Tips and Tricks

PHPCon2002

October 24, 2002. Milbrae, CA

Rasmus Lerdorf <rasmus@php.net>

Optimization

Don't use a regex if you don't have to

PHP has a rich set of string manipulation functions - use them!

```
BAD: <?  $new = ereg_replace("-","_",$str); ?>

GOOD:<?  $new = str_replace("-","_",$str);  ?>

BAD: <?  preg_match('/(\..*?)$/',$str,$reg);?>

GOOD:<?  substr($str,strrpos($str,'.'));    ?>
```

Use References if you are passing large data structs around to save memory

There is a tradeoff here. Manipulating references is actually a bit slower than making copies of your data, but with references you will be using less memory. So you need to determine if you are cpu or memory bound to decide whether to go through and look for places to pass references to data instead of copies.

Use Persistent Database connections

Some database are slower than others at establising new connections. The slower it is, the more of an impact using persistent connections will have. But, keep in mind that persistent connections will sit and tie up resources even when not in use. Watch your resource limits as well. For example, by default Apache's

Using MySQL? Check out mysql_unbuffered_query()

Use it exactly like you would mysql_query(). The difference is that instead of waiting for the entire query to finish and storing the result in the client API, an unbuffered query makes results available to you as soon as possible and they are not allocated in the client API. You potentially get access to your data quicker, use a lot less memory, but you can't use mysql_num_rows() on the result resource and it is likely to be slightly slower for small selects.

Hey Einstein!

Don't over-architect things. If your solution seems complex to you, there is probably a simpler and more obvious approach. Take a break from the computer and go out into the big (amazingly realistic) room and think about something else for a bit.

Adding an extension

Problem

You need PHP's built-in ftp functions for the ultra-cool script you are writing, but your service provider does not have PHP compiled with the --enable-ftp option.

Solution

If you have a shell account on a system with the same operating system as your web server, grab the PHP source tarball and build using:

```
--with-apxs --enable-ftp=shared
```

You can check which flags your provider used by putting a phpinfo() call in a script on your server.

```
<?phpinfo()?>
```

Once compiled, you will find a "modules/ftp.so" file which you can copy to your web server and enable either by putting:

```
extension=ftp.so
```

in your php.ini file or by adding this to the top of your script:

```
<?php dl("ftp.so") ?>
```

Cookie Expiry

Problem

Short expiry cookies depend on users having their system clocks set correctly.

Solution

Don't depend on the users having their clocks set right. Embed the timeout based on your server's clock in the cookie.

```
<?php
  $value = time()+3600 . ':' . $variable;
  SetCookie('Cookie_Name',$value);
?>
```

Then when you receive the cookie, decode it and determine if it is still valid.

```
<?php
list($ts,$variable) = explode(':',$Cookie_Name,2);
if($ts < time()) {
   ...
} else {
    SetCookie('Cookie_Name','');
}
?>
```

Client/Server Request/Response

HTTP is a simple client/server protocol with stateless request/response sequences.

The Client HTTP Request

7 possible HTTP 1.1 request types: GET, PUT, POST, DELETE, HEAD, OPTIONS and TRACE. Any number of HTTP headers can accompany a request.

```
GET /filename.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Charset: iso-8859-1,*,utf-8
Accept-Encoding: gzip
Accept-Language: en
Connection: Keep-Alive
Host: localhost
User-Agent: Mozilla/4.77 [en] (X11; U; Linux 2.4.5-pre4 i686; Nav)
```

The Server HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 21 May 2001 17:01:51 GMT
Server: Apache/1.3.20-dev (Unix) PHP/4.0.7-dev
Last-Modified: Fri, 26 Jan 2001 06:08:38 GMT
ETag: "503d3-50-3a711466"
Accept-Ranges: bytes
Content-Length: 80
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Keep-Alive

When a keep-alive request is granted the established socket is kept open after each keep-alive response. Note that a keep-alive response is only possible when the response includes a content-length header.

```
request 1
request 2
request 3
request 4
```

```
20 bytes
120 bytes
60 bytes
?? bytes
```

You cannot rely on the keep-alive feature for any sort of application-level session state maintenance.

Using Output Buffering to get content-length

```
<?php
    ob_start();
    echo "Your Data";
    $l = ob_get_length();
    Header("Content-length: $l");
    ob_end_flush();
?>
```

You will have to weigh the trade-off between the extra cpu and memory that output buffering takes against the increased effciency of being able to use keep-alive connections for your dynamic pages.

Connection Handling

PHP maintains a connection status bitfield with 3 bits:

- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT

By default a PHP script is terminated when the connection to the client is broken and the ABORTED bit is turned on. This can be changed using the ignore_user_abort() function. The TIMEOUT bit is set when the script timelimit is exceed. This timelimit can be set using set_time_limit().

```
<?php
set_time_limit(0);
ignore_user_abort(true);
/* code which will always run to completion */
?>
```

You can call connection_status() to check on the status of a connection.

```
<?php
ignore_user_abort(true);
echo "some output";
if(connection_status()==0) {
    // Code that only runs when the connection is still alive
} else {
    // Code that only runs on an abort
}
?>
```

You can also register a function which will be called at the end of the script no matter how the script was terminated.

```
<?php
function foo() {
    if(connection_status() & 1)
        error_log("Connection Aborted",0);
    if(connection_status() & 2)
        error_log("Connection Timed Out",0);
    if(!connection_status())
        error_log("Normal Exit",0);
}
register_shutdown_function('foo');
?>
```

Variable variables

A variable variable looks like this: $$var

So, if $var = 'foo' and $foo = 'bar' then $$var would contain the value 'bar' because $$var can be thought of as $'foo' which is simply $foo which has the value 'bar'.

Variable variables sound like a cryptic a useless concept, but they can be useful sometimes. For example, if we have a configuration file consisting of configuration directives and values in this format:

```
foo=bar
abc=123
```

Then it is very easy to read this file and create corresponding variables:

```
<?php
$fp = fopen('config.txt','r');
while(true) {
    $line = fgets($fp,80);
    if(!feof($fp)) {
        if($line[0]=='#' || strlen($line)<2) continue;
        list($name,$val)=explode('=',$line,2);
        $$name=trim($val);
    } else break;
}
fclose($fp);
?>
```

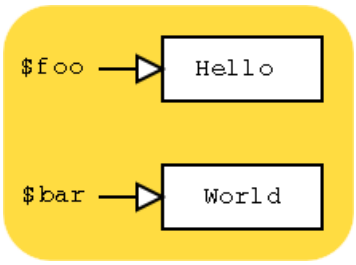
Along the same lines as variable variables, you can create compound variables and variable functions.

```
<?php
  $str = 'var';
  $var_toaster = "Hello World";
  echo ${$str.'_toaster'};

  $str();  // Calls a function named var()
  ${$str.'_abc'}();  // Calls a function named var_abc()
?>
```

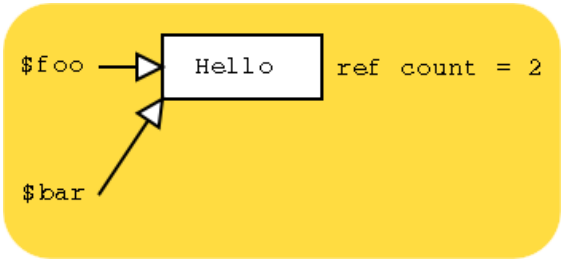
References

References are not pointers!

```
<?php
    $foo = 'Hello';
    $bar = 'World';
?>
```




```
<?php
    $bar = & $foo;
?>
```



