Developing a User Personalization System with PHP and Cookies

By Duncan Lamb

Table of Contents

Introduction.....1

Grabbing Headlines.....2

User Login.....4

Reading from Cookies.....6

Conclusion.....8

Introduction

Everyone is looking for ways to make their sites attract more repeat traffic. Even after you have content, bulletin boards and search functions, there's still more you can do. As you probably know, different people that come to your site probably come for different parts of your site, but there's no way to please everyone, is there?

This tutorial will show you a way to give it a shot. Using mainly PHP3 and MySQL, we'll construct a small site that uses cookies to detect specific users who have visited the site before, and customize a page for them based on things they have indicated they would like to see, and do it without prompting. In other words, we're going to construct a smart, responsive site with techniques easily adaptable to just about any concept for a site you can think of.

In this example, we will take some headlines from a couple of sites some people may visit often, then make a form to collect logins and preferences, and store it all in a database. Every time the user returns, a script reads their cookie, retrieves their preferences, and builds a page showing what they want to see. We'll start out with a smattering of Perl to help us automate collecting those headlines.

Grabbing Headlines

To grab headlines of some popular news sites, we'll use Perl, undoubtedly the workhorse when it comes to searching through text (or html) files. The script is easily adaptable for use on other sites, as you'll see later on.

The way we will grab the headlines is to fetch the page they reside on, then parse the html, looking for a pattern which indicates the beginning of a headline (normally a font size or color declaration). Then the script will save the text after that match, and before a match indicating the end of the headline.

Probably the easiest example to use are the headlines off of Slashdot. Slashdot encourages this sort of thing (within reason) by providing a page with the essential information describing each article. You can view this file here: http://slashdot.org/slashdot.xml

Here's the whole script:

```
#!/usr/bin/perl $pagename="Slashdot"; $newsurl ="www.slashdot.org/slashdot.xml";
$homeurl="http://slashdot.org/"; $file="slashdot.lnk"; $before = "<title>"; $after = "<";
$webdog = "story"; #don't search till this is found #$' is post match, $` is prematch. @lines =
`perl webget.pl −q $newsurl`; #First line: make it proper home page URL @headlines[0] =
"<a href=\"$homeurl\" class=\"newstable\"><b>$pagename</b><font size=1>\n"; $found =
0; $count = 0; foreach $line (@lines) { if ($line =~ /$webdog/i) { $found = 1;} if (($found)
and ($line =~ /$before/i)) { $_ = $'; #grabs everything after match /$after/i; $headline = $`;
push (@headlines,("<br>".$headline."\n")); #make all font changes, colors, etc. on this line
$count++; last if ($count == 5); } } push (@headlines,"</font></a>"); #print @headlines;
open (FILE, ">$file"); foreach $headline(@headlines){ print FILE $headline; } close FILE;
```

This file constructs a very small file with just the headlines that fits nicely into a table cell. Lets step through this file a chunk at a time to better understand what is going on.

Lines 3−9 define the variables we'll use to grab the headlines. `$before` and `$after` hold stings of text which are before and after the headlines −− the text between these two matches on the same line will be grabbed as the headline. `$webdog` is a variable put in to make searches a little faster the real search doesn't start until this tag is reached.

Line 13 makes a system call to the webget script (with the "quiet" modifier) and puts the results in an array, `@lines`. I chose `webget.pl` because it is a single script which requires no external libraries, and is freely available here: http://www.ee.calpoly.edu/~jcline/ee/webget.pl. Similar results can be achieved using the LWP library and its "GET" function. At this point, the entire html file of the remote news page is the `@lines` array, and now we can begin to manipulate it as we wish.

Line 16 sets array element `$headlines[0]` with the title of the page, font sizes, etc. On 18−19, `$found` is a flag for if the search for the headlines can begin, while `$count` will keep track of how many headlines have been found.

Line 18 begins the loop that searches line by line through the page for headlines. Each line is checked to see if

there is a match for our `$webdog` variable. If there is no match, the next line is checked. When a match is finally found, the `$found` flag is changed to "1".

Once the `$found` flag is set, the script looks for the `$before` text in each line (Line 21). If the text matches, we grab all the text **after** that match with the `$'` function (also called `$POSTMATCH`). The two functions used here a very useful, and allow the script to grab a string we don't necessarily have a match for (like constantly changing headlines). Here are the functions:

`$'` grabs text **after** a successful match. `$`` grabs text **before** a successful match.

Using these together allow us to grab a headline by knowing the html tags on either side. On Line 22 we place the text after the first match into the default variable (`$_`), match the tag after the headline, then strip the headline (which is before that last match) and put it in the `$headline` variable. Whew!

Now that the headline is in an easily handled variable, we just push it onto the `@headlines` array and add a
 for readability (Line 25).

Line 26 and 27 are used to limit the number of headlines we grab. Once a limit to the number of headlines is reached (in this case 5), the script breaks the loop. If this limit is never reached, the loop will end after all the lines in the file have been examined.

