct." So we change:

```
echo $count;
```

To this:

```
$total = count($data);
echo "<br>\nYou got $count correct out of $total.";
```

Copyright © 2003 Robert Plank

The result is:

You got question 1 right.
You got question 2 wrong.
You got question 3 right.
You got question 4 wrong.
You got question 5 wrong.

You got 2 correct out of 5.

And what about a percentage? That's easy too... just a little bit of simple math. Change that block to:

```
$total = count($data);
$percentage = ($count / $total) * 100;
echo "<br>\nYou got $percentage percent out of $total questions.";
```

If you understand how to get a percentage of your score, you divide the number you got right by the total number of questions we had... and multiply that by 100. The asterisk (*) is the multiplication symbol and the slash (/) is our division symbol.

Done.

```
<?php

$sep = ": ";
$answer = "answers.txt";

$data = file($answer);
$count = 0;

for ($i=0;$i<count($data);$i++) {
  $data[$i] = trim($data[$i]);
  if ($data[$i] == $question[$i]) {
     echo "You got question ".($i+1)." right.<br>\n";
     $count++;
```

Copyright © 2003 Robert Plank

```
  }
  else { echo "You got question ".($i+1)." wrong.<br>\n"; }
}

$total = count($data);
$percentage = ($count / $total) * 100;
echo "<br>\nYou got $percentage percent out of $total questions.";

?>
```

That example script is done with. We aren't going back to it anymore. Now it's time for some delicious loops. No, not Fruit Loops, just plain ole loops.

When you need the same thing done over and over and over... don't worry, that's what computers are good at. Take something like this for example:

```
<?php

echo "hello ";
echo "hello ";
echo "hello ";
echo "hello ";
echo "hello ";

?>
```

Isn't there an easier way to do that? Well, of course... that's the bread and butter of loops. In PHP you'll find three kinds of loops: for loops, while loops, and do-while loops. Once you get the hang of them you'll see that all three are nearly identical.

You've already seen us use for loops to pass through arrays, doing something like this:

```
for ($i=0;$i<count($something);$i++) {
  echo "some text";
}
```

Copyright © 2003 Robert Plank

Let's pay special attention that first line:
```
for ($i=0;$i<count($something);$i++)
```

And now to look at the parameters of for.

($i=0;$i<count($something);$i++)

Each step is separated by colons. The first piece:

$i=0;

Tells PHP to set the value of $i to zero when we start this. That's all there is to it.

The next piece...

$i<count($something)

Yum, another one of those lovely inequalities. This tells us that we should remain in this loop only as long as this works. I know this can be a little confusing... but if this piece was something like...

$i<10

This says to keep us in the loop as long as the value of $i is less than 10. If $i is 10, or higher than 10, the loop is over and we continue through the program.

So why do we stick count() in there when looping through arrays? Well, say we have an indexed array of 12 elements. The value from count() we get is 12, but each element is in the array starting from zero, so we need to count from 0 to 11.

Well hey, it's starting to fit together isn't it? Start at 0, continue as long as we are below 12... but aren't we missing something here?

Yes, we are. And that's the third piece.

Copyright © 2003 Robert Plank

$i++

I think I used this once before, and this means "increase the value of $i by one". Saying:

```
$i++;
```

...is the exact same thing as saying:

```
$i = $i + 1;
```

So what's the third piece do? Whatever is in the third piece, gets performed every time we go through the loop.

Now that we know what everything does, let's pretend we're using a block of code like this:

```
for ($i=0;$i<20;$i++) {
  echo $i." ";
}
```

What will this code do? Well, it'll start at 0. The second piece says, keep looping until we reach 20, and the third piece says to increase $i by one each time we make another pass through the loop.

It goes through the loop once and $i is 0. It echoes $i so we see "0". Increase $i by one. Now $i is 1, we echo it so we see "1".

This continues until we reach 19 (remember, we continue as long as $i is LESS THAN, not equal to, 20) and then we're out of the loop. What gets outputted is this:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Want to count from 1 to 50, showing each step as it goes through? Do this:

```
for ($i=1;$i<=50;$i++) {
```

Copyright © 2003 Robert Plank

```
  echo $i." ";
}
```

Here, we begin counting at 1 and continue as long as $i is less than or equal to 50. If we get to 51, we're bust.

Remember how I told you there were THREE kinds of loops in PHP? Heh heh. Don't worry, these other two are amazingly simple.

Let's take our for loop up there, that counts from 1 to 50, and turn it into a while loop:

```
$i=0;

while ($i<=50) {
  echo $i." ";
  $i++;
}
```

While loops take up lots more space, but they're easier to understand (probably also easier for you to make at the moment). The only thing a while loop needs to do its job is the information that would normally go into the 2nd chunk of a for loop. That is, "Keep at it until this doesn't hold any longer."

So, outside the while loop we first set $i to 0. Then, we tell the loop to keep going until $i is no longer less than or equal to 50. (Meaning $i will have to be LARGER than 50 not to continue.)

If we just left it at that, though, our loop would never end. Hence the need for the $i++; you see in there, to increase $i by one each time. (Of course nothing's stopping you from saying "$i = $i + 2;", in any loop...)

Two down, one to go. The last kind of a loop is what's called a "do-while" loop. Probably a big change, huh? A totally different way of writing a loop...

Well, no, not at all. You see that while loop code above us? I'll change it into a do-while loop for ya:

Copyright © 2003 Robert Plank

```
$i=0;

do {
  echo $i." ";
  $i++;
}
while ($i<=50);
```

Poof.

The only difference between a while loop and a do-while loop is just that the while() part goes after the loop.

So what the hell are loops good for anyway? It's just counting. Yes, but counting with which you can do very cool things. Like what? Well hey, when I first "learned" PHP, the first lil' thing I made was a cool calendar. Sound fun? It is.

Let's go.