Returning References

Passing arguments to a function by reference

```
<?php
function inc(& $b) {
    $b++;
}
$a = 1;
inc($a);
echo $a;
?>
```

Output:

```
2
```

A function may return a reference to data as opposed to a copy

```
<?php
function & get_data() {
    $data = "Hello World";
    return $data;
}
$foo = & get_data();
?>
```

debug_backtrace

debug_backtrace() is a new function in PHP 4.3

Custom error handler

```
<?
    function myErrorHandler ($errno, $errstr, $errfile, $errline) {
        echo "$errno: $errstr in $errfile at line $errline\n";
        echo "Backtrace\n";
        $trace = debug_backtrace();
        foreach($trace as $ent) {
            if(isset($ent['file'])) echo $ent['file'].':';
            if(isset($ent['function'])) {
                echo $ent['function'].'(';
                if(isset($ent['args'])) {
                    $args='';
                    foreach($ent['args'] as $arg) { $args.=$arg.','; }
                    echo rtrim($args,',');
                }
                echo ') ';
            }
            if(isset($ent['line'])) echo 'at line '.$ent['line'].' ';
            if(isset($ent['file'])) echo 'in '.$ent['file'];
            echo "\n";
        }
    }

    set_error_handler('myErrorHandler');

    include 'file2.php';

    test2(1,0);
?>
```

Custom error handler

```
<?
    function test1($b,$a) {
        $a/$b;
    }

    function test2($a,$b) {
        test1($b,$a);
    }
?>
```

Safe Mode

Safe Mode is an attempt to solve the shared-server security problem. It is architecturally incorrect to try to solve this problem at the PHP level, but since the alternatives at the web server and OS levels aren't very realistic, many people, especially ISP's, use safe mode for now.

The configuration directives that control safe mode are:

```
safe_mode = Off
open_basedir =
safe_mode_exec_dir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH
disable_functions =
```

When safe_mode is on, PHP checks to see if the owner of the current script matches the owner of the file to be operated on by a file function.

For example:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Running this script.php

```
<?php
readfile('/etc/passwd');
?>
```

results in this error when safe mode is enabled:

```
<b>Warning</b>:  SAFE MODE Restriction in effect.  The script whose uid is 500 is
not allowed to access /etc/passwd owned by uid 0 in <b>/docroot/script.php</b> on
line <b>2</b>
```

If instead of safe_mode, you set an open_basedir directory then all file operations will be limited to files under the specified directory. For example (Apache httpd.conf example):

```
<Directory /docroot>
php_admin_value open_basedir /docroot
</Directory>
```

If you run the same script.php with this open_basedir setting then this is the result:

```
<b>Warning</b>: open_basedir restriction in effect. File is in wrong directory in
<b>/docroot/script.php</b> on line <b>2</b>
```

You can also disable individual functions. If we add this to our php.ini file:

```
disable_functions readfile,system
```

Then we get this output:

```
<b>Warning</b>:  readfile() has been disabled for security reasons in
<b>/docroot/script.php</b> on line <b>2</b>
```

Watch for uninitialized variables

```
<?php
    if($user=='rasmus') {
        $ok = true;
    }

    if($ok) {
        echo "$user logged in";
    }
?>
```

Catch these by setting the error_reporting level to E_ALL. The above script would generate this warning (assuming $user is set):

```
<b>Warning</b>:  Undefined variable:  ok in <b>script.php</b> on line <b>6</b>
```

You can of course also turn off register_globals, but that addresses the symptom rather than the problem.

Never trust user data!

```
<?php
    readfile($filename);
?>
```

Turning off register_globals doesn't make this any more secure. The script would instead look like this:

```
<?php
    readfile($HTTP_POST_VARS['filename']);
?>
```

The only way to secure something like this is to be really paranoid about cleaning user input. In this case if you really want the user to be able to specify a filename that gets used in any of PHP's file functions, do something like this:

```
<?php
    $doc_root = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];
    $filename = realpath($filename);
    readfile($doc_root.$filename);
?>
```

You may also want to strip out any path and only take the filename component. An easy way to do that is to use the basename() function. Or perhaps check the extension of the file. You can get the extension using this code:

```
<?php
    $ext = substr($str,strrpos($str,'.'));
?>
```

Again, never trust user data!

```
<?php
    system("ls $dir");
?>
```

In this example you want to make sure that the user can't pass in $dir set to something like: ".;cat /etc/passwd" The remedy is to use escapeshellarg() which places the argument inside single quotes and escapes any single quote characters in the string.

```
<?php
    $dir=escapeshellarg($dir);
    system("ls $dir");
?>
```

Beyond making sure users can't pass in arguments that executes other system calls, make sure that the argument itself is ok and only accesses data you want the users to have access to.

Security

Many users place code in multiple files and include these files:

```
<?php
    require 'functions.inc';
?>
```

Or perhaps

```
<?php
    require 'functions.php';
?>
```

Both of these can be problematic if the included file is accessible somewhere under the DOCUMENT_ROOT directory. The best solution is to place these files outside of the DOCUMENT_ROOT directory where they are not accessible directly. You can add this external directory to your include_path configuration setting.

Another option is to reject any direct requests for these files in your Apache configuration. You can use a rule like this in your "httpd.conf" file:

```
<Files ~ "\.inc$">
    Order allow,deny
    Deny from all
</Files>
```

Take this standard file upload form:

```
<FORM ENCTYPE="multipart/form-data" ACTION="upload.php" METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="100000">
Send this file: <INPUT NAME="myfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

The correct way to put the uploaded file in the right place:

```
<?php
    /* Not under DOCUMENT_ROOT */
    $destination = "/some/path/$myfile_name";

    move_uploaded_file($myfile, $destination);
?>
```

If you are uploading files to be placed somewhere under the DOCUMENT_ROOT then you need to be very paranoid in checking what you are putting there. For example, you wouldn't want to let people upload arbitrary PHP scripts that they can then browse to in order to execute them. Here we get paranoid about checking that only image files can be uploaded. We even look at the contents of the file and ensure that the file extension matches the content.

```
<?php
    $type = $HTTP_POST_FILES['myfile']['type'];
    $file = $HTTP_POST_FILES['myfile']['tmp_name'];
    $name = $HTTP_POST_FILES['myfile']['name'];
    $types = array(0,'.gif','.jpg','.png','.swf');
    list(,,$type) = getimagesize($file);
    if($type) {
        $name = substr($name,0,strrpos($str,'.'));
        $name .= $types[$type];
    }
    move_uploaded_file($myfile, "$DOCUMENT_ROOT/images/$name");
?>
```

Starting a Session

To start a session use session_start() and to register a variable in this session use the $_SESSION array.

```
<?php
  session_start();
  $_SESSION['my_var'] = 'Hello World';
?>
```

If register_globals is enabled then your session variables will be available as normal variables on subsequent pages. Otherwise they will only be in the $_SESSION array.