Line 30 on are for housekeeping. The open font and link tags are closed on line 30. Line 31 is commented out for normal operations, but is very useful when debugging your match choices at the command line. After that, we print the `@headlines` array to the file we specified in `$file` at the beginning of the script.

Note that this is a very simple script that takes you right to the news page. It would be fairly easy to have a separate url for each story, or even more features. And it's easy to customize for other sites by changing the variables at the top of the script.

The script creates a file for two reasons: the content will be used many times, and site owners usually don't mind a query every hour or so, but it could be a bit much if your site gets a lot of traffic, and this script is run every time. So the best thing to do with this script, and others you customize to get news from other sites, is to put them in a cron job. After you have a few ascripts, each regularly polling a site, you should have several small text files being regularly produced with their headlines. To keep everything organized, put all of your headline−grabbing scripts into a subdirectory called "news".

In this example, we will take some headlines from a couple of sites some people may visit often, then make a form to collect logins and preferences, and store it all in database. Every time the user returns, a script reads his cookie, retrieves his preferences, and builds a page showing what they want to see. We'll start out with a smattering of Perl to help us automate collecting those headlines.

First, lets create the table in mysql, in a database named "`project`":

```
CREATE TABLE users ( login char(16) NOT NULL, password char(10) NOT NULL,
lastlogin date DEFAULT '0000−00−00' NOT NULL, news1 char(20), news2 char(20), news3
char(20), PRIMARY KEY (login) );
```

User Login

Now a form should be built to enter information, make choices, and then enter the data in the database. Here's a simple form with a few more choices available:

```
<HTML> <HEAD> <TITLE>Login now!</TITLE> </HEAD> <BODY> <br>Choose the
headlines you wouldlike to see!<br> <form method=post action="newuser.php3"> <br>
Please pick a login name:<input name="login" type="TEXT" value=""> <br> Now pick a
password:<input name="password" type="TEXT" value=""> <br><br> <br>News Choice
1:<SELECT NAME="news1" size=5 value=""> <option value="slashdot.lnk">Slashdot
<option value="freshmeat.lnk">Freshmeat <option
value="voodooextreme.lnk">VoodooExtreme <option value="devshed.lnk">DevShed
<option value="bluesnews.lnk">Blue's News </select> <br>News Choice 2:<SELECT
NAME="news2" size=5 value=""> <option value="slashdot.lnk">Slashdot <option
value="freshmeat.lnk">Freshmeat <option value="voodooextreme.lnk">VoodooExtreme
<option value="devshed.lnk">DevShed <option value="bluesnews.lnk">Blue's News
</select> <br>News Choice 3:<SELECT NAME="news3" size=5 value=""> <option
value="slashdot.lnk">Slashdot <option value="freshmeat.lnk">Freshmeat <option
value="voodooextreme.lnk">VoodooExtreme <option value="devshed.lnk">DevShed
<option value="bluesnews.lnk">Blue's News </select> <br><input type="submit"
name="Submit" value=" Submit "> <br></FORM> </BODY> </HTML>
```

The action in the form is "newuser.php3." Once data has been entered in the form, this script inserts a new row of data in the database.

```
<? SetCookie("visitor",$login,time()+864000); ?> <HTML>
<HEAD><TITLE>Thanks!</TITLE></HEAD> <BODY> <? mysql_connect("localhost",
"username", "pass") or DIE("Unable to connect to database"); @mysql_select_db("project")
or die("Unable to select database"); $result = mysql_query("Select * from users where
login='$login'"); if(!mysql_numrows($result)) { #Make a new login − entire page follows
$result = mysql_query("insert into users values ( '$login', '$password', NOW(), '$news1',
'$news2', '$news3')"); ?> <br> Thanks for logging in!<br><br> To see what the page looks
like with your personal settings, please <a href="index.php3">return</a>. <? } else { echo
"Whoa! Someone has already picked that username!<br><br>"; echo "Please hit the back
button and pick a unique login."; } ?> </body> </html>
```

Notice that "Setcookie" command? We'll talk about on the next page. For now, notice how the database is checked for duplicate logins first (lines 10), then a new row is created (line 12). Afterwards, a link to the start page is given.

At the end of the script, the user gets some feedback if his login is not unique.

Normally to pass variables to any sort of script, variables have to be added to the end of the URL, but what we want is a simple solution, so whenever a user comes to the page, **their** preferences will be displayed without

any mumbo jumbo at the end of the url. We still need a way to pass our scripts information about the user. The best way to do so is with cookies, and nothing handles cookies as easily as PHP3.

Reading from Cookies

Cookies are an easy, transparent way to store data on a user's computer, and to retrieve that data every time the user returns. Used wisely so they don't store sensitive data, they can allow seamless interactivity for any web application.