Assignment

You've seen a ton of ways to count *up* with all kinds of loops, now I want you to see if you can figure out how to make them count *down*.

QUIZ

1. What does $george++ do?
 A: Nothing, that causes at error.
 B: Doubles the value of $george.

Copyright © 2003 Robert Plank

C: Decreases the value of $george by 1.
D: Increases the value of $george by 1.

2. Let's say the value of $truck is 5. Consider this code:
$truck = $truck * 10;
What is the value of $truck now?
A: 50.
B: 15.
C: 510.
D: 2.

3. How many different kinds of loops did we cover in Chapter 14?
A: One.
B: Two.
C: Three.
D: Four.

4. The slash (/) symbol does:
A: Division.
B: Modulus.
C: Multiplication.
D: Exponentation.

5. Consider the code:
for ($c=1;$c<20;$c=$c*2) {
echo "$c ";
}[/php]

This would give you:
A: 1 2 4 8 16 32
B: 1 2 4 8 16
C: 1 2 3 4 5 ... 20
D: 1 2 3 4 5 ... 19

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 15 A Calendar With PHP

So, I promised that I'd show you how to use your newfound knowledge in loops to make a calendar, just like the one I made when I was a PHP newbie.

Well, sort of like mine. Mine had some extra mySQL database stuff in it, that took me a long time to get working the way I wanted it to. (Hey, cut me a break, it was my first shot at PHP.) I'll get you there eventually, let's just take this one step at a time...

We're to make a calendar. What good is a calendar? You can use the calendar as part of a database and use it to announce events, use it as a task manager. If you run a newsletter you can use it to link to archived issues. Think up anything that uses a time or date, and you can use a calendar with it.

First we need to think about how we're going to approach the problem. Since this is going to be output in HTML, we'll need to put everything into an HTML table. If you don't know the HTML markup for tables, that's okay, but it really helps you to understand this if you do.

The simplest thing we can start off doing is drawing a calendar for a single month. So, what we need to know from PHP's date() function is, how many days are in the month, and on what day of the week does our month begin (for example, September 1st, 2003, begins on a Monday).

Once we have this information we can draw out our calendar. Once we have this we'll be able to create a calendar for any month of the year, for any year from 1970 to 2038.

A quick review of HTML tables in case you aren't educated on them... the table itself is opened and closed using the <table> and </table> tags. Each row is defined with the <tr> tag (TR stands for Table Row), and each cell within that

Copyright © 2003 Robert Plank

row is defined using <td> (TD stands for Table Cell... and yes, I know "cell" doesn't start with a D).

You must have the same number of cells in each row on the same table, or the table will look broken on some browsers. Here's a sample table, 2 columns wide and 3 rows long. I've made the table's border equal to "1" so that you can actually see the table.

```
<table border="1">
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>

  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>

  <tr>
    <td>Row 3, Column 1</td>
    <td>Row 3, Column 2</td>
  </tr>
</table>
```

And of course if I want to produce a 2 x 3 table with empty cells, I'd put " " inside each cell. Why? Well, some browsers don't like empty cells, so they don't show the cell at all. This is bad. But   (which is like a space that is treated as a real character) is our way of making the tables *look* empty, while allowing them to remain visible.

Our 2 x 3 *blank* table:

```
<table border="1">
  <tr>
    <td> </td>
    <td> </td>
```

Copyright © 2003 Robert Plank

```
  </tr>

  <tr>
    <td> </td>
    <td> </td>
  </tr>

  <tr>
    <td> </td>
    <td> </td>
  </tr>
</table>
```

Now, the next step is figuring out how we can draw a table of any given size. What I mean by this is, how can we just tell PHP that we want, for example, 7 columns and 4 rows of table? Using loops?

I'll give you a second to think about it.

Stumped? Yes, no? If you think back to our tip of the day script (chapter 9) it might seem a little easier.

Back then I introduced modulus, otherwise known as remainder, which as you remember was useful in a very specific way. Back then it was good for letting our tip of the day change only *once* per day. This time it'll be useful for keeping the dates of the month (1, 2, 3, etc.) within the boundary of days of the month (Monday, Tuesday, Wednesday, etc.)

Let's start by saying how many rows and how many columns we'd like. Let's just call these variables $rows and $cols so we know what they are.

```
<?php

$cols = 7;
$rows = 4;

?>
```

Copyright © 2003 Robert Plank

Then, start the table:

```
echo "<table border=\"1\">\n";
```

Next, multiply the number of columns and rows to figure out what the total number of cells in the table will be. (This will come in handy later.)

```
$total = $cols * $rows;
```

Begin our loop.

```
for ($i=0;$i<$total;$i++) {
```

When creating HTML tables we can't just say how many columns we'd like. We have to specify where the breaks are. We have to do something like this:

```
if ($i % $cols == 0) { echo "<tr>\n"; }
```

Here we say, that if the value of $i at this point (remember we're counting from 0 to 27 here) has a remainder zero, create a new row.

When does a number have remainder zero? When it's divisible by the number of columns. In our case, 7.

In a 4 x 7 table, those numbers are: 0, 7, 14, 21... hey... that makes the table break and start in a new row after every 7th cell. Just what we want!

When we start a new row though, we want to end the previous one. So we have to put this line ABOVE the previous one:

```
if ($i % $cols == 0) { echo "</tr>\n\n"; }
```

Now, put this line in after it:

```
echo "<td>$i</td>";
```

Copyright © 2003 Robert Plank

This puts the value of $i (our counter) inside each cell.

End the loop (meaning put the other bracket in there. And of course end the table.

```
echo "</table>";
```

Our code up to this point:

```
<?php

$cols = 7;
$rows = 4;

$total = $cols * $rows;

echo "<table border=\"1\">\n";
for ($i=0;$i<$total;$i++) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }
  echo "<td>$i</td>";
}

echo "</table>";

?>
```

Upload and try it out. You should have a table with 4 rows. The top row counts from 0 to 6, the second from 7 to 13, all the way up to 27. (Making a total of 28 since we started counting at 0).

