How to set and remove cookies in PHP

This article explains how to set and remove cookies in PHP '

Introduction

Websites using PHP handle user logins usually with the use of cookies. Unlike sessions in PHP, the cookies can be set to last for an extensively long time, even years.

In a cookie it is possible to store user related information and session specific parameters. The idea is that the data is stored persistently until expiration date.

For a website this would simply mean e.g. a possibility for the user to return to the webpage with the login still being valid "keep me logged in". When the user would require an logout, we would then need to remove the cookie. In this article we will show how to both operations.

Setting a cookie

We will use the setcookie() function to do this. The parameters used in this example are:

"name " This will be the name of the cookie. "value" The value of the cookie. This value is stored on the clients computer; do not store sensitive information. Assuming the name is 'cookiename', this value is retrieved through $_COOKIE['cookiename'] expire The time the cookie expires. This is a Unix timestamp so is in number of seconds since the epoch. In other words, you'll most likely set this with the time() function plus the number of seconds before you want it to expire. Or you might use mktime(). time()+60*60*24*30 will set the cookie to expire in 30 days. If set to 0, or omitted, the cookie will expire at the end of the session (when the browser closes).

Note: You may notice the expire parameter takes on a Unix timestamp, as opposed to the date format Wdy, DD-Mon-YYYY HH:MM:SS GMT, this is because PHP does this conversion internally.

Example code

The specific information of the website setting the cookie relays user computer and the website configuration or other user related information. Instead to PHP sessions the cookies they will offer a long restoration time, even years.

We will use two files in this example, the first one is to set the cookie and the second one is to view the cookie status.

In this example code we will use the setcookie(). The session cookies are value of seconds, so e.g. value 10 means the cookie will expire in 10 seconds if not reset. We will select the variable to be called 'username'.

We use the time function to get the timestamp first. When we then set the cookie we add the seconds we want to the timestamp, against what the checking will happen.

setcookie.php

```
<?php
setcookie('username', 'maveric', time()+10);
?>
```

viewcookie.php

To view anytime the cookie status we need this piece of code:

```
<?php
echo $_COOKIE['username'];
```

You can save this piece of code to a file and name it e.g. "viewcookie.php" or anything you want.

(C) Copyright Nokia 2014. All rights reserved.

So if you set the cookie time to e.g. 10 seconds and try to view it before and after the given time has run out, you should see the effect.

This works exactly as the webpages usually use this when the user logs in and e.g. has a checkbox for "keep me logged in" then that will set the cookie for until the user reverts this. Cookies are not encrypted by default, so caution should be exercised with what data to store in them.

Removing a cookie

In this part we can use the previous files, with a small editing.

To remove a cookie happens simply by resetting the cookie. Instead of setting the cookie the value for time will be put to minus value for the previous plus value.

Example code

In the section for setting a cookie we had added the value of 10 seconds to the timestamp. Now we will just do the opposite, we will reduce it from the timestamp to immediately expire, or "remove" the cookie.

```
<?php

setcookie('username', 'maveric', time()-10);

?>
```

This can be attached to e.g. a "logout" button on the webpage.

Summary

The cookies can be created and removed.

Caution should be exercised when storing sensitive data into a cookie; they are not encrypted by default.

Cookies are different to the PHP sessions.

Tested with

PHP 5.4

(C) Copyright Nokia 2014. All rights reserved.