```
<?php
  session_start();
  echo $_SESSION['my_var'];
?>
```

Session Configuration

Default session settings are set in your php.ini file:

```
session.save_handler = files    ; Flat file backend
session.save_path=/tmp          ; where to store flat files
session.name = PHPSESSID        ; Name of session (cookie name)
session.auto_start = 0          ; init session on req startup
session.use_cookies = 1         ; whether cookies should be used
session.use_only_cookies = 0    ; force only cookies to be used
session.cookie_lifetime = 0     ; 0 = session cookie
session.cookie_path = /         ; path for which cookie is valid
session.cookie_domain =         ; the cookie domain
session.serialize_handler = php ; serialization handler (wddx|php)
session.gc_probability = 1      ; garbage collection prob.
session.gc_dividend    = 100    ; If 100, then above is in %
session.gc_maxlifetime = 1440   ; garbage collection max lifetime
session.referer_check =         ; filter out external URL\'s
session.entropy_length = 0      ; # of bytes from entropy source
session.entropy_file =          ; addtional entropy source
session.use_trans_sid = 1       ; use automatic url rewriting
url_rewriter.tags = "a=href,area=href,frame=src,input=src"
session.cache_limiter = nocache ; Set cache-control headers
session.cache_expire = 180      ; expiry for private/public caching
```

Cache-control is important when it comes to sessions. You have to be careful that end-user client caches aren't caching invalid pages and also that intermediary proxy-cache mechanisms don't sneak in and cache pages on you. When cache-limiter is set to the default, no-cache, PHP generates a set of response headers that look like this:

```
HTTP/1.1 200 OK
Date: Sat, 10 Feb 2001 10:21:59 GMT
Server: Apache/1.3.13-dev (Unix) PHP/4.0.5-dev
X-Powered-By: PHP/4.0.5-dev
Set-Cookie: PHPSESSID=9ce80c83b00a4aefb384ac4cd85c3daf; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Connection: close
Content-Type: text/html
```

For cache_limiter = private the cache related headers look like this:

```
Set-Cookie: PHPSESSID=b02087ce4225987870033eba2b6d78c3; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
```

For cache_limiter = public they look like this:

```
Set-Cookie: PHPSESSID=37421e3d0283c667f75481745b25b9ad; path=/
Expires: Tue, 12 Feb 2001 13:57:16 GMT
Cache-Control: public, max-age=10800
```

Custom Backend

You can change the session backend datastore from a script using session_module_name().

```
<?php
  session_module_name("files");  // ASCII files

  session_module_name("mm");     // Shared memory

  session_module_name("user");   // Custom session backend
?>
```

You can also define your own custom session backend datastore using the session_set_save_handler() function.

```
<?php
  session_set_save_handler("myOpen", "myClose",
                           "myRead", "myWrite",
                           "myDestroy", "myGC");
?>
```

You would then write these 6 functions.

Custom Backend

Let's have a look at an actual custom session backend. This uses MySQL to store the session data. We could set these right in the script, but let's make use of Apache's httpd.conf file to set our custom save handler for a portion of our web site.

```
<Directory "/var/html/test">
    php_value session.save_handler user
    php_value session.save_path mydb
    php_value session.name sessions
</Directory>
```

The MySQL schema looks like this:

```
CREATE TABLE sessions (
    id char(32) NOT NULL,
    data text,
    ts timestamp,
    PRIMARY KEY (id)
)
```

We can now write our handler. It looks like this:

```
<?php
function open($db,$name) {
    global $table;
    mysql_connect('localhost');
    mysql_select_db($db);
    $table = $name;
    return true;
}

function close() {
    mysql_close();
    return true;
}

function read($id) {
    global $table;
    $result = mysql_query("select data from $table where id='$id'");
    if($result && mysql_num_rows($result)) {
        return mysql_result($result,0);
    } else {
        error_log("read: ".mysql_error()."\n",3,"/tmp/errors.log");
        return "";
    }
}

function write($id, $data) {
    global $table;
    $data = addslashes($data);
    mysql_query("replace into $table (id,data) values('$id','$data')")
        or error_log("write: ".mysql_error()."\n",3,"/tmp/errors.log");
    return true;
}

function destroy($id) {
    global $table;
    mysql_query("delete from $table where where id='$id'");
}

function gc($max_time) {
    global $table;
    mysql_query(
      "delete from $table where UNIX_TIMESTAMP(ts)<UNIX_TIMESTAMP()-$max_time")
```

```
      or error_log("gc: ".mysql_error()."\n",3,"/tmp/errors.log");
    return true;
}

session_set_save_handler('open','close','read','write','destroy','gc');
?>
```

Our PHP files under /var/html/test then simply need to look something like this:

```
<?php
  require 'handler.php';

  session_start();
  session_register('var');
  $var = "Hello World";
?>
```

$PATH_INFO

$PATH_INFO is your friend when it comes to creating clean URLS. Take for example this URL:

```
http://www.company.com/products/routers
```

If the Apache configuration contains this block:

```
<Location "/products">
  ForceType application/x-httpd-php
</Location>
```

Then all you have to do is create a PHP script in your DOCUMENT_ROOT named 'products' and you can use the $PATH_INFO variable which will contain the string, '/routers', to make a DB query.

ErrorDocument

Apache's ErrorDocument directive can come in handy. For example, this line in your Apache configuration file:

```
ErrorDocument 404 /error.php
```

Can be used to redirect all 404 errors to a PHP script. The following server variables are of interest:

- $REDIRECT_ERROR_NOTES - File does not exist: /docroot/bogus
- $REDIRECT_REQUEST_METHOD - GET
- $REDIRECT_STATUS - 404
- $REDIRECT_URL - /docroot/bogus

Don't forget to send a 404 status if you choose not to redirect to a real page.

```
<? Header('HTTP/1.0 404 Not Found'); ?>
```

Interesting uses

- Search for closest matching valid URL and redirect
- Use attempted url text as a DB keyword lookup
- Funky caching

An interesting way to handle caching is to have all 404's redirected to a PHP script.

```
ErrorDocument 404 /generate.php
```

Then in your generate.php script use the contents of $REDIRECT_URI to determine which URL the person was trying to get to. In your database you would then have fields linking content to the URL they affect and from that you should be able to generate the page. Then in your generate.php script do something like:

```
<?php
    $s = $REDIRECT_URI;
    $d = $DOCUMENT_ROOT;
    // determine requested uri
    $uri = substr($s, strpos($s,$d) + strlen($d) + 1);
    ob_start();  // Start buffering output
    // ... code to fetch and output content from DB ...
    $data = ob_get_contents();
    $fp = fopen("$DOCUMENT_ROOT/$uri",'w');
    fputs($fp,$data);
    fclose($fp);
    ob_end_flush(); // Flush and turn off buffering
?>
```

So, the way it works, when a request comes in for a page that doesn't exist, generate.php checks the database and determines if it should actually exist and if so it will create it and respond with this generated data. The next request for that same URL will get the generated page directly. So in order to refresh your cache you simply have to delete the files.

Creating a PNG with a TrueType font

```
<?
Header("Content-type: image/png");
$im = ImageCreate(630,80);
$blue = ImageColorAllocate($im,0x5B,0x69,0xA6);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
ImageTTFText($im, 45, 0, 10, 57, $black, "CANDY", $text);
ImageTTFText($im, 45, 0, 6, 54, $white, "CANDY", $text);
ImagePNG($im);
?>

<IMG src="txt.php?text=<?echo urlencode($text)?>">
```

Color Handling

For images with an 8-bit indexed palette it can be tricky to manage colors.

