PHP Caching

Introduction

In this article I will try to give a view of what is the custom caching with php, why and how we can use it.

In the modern days, most of the sites are database driven. That means that your site is actually an application which retrieves data from a DBMS (database managment system, eg MySQL) , parses the data and shows the result to the user. Most of these data are usually don't change frequently or don't change at all, and the reason that we use the database is that we can easilly update the site and the content.

A problem that this process creates is the server overhead. Every time we execute a query in the database, the instance of our script will call the DBMS, and then the DBMS will send the results of the query. This is time consuming, and especcially for sites with heavy traffic is a real big problem.

How we can solve this problem?

There are two ways to solve this if you want to make your site faster. First is optimizing the queries, but we will not talk about this at the present article. The second and most valuable is using some kind of custom caching technique.

Custom caching with php

First let me explain the idea behind custom caching. When we have dynamic pages that their data is not updated frequently, we can use a 'system' that will be able to create the page, and then store it for later use. That means that after the page's creation, our application will not run the queries again in order to display the page, but it will show the cached one. Of course this system must be able to keep the cached pages for a time period that we will set.

Let's code it

Here is a simple class that will do the job. Let's see the code first :

Code:
```
class cache
{
var $cache_dir = './tmp/cache/';//This is the directory where the cache files will be stored;
var $cache_time = 1000;//How much time will keep the cache files in seconds.

var $caching = false;
var $file = '';

function cache()
{
//Constructor of the class
$this->file = $this->cache_dir . urlencode( $_SERVER['REQUEST_URI'] );
if ( file_exists ( $this->file ) && ( fileatime ( $this->file ) + $this->cache_time ) > time() )
{
//Grab the cache:
$handle = fopen( $this->file , "r");
do {
$data = fread($handle, 8192);
if (strlen($data) == ) {
break;
}
echo $data;
} while (true);
fclose($handle);
exit();
}
else
{
//create cache :
$this->caching = true;
ob_start();
}
}

function close()
{
//You should have this at the end of each page
if ( $this->caching )
{
//You were caching the contents so display them, and write the cache file
$data = ob_get_clean();
echo $data;
$fp = fopen( $this->file , 'w' );
fwrite ( $fp , $data );
fclose ( $fp );
}
}
}


//Example :
$ch = new cache();
echo date("D M j G:i:s T Y");
$ch->close();
?>
```

Now let me explain :

function cache()

This is the constructor function of the class. The job of this function is to check if there is a cached file for the page that we want, or it should create it. Here is how this is done :

$this->file = $this->cache_dir . urlencode($_SERVER['REQUEST_URI']);

This line creates the file name of our cached page. So the cached file will be something like /path/to/cache/dir/request_uri

if (file_exists ($this->file) && (fileatime ($this->file) + $this->cache_time) > time())

Here we check if there is a cached version of this page, and if the file must be recreated because it has expired. If the file is cached, it will show the cached page and the exit. I will explain later why exit. If the cached file must be created this code will be executed :

$this->caching = true;
ob_start();

The first statement indicates to the close() function that it is creating the cache file, and the ob_start() will start buffering the output. The buffer's data will be used later by the close() function to save the cache file.

function close()

This function must be called from the end of your script, and it will do the rest of the job. Actually it is needed only when we are in the process of caching that's why it starts with the statement if ($this->caching)
Let me explain what is happening here :

$data = ob_get_clean();

Here we get all the data from the output buffer while we unset it, and put the data in the $data variable. The four statements that folow up are showing the data and then write the cache file.

Troubleshooting

This is a very simple class, and the purpose is to learn how you can implement a caching solution for your site. The obligation using this class is that you must use it only in this form :

Code:
```
$a = new cache();
....
....
....
$a->close();
?>
```

If you have code after the $a->close() statement, the class will not work right. This is because of the exit() statement in the cache() function.

Of course you can take this code and make it work for your own needs.

A quick solution is to remove the exit() statement in the cache() function and then use the class this way :

Code:
```
$a = new cache();
if ( $a->caching )
{
....
....
....
}
$a->close();
?>
```

Hope this helped.

Author: Nikolas
Copied from: http://www.webdigity.com/index.php/topic,862.0.PhP+Caching.html