I see two problems here, however. If you look at the HTML source, firstly there's something like this:

```
<table border="1">
</tr>
```

Copyright © 2003 Robert Plank

What?! We're ending a row before we've even started one! Bad HTML. Very bad. Fortunately it's easy to fix.

See this line?

```
if ($i % $cols == 0) { echo "</tr>\n\n"; }
```

Just change it to...

```
if ($i > 0 && $i % $cols == 0) { echo "</tr>\n\n"; }
```

This says if $i is a number bigger than 0 *AND* it divides into 7, end the current row. This way, at 0, a new row will be created, but a row won't be closed beforehand (because there is none).

The second part, is at the end, which is just a simple matter of ending the last row in the table.

Find this line:

```
echo "</table>";
```

Put this line ABOVE it:

```
echo "</tr>\n";
```

All right. Now it's fixed. Still not done yet, though.

```
<?php

$cols = 7;
$rows = 4;

$total = $cols * $rows;

echo "<table border=\"1\">\n";
for ($i=0;$i<$total;$i++) {
```

Copyright © 2003 Robert Plank

```
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }
  echo "<td>$i</td>";
}
echo "</tr>\n";
echo "</table>";

?>
```

As I stated earlier, we need some way of finding out how many days are in the month. That's somewhat easy with the date() function, once you get the hang of it.

First we have to say what month and year we're going to do. I'm going to do September of this year, since that's the month my birthday lands on. In PHP, the month goes from 01 to 12. September is the 9th month of the year, so it's 09.

```
$month = "09";
$year = "2003";

$total = date("t", mktime(0,0,0,$month, 1, $year));
```

This tells us how many days are in the current month. If you want to go and tear this line of code apart, head on over to the PHP web site and look at the documentation on the date() and mktime() functions.

But basically, I tell the date() function: "We're going to be looking at September 1, 2003 for the moment. During this time, how many days are in the month?" And it tells us... in our case, 30.

How is this applied to our example? Well, if you look at that above code carefully you'll see that we didn't even need to give it the number of rows.

All we *needed* was the total, which we were able to get by multiplying the number of rows by the number of columns. Plus, since this is a calendar, the number of days in the week will *always* be seven, so we don't even have to touch that.

Copyright © 2003 Robert Plank

So, you take this line:

```
$total = date("t", mktime(0,0,0,$month, 1, $year));
```

And put it in the place of that other line with $total. You can completely get rid of the line where we define $rows (not the line where we define $cols though!!)

This is what we have up until now:

```
<?php

$cols = 7;

$month = "09";
$year = "2003";

$total = date("t", mktime(0,0,0,$month, 1, $year));

echo "<table border=\"1\">\n";
for ($i=0;$i<$total;$i++) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }
  echo "<td>$i</td>";
}
echo "</tr>\n";
echo "</table>";

?>
```

Oops, but if you tried it you'd see that the days start counting on 0 and not 1. That's not right!! So change this line:

```
echo "<td>$i</td>";
```

To this:

Copyright © 2003 Robert Plank

```
echo "<td>".($i+1)."</td>";
```

And here is our work so far:

```
<?php

$cols = 7;

$month = "09";
$year = "2003";

$total = date("t", mktime(0,0,0,$month, 1, $year));

echo "<table border=\"1\">\n";
for ($i=0;$i<$total;$i++) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }
  echo "<td>".($i+1)."</td>";
}
echo "</tr>\n";
echo "</table>";

?>
```

Upload it and try. Everything *looks* right, for a moment, until we realize that not all months start on Sunday. Whoops.

We need to figure out what day of the week the 1st will be on. We need another similar to that line where we figured out the number of days in the month. A line like this:

```
$firstday = date("w", mktime(0,0,0,$thismonth, 1, $year));
```

This will tell you the day of the week of any date... for example, September 5th or 23rd. But, this time we asked, "What day of the week is it on September 1st?"

Now, PHP tells this to us in any number from 0 to 6, where 0 means Sunday and

Copyright © 2003 Robert Plank

6 means Saturday. This actually works out quite well for us because *our* calendar starts at 0 as well. So, we know when to start counting.

First of all, that for loop isn't going to do it for us anymore. It needs to be a while loop now. (We learned how to change a for loop to a while loop in article 14.) So we take it:

```
for ($i=0;$i<$total;$i++) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }
  echo "<td>".($i+1)."</td>";
}
```

And make it into this:

```
$i=0;
while ($i<$total) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }
  echo "<td>".($i+1)."</td>";
  $i++;
}
```

I hope you'll still with me here. Because next, I want you to take this line:

```
echo "<td>".($i+1)."</td>";
```

And change it to this:

```
echo "<td>".($i-$firstday+1)."</td>";
```

Get what this is? In the month of September, for instance, the value of $firstday is 1. We begin counting, 0 minus 1 equals -1, 1 minus 1 equals 0, which is where we start. This way we actually start when the month starts as well.

Of course we add the one in there because we want to count the days of the month as 1, 2, 3, etc. and not 0, 1, 2...

Copyright © 2003 Robert Plank

In order to hide the dates there with negative numbers, we change that little line yet again to:

```
if ($i < $firstday) { echo "<td> </td>"; }
else { echo "<td>".($i-$firstday+1)."</td>"; }
```

So if we aren't yet ready to show the dates, meaning it's not that day of the week yet, we just put an empty cell in there.

