Creating a Who s Online script with PHP

By Dennis Pallett

Introduction

In this tutorial I will show you how to create your own "Who's Online" script, which is often found on message boards and other community scripts. We will first create a really simple script, and after that add a few more features like the location of each visitor. But first, let me explain how our "Who's Online" script is going to work.

Our script will store a visitor's IP address and last active timestamp in a database, and update it every time the visitor requests one of our pages. The script will actually be a web bug, and run completely in the background. If you want to know more about web bugs, have a look at this SitePoint Blog entry by Harry Fuecks, but in a nut shell it's a small 1x1 image that actually executes PHP, and returns an image. So all our pages will point to a PHP file, like so:

```
<img src="whosonline.php" alt="" />
```

And the whosonline.php file will execute some code, and return a 1x1 gif image. Let's get started with creating a simple version of our "Who's Online" script.

A simple version

Our web bug has to do four things:
- Check if a visitor has already been 'registered'
- Either insert a new visitor or update an existing visitor
- Remove all old visitors who are no longer 'online'
- Serve a 1x1 GIF image, and make sure the web bug doesn't delay the calling page

The first three things are pretty easy, and the following database scheme:

```
CREATE TABLE `online` (
  `onlineid` int(5) NOT NULL AUTO_INCREMENT,
  `ipaddress` varchar(255) NOT NULL DEFAULT '',
  `lastactive` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY  (`onlineid`)
) TYPE=MyISAM AUTO_INCREMENT=;
```

With the following code does exactly what we want:

```
<?php
```


Copyright (c) 2007 PHPit.net

```
 include ('mysql.php');

# Timeout - how long should it take before visitors are no longer 'online'? (in minutes)
 define ('TIMEOUT', 20);

// Check if visitor is already in the table
 $ipaddress = ss($_SERVER['REMOTE_ADDR']);
 $lastactive = time();
 $intable = $db->query_first ("SELECT onlineid FROM online WHERE ipaddress = '$ipaddress'");

if ($intable == false) {
        // Insert new visitor
        $db->query ("INSERT INTO online (ipaddress, lastactive) VALUES ('$ipaddress', $lastactive)");
 } else {
        // Update exisiting visitor
        $db->query ("UPDATE online SET lastactive = $lastactive WHERE ipaddress = '$ipaddress'");
 }

// Remove any inactive visitors
 $inactive = time()-(60*60*TIMEOUT);
 $db->query ("DELETE FROM online WHERE lastactive < $inactive");
 ?>
```

(Note that you can get the 'mysql.php' file here)

I don't think it needs any explanation, as it's fairly easy to understand. The only thing I'd like to mention is that to know which visitors should be removed, the script first calculates what the time was 20 minutes ago (which is the timeout time I've set), and any visitor whose last active time is earlier than that is no longer considered active

The only thing that's missing from the above is serving the 1x1 GIF image, and making sure the image doesn't delay anything, and that's actually quite important. Although the above code probably doesn't have any noticeable delay, it's always a good thing to optimize your code as much as possible (within reason).

If you've read the SitePoint blog entry about web bugs, you will have seen that they've given the code to make sure the image doesn't delay anything. What they basically do is first send out the image, close the connection with the browser, and then actually start the real work, which in our case is the code above. It looks something like this:

```
// Make sure image is sent to the browser immediately
 ob_implicit_flush(TRUE);

 // keep running after browser closes connection
 @ignore_user_abort(true);

 sendGIF();

function sendGIF(){
    $img = base64_decode('R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAEALAAAAAABAAEAAAIBTAA7');
    header('Content-Type: image/gif');
    header('Content-Length: '.strlen($img));
    header('Connection: Close');
    print $img;
```


Copyright (c) 2007 PHPit.net

```
    // Browser should drop connection after this
    // Thinks it's got the whole image
}
```

This code first makes sure the image gets sent out, and then sets the ignore_user_abort option to true, which causes the script to run on, even after the connection is closed, which is exactly what we want. It then sends the GIF image, using some standard headers and data.

If you combine the above two code snippets together (the second snippet first, then the first snippet), and we've got our 'whosonline.php' file.

All we need now is some simple code to display everyone who's online, and we've got ourselves a simple "Who's Online" script. I'm using the following code to display who's online:

```
<?php
 include ('mysql.php');

// Include counter (I'm only using one page)
 echo '<img src="whosonline.php" alt="" />';

echo '<h2>Currently Online:</h2>';

$online = $db->sql_query ("SELECT onlineid, lastactive, ipaddress FROM online ORDER BY lastactive");

if (count($online) == ) {
        echo 'Nobody';
        die();
}

echo '<table>';
 echo '<th>IP Address</th><th>Last Active</th>';
 foreach ($online as $visitor) {
        echo '<tr>';

       echo '<td>' . $visitor['ipaddress'] . '</td>';
        echo '<td>' . date('g:i:s', $visitor['lastactive']) . '</td>';

       echo '</tr>';
}
 echo '</table>';

?>
```

(View Live Demo)

Let's have a look at adding a new feature: how to include the location of the visitor.

Where am I?

