

Output Buffering for Web Developers, a Beginner's Guide

Sunday, January 25th, 2009

If you're not using [PHP's output buffering](#), you should be. And if you are, you may not be using it to its potential.

In this article written specifically for web developers, I'll make a case for output buffering and show you how to get started within seconds. This article is the beginning of a series in which I'll share with you the awesome potential of output buffering.

Simple explanation of output buffering for Web developers

Without output buffering, your HTML is sent to the browser in pieces as PHP processes through your script. **With output buffering**, your HTML is stored in a variable and sent to the browser as one piece at the end of your script. *Can you already begin to see the performance advantages and post processing opportunities?*

Advantages of output buffering for Web developers

- Turning on output buffering alone decreases the amount of time it takes to download and render our HTML because it's not being sent to the browser in pieces as PHP processes the HTML.
- All the fancy stuff we can do with PHP strings, we can now do with our whole HTML page as one variable.
- If you've ever encountered the message "Warning: Cannot modify header information - headers already sent by (output)" while setting cookies, you'll be happy to know that output buffering is your answer.

Here's a "hello world" of PHP output buffering

[view sourceprint?](#)

```
01.<?php
02.// start output buffering at the top of our script with this simple
command
03.ob_start();
04.??>
05.
06.<html>
07.<body>
08.<p>Hello world!</p>
09.</body>
10.</html>
11.
12.<?php
13.// end output buffering and send our HTML to the browser as a whole
14.ob_end_flush();
15.??>
```

It's that simple! Just by doing this our webpages appear less choppy as they render. Now let's take it one step further.

The next step: compress the output

In the code below, `ob_start()` is changed to `ob_start('ob_gzhandler')`. That one simple change compresses our HTML, resulting in a smaller HTML download size for most browsers.

[view sourceprint?](#)

```
01.<?php
02.// start output buffering at the top of our script with this simple
   command
03.// we've added "ob_gzhandler" as a parameter of ob_start
04.ob_start('ob_gzhandler');
05.?>
06.
07.<html>
08.<body>
09.<p>Hello world!</p>
10.</body>
11.</html>
12.
13.<?php
14.// end output buffering and send our HTML to the browser as a whole
15.ob_end_flush();
16.?>
```

One more step further: custom post processing

In the code below, `ob_start()` is changed to `ob_start('ob_postprocess')`. `ob_postprocess()` is a function we define below used to make changes to the HTML before it is sent to the browser. Instead of `Hello world!`, the user will see `Aloha world!`

[view sourceprint?](#)

```
01.<?php
02.// start output buffering at the top of our script with this simple
   command
03.// we've added "ob_postprocess" (our custom post processing
   function) as a parameter of ob_start
04.ob_start('ob_postprocess');
05.?>
06.
07.<html>
08.<body>
09.<p>Hello world!</p>
10.</body>
11.</html>
12.
13.<?php
14.// end output buffering and send our HTML to the browser as a whole
15.ob_end_flush();
16.
17.// ob_postprocess is our custom post processing function
18.function ob_postprocess($buffer)
19.{
20.// do a fun quick change to our HTML before it is sent to the
   browser
21.$buffer = str_replace('Hello', 'Aloha', $buffer);
```

```
22.// "return $buffer;" will send what is in $buffer to the browser,  
which includes our changes  
23.return $buffer;  
24.}  
25.??>
```

Bonus tip: you can set cookies at any time with output buffering on

Since HTML is not sent directly to the browser, we can set cookies anywhere in our scripts without worrying about anything. Just turn it on like we did in step 1 and voilà! No more cookie problems.

Finished for now, subscribe for more

There you have it! Three simple steps into output buffering to get you started. First we just turned it on with `ob_start()`, then we added automatic compression with `ob_start('ob_gzhandler')`, and finally we made `ob_postprocess()` — a custom function to modify our HTML before output.

<http://web.archive.org/web/20101216035343/http://dev-tips.com/featured/output-buffering-for-web-developers-a-beginners-guide>