We adjust the limit so it actually counts the entire month:

```
while ($i<$total) {
```

Becomes:

```
while ($i<$total+$firstday) {
```

What we have now is:

```
<?php

$cols = 7;

$month = "09";
$year = "2003";

$total = date("t", mktime(0,0,0,$month, 1, $year));
$firstday = date("w", mktime(0,0,0,$month, 1, $year));

echo "<table border=\"1\">\n";
while ($i<$total+$firstday) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }

  if ($i < $firstday) { echo "<td> </td>"; }
  else { echo "<td>".($i-$firstday+1)."</td>"; }
```

Copyright © 2003 Robert Plank

```
  $i++;
}

echo "</tr>\n";
echo "</table>";

?>
```

One thing you may notice about this is that the last row is missing some cells. So, what we've got to do to correct this is to create another while loop afterwards, and just keep increasing $i by one and throwing an empty table in there, until we reach the end of the row and can close it. Like this:

```
while ($i % $cols > 0) {
  echo "<td> </td>";
  $i++;
}
```

As long as $i isn't a multiple of 7 (or zero) we'll have to fill in those empty rows up until the end.

You've just made a calendar for the month of September 2003. Try changing the value of $month or year and it can show other months and dates. In fact, here's some extra goodies I've added that will show you the month and year at the top:

```
<?php

$cols = 7;

$month = "09";
$year = "2003";

$monthlist = array("January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December");

$total = date("t", mktime(0,0,0,$month, 1, $year));
$firstday = date("w", mktime(0,0,0,$month, 1, $year));
```

Copyright © 2003 Robert Plank

```
echo "<table border=\"1\">\n";

echo "<tr><td colspan=\"$cols\">".$monthlist[$month-1]."
".$year."</td></tr>\n";
while ($i<$total+$firstday) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }

  if ($i < $firstday) { echo "<td> </td>"; }
  else { echo "<td>".($i-$firstday+1)."</td>"; }
  $i++;
}

while ($i % $cols > 0) {
  echo "<td> </td>";
  $i++;
}
echo "</tr>\n";
echo "</table>";

?>
```

Assignment

Look up the is_int() function on php.net and figure out how to make this same calendar, using division and is_int() instead of modulus.

QUIZ

1. If you have a cell in a table that you want empty but you still want to *appear*, what should you fill it with? (Hint: non-breaking space)

Copyright © 2003 Robert Plank

A: <!-- space -->
B:
C: "
D: Fill it with just a plain space.

2. How do we figure out what day of the week a month starts on?
A: Guess.
B: Use date(), and check the day of the week for the 1st of that month.
C: Use date(), and check the day of the week for the last of that month.
D: Use the built-in function dayofweek().

3. How is a quote represented within a quote?
A: It isn't.
B: \"
C: /"
D: ""

4. True or false: A statement such as this is legal:
echo $myarray[$somevalue+3];
A: True.
B: False.

5. What function tells how many days are in a particular month?
A: date()
B: mkdate()
C: mktime()
D: time()

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 16 Functions With PHP

I've shown you how to use dozens of functions during your quest to become a master at PHP. For instance, aren't you so glad there's a mktime() function to compute the time for you? Good luck trying to do it manually... eek...

But first, to wrap up last week's calendar.

So you're thinking, I've spent all this time making a calendar, so now what? What good is it to me? Well, since we haven't even begun to cover mySQL you're going to have to look into that on your own. BUT, there is time to show you a really simple way to do it.

This isn't the most "correct" way, but it'll get you started.

This is the calendar from last week:

```
<?php

$cols = 7;

$month = "09";
$year = "2003";

$monthlist = array("January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December");

$total = date("t", mktime(0,0,0,$month, 1, $year));
$firstday = date("w", mktime(0,0,0,$month, 1, $year));

echo "<table border=\"1\">\n";
```

Copyright © 2003 Robert Plank

```
echo "<tr><td colspan=\"$cols\">".$monthlist[$month-1]."
".$year."</td></tr>\n";
while ($i<$total+$firstday) {
  if ($i % $cols == 0) { echo "</tr>\n\n"; }
  if ($i % $cols == 0) { echo "<tr>\n"; }

  if ($i < $firstday) { echo "<td> </td>"; }
  else { echo "<td>".($i-$firstday+1)."</td>"; }
  $i++;
}

while ($i % $cols > 0) {
  echo "<td> </td>";
  $i++;
}
echo "</tr>\n";
echo "</table>";

?>
```

Now, let's pretend you had, say, newsletter archives. And, for the sake of argument let's say all your archives were named by date. There are a million ways you could do this, but here's an example filename for September 23, 2003:

2003.09.23.html

Lucky for us, there's a built-in function PHP called file_exists(). As you might guess, it takes a file name and then looks to see if the file is actually there. For the purposes of this example I'm assuming that the files which contain your archives exist in the same directory this calendar script is running.

First I'll set a variable called $thefile in each loop which sets the hypothetical file name to that format. (So as it goes through the month, $thefile will be 2003.09.1.html, 2003.09.2.html, etc.)

```
$thefile = $year.".".$month.".".($i-$firstday+1).".html";
```

Copyright © 2003 Robert Plank

Now that we know what file name to look for, it's just a simple matter of using file_exists() with it.

See this block?

```
if ($i < $firstday) { echo "<td> </td>"; }
else { echo "<td>".($i-$firstday+1)."</td>"; }
```

Make it this:

```
if ($i < $firstday) { echo "<td> </td>"; }
elseif (file_exists($thefile)) {
  echo "<td><a href=\"$thefile\">".($i-$firstday+1)."</a></td>";
}
else { echo "<td>".($i-$firstday+1)."</td>"; }
```

So, if we've begin showing the dates, first we check to see if the appropriate filename exists. If it does, then show the date number *with a link to the article*.

Otherwise, just show the article date.

But here's the good part. Now we're going to turn this calendar into a function! Oh yes, it'll be so much fun!

Start off by removing the lines that set $month and $year.

Next, put this code in:

```
function drawCalendar($month,$year) {
}
```

Now put all that code of yours inside the brackets.