```
<?
$im  = ImageCreate(300,256);
for($r=0; $r<256; $r++) {
    $col = ImageColorAllocate($im,$r,0,0);
    ImageLine($im, 0,$r, 100, $r, $col);
}
for($g=0; $g<256; $g++) {
    $col = ImageColorAllocate($im,0,$g,0);
    ImageLine($im, 100,255-$g, 200, 255-$g, $col);
}
for($b=0; $b<256; $b++) {
    $col = ImageColorAllocate($im,0,0,$b);
    ImageLine($im, 200,$b, 300, $b, $col);
}
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Output:

For paletted images the following functions can be useful:

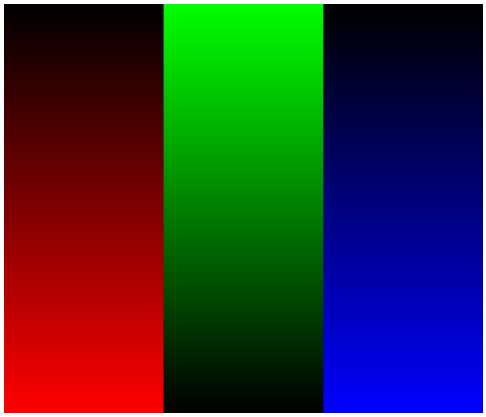
- ImageColorClosest
- ImageColorExact
- ImageColorDeallocate

Colour Handling

For Truecolor images we have no such issues.

```
<?
$im  = ImageCreateTruecolor(300,256);
for($r=0; $r<256; $r++) {
    $col = ImageColorAllocate($im,$r,0,0);
    ImageLine($im, 0,$r, 100, $r, $col);
}
for($g=0; $g<256; $g++) {
    $col = ImageColorAllocate($im,0,$g,0);
    ImageLine($im, 100,255-$g, 200, 255-$g, $col);
}
for($b=0; $b<256; $b++) {
    $col = ImageColorAllocate($im,0,0,$b);
    ImageLine($im, 200,$b, 300, $b, $col);
}
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

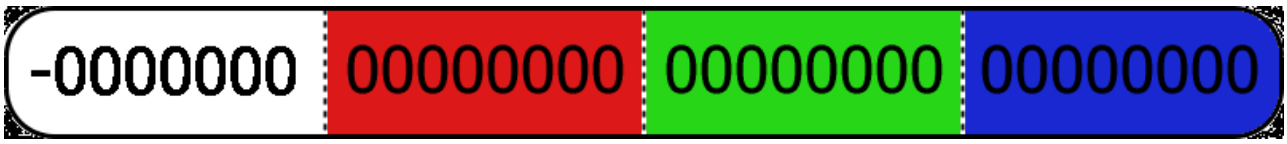
Output:



Truecolor Colors

Truecolor color handling

For Truecolor images the colors are actually simple 31-bit longs. Or, think of them as being composed of 4 bytes arranged like this:

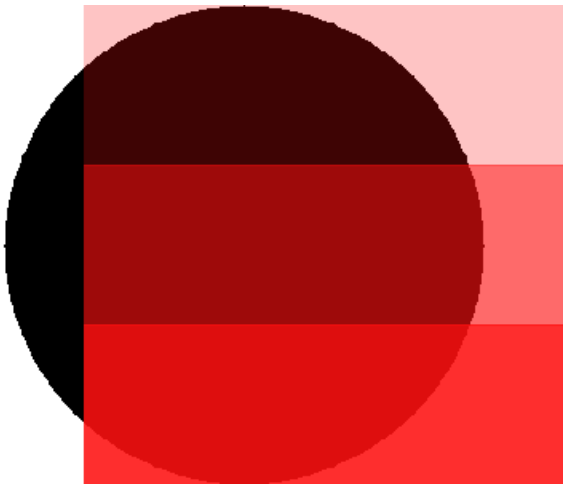


The highest or leftmost bit in the alpha channel is not used which means the alpha channel can only have values from 0 to 127. You can use the ImageColorAllocate() as with paletted images, but you can also specify the color directly.

For example:

```
<?
$im  = ImageCreateTruecolor(400,300);
ImageFilledRectangle($im,0,0,399,299,0x00ffffff);
ImageFilledEllipse($im,200,150,300,300,0x00000000);
ImageAlphaBlending($im,true);
ImageFilledRectangle($im,100,0,400,100,0x60ff1111);
ImageFilledRectangle($im,100,100,400,200,0x30ff1111);
ImageFilledRectangle($im,100,200,400,300,0x10ff1111);
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Output:



This example could also be written like this:

```
<?php
$im  = ImageCreateTruecolor(400,300);
$white = ImageColorAllocate($im,255,255,255);
ImageFilledRectangle($im,0,0,399,299,$white);
$black = ImageColorAllocate($im,0,0,0);
ImageFilledEllipse($im,200,150,300,300,$black);
ImageAlphaBlending($im,true);
$col = ImageColorResolveAlpha($im,0xff,0x11,0x11,0x60);
ImageFilledRectangle($im,100,0,400,100,$col);
$col = ImageColorResolveAlpha($im,0xff,0x11,0x11,0x30);
ImageFilledRectangle($im,100,100,400,200,$col);
$col = ImageColorResolveAlpha($im,0xff,0x11,0x11,0x10);
```

```
ImageFilledRectangle($im,100,200,400,300,$col);
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Truecolor Color Handling

Given the nature of the way truecolor colors are constructed, we can rewrite our color testing strip using PHP's bitshift operator:

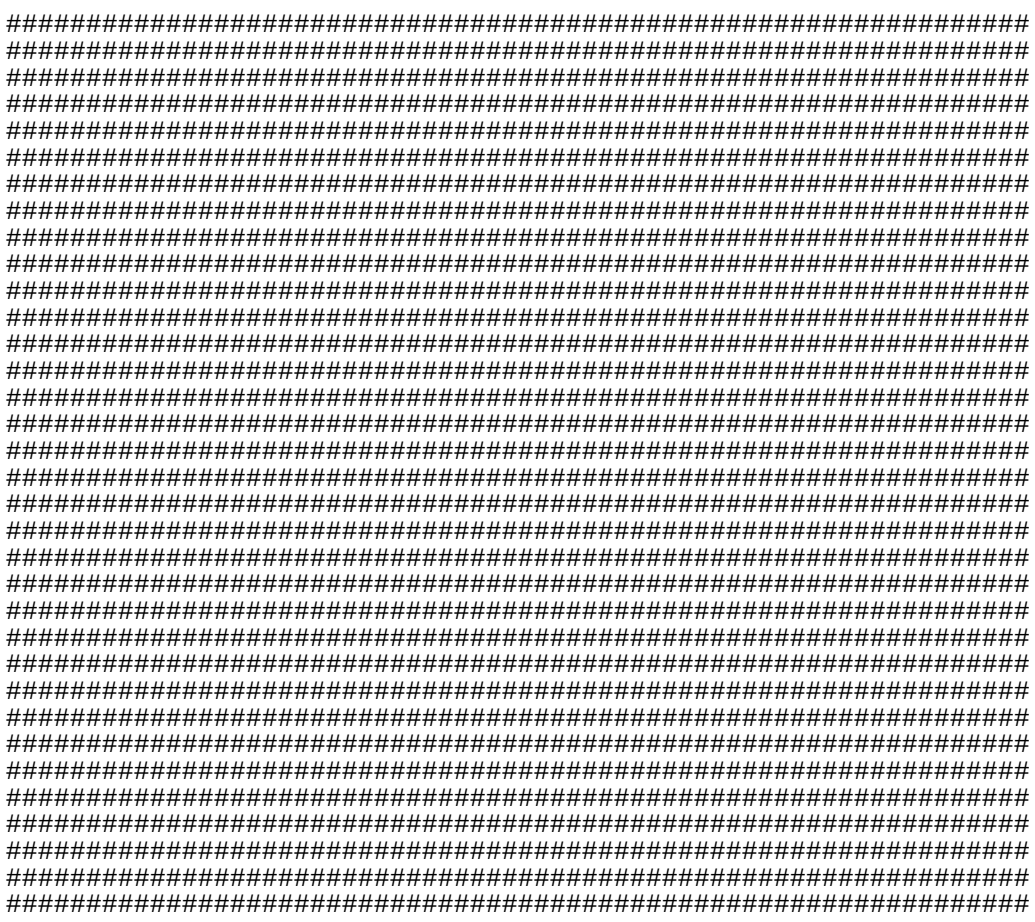
```
<?
$im = ImageCreateTrueColor(256,60);
for($x=0; $x<256; $x++) {
    ImageLine($im, $x, 0, $x, 19, $x);
    ImageLine($im, 255-$x, 20, 255-$x, 39, $x<<8);
    ImageLine($im, $x, 40, $x, 59, $x<<16);
}
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Output:

Using ImageColorAt()

```
<? $image = "presentations/slides/intro/php-tiny.jpg"; ?>
<img src="<?=$image?>"><br />
<?
    $im = imagecreatefromjpeg($image);
    $dx = imagesx($im);
    $dy = imagesy($im);
    for($y = 0; $y < $dy; $y++) {
        for($x=0; $x < $dx; $x++) {
            $col = imagecolorat($im, $x, $y);
            $rgb = imagecolorsforindex($im,$col);
            printf('<font color=#%02x%02x%02x>#</font>',
              $rgb['red'],$rgb['green'],$rgb['blue']);
        }
        echo "<br />\n";
    }
    imagedestroy($im);
?>
```

Output:



CreateFrom and Bounding Box Math

```
<?
Header("Content-type: image/png");
$font = 'phpi';
if(!$si) $si = 66;
$im = ImageCreateFromPNG('php-blank.png');
$tsize = imagettfbbox($si,0,$font,$text);
$dx = abs($tsize[2]-$tsize[0]);
$dy = abs($tsize[5]-$tsize[3]);
$x = ( imagesx($im) - $dx ) / 2;
$y = ( imagesy($im) - $dy ) / 2 + 3*$dy/4;
$blue = ImageColorAllocate($im,0x5B,0x69,0xA6);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
ImageAlphaBlending($im,true);
ImageTTFText($im, $si, 0, $x, $y, $white, $font, $text);
ImageTTFText($im, $si, 0, $x+2, $y, $white, $font, $text);
ImageTTFText($im, $si, 0, $x, $y+2, $white, $font, $text);
ImageTTFText($im, $si, 0, $x+2, $y+2, $white, $font, $text);
ImageTTFText($im, $si, 0, $x+1, $y+1, $black, $font, $text);
ImagePNG($im);
?>

<IMG src="txt2.php?text=<?echo urlencode($text)?>&si=<?echo $si?>">

Text:  Size:
```

Scaling and the Alpha Channel

```
<?
Header('Content-Type: image/png');
$height = 600;
$txt = 'Carl in his strawberry hat';
$size = ImageTTFBbox(25,0,'timesi',$txt);
$txt_w = abs($size[2]-$size[0]);
$txt_h = abs($size[6]-$size[1]);
$bg = ImageCreateFromJpeg('img_resize.jpg');
$img_width = imagesx($bg);
$img_height = imagesy($bg);
$width = ($height)/$img_height * $img_width;
$sizing =  "Original image size $img_width x $img_height\r\n";
$sizing .= "     New image size $width x $height";
$im = ImageCreateTrueColor($width,$height);
ImageAlphaBlending($im,false);
ImageCopyResampled($im,$bg,0,0,0,0,$width,$height,$img_width,$img_height);
$white = ImageColorAllocate($im,255,255,255);
ImageTTFText($im,15,0,20,20,$white,'couri',$sizing);
ImageDestroy($bg);
$col = ImageColorResolveAlpha($im,10,10,10,50);
ImageAlphaBlending($im,true);
$box = ($width-$txt_w)/2;
ImageFilledRectangle($im,$box-10,$height-$txt_h-30,$width-$box,$height-5,$col);
ImageAlphaBlending($im,false);
$yellow = ImageColorAllocate($im,255,255,10);
ImageTTFText($im,25,0,$box,$height-$txt_h-5,$yellow,'timesi',$txt);
ImagePNG($im);
?>
```

Output:

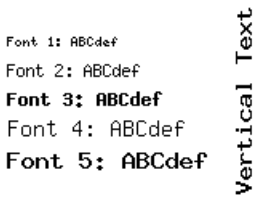

Original image size 1536 x 1024
New image size 900 x 600
Carl in his strawberry hat

Built-in Fonts

GD comes with 5 built-in fonts. They aren't all that useful.

```
<?
$im  = ImageCreate(175,125);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
ImageString($im,1,10,20,"Font 1: ABCdef",$black);
ImageString($im,2,10,35,"Font 2: ABCdef",$black);
ImageString($im,3,10,53,"Font 3: ABCdef",$black);
ImageString($im,4,10,70,"Font 4: ABCdef",$black);
ImageString($im,5,10,90,"Font 5: ABCdef",$black);
ImageStringUp($im,5,150,118,"Vertical Text",$black);
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Output:



TrueType Fonts

You can use any TrueType Font that includes a Unicode mapping table. Fonts such as Wingdings will not work.

```
<?
$im  = ImageCreate(600,7150);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
$dir = opendir('/usr/share/fonts/truetype');
$y=30;
while($file = readdir($dir)) {
    if(substr($file,strrpos($file,'.'))=='.ttf') {
        ImageString($im,5,5,$y-20,substr($file,0,-4),$black);
        ImageTTFText($im,30,0,100,$y,$black, substr($file,0,-4),"ABCdéf123");
        $y+=40;
    }
}
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Output:

abalc ABCdéf123

a_d_mono 

Adresack

aggstock 

ariblk **ABCdéf123**

arnari *ABCdéf123*

arnar ABCdéf123

bkant ABCdéf123

Bookosbi ***ABCdéf123***

bookosb **ABCdéf123**

Bookosi *ABCdéf123*

Bookos ABCdéf123

calist ABCdéf123

comicbd **ABCdéf123**

comic ABCdéf123

coprgtb **ABCDÉF123**

coprgtl ABCDÉF123

courbd **ABCdéf123**

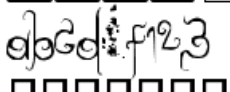
courbi ***ABCdéf123***

couri *ABCdéf123*

cour ABCdéf123

dc_sans 

dc_serif 

dilate 

dirtydoz 

Dotmatrx ABCdéf 123

dronecat 

earth 

Eklekti0 ABCDÉF123

electroh 

Eroded2020 

Reading EXIF Headers from a JPEG

```
<?php
$data = exif_read_data('presentations/slides/intro/img_resize.jpg');
foreach($data as $key=>$val) {
    if(is_array($val)) {
        foreach($val as $k=>$v) {
            echo $key."[$k]: $v<br />\n";
        }
    } else
        echo "$key: ".@substr($val,0,40)."<br />\n";
}
?>
```