To include the location of the visitor, the parent page (and not the web bug) must get the location, because if you try to get the location with the web bug, you will always get the location of the web bug (in our case 'whosonline.php'), which is quite useless.

Copyright (c) 2007 PHPit.net

Before writing any code whatsoever, let's first add a new field to our online table:

```
ALTER TABLE `online` ADD `location` VARCHAR( 255 ) NOT NULL ;
```

Next thing we have to add is the code to get the location of the visitor. All we need to do is simply pass the location through a query string, like so:

```
<img src="locationcounter.php?location=locationdisplay.php" alt="" />
```

And the counter now has to store the location as well, which is simply a matter of changing the queries a bit, like so:

```
// Check if visitor is already in the table
 $ipaddress = ss($_SERVER['REMOTE_ADDR']);
 $lastactive = time();
 $intable = $db->query_first ("SELECT onlineid FROM online WHERE ipaddress = '$ipaddress'");
 if (!isset($_GET['location'])) { $location = ''; } else { $location = ss($_GET['location']); }

if ($intable == false) {
        // Insert new visitor
        $db->query ("INSERT INTO online (ipaddress, lastactive, location) VALUES ('$ipaddress', $lastactive,
'$location')");
 } else {
        // Update exisiting visitor
        $db->query ("UPDATE online SET lastactive = $lastactive, location = '$location' WHERE ipaddress =
'$ipaddress'");
 }
```

You might want to update the display page as well to include the location. To view a live demo, go to page 1, page 2, page 3, and view the display page (this time the display doesn't include the web bug).

This also demonstrates the reason why we used a web bug, instead of simply including a PHP file. Our "Who's Online" script also works with regular HTML pages, or any other page that can include images, but if we had used a PHP include, then only PHP pages could have used the who's online script. A small but very useful advantage indeed!

Automatically getting page titles

At the moment our location display only shows page URL's, but what if we could show the page title of each URL? That would certainly look a lot better, and more professional.

There are several ways of doing this. The easiest way is to include another query string, called title, and let each page pass its own title to the web bug. But you might not want to do this, or you're just too lazy to do it all manually. That's why our web bug will automatically fetch each page, and get its title. For this we're going to use the standard file_get_contents() function and some regular expression magic. Please be note though that opening URL's might be blocked for you, and it's possible that you have to open URL's in a different way (e.g. using the cURL library).

The first thing our web bug has to do is check whether the location passed is a full URL or a relative filename. The below code does what we want:

```
// Relative or not?
```


Copyright (c) 2007 PHPit.net

```
 if (substr($location, , strlen('http://')) != 'http://') {
                // Yes, relative, create absolute URL
        $location = 'http://' . $_SERVER['HTTP_HOST'] . dirname($_SERVER['REQUEST_URI']) . '/' . $location
;
 }
```

This code checks if the location starts with http://, and if it doesn't, it creates an absolute URL by adding the host, path and relative filename together.

Next thing we must do is getting the page's HTML, and extracting the title using regular expressions, like so:

```
$html = file_get_contents($location);

if (preg_match ('/<title>(.*?)<\/title>/is', $html, $match) == false) {
        // No title, just exit
        return false;
 } else {
        $title = $match['1'];
 }
```

And that's it. All we have to add now is some code to insert the title into the table, which has a new field as well:

```
ALTER TABLE `online` ADD `title` VARCHAR( 255 ) NOT NULL ;
```

The database code now looks like this:

```
if ($intable == false) {
        // Insert new visitor
        $db->query ("INSERT INTO online (ipaddress, lastactive, location, title) VALUES ('$ipaddress',
$lastactive, '$location', '$title')");
 } else {
        // Update exisiting visitor
        $db->query ("UPDATE online SET lastactive = $lastactive, location = '$location', title = '$title' WHERE
ipaddress = '$ipaddress'");
 }
```

And that's all. Go to page 1, page 2, page 3 again, and click here to view a live demo of the above code.

Conclusion

In this tutorial I have shown you how to create your own "Who's Online" script, just like the one that's often found in message boards. The script we've created in this tutorial can easily be extended to support more features, e.g. the ability for visitors to register their name or a page history for each visitor.

One thing I'd like to mention though is that the getting page titles code isn't very efficient at the moment, and if you're going to use it on your website, you should really implement some sort of caching mechanism. At the moment every time a visitor views one of your pages, the page will be hit a second time by the web bug, effectively doubling your server load.

Another thing to take note of is that I used REQUEST_URI quite often in this tutorial, and IIS doesn't support this. If you're using Apache on either Windows or Linux there aren't any problems, but all the code in this

Copyright (c) 2007 PHPit.net

tutorial won't work properly because of IIS' lack of support for REQUEST_URI. To fix this you have to write a small function that will get the current URL.

If you need any more help, or have any comments on this tutorial, head over to the PHPit Forums or leave them below.

Click here to download the demo's

About this PDF

You may distribute this PDF in any way you like, as long as you don't modify it in any way. You can ONLY distribute the unchanged original PDF.

For more information, contact us at support@pallettgroup.com.

Copyright (c) 2007 PHPit.net