Here's what we got:

```
<?php
```

Copyright © 2003 Robert Plank

```
function drawCalendar($month,$year) {

  $cols = 7;

  $monthlist = array("January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December");

  $total = date("t", mktime(0,0,0,$month, 1, $year));
  $firstday = date("w", mktime(0,0,0,$month, 1, $year));

  echo "<table border=\"1\">\n";

  echo "<tr><td colspan=\"$cols\">".$monthlist[$month-1]."
".$year."</td></tr>\n";
  while ($i<$total+$firstday) {
    if ($i % $cols == 0) { echo "</tr>\n\n"; }
    if ($i % $cols == 0) { echo "<tr>\n"; }

    $thefile = $year."-".$month."-".($i-$firstday+1).".html";

    if ($i < $firstday) { echo "<td> </td>"; }
    elseif (file_exists($thefile)) {
      echo "<td><a href=\"$thefile\">".($i-$firstday+1)."</a></td>";
    }
    else { echo "<td>".($i-$firstday+1)."</td>"; }
    $i++;
  }

  while ($i % $cols > 0) {
    echo "<td> </td>";
    $i++;
  }
  echo "</tr>\n";
  echo "</table>";
}
```

Copyright © 2003 Robert Plank

```
?>
```

If you run it, nothing will happen, because it's inside a function now. The code inside of that doesn't run automatically now. Think of it like the code for that calendar is now inside a box, and we have to take the lid of the box off first to see what's inside.

This isn't any old box, however. This is a magic box. We can put things into the box, and, depending on what we put in, different things will come out. Look at that first line we put in there for example:

function drawCalendar($month,$year)

Why did we take those lines out of our script that defined what $month and $year were? Because, we'll be putting those *into* the function we've just called "drawCalendar".

We're making our code reusable.

Try it out... put a line like this at the top of the file:

```
drawCalendar("09", "2003");
```

This calls our function called drawCalendar, giving the parameter $month a value of "09" and the parameter year a value of "2003".

Now, change it to:

```
drawCalendar("10", "2003");
```

Or:

```
drawCalendar("5", "2015");
```

Are you beginning to see now? We could use that calendar function over and over and over... say, for example, loop through the 12 months of the year, and be able to show one page full of twelve small calendars for the entire year. You can

Copyright © 2003 Robert Plank

do that on your own time... ;-)

Pretty neat, though, huh? Sure beats writing out all that code again and again.

Function names are case-sensitive. So, if you have a function called "drawCalendar" and you try to say "drawcalendar("5", "2015");", it won't work. They need to look exactly the same.

That takes care of functions that modify things. But there are other sorts of functions. Ones that, instead of going out to do something, give something back. Think of a function like someone who will do whatever you want. Now, you can, for example, send someone out to pick up groceries for you. Or take your car down to the Jiffy-Lube to get your oil changed. These are like those modifying functions.

We can also have functions that access things. For example, you might have one of your incarnated functions read the newspaper and tell you what's in it. The function didn't change anything -- it didn't write on the newspaper or tear it into shreds... it just passively looked for you.

How do we make functions like this? Well, pretty much in the same way as with our last example. This is a simple function called readthis() that reads the contents of a file, and then returns that data.

```
function readthis($file) {
  $fp = fopen($file, "r");
  $data = fread($fp, filesize($file));
  fclose($fp);
  return $data;
}
```

What good is this? Well, if we just did:

```
readthis("somefile.txt");
```

Nothing would happen. We need to put this data somewhere. Put if we did something like this:

Copyright © 2003 Robert Plank

```
$sometext = readthis("somefile.txt");
```

This would read the contents of a file called "somefile.txt" and put it all into a variable called $sometext. Neat, huh?

We could also do:

```
echo readthis("somefile.txt");
```

And it would output that file right to the browser. Useful, isn't it?

Assignment

Look up functions on php.net. Add default values to the parameters of the calendar function so that if the function is called with no parameters, such as:

drawCalendar()

It defaults to the current month and year. Use date() and mktime() to compute the current month and year.

QUIZ

1. True or false: Is it possible to have functions within functions? (Look up the documentation on functions at php.net.)
 A: True.
 B: False.

2. Which of the following is NOT TRUE about using functions in your scripts?
 A: They provide reusable code.
 B: Functions make it easier to iterate (loop) through things.
 C: Using functions gives you a faster compile time.
 D: Data returned from a function can be placed into a variable.

Copyright © 2003 Robert Plank

3. True or false: Functions can accept arrays or objects as parameters.
A: True.
B: False.

4. True of false: Functions can only return strings or integers.
A: True.
B: False.

5. True or false: The function file_exists() automatically creates the file if it does not exist.
A: True.
B: False.

Click Here For Correct Answers

Copyright © 2003 Robert Plank

Chapter 17 Saving With PHP

It's time for the 17th and final chapter.

John Calder of theEzine.net gave this idea to me. He runs a newsletter (theEzine.net) and had a ton of archived newsletters he needed to turn into HTML format. (John actually runs a similar script to that calendar example you saw in article #15...)

So, say you're in a similar predicament. You have a lot of newsletter archives... or some other sort of documents that you need turned into HTML. Of course, you could do a little search and replace, but if you have a LOT of documents to save as HTML, that's quite a bit of work. I don't know about you, but if those were my newsletter archives, I might want to add a header and footer... not only that, but maybe a title tag, a header, maybe some sort of sub header...

So, we're going to make something like that. Ours is going to make it totally based on templates (similar to the affiliate script) so you can use it with anything, add your own variables and so on.

BEFORE WE START: Keep in mind that if you're on a Unix host you'll have to chmod the folder these files will be written in, to 0777.

I'm going to start off my header.txt like this:

```
<html>
<head>
<title><%title%></title>
</head>

<body>
<h1><%header%></h1>
```

Copyright © 2003 Robert Plank

<%subheader%>

And my footer.txt will contain:

</body>
</html>

Now, the first step is to read our header and footer templates and take those <%variable%> things out of them. It is at these spots where we will actually be filling things in later.