To create a cookie on a user's computer, use the following syntax in PHP3:

```
setcookie(cookie name, value, expire time, string path, string domain, secure flag);
```

The rules for using this function are:

1. You must declare the cookie before the HTML header in your script, since it is sent as part of the header. That means before the `<HTML>` tag.
2. You can skip entries with "", and leave off the last entries if not needed.
3. Just using "`setcookie(cookie name)`" or setting the expiration time to 0 will delete the cookie off the user's system.
4. The secure flag is a 0 or 1, and identifies whether the cookie should be transmitted over an SSL connection.
5. Expire time is expressed in seconds, and normally defined with `time()+` a number of seconds, such as `time()+86400`, if the cookie should expire in 24 hours.

Values saved in cookies by a particular site are automatically sent to that site in every header. PHP3 automatically decodes the variable and assigns it to a variable, so to access its value in PHP3, just use the variable as you would normally −− it already exists. For example, we can grab the cookie and print out the customized page we have been building for this user. The following script will grab the appropriate row from the database, and print out the page with all of their settings.

```
<? if ($visitor){ mysql_connect("localhost", "username", "pass") or DIE("Unable to connect
to database"); @mysql_select_db("project") or die("Unable to select database"); $result =
mysql_query("select * from users where login='$visitor'"); $row =
mysql_fetch_array($result); ?> <HTML> <HEAD><TITLE>Welcome to your
headlines!</TITLE> </HEAD> <BODY> <H1><CENTER>Welcome <? echo $visitor;
?>!</CENTER></H1> <br><br> The news for today is:<br> <table> <tr> <td><? include
("news/$row[news1]"); ?></td> <td><? include ("news/$row[news2]"); ?></td> <td><?
include ("news/$row[news3]"); ?></td> </tr> </table> <? } else { ?> <HTML> <HEAD>
<TITLE>Login now!</TITLE> </HEAD> <BODY> <br>Choose the headlines you would
like to see!<br> <form method=post action="newuser.php3"> <br> Please pick a login
name:<input name="login" type="TEXT" value=""> <br> Now pick a password:<input
name="password" type="TEXT" value=""> <br><br> <br>News Choice 1:<SELECT
NAME="news3" size=5 value=""> <option value="slashdot.lnk">Slashdot <option
value="freshmeat.lnk">Freshmeat <option value="voodooextreme.lnk">VoodooExtreme
<option value="devshed.lnk">DevShed <option value="bluesnews.lnk">Blue's News
</select> <br><input type="submit" name="Submit" value=" Submit "> <br></FORM> <? }
?> </BODY> </HTML>
```

This script branches between two different possibilities: the first section is executed if the user has a cookie set, at which time their persoanl settings are retrieved and used to build a page, while the last part of the script is executed when no cookie is set (this is the same form used on the last page). By making this our index file, users will automatically get the page intended for them.

PHP3 does all the dirty work, automatically reading the headers sent from the client and placing the cookie in a variable. Note how the `$visitor` variable will have a value and be true if the cookie has been set. Another way to test for a cookie is with the "isset" command, as in:

```
if (isset($visitor)){
```

PHP3 does all the dirty work, automatically reading the headers sent from the client and placing the cookie in a variable. Nothing could be easier!

If a cookie has been set, the script reads its value, grabs the appropriate row from the database, and includes the appropriate headlines. Since values stored for news choices are named the same as the filenames they are stored in, putting them into the page is as easy as:

```
include ("news/$row[news1]");
```

(Remember, we put those headlines in the "news/" subdirectory.)

The only other bit of code in this script is where the user's login name is printed out in a welcome message:

```
Welcome <? echo $visitor; ?>!
```

For an even briefer version for the same results, PHP3 will accept the ASP standby of:

```
<%= $visitor %>
```

As a matter of fact, it is only lightly documented, but the <? and ?> tags are interchangeable with the ASP tags <% and %>.

Conclusion

This is just a brief example which could be made much more robust by adding more news sources, or adding other features, such as the ability to change your choices (actually that should be a required feature, edited out here for brevity). Using these techniques, it would be fairly straightforward to pull all manner of data from a common database based on the visiting user, and print it out to their browser. It is precisely this sort of process which enables Yahoo or Excite to show your favorite stocks on their homepage, Amazon to make a pretty good guess on books you might like to read, and any number of news sites which will show current weather for your area.

Think about what people would enjoy seeing at your web site or how other web sites could improve. I personally think it would be great if everytime I went to Pricewatch, it would show me up front the best prices for whatever hardware I'm interested in that week (after I specify the hardware, that is). I'd also get a kick out of having links to updates for my favorite Football teams right on ESPN's front page, so I don't have to click through 5 pages to get there.

Making the internet experience friendlier and more intuitive is the best way to keep visitors coming back to your site. If nothing else, I'd recommend you mess around with these features yourself, because these techniques can work magic, and will definitely be in more demand in the future. Many ideas are out there which haven't been exploited yet, and many niche subjects yearn for this kind of feature.