Output:

```
FileName: img_resize.jpg
FileDateTime: 1027351588
FileSize: 669158
FileType: 2
MimeType: image/jpeg
SectionsFound: ANY_TAG, IFD0, THUMBNAIL, EXIF
COMPUTED[html]: width="1536" height="1024"
COMPUTED[Height]: 1024
COMPUTED[Width]: 1536
COMPUTED[IsColor]: 1
COMPUTED[ByteOrderMotorola]: 0
COMPUTED[ApertureFNumber]: f/4.0
COMPUTED[FocusDistance]: 1.07m
COMPUTED[Thumbnail.FileType]: 8
COMPUTED[Thumbnail.MimeType]: image/tiff
COMPUTED[Thumbnail.Height]: 64
COMPUTED[Thumbnail.Width]: 96
Make: Eastman Kodak Company
Model: KODAK DC265 ZOOM DIGITAL CAMERA (V01.00)
Orientation: 1
XResolution: 150/1
YResolution: 150/1
ResolutionUnit: 2
YCbCrPositioning: 1
Exif_IFD_Pointer: 190
THUMBNAIL[ImageWidth]: 96
THUMBNAIL[ImageLength]: 64
THUMBNAIL[BitsPerSample]: Array
THUMBNAIL[Compression]: 1
THUMBNAIL[PhotometricInterpretation]: 2
THUMBNAIL[StripOffsets]: 1748
THUMBNAIL[Orientation]: 1
THUMBNAIL[SamplesPerPixel]: 3
THUMBNAIL[RowsPerStrip]: 64
THUMBNAIL[StripByteCounts]: 18432
THUMBNAIL[XResolution]: 72/1
THUMBNAIL[YResolution]: 72/1
THUMBNAIL[PlanarConfiguration]: 1
THUMBNAIL[ResolutionUnit]: 2
ExposureTime: 1/250
FNumber: 400/100
ExifVersion: 0200
DateTimeOriginal: 1999:01:31 04:17:59
ComponentsConfiguration:
```

Fetching an embedded thumbnail

```
<?
```

```
Header('Content-type: image/tiff');
echo exif_thumbnail('p0004557.jpg');
?>
```

Super-cool Dynamic Image Generator

Want to be cooler than all your friends? Well here it is!

First, set up an ErrorDocument 404 handler for your images directory.

```
<Directory /home/doc_root/images>
    ErrorDocument 404 /images/generate.php
</Directory>')
```

Then generate.php looks like this:

```
<?php
$filename = basename($_SERVER['REDIRECT_URL']);
if(preg_match('/^([^_]*?)_([^_]*?)_([^_]*?)\.(.*?)$/',$filename, $reg)) {
    $type = $reg[1];
    $text = $reg[2];
    $rgb = $reg[3];
    $ext = $reg[4];
}

if(strlen($rgb)==6) {
    $r = hexdec(substr($rgb,0,2));
    $g = hexdec(substr($rgb,2,2));
    $b = hexdec(substr($rgb,4,2));
} else $r = $g = $b = 0;

    switch(strtolower($ext)) {
        case 'jpg':
            Header("Content-Type: image/jpg");
            break;
        case 'png':
        case 'gif': /* We don't do gif - send a png instead */
            Header("Content-Type: image/png");
            break;
        default:
            break;
    }

    switch($type) {
        case 'solid':
            $im = imagecreatetruecolor(80,80);
            $bg = imagecolorallocate($im, $r, $g, $b);
            imagefilledrectangle($im,0,0,80,80,$bg);
            break;
        case 'button':
            $si = 32; $font = "php";
            $im = imagecreatefrompng('blank_wood.png');
            $tsize = imagettfbbox($si,0,$font,$text);
            $dx = abs($tsize[2]-$tsize[0]);
            $dy = abs($tsize[5]-$tsize[3]);
            $x = ( imagesx($im) - $dx ) / 2;
            $y = ( imagesy($im) - $dy ) / 2 + $dy;
            $white = ImageColorAllocate($im,255,255,255);
            $black = ImageColorAllocate($im,$r,$g, $b);
            ImageTTFText($im, $si, 0, $x, $y, $white, $font, $text);
            ImageTTFText($im, $si, 0, $x+2, $y, $white, $font, $text);
            ImageTTFText($im, $si, 0, $x, $y+2, $white, $font, $text);
            ImageTTFText($im, $si, 0, $x+2, $y+2, $white, $font, $text);
            ImageTTFText($im, $si, 0, $x+1, $y+1, $black, $font, $text);
        break;
    }
    Header("HTTP/1.1 200 OK");
    $dest_file = dirname($_SERVER['SCRIPT_FILENAME']).'/'.$filename;
    switch(strtolower($ext)) {
```

```
            case 'png':
            case 'gif':
                @ImagePNG($im,$dest_file);
                ImagePNG($im);
                break;
            case 'jpg':
                @ImageJPEG($im,$dest_file);
                ImageJPEG($im);
                break;
        }
    ?>
```

The URL, http://localhost/images/button_test_000000.png produces this image:

A PDF Invoice

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf);
pdf_set_info($pdf, "Author","Rasmus Lerdorf");
pdf_set_info($pdf, "Title","Sample Invoice");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Sample Invoice");

$sizes = array('a4'=>'595x842', 'letter'=>'612x792', 'legal'=>'612x1008');

if(!isset($type)) $type='letter';
list($x,$y) = explode('x',$sizes[$type]);

$items = array(array('Our special low-cost widget that does everything','299.99'),
               array('Our special high-cost widget that does more','1899'),
               array('A blue widget','29.95'),
               array('And a red widget','49.95'),
               array('A yellow widget that makes noise','49.9'),
               array('And one that doesn\'t','999.95'),
              );

pdf_begin_page($pdf, $x, $y);

$im = pdf_open_jpeg($pdf, "php-big.jpg");
pdf_place_image($pdf, $im, 5, $y-72, 0.5);
pdf_close_image ($pdf,$im);

pdf_set_value($pdf, 'textrendering', 0); // fill

pdf_set_font($pdf, "Helvetica" , 12, winansi);
pdf_show_xy($pdf, 'Generic Evil Company Inc.',145,$y-20);
pdf_continue_text($pdf, '123 Main Street');
pdf_continue_text($pdf, 'Dark City, CA 98765');

pdf_set_font($pdf, "Helvetica" , 10, winansi);
pdf_show_xy($pdf, 'Helpless Customer Ltd.',20,$y-100);
pdf_continue_text($pdf, '2 Small Street');
pdf_continue_text($pdf, 'Little Town, ID 56789');

pdf_set_font($pdf, "Helvetica" , 10, winansi);
pdf_show_xy($pdf, 'Terms: Net 30',150,$y-100);
pdf_continue_text($pdf, 'PO #:  12345');

pdf_set_font($pdf, "Helvetica-Bold" , 30, winansi);
pdf_show_xy($pdf, "* I N V O I C E *",$x-250,$y-112);

pdf_setcolor($pdf,'fill','gray',0.9,0,0,0);
pdf_rect($pdf,20,80,$x-40,$y-212);
pdf_fill_stroke($pdf);