For now let's make a variable called $thefile and have it equal to the value of $header. You'll see why I'm doing this in a sec.

```
$thefile = $header;
```

Then of course read the contents of the file just as we've done so many times before.

```
$fp = fopen($thefile,"r");
$contents = fread($fp,filesize($thefile));
fclose($fp);
```

At this point I want an empty array. Let's call it $new.

```
$new = array();
```

This is the array we'll be putting the variable names into. (For example, "subheader" and "header".)

Then, explode the string by <%. Why is this? Well, this way the array will be separated each time there is a <%.

Right now the array called $contents looks like this.

Array
(

Copyright © 2003 Robert Plank

```
  [0] => <html>
<head>
<title>
  [1] => title%></title>
</head>

<body>
<h1>
  [2] => header%></h1>
<b>
  [3] => subheader%></b><br><br>
)
```

Eww! What a mess. But we're on the right track. Think about it though... we've already split the array up where ever there's an <%... so if we get rid of everything after the %> on each line (where the variable name ends)... all that will be left are our variable names.

Here, I'll show you.

Loop through the array and as we go, take out everything after the %>. Leave only the variable name.

```
for ($i=0;$i<count($contents);$i++) {
  $contents[$i] = eregi_replace("%>.*$","",$contents[$i]);
}
```

Only, instead of putting each piece back into $contents, I want to put it into another array... that array we created earlier called $new.

Now here's a useful trick if you want to keep piling things onto an array. Right now our array is of size 0, right? Because there are 0 elements inside the array. So the value of count($new); is 0.

If we fill something into the element 0 of the array $new, count($new) will give us 1. See how that works out?

Copyright © 2003 Robert Plank

So, if you want to add something to the end of an indexed array, for example, at the end of our array called $new just add it to:

$new[count($new)]

Pretty neat, huh? I think so. So, as I said I want to add each piece into our array called $new.

```
for ($i=0;$i<count($contents);$i++) {
  $new[count($new)] = eregi_replace("%>.*$","",$contents[$i]);
}
```

Here's $new now:

```
Array
(
   [0] => <html>
<head>
<title>
   [1] => title
   [2] => header
   [3] => subheader
)
```

Oops! Now that first element in the array shouldn't be in there. So, we need to put an if-statement inside our loop to make sure that we ONLY want to do this if we're really dealing with a variable here. (Meaning there's a "%>" in there somewhere.

If you remember back to article 10 where we did regular expressions, this is pretty easy. So we add this if-statement around it:

```
if (eregi("%>",$contents[$i]))
```

And this is our entire loop:

```
for ($i=0;$i<count($contents);$i++) {
```

Copyright © 2003 Robert Plank

```
  if (eregi("%>",$contents[$i])) {
    $new[count($new)] = eregi_replace("%>.*$","",$contents[$i]);
  }
}
```

If you ran this code, the array $new would look like this:

```
Array
(
   [0] => title
   [1] => header
   [2] => subheader
)
```

Hey! That's just what we want.

Uh oh! But this only takes care of the header. What if there are variables in the footer. Do we have copy and paste this code so it's there twice? No, make a function. Because I said so, that's why.

I'm going to call the function getVars(). All we need to do is put function getVars() { ... } around our code, and then add a return $new to the end.

This is everything:

```
<?php

$header = "header.txt";
$footer = "footer.txt";

function getVars($thefile) {

  $fp = fopen($thefile,"r");
  $contents = fread($fp,filesize($thefile));
  fclose($fp);

  $new = array();
```

Copyright © 2003 Robert Plank

```
  $contents = explode("<%",$contents);

  for ($i=0;$i<count($contents);$i++) {
    if (eregi("%>",$contents[$i])) {
       $new[count($new)] = eregi_replace("%>.*$","",$contents[$i]);
    }
  }

  return $new;

}

?>
```

So how do we use this now? Simple:

```
$headerlist = getVars($header);
$footerlist = getVars($footer);
```

And there it is. Now $headerlist is an array containing the names of all the variables in the header. Likewise with the footer.

These are two separate arrays. We want everything in one array. array_merge() to the rescue.

```
$list = array_merge($headerlist,$footerlist);
```

I don't want duplicates in the array called $list so we'll use array_unique() to eliminate any duplicate values.

```
$list = array_unique($list);
```

And then fill in any empty spaces in the array if anything was taken out.

```
$list = array_values($list);
```

Copyright © 2003 Robert Plank

It's time to generate our HTML form. This is going to use the variable list from $list to make a series of text fields where we can fill in the info we need later.

First I want to post this data to a file we haven't made yet, called write.php, and also send a hidden value named $redirect to send us right back here where we started.

```
echo "<form action=\"write.php\" method=\"post\">\n";
echo "<input type=\"hidden\" name=\"redirect\" value=\"save.php\">\n";
```

Oh, I also want to pass the values of $header and $footer to this other script.

```
echo "<input type=\"hidden\" name=\"header\" value=\"$header\">\n";
echo "<input type=\"hidden\" name=\"footer\" value=\"$footer\">\n";
```

Start a table.

```
echo "<table>\n";
```

Now let's loop through the array, and as we do, make a new row and put in two columns: column one will be some text saying what the variable is we want to fill, while column two will be the text field we will actually fill in.

```
for ($i=0;$i<count($list);$i++) {
  echo "<tr>\n";
  echo "<td>$list[$i]:</td>\n";
  echo "<td><input type=\"text\" name=\"list[".$list[$i]."]\"></td>\n";
  echo "</tr>\n\n";
}
```

I want to add in another row with textarea field where we paste the newsletter into.

```
echo "<tr>\n";
echo "<td colspan=\"2\"><textarea name=\"contents\" cols=\"50\"
rows=\"8\"></textarea></td>\n";
echo "</tr>\n\n";
```

Copyright © 2003 Robert Plank

Then, another text field asking us the filename to save this as:

```
echo "<tr>\n";
echo "<td>Save As:</td>\n";
echo "<td><input type=\"text\" name=\"saveas\"></td>\n";
echo "</tr>\n\n";
```

Close the table.

```
echo "</table>\n";
```

Add a submit button.

```
echo "<input type=\"submit\" name=\"submit\" value=\"Save\">";
```

Close the form.

```
echo "</form>";
```

There you are.

