PHP Tips and Tricks

LCA 2004

Jan.14, 2004. Adelaide

Rasmus Lerdorf <rasmus@php.net>

http://lerdorf.com/lca04.pdf

Many Things

PHP is Many Things to Many People

- The BASIC of the Web
- Web Template System
- General-purpose Scripting Language
- Advanced Application Framework
- Application Server?

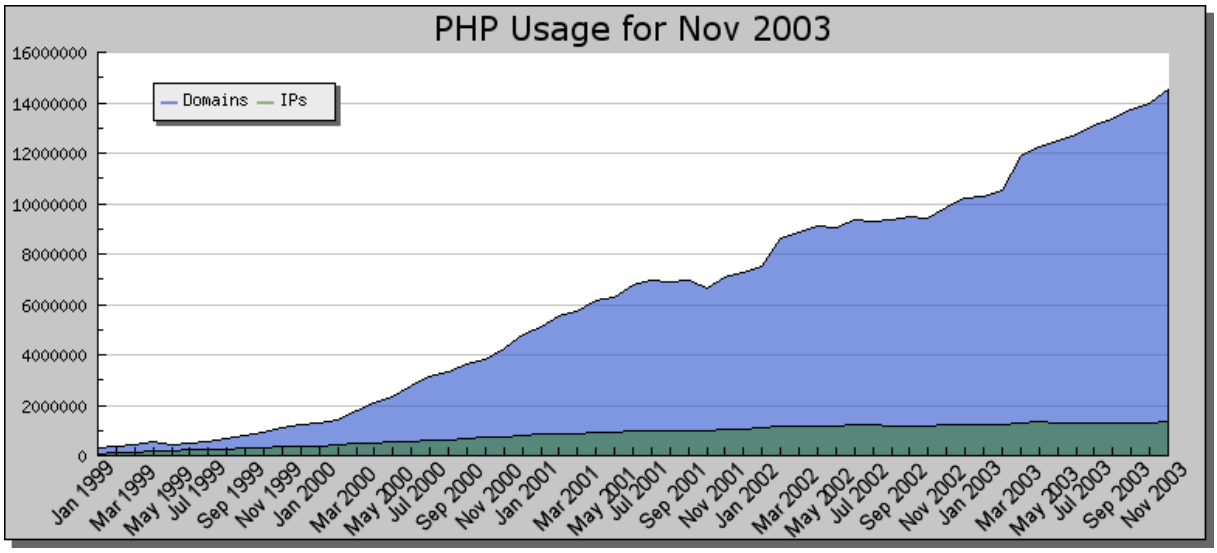
Template System

- A mechanism to separate logic from layout.
- The defining characteristic is where and how the intersection between logic and layout is done.
- PHP is a general-purpose templating system.
- Any general-purpose templating system will eventually become PHP.

PHP is Big!

January 2004 Netcraft Report

- 46,067,743 Domains queried
- 14,699,098 Domains. 1,330,821 IP addresses
- PHP installed on 32% of all domains



Source: Netcraft

January 2004 Apache Module Report

- 9,286,904 Apache Servers surveyed
- 5,010,035 (53.95%) PHP
- 2,608,258 (28.09%) OpenSSL
- 2,527,385 (27.21%) mod_ssl
- 1,845,621 (19.87%) Frontpage
- 1,597,830 (17.21%) mod_perl
- 411,443 (4.43%) DAV
- 380,012 (4.09%) mod_throttle
- 360,440 (3.88%) mod_log_bytes
- 355,235 (3.83%) mod_bwlimited
- 327,263 (3.52%) mod_jk
- 233,404 (2.51%) mod_fastcgi
- 222,410 (2.39%) AuthMySQL

Source: SecuritySpace.com

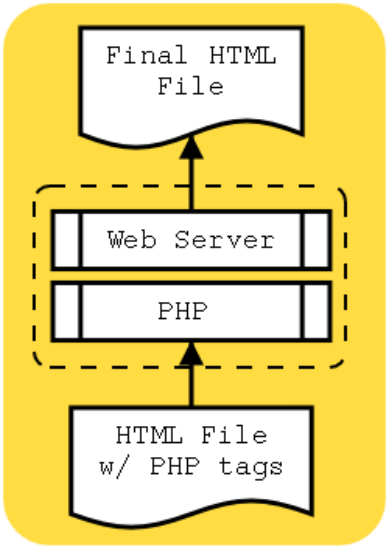
Killer Apps!