$offset = 184; $i=0;
while($y-$offset > 80) {
    pdf_setcolor($pdf,'fill','gray',($i%2)?0.8:1,0,0,0);
    pdf_setcolor($pdf,'stroke','gray',($i%2)?0.8:1,0,0,0);
    pdf_rect($pdf,21,$y-$offset,$x-42,24);
    pdf_fill_stroke($pdf);
    $i++; $offset+=24;
}

pdf_setcolor($pdf,'fill','gray',0,0,0,0);
pdf_setcolor($pdf,'stroke','gray',0,0,0,0);
pdf_moveto($pdf, 20,$y-160);
pdf_lineto($pdf, $x-20,$y-160);
```

```
pdf_stroke($pdf);

pdf_moveto($pdf, $x-140,$y-160);
pdf_lineto($pdf, $x-140,80);
pdf_stroke($pdf);

pdf_set_font($pdf, "Times-Bold" , 18, winansi);
pdf_show_xy($pdf, "Item",30,$y-150);
pdf_show_xy($pdf, "Price",$x-100,$y-150);


pdf_set_font($pdf, "Times-Italic" , 15, winansi);

$offset = 177;
foreach($items as $item) {
    pdf_show_xy($pdf, $item[0],30,$y-$offset);
    pdf_show_boxed($pdf, '$'.number_format($item[1],2), $x-55, $y-$offset, 0, 0,
'right');
    $offset+=24;
    $total += $item[1];
}

pdf_set_font($pdf, "Times-Bold" , 17, winansi);
$offset+=24;
pdf_show_xy($pdf, 'Total',30,$y-$offset);
pdf_show_boxed($pdf, '$'.number_format($total,2), $x-55, $y-$offset, 0, 0,
'right');

pdf_end_page($pdf);
pdf_close($pdf);

$data = pdf_get_buffer($pdf);
header('Content-type: application/pdf');
header("Content-disposition: inline; filename=invoice.pdf");
header("Content-length: " . strlen($data));
echo $data;
?>
```

See http://www.opaque.net/ming/

```
<?
    $s = new SWFShape();
    $fp = fopen('php-big.jpg','r');
    $jpg = new SWFBitmap($fp);
    $w = $jpg->getWidth(); $h = $jpg->getHeight();

    $f = $s->addFill($jpg);
    $f->moveTo(-$w/2, -$h/2);
    $s->setRightFill($f);

    $s->movePenTo(-$w/2, -$h/2);
    $s->drawLine($w, 0);
    $s->drawLine(0, $h);
    $s->drawLine(-$w, 0);
    $s->drawLine(0, -$h);

    $p = new SWFSprite();
    $i = $p->add($s);

    for($step=0; $step<360; $step+=2) {
        $p->nextFrame();
        $i->rotate(-2);
    }

    $m = new SWFMovie();
    $i = $m->add($p);
    $i->moveTo(230,120);
    $m->setRate(100);
    $m->setDimension($w*1.8, $h*1.8);

    header('Content-type: application/x-shockwave-flash');
    $m->output();
?>
```

Output:

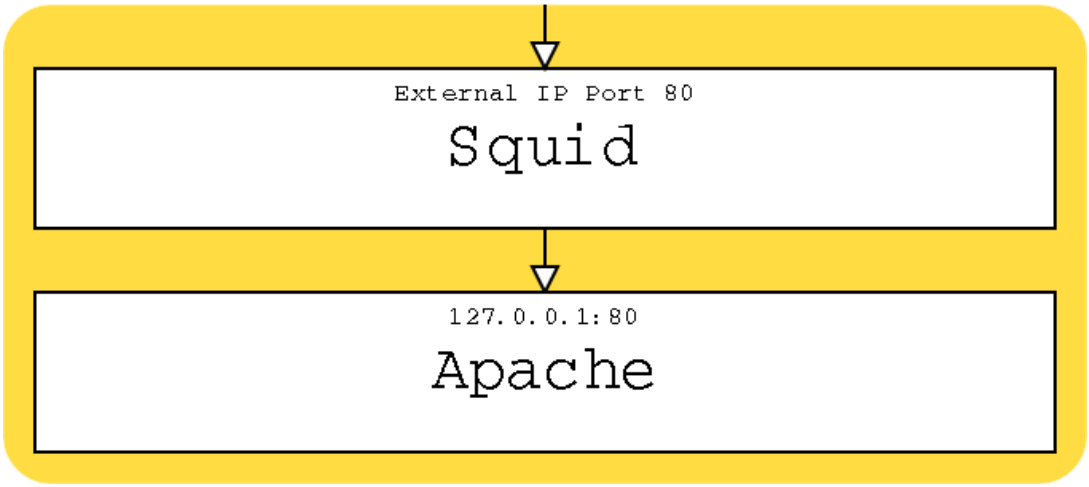
Flash + RSS/XML

```
<?php
require 'XML/RSS.php';

$r =& new XML_RSS('slashdot.rdf');
$r->parse();

$allItems = $r->getItems();
$itemCount = count($allItems);
$width = 1000;
$m = new SWFMovie();
$m->setDimension($width, 70);
$m->setBackground(0xcf, 0xcf, 0xcf);

$f = new SWFFont("../../../fonts/Techno.fdb");

$hit = new SWFShape();
$hit->setRightFill($hit->addFill(0,0,0));
$hit->movePenTo(-($width/2), -30);
$hit->drawLine($width, 0);
$hit->drawLine(0, 60);
$hit->drawLine(-$width, 0);
$hit->drawLine(0, -60);
$x = 0;

// build the buttons
foreach($allItems as $Item) {

    $title = $Item['title'];
    $link = $Item['link'];

    // get the text
    $t = new SWFText();
    $t->setFont($f);
    $t->setHeight(50);
    $t->setColor(0,0,0);
    $t->moveTo(-$f->getWidth($title)/2, 25);
    $t->addString($title);

    // make a button
    $b[$x] = new SWFButton();
    $b[$x]->addShape($hit, SWFBUTTON_HIT);
    $b[$x]->addShape($t, SWFBUTTON_OVER | SWFBUTTON_UP | SWFBUTTON_DOWN);
    $b[$x++]->addAction(new SWFAction("getURL('$link','_new');"), SWFBUTTON_MOUSEUP);
}


// display them
for($x=0; $x<$itemCount; $x++) {

    $i = $m->add($b[$x]);
    $i->moveTo($width/2,30);

    for($j=0; $j<=30; ++$j) {
        $i->scaleTo(sqrt(sqrt($j/30)));
        $i->multColor(1.0, 1.0, 1.0, $j/30);
        $m->nextFrame();
    }