Here is the sample script save.php:

```
<?php

$header = "header.txt";
$footer = "footer.txt";

$headerlist = getVars($header);
$footerlist = getVars($footer);

$list = array_merge($headerlist,$footerlist);
$list = array_values(array_unique($list));

echo "<form action=\"write.php\" method=\"post\">\n";
echo "<input type=\"hidden\" name=\"redirect\" value=\"save.php\">\n";
```

Copyright © 2003 Robert Plank

```
echo "<table>\n";
for ($i=0;$i<count($list);$i++) {
  echo "<tr>\n";
  echo "<td>$list[$i]:</td>\n";
  echo "<td><input type=\"text\" name=\"list[".$list[$i]."]\"></td>\n";
  echo "</tr>\n\n";
}

echo "<tr>\n";
echo "<td colspan=\"2\"><textarea name=\"contents\" cols=\"50\"
rows=\"8\"></textarea></td>\n";
echo "</tr>\n\n";

echo "<tr>\n";
echo "<td>Save As:</td>\n";
echo "<td><input type=\"text\" name=\"saveas\"></td>\n";
echo "</tr>\n\n";

echo "</table>\n";

echo "<input type=\"submit\" name=\"submit\" value=\"Save\">";

echo "</form>";


function getVars($thefile) {

  $fp = fopen($thefile,"r");
  $contents = fread($fp,filesize($thefile));
  fclose($fp);

  $new = array();

  $contents = explode("<%",$contents);

  for ($i=0;$i<count($contents);$i++) {
```

Copyright © 2003 Robert Plank

```
    if (eregi("%>",$contents[$i])) {
       $new[count($new)] = eregi_replace("%>.*$","",$contents[$i]);
    }
  }

  return $new;

}

?>
```

What's left? Well, we submitted this to write.php so we sure have to do something with it. Start over in a new file.

I've given you our function from chapter 16, to make your reading a file into a variable easier. Use it.

```
<?php

function readthis($filename) {
  $fp = fopen($filename,"r");
  $data = fread($fp,filesize($filename));
  fclose($fp);
  return $data;
}

?>
```

The variables $header and $footer that we passed were just filenames, now. So use the readthis() function to read the contents of the file $header into a variable called $headtext. Likewise for $footer, which I want you to put into $foottext.

```
$headtext = readthis($header);
$foottext = readthis($footer);
```

We have to loop through the array we've posted called $list. If you recall how we looped through associative arrays in chapter 10, it's like this:

Copyright © 2003 Robert Plank

foreach($data as $key => $value)

As we do this, we have to do an eregi_replace() on all the variables in the header and footer, as we loop through this array.

In fact I think we did almost this exact same thing in article 10. So let's just steal code from that article:

```
foreach($data as $key => $value) {
  $text = eregi_replace("<%".$key."%>",$value,$text);
}
```

Only this needs to be changed to match the variable names we have now. Plus, we have to do one replace on the header and another on the footer. So this becomes:

```
foreach($list as $key => $value) {
  $headtext = eregi_replace("<%".$key."%>",$value,$headtext);
  $foottext = eregi_replace("<%".$key."%>",$value,$foottext);
}
```

Oh, but recall that we're changing text into HTML, so we want to use the function nl2br() which takes all the newlines in our text file and puts a break (
) so it looks okay in our browsers. This is to be used on $contents only since we assume that our header and footer are already in HTML.

```
$contents = nl2br($contents);
```

Now, mash $headtext, $contents, and $foottext together. Put this all into a variable called $text.

```
$text = $headtext.$contents.$foottext;
```

Now, let's output text to see what we've got.

```
echo $text;
```

Copyright © 2003 Robert Plank

This is what you should have so far, in write.php:

```
<?php

function readthis($filename) {
  $fp = fopen($filename,"r");
  $data = fread($fp,filesize($filename));
  fclose($fp);
  return $data;
}

$headtext = readthis($header);
$foottext = readthis($footer);

foreach($list as $key => $value) {
  $headtext = eregi_replace("<%".$key."%>",$value,$headtext);
  $foottext = eregi_replace("<%".$key."%>",$value,$foottext);
}

$text = $headtext.$contents.$foottext;

echo $text;

?>
```

Everything looks good, right? We just need to write this to a file instead of outputting it to the browser. Take out the line that says echo $text;.

The file name was (hopefully) submitted into write.php as the variable name $saveas, will be the place where we'll save the file to.

You've done this before, of course. Open whatever $saveas is set to, for writing.

```
$fp = fopen($saveas,"w");
```

Then use fwrite() to shovel $text into that file.

Copyright © 2003 Robert Plank

```
fwrite($fp,$contents);
```

Close the file.

```
fclose($fp);
```

It won't work otherwise, so consider yourself warned.

Lastly, we want to redirect our people to whatever the value of $redirect is, and then we die.

```
header("Location:$redirect"); die();
```

So, here's our stuff:

```
<?php

function readthis($filename) {
  $fp = fopen($filename,"r");
  $data = fread($fp,filesize($filename));
  fclose($fp);
  return $data;
}

$headtext = readthis($header);
$foottext = readthis($footer);

foreach($list as $key => $value) {
  $headtext = eregi_replace("<%".$key."%>",$value,$headtext);
  $foottext = eregi_replace("<%".$key."%>",$value,$foottext);
}

$contents = nl2br($contents);

$text = $headtext.$contents.$foottext;
```

Copyright © 2003 Robert Plank

```
$fp = fopen($saveas,"w");
fwrite($fp,$contents);
fclose($fp);

header("Location:$redirect"); die();

?>
```

Done.