Thousands of cool PHP apps out there. Some better than others. Here are some I use:

- Gallery - http://gallery.menalto.com/
- SquirrelMail - http://squirrelmail.org
- FudForum - http://fud.prohost.org/
- Serendipity - http://s9y.org/
- Cacti - http://www.raxnet.net/products/cacti/

PHP is a Server-side language

Even though it is embedded in HTML files much like the client-side Javascript language, PHP is server-side and all PHP tags will be replaced by the server before anything is sent to the web browser.



So if the HTML file contains:

```
<html>
<?php echo "Hello World"?>
</html>
```

What the end user would see with a "view source" in the browser would be:

```
<html>
Hello World
</html>
```

Embedding PHP

The 4 available tag styles

```
<html>
<body>
<? echo 'Short Tags - Most common' ?>
<br />
<?php echo 'Long Tags - Portable' ?>
<br />
<= 'ASP Tags' >
<br />
<script language="php">
 echo 'Really Long Tags - rarely used';
</script>
<br />
</body>
</html>
```

Output:

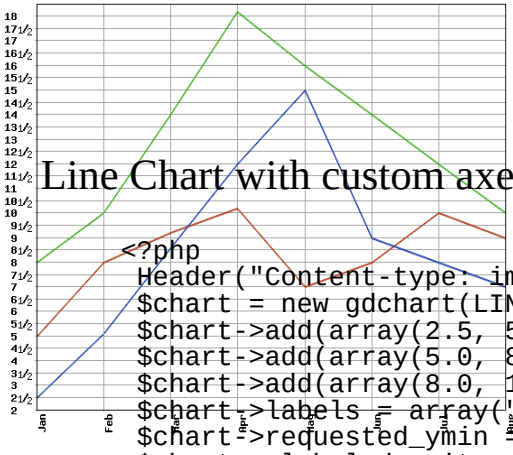
```
Short Tags - Most common
Long Tags - Portable
ASP Tags
Really Long Tags - rarely used
```

gdchart is a fast graphing extension written entirely in C. It can handle most common types of graphs with the only major missing feature being that it currently doesn't handle legends. Either create your own legend beside your graph, or maybe even layer it into the background image of your gdchart graph if you have a static legend.

Line Chart

```
<?php
 Header("Content-type: image/png");
 $chart = new gdchart(LINE);
 $chart->add(array(2.5, 5.1, 8.6, 12.0, 15, 9, 8, 7));
 $chart->add(array(5.0, 8.0, 9.2, 10.2, 7, 8, 10, 9));
 $chart->add(array(8.0, 10.0, 14.0, 18.2, 16, 14, 12, 10));
 $chart->labels = array("Jan","Feb","Mar","Apr","May","Jun","Jul", "Aug");
 $chart->colors = array(0x1133aa, 0xaa3311, 0x33aa11);
 $chart->out(640,480,IMG_PNG);
?>
```

Output:



Line Chart with custom axes

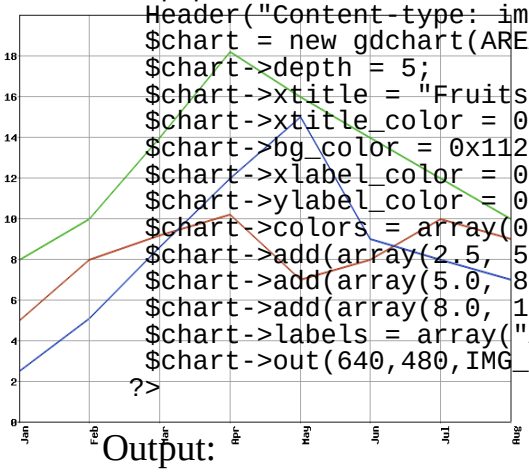
```
<?php
 Header("Content-type: image/png");
 $chart = new gdchart(LINE);
 $chart->add(array(2.5, 5.1, 8.6, 12.0, 15, 9, 8, 7));
 $chart->add(array(5.0, 8.0, 9.2, 10.2, 7, 8, 10, 9));
 $chart->add(array(8.0, 10.0, 14.0, 18.2, 16, 14, 12, 10));
 $chart->labels = array("Jan","Feb","Mar","Apr","May","Jun","Jul", "Aug");
 $chart->requested_ymin = 0;
 $chart->ylabel_density = 25;
 $chart->ylabel_fmt = "%.0f";
 $chart->colors = array(0x001133aa, 0x00aa3311, 0x33aa11);
 $chart->out(640,480,IMG_PNG);
?>
```

