PHPit Totally PHP

PHP and Cookies; a good mix!

By Dennis Pallett

Cookies have long been used in PHP scripts, and are a very useful function. But what exactly are cookies? Maybe you have used then, but you still don't know exactly what they are. Or you are completely new to cookies? It doesn't matter, because in this tutorial I will show you exactly what cookies are, and what they are used for.

Cookies in a nutshell

Cookies are small pieces of information that is stored on the computer of your visitors. Each browser handles it differently, but most simply store the information in a small text file. Internet Explorer has a special folder, which can be found in your C:\Windows or C:\Windows\System32 folder. You can delete all your cookies, by going to the Options and 'Clearing Cookies' or deleting them by hand. I don't recommend this though.

Almost every website uses cookies. If you go to Amazon.com, you will get several cookies. The same goes for CNN.com. Even Google uses cookies! They are extremely useful for (temporarily) storing information. For example, if you have a login system for your visitors, you could save their userid and password (very heavily encrypted!) so they are automatically logged in the next time they visit your website.

Or you could remember their last visit, and highlight everything that is new. And that's just the beginning.

Using cookies in PHP is extremely easy. In fact, there is nothing to it, because of PHP's inbuilt setcookie() function. Have a look at the documentation, and then try the following example:

```
<?php

// Set a cookie
 // Cookie name: name
 // Cookie value: Dennis Pallett
 // Cookie expire: in 24 hours

setcookie ('name', 'Dennis Pallett', time() + (60*60*24));
 ?>
```

If you run the code above, then a cookie will be set. That's all. The cookie name and value are pretty obvious. The cookie expire is when the cookie expires, or goes away. Simply use the time() function and add the number of seconds you want to have the cookie available to it. In the example I added 60*60*24=86400 seconds, or 24 hours.

If you have looked at the documentation, you probably noticed there are additional arguments. As the documentation says, the path is to limit a cookie to a specific path on your web server. This is often used when you run multiple instances of the same script in separate directories. You can safely omit this argument when it doesn't matter if the cookie is available site-wide.

There is also the domain argument. This can be used to limit the cookie to a specific sub-domain, e.g. test.example.com. You can also safely ignore this argument, or set it to .example.com (note the beginning period, this is essential!).

Finally, there is also the secure argument. This argument is only used for cookies that are sent over a secure HTTPS connection (SSL). Just ignore this argument, unless you're working with a secure connection.

One thing that should be mentioned is that cookies must be set, before you display any HTML/text. It's probably best if you turn on output buffering by putting ob_start() at the top of your page.

Now that you have set a cookie, you probably want to retrieve the value as well. After all, that is the whole point of using cookies. Thankfully, as PHP is ever so easy, you can retrieve the same way as you retrieve a GET value. See the following example to retrieve the value of the previous example:

```
<?php
 echo 'Your name is ' . $_COOKIE['name'];
 ?>
```

This should print "Your name is Dennis Pallett". There's nothing more to it. It's just that easy!

Finally, one thing you probably want to do as well is remove cookies. This is as easy as setting them. Simply change the value of the cookie to FALSE, and change the expire date to -3000 seconds. See the following example:

```
<?php
 setcookie ('name', FALSE, time()-1000);
 ?>
```

Before you start using cookies, you must make sure your visitor has cookies enabled. This can be done with a simply PHP checking script. Unfortunately, the PHP page needs to reload to check for cookies. But this can be done very transparently, and your visitor should hardly notice anything.

The following example will first set a test cookie, then reload the page, and finally check whether cookies are enabled.

```
<?php
 error_reporting (E_ALL ^ E_WARNING ^ E_NOTICE);

// Check if cookie has been set or not
 if ($_GET['set'] != 'yes') {
        // Set cookie
        setcookie ('test', 'test', time() + 60);

       // Reload page
        header ("Location: checkcookies.php?set=yes");
 } else {
        // Check if cookie exists
        if (!empty($_COOKIE['test'])) {
              echo "Cookies are enabled on your browser";
        } else {
                echo "Cookies are <b>NOT</b> enabled on your browser";
        }
 }
 ?>
```

Run the code above, and see what the output is. Check if cookies are enabled in your browser. If they're not enabled, then you can enable them by going to your browser's options. Unfortunately, this is different from each browser, so I can't give you exact instructions. But Google can.

One feature of cookies that is often missed in articles is the ability to story arrays. Cookies can be used to store multi-dimensional arrays, which can be extremely useful to store data.

Consider the following code;

```
<?php
 setcookie ("name[first]", "Dennis", time() + (60*60*24));
 setcookie ("name[last]", "Pallett", time() + (60*60*24));
 ?>
```

You can then display these two cookies using the following code:

```
<?php
 echo "First Name: " . $_COOKIE['name']['first'];
 echo "Last Name: " . $_COOKIE['name']['last'];
 ?>
```

The cookie 'name' is an array, and has multiple values. You can even go deeper and have multi-dimensional arrays, e.g. $_COOKIE['name']['test']['something']['value']. You could store whole arrays of data in cookies. But beware that you don't store too much data, there are certain size limits to cookies.

Cookies are really versatile, and can be used for a lot of different purposes. Many websites use cookies, and cookies can really make your website more personalized. Using cookies in PHP isn't hard at all, and you should be able to use them without any difficulty.

Before actively using cookies in your website, you must check whether the visitor has enabled them in their browser. If they don't have cookies enabled, you must either redirect to a non-cookies version of your website, or you can make sure your website also works without cookies.

You can download a sample script at http://www.phpit.net/demo/php%20and%20cookies/logger.zip, where cookies are used in a (somewhat) practical way. In this example, there is a logging module, called log.php and a display module, called history.php. Basically, you include the log.php in other PHP pages, and then you can view history.php to lookup all the pages you have viewed and how often. The example uses arrays, and stores them in cookies.

The examples in this article can be downloaded at http://www.phpit.net/demo/php%20and%20cookies/examples.zip.

If you have a really unique practical way of using cookies, please let me know at dennis [AT] nocertainty [DOT] com. I'd really like to hear about interesting ways of using cookies.

About this PDF