Assignment

What happens if you try to make a new text file but that folder isn't chmoded to 0777?

QUIZ

1. True or false: It's possible to create an empty array.
 A: True.
 B: False.

2. To connect two arrays together, use:
 A: array_values()
 B: array_merge()
 C: array_unique()
 D: None of the above.

3. To convert an associative array to an indexed array, or to "fill in" the missing elements in an indexed array, use:
 A: array_values()
 B: array_merge()
 C: array_unique()

Copyright © 2003 Robert Plank

D: None of the above.

4. To eliminate duplicates from an array, use:
A: array_values()
B: array_merge()
C: array_unique()
D: None of the above.

5. To search through an array, use:
A: array_values()
B: array_merge()
C: array_unique()
D: None of the above.

Click Here For Correct Answers

This concludes our tutorial. Hopefully, you have learned enough PHP to bravely go out and conquer new worlds!

Copyright © 2003 Robert Plank

Quiz Answers

Chapter 1

1. [B] [text]$onion[/text]
2. [A] True.
3. [A] [text]echo "$one two $three";[/text]
4. [D] [text]http://www.your.host/sales.php?monkeys=20&fork=tasty[/text]
5. [A] I won't see anything.

Chapter 2

1. [D] I won't see anything because the filename doesn't end in ".php"
2. [A] [text]include("hello.txt");[/text]
3. [C] Causes the script to give up right there and then.
4. [B] False.
5. [A] True.

Chapter 3

1. [A] The variable $email, containing whatever is written in the text box.
2. [A] a
3. [C] It redirects to [text]http://www.simplephp.com[/text]
4. [A] True.
5. [B] Tell the recipient's e-mail client that the message is in plain text.

Chapter 4

1. [A] True.
2. [B] Base 64
3. [C] [text]fopen()[/text] and [text]fread()[/text]
4. [A] ".=" adds onto the end of a variable's value while "=" just sets a new value.
5. [A] True.

Copyright © 2003 Robert Plank

Chapter 5

1. [A] True.
2. [A] PHP scripts run only on the server, while JavaScript is run on the visitor's machine.
3. [C] [text]alert()[/text]
4. [D] Something the author made up.
5. [C] The plus sign.

Chapter 6

1. [A] True.
2. [A] True.
3. [A] True.
4. [D] The period.
5. [C] [text]urlencode()[/text] and [text]urldecode()[/text]

Chapter 7

1. [A] element
2. [B] Only once in the script.
3. [A] True.
4. [B] Zero.
5. [A] True.

Chapter 8

1. [C] Reads a file into an array separated by each line.
2. [A] print readable
3. [B] It loops through each element in the array [text]$myarray[/text]
4. [D] [text]explode()[/text]
5. [A] Puts a [text]
[/text] tag at the end of each line so a file is readable in HTML.

Chapter 9

1. [A] Returns the number of seconds since January 1, 1970.

Copyright © 2003 Robert Plank

2. [C] Always rounds down.
3. [A] Allows us to pass data from an HTML document into a PHP script, without that data being displayed to the user.
4. [B] The remainder of some number.
5. [D] Takes a date or length of time written in plain English and converts it to Unix time.

Chapter 10

1. [C] Indexed arrays contain numbered elements (0, 1, 2, etc.) while associative arrays contain non-numbered elements (like "toast", "bacon", etc.)
2. [A] [text]ereg()[/text] only finds a match,[text]ereg_replace()[/text] makes a replacement at that match.
3. [A] [text]eregi()[/text] doesn't care if a matching pattern is uppercase or lowercase.
4. [B] Associative arrays.
5. [A] Everything to the right of the question mark in the URL (eg, [text]script.php?junk[/text])

Chapter 11

1. [B] Writing only.
2. [A] "[text]\r[/text]" means carriage return, "[text]\n[/text]" means newline.
3. [B] Chmod the file to 0777.
4. [A] [text]in_array()[/text]
5. [C] Eliminates whitespace (extra spaces, line feeds, carriage returns, etc.) from the ends of a string.

Chapter 12

1. [C] Set the cookie's date to a point in the past.
2. [B] Formats a Unix timestamp into a readable date.
3. [D] [text]gmdate()[/text] gives us a formatted date in Greenwich Mean Time, [text]date()[/text] is only the local time.
4. [A] True.
5. [A] True.

Copyright © 2003 Robert Plank

Chapter 13

1. [B] False.
2. [B] False.
3. [B] The size of the file [text]$somefile[/text], in bytes.
4. [C] [text]file()[/text] separates each line of a file but doesn't get rid of newline characters.
5. [A] A total of the number of elements contained within [text]$myarray[/text].

Chapter 14

1. [D] Increases the value of $george by 1.
2. [A] 50.
3. [C] Three.
4. [A] Division.
5. [B] 1 2 4 8 16

Chapter 15

1. [B] [text] [/text]
2. [B] Use [text]date()[/text], and check the day of the week for the 1st of that month.
3. [B] [text]\"[/text]
4. [A] True.
5. [A] [text]date()[/text]

Chapter 16

1. [A] True.
2. [C] Using functions gives you a faster compile time. (remember, this is the INCORRECT answer of the bunch)
3. [A] True.
4. [B] False.
5. [B] False.

Copyright © 2003 Robert Plank

Chapter 17

1. [A] True.
2. [B] [text]array_merge()[/text]
3. [A] [text]array_values()[/text]
4. [C] [text]array_unique()[/text]
5. [D] None of the above.

Another eBookWholesaler Publication

Copyright © 2003 Robert Plank