Output:

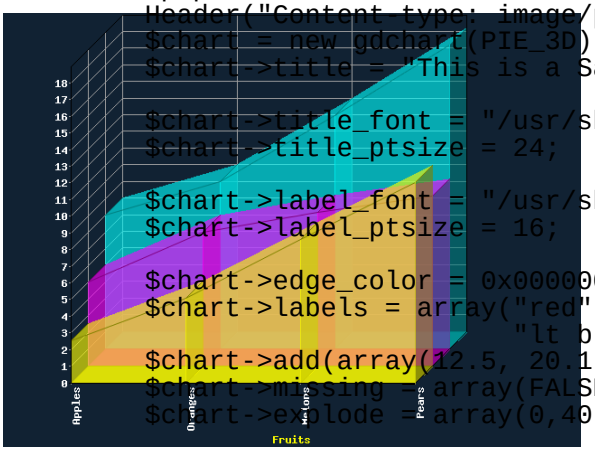
3D Area Chart

```
<?php
 Header("Content-type: image/png");
 $chart = new gdchart(AREA_3D);
 $chart->depth = 5;
 $chart->xtitle = "Fruits";
 $chart->xtitle_color = 0xffff00;
 $chart->bg_color = 0x112233;
 $chart->xlabel_color = 0xffffff;
 $chart->ylabel_color = 0xffffff;
 $chart->colors = array(0x30ffff00, 0x30ff00ff, 0x3000ffff);
 $chart->add(array(2.5, 5.1, 8.6, 12.0));
 $chart->add(array(5.0, 8.0, 9.2, 10.2));
 $chart->add(array(8.0, 10.0, 14.0, 18.2));
 $chart->labels = array("Apples","Oranges","Melons","Pears");
 $chart->out(640,480,IMG_PNG);
?>
```

Output:



Pie Chart

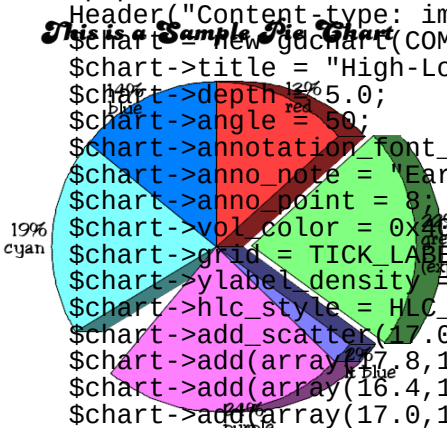



```
<?php
 Header("Content-type: image/png");
 $chart = new gdchart(PIE_3D);
 $chart->title = "This is a Sample Pie Chart";

 $chart->title_font = "/usr/share/fonts/truetype/CANDY.ttf ";
 $chart->title_ptsize = 24;

 $chart->label_font = "/usr/share/fonts/truetype/Jester.ttf";
 $chart->label_ptsize = 16;

 $chart->edge_color = 0x000000;
 $chart->labels = array("red","green\r\n(exploded)",
                        "lt blue","purple","missing","cyan","blue");
 $chart->add(array(12.5, 20.1, 2.0, 22.0, 5.0, 18.0, 13.0));
 $chart->missing = array(FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE);
 $chart->explode = array(0,40,0,0,0,0,0);

 $chart->pie_depth = 30;
 $chart->perspective = 0;
 $chart->pie_angle = 90;
 $chart->label_line = false;
 $chart->percent_labels = LABEL_ABOVE;

 $chart->out(640,480,IMG_PNG);
?>
```

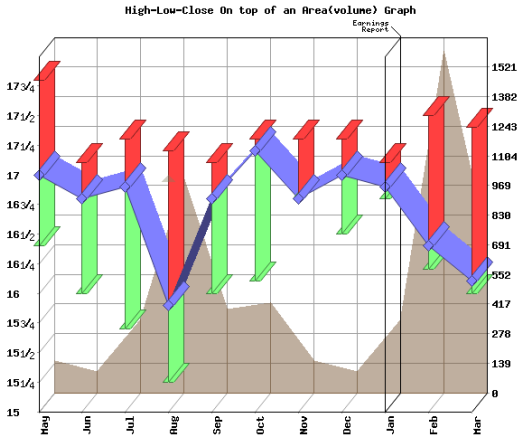
Output:

Combo High-Low-Close Chart

```
<?php
 Header("Content-type: image/png");
 $chart = new gdchart(COMBO_HLC_AREA_3D);
 $chart->title = "High-Low-Close On top of an Area(volume) Graph";
 $chart->depth = 5.0;
 $chart->angle = 50;
 $chart->annotation_font_size = FONT_TINY;
 $chart->anno_note = "Earnings\nReport";
 $chart->anno_point = 8;
 $chart->vol_color = 0x40806040;
 $chart->grid = TICK_LABELS;
 $chart->ylabel_density = 40;
 $chart->hlc_style = HLC_CONNECTING | HLC_I_CAP | HLC_DIAMOND;
 $chart->add_scatter(17.0, 3, SCATTER_TRIANGLE_UP, 0x50808060, 30);
 $chart->add(array(17.8,17.1,17.3,17.2,17.1,17.3,17.3,17.3,17.1,17.5,17.4));
 $chart->add(array(16.4,16.0,15.7,15.25,16.0,16.1,16.8,16.5,16.8,16.2,16.0));
 $chart->add(array(17.0,16.8,16.9,15.9,16.8,17.2,16.8,17.0,16.9,16.4,16.1));
 $chart->add_combo(
   array(150.0,100.0,340.0,999.0,390.0,420.0,150.0,100.0,340.0,1590.0,700.0));
 $chart->labels =
array("May","Jun","Jul","Aug","Sep","Oct","Nov","Dec","Jan","Feb","Mar","Apr");
 $chart->out(640,480,IMG_PNG);
?>
```



Output:


High-Low-Close On top of an Area(volume) Graph
Earnings Report
17 3/4
17 1/2
17 1/4
17
16 3/4
16 1/2
16 1/4
16
15 3/4
15 1/2
15 1/4
15
1521
1382
1243
1104
969
830
691
552
417
278
139
0
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar

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/

```
<?php
    $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);
    $m->setDimension($w1.8, $h1.8);

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

Output:

```
FWS  %  x  e      H  d* C ÿÿÿ•       ÿØÿÛ C
                      $.'
",#  (7),01444 '9=828••›™ KP÷Ô]|Q#{&QŠol¦ÕÚ^C‘N {k» Hx’>ÅKÇÃõ-'Ü ViUtAý ½š M+LI eµ— )ie
•¼_áÙÐRâµä šYNóÒy…f÷œPëë*–ñq{ÐÂ6 ü)•ªSè†Ò.óqZxè†•ƒýåîëËãTËŽ0a  Šæ=Ê ¬’;~ Q™s•4•ã„6w6“’•-
´àüSmÆÑï‰s A·µ
Ÿ ²êŠƒ™ä•íî¬ ¶m ° h®÷;  ¸Ü ˆ“ÊRÚuÏg ×>¹NSK,t’Hæ¥)]s0¥)@
R” ¥)@  •¡´W £¬(-¥• s r
õ×b”¹Õ    ö‹)5Ñc±iJïeÊ=Ñµ\  *Zµ\Oï}.ÞúÕðþ)´âh¼5ºJT°3q•lq¾±ïÝX ÛC©ÕZB‡MG gÚ¥&uªCÍ
```

```
prompt:~> pear list
Installed packages:
===================
+----------------+----------+--------+
| Package        | Version  | State  |
| Archive_Tar    | 0.9      | stable |
| Console_Getopt | 0.11     | beta   |
| DB             | 1.2      | stable |
| Mail           | 1.0      | stable |
| Net_Sieve      | 0.8      | stable |
| Net_Socket     | 1.0.1    | stable |
| PEAR           | 0.91-dev | beta   |
| XML_Parser     | 1.0      | stable |
| XML_RPC        | 1.0.3    | stable |
| XML_RSS        | 0.9.1    | stable |
| XML_Tree       | 1.1      | stable |
+----------------+----------+--------+

