PHP performance tips

* Upgrade your version of PHP
* Use caching

Making use of a caching module, such as Memcache, or a templating system which supports caching, such as Smarty, can help to improve the performance of your website by caching database results and rendered pages.

* Use output buffering

PHP uses a memory buffer to store all of the data that your script tries to print. This buffer can make your pages seem slow, because your users have to wait for the buffer to fill up before it sends them any data. Fortunately, you can make some changes that will force PHP to flush the output buffers sooner, and more often, making your site feel faster to your users.

* Output Buffering Control
* Don't copy variables for no reason

Sometimes PHP novices attempt to make their code "cleaner" by copying predefined variables to variables with shorter names before working with them. What this actually results in is doubled memory consumption (when the variable is altered), and therefore, slow scripts. In the following example, if a user had inserted 512KB worth of characters into a textarea field. This implementation would result in nearly 1MB of memory being used.

$description = strip\_tags($\_POST['description']);

echo $description;

There's no reason to copy the variable above. You can simply do this operation inline and avoid the extra memory consumption:

echo strip\_tags($\_POST['description']);

* Avoid doing SQL queries within a loop

A common mistake is placing a SQL query inside of a loop. This results in multiple round trips to the database, and significantly slower scripts. In the example below, you can change the loop to build a single SQL query and insert all of your users at once.

foreach ($userList as $user) {

$query = 'INSERT INTO users (first\_name,last\_name) VALUES("' . $user['first\_name'] . '", "' . $user['last\_name'] . '")';

mysql\_query($query);

}

Produces:

INSERT INTO users (first\_name,last\_name) VALUES("John", "Doe")

Instead of using a loop, you can combine the data into a single database query.

$userData = array();

foreach ($userList as $user) {

$userData[] = '("' . $user['first\_name'] . '", "' . $user['last\_name'] . '")';

}

$query = 'INSERT INTO users (first\_name,last\_name) VALUES' . implode(',', $userData);

mysql\_query($query);

Produces:

INSERT INTO users (first\_name,last\_name) VALUES("John", "Doe"),("Jane", "Doe")...

* Avoid writing naive setters and getters

When writing classes in PHP, you can save time and speed up your scripts by working with object properties directly, rather than writing naive setters and getters. In the following example, the dog class uses the setName() and getName() methods for accessing the name property.

class dog {

public $name = '';

public function setName($name) {

$this->name = $name;

}

public function getName() {

return $this->name;

}

}

Notice that setName() and getName() do nothing more than store and return the name property, respectively.

$rover = new dog();

$rover->setName('rover');

echo $rover->getName();

Setting and calling the name property directly can run up to 100% faster, as well as cutting down on development time.

$rover = new dog();

$rover->name = 'rover';

echo $rover->name;