    for($j=0; $j<=30; ++$j) {
        $i->scaleTo(sqrt(sqrt(1+($j/30))));
        $i->multColor(1.0, 1.0, 1.0, (30-$j)/30);
        $m->nextFrame();
    }
```

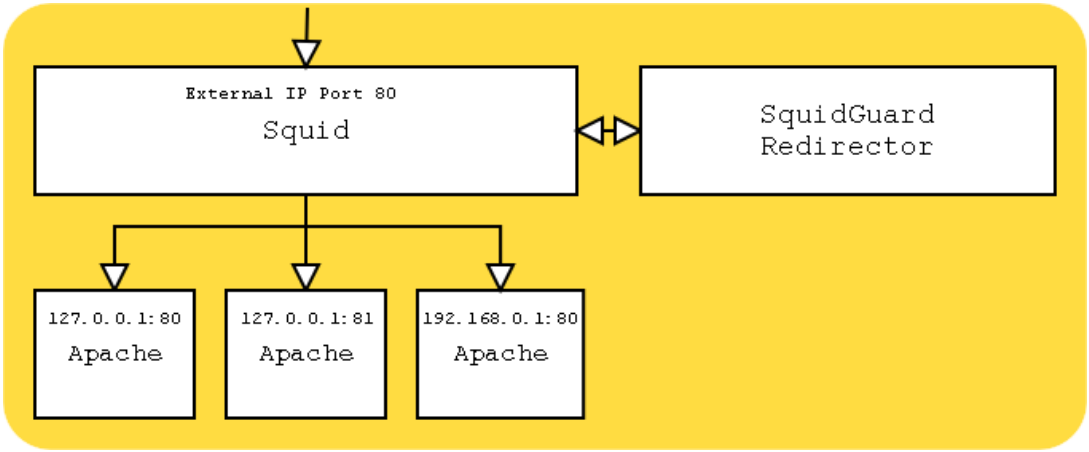
```
    $m->remove($i);

}

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Output:

For really busy sites, a reverse proxy like Squid is magical! Either run it as a single-server accelerator:



Or as a front-end cache to a number of local or remote servers:



Note:

Watch out for any use of $REMOTE_ADDR in your PHP scripts. Use $HTTP_X_FORWARDED_FOR instead.

Squid Configuration

Make it listen to port 80 on our external interface:

```
http_port 198.186.203.51:80
```

If we don't do cgi-bin stuff, comment these out:

```
#acl QUERY urlpath_regex cgi-bin
#no_cache deny QUERY
```

If we have plenty of RAM, bump this up a bit:

```
cache_mem 16MB
maximum_object_size 14096 KB
```

Specify where to store cached files (size in Megs, level 1 subdirs, level 2 subdirs)

```
cache_dir ufs /local/squid/cache 500 16 256
```

Get rid of the big store.log file:

```
cache_store_log none
```

Set our SNMP public community string:

```
acl snmppublic snmp_community public
```

Get rid of "allow all" and use list of hosts we are blocking (1 ip per line):

```
#http_access allow all
acl forbidden src "/local/squid/etc/forbidden"
http_access allow !forbidden
```

Set user/group squid should run as:

```
cache_effective_user squid
cache_effective_group daemon
```

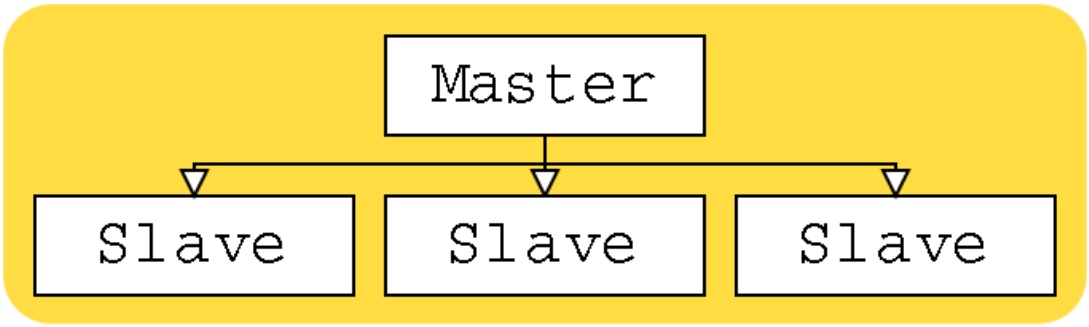
Single-server reverse proxy setup (set up Apache to listen to port 80 on the loopback):

```
httpd_accel_host 127.0.0.1
httpd_accel_port 80
httpd_accel_single_host on
httpd_accel_uses_host_header on
```

Only allow localhost access through snmp:

```
snmp_access allow snmppublic localhost
```

As of version 3.23.15 (try to use 3.23.29 or later), MySQL supports one-way replication. Since most web applications usually have more reads than writes, an architecture which distributes reads across multiple servers can be very beneficial.



In typical MySQL fashion, setting up replication is trivial. On your master server add this to your "my.cnf" file:

```
[mysqld]
log-bin
server-id=1
```

And add a replication user id for slaves to log in as:

```
GRANT FILE ON *.* TO repl@"%" IDENTIFIED BY 'foobar';
```

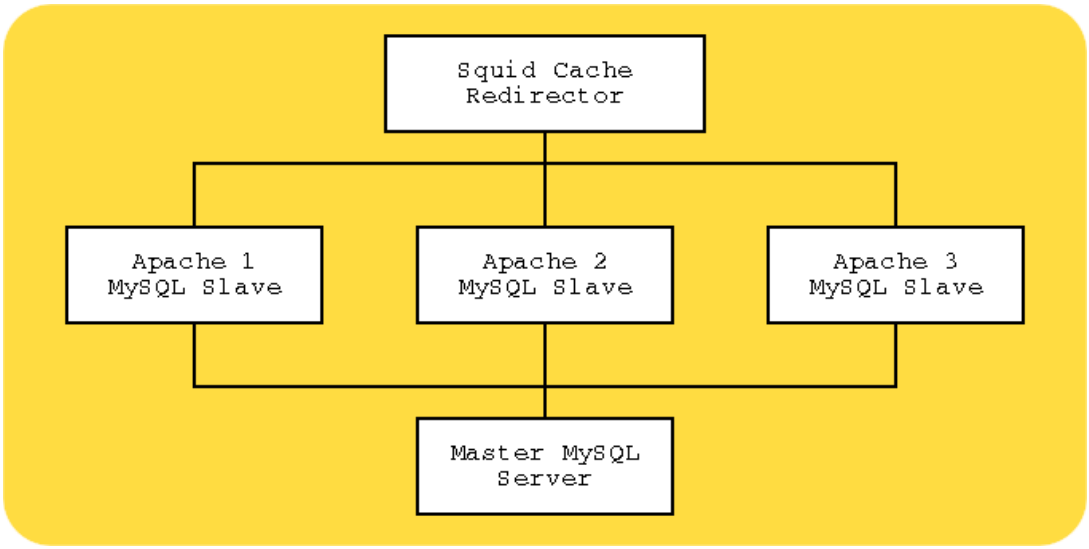
Then on your slave servers:

```
[mysqld]
set-variable = max_connections=200
log-bin
master-host=192.168.0.1
master-user=repl
master-password=foobar
master-port=3306
server-id=2
```

Make sure each slave has its own unique server-id. And since these will be read-only slaves, you can start them with these options to speed them up a bit:

```
--skip-bdb --low-priority-updates --delay-key-write-for-all-tables
```

Stop your master server. Copy the table files to each of your slave servers. Restart the master, then start all the slaves. And you are done. Combining MySQL replication with a Squid reverse cache and redirector and you might have an architecture like this:



You would then write your application to send all database writes to the master server and all reads to the local slave. It is also possible to set up two-way replication, but you would need to supply your own application-level logic to maintain atomicity of distributed writes. And you lose a lot of the advantages of this architecture if you do this as the writes would have to go to all the slaves anyway.

Resources

Home Page: http://www.php.net

Manual: http://php.net/manual

Tutorial: http://php.net/tut.php

Books: http://php.net/books.php

Index