prompt:~> pear remote-list
Available packages:
===================
+----------------------+---------+
| Package              | Version |
| Archive_Tar          | 0.9     |
| Auth                 | 1.0.2   |
| Auth_HTTP            | 1.0.1   |
| Benchmark            | 1.1     |
| Cache                | 1.5.1   |
| Config               | 0.3.1   |
| Crypt_CBC            | 0.3     |
| Crypt_Rc4            | 0.1     |
| Date                 | 1.1     |
| DB                   | 1.3     |
| DB_ado               | 1.1     |
| DB_Pager             | 0.7     |
| File                 | 1.0.2   |
| File_Find            | 0.1     |
| File_SearchReplace   | 1.0     |
| HTML_Common          | 1.0     |
| HTML_QuickForm       | 2.3     |
| HTML_Table           | 1.1     |
| HTML_TreeMenu        | 1.0.3   |
| HTTP                 | 1.1     |
| HTTP_Request         | 1.0     |
| HTTP_Upload          | 0.8     |
| Log                  | 1.2     |
| Mail                 | 1.0     |
| Mail_Mime            | 1.2     |
| Net_CheckIP          | 1.0.1   |
| Net_Curl             | 0.1     |
| Net_Dig              | 0.1     |
| Net_Geo              | 1.0     |
| Net_NNTP             | 0.1     |
| Net_Ping             | 1.0.1   |
| Net_POP3             | 1.1     |
| Net_Portscan         | 1.0.1   |
| Net_Sieve            | 0.8     |
| Net_SMTP             | 1.0     |
| Net_Socket           | 1.0.1   |
| Net_URL              | 1.0.3   |
| Net_UserAgent_Detect | 1.0     |
| Numbers_Roman        | 0.1     |
| Pager                | 1.0.4   |
| Payment_Clieop       | 0.1     |
| PEAR                 | 0.9     |
| PHPUnit              | 0.3     |
| Science_Chemistry    | 1.0.2   |
| System_Command       | 1.0     |
| XML_CSSML            | 1.1     |
| XML_fo2pdf           | 0.97    |
| XML_image2svg        | 0.1     |
```

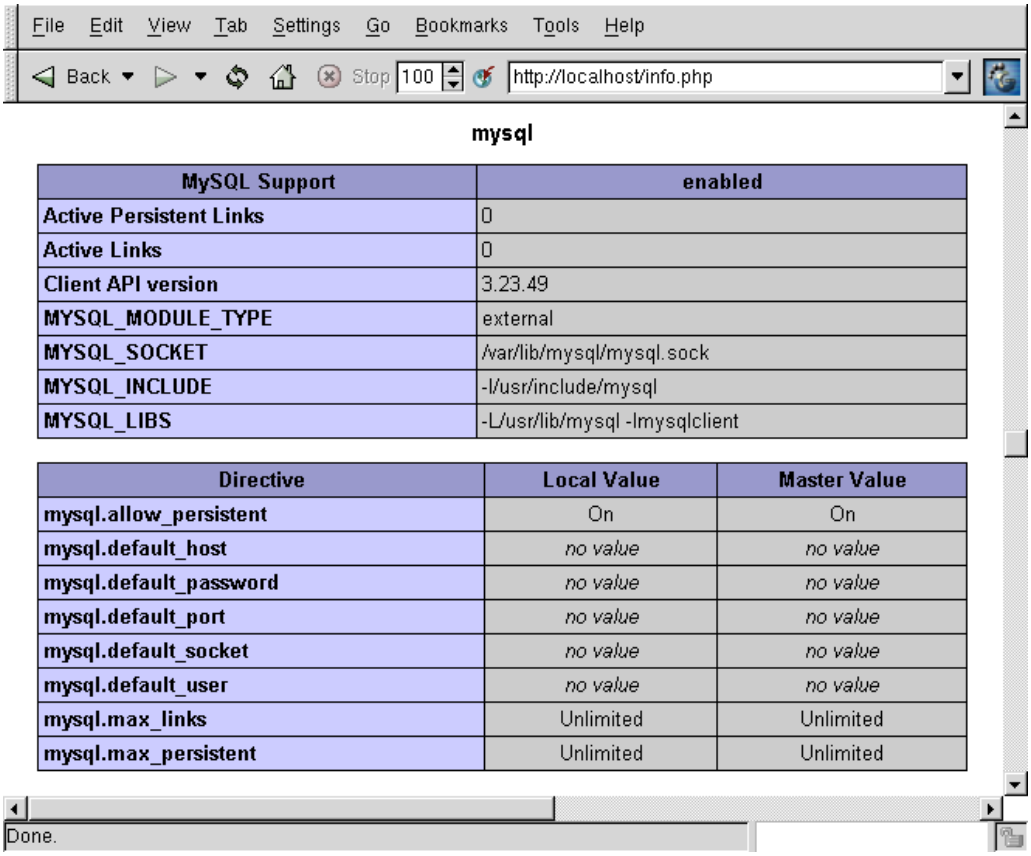
```
| XML_Parser           | 1.0     |
| XML_RPC              | 1.0.3   |
| XML_RSS              | 0.9.1   |
| XML_Transformer      | 0.3     |
| XML_Tree             | 1.1     |
+----------------------+---------+

prompt:~> pear list-upgrades
Available Upgrades (stable):
============================
+---------+---------+------+
| Package | Version | Size |
| DB      | 1.3     | 58kB |
+---------+---------+------+

prompt:~> pear upgrade DB
downloading DB-1.3.tgz ...
...done: 59,332 bytes
upgrade ok: DB 1.3

prompt:~> pear install PHPUnit
downloading PHPUnit-0.3.tgz ...
...done: 7,284 bytes
install ok: PHPUnit 0.3
```

- PHP Extension Code Library
- Shared extensions
- Makes use of the 'pear' tool
- Future home for all extensions

```
[derick@kossu nl]$ sudo pear -v install
http://files.derickrethans.nl/xdebug-1.2.0.tgz
downloading xdebug-1.2.0.tgz ...
...done: 58,608 bytes
28 source files, building
running: phpize
PHP Api Version        : 20020918
Zend Module Api No     : 20020429
Zend Extension Api No  : 20021010
building in /var/tmp/pear-build-root/xdebug-1.2.0
running: /tmp/tmpXOqy3A/xdebug-1.2.0/configure
running: make
xdebug.so copied to /tmp/tmpXOqy3A/xdebug-1.2.0/xdebug.so
install ok: xdebug 1.2.0
```

Check your PHP Setup for MySQL support

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



If not enabled

Some possible ways to enable MySQL support:

```
apt-get install php-mysql

rpm -Uvh php-mysql-4.2.2-1.i386.rpm



./configure --with-mysql=shared,/usr
cp modules/mysql.so /usr/local/lib/php

extension_dir=/usr/local/lib/php
extension=mysql.so
```

Sanity Check

Make sure MySQL is running

```
prompt:~> mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
| test      |
+-----------+
```

Or with the latest PHP

```
<? echo mysql_stat() ?>
```

Output:

```
Uptime: 1169  Threads: 1  Questions: 86  Slow queries: 15  Opens: 16  Flush
tables: 1  Open tables: 1 Queries per second avg: 0.074
```

Connecting to MySQL

The simple connection

```
<?
$conn = mysql_connect('localhost');
echo $conn;
?>
```

Output:

```
Resource id #91
```

Other variations

```
<?
mysql_connect('db.domain.com:33306','rasmus','foobar');
mysql_connect('localhost:/tmp/mysql.sock');
mysql_connect('localhost','rasmus','foobar',
              true,MYSQL_CLIENT_SSL|MYSQL_CLIENT_COMPRESS);
?>
```

Persistent Connections

The simple connection

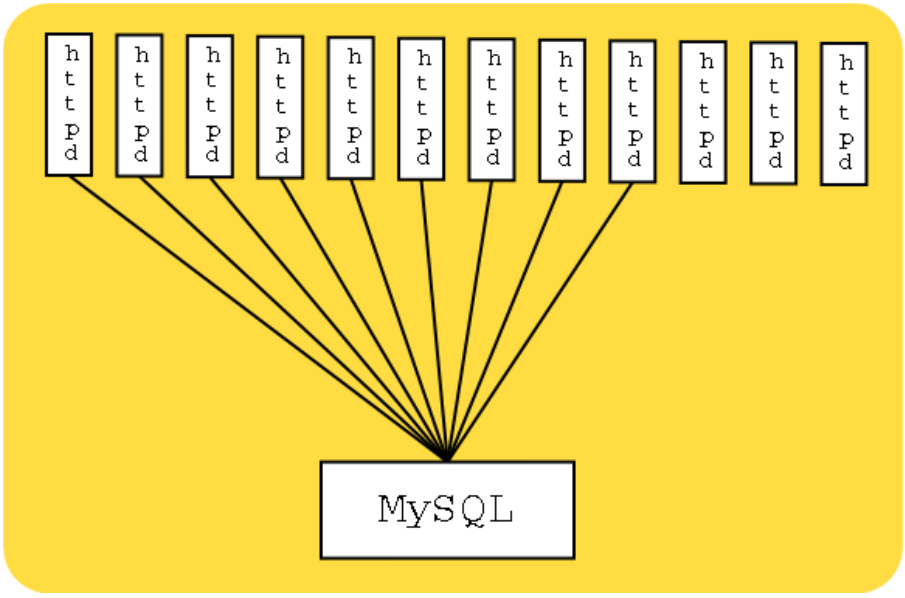
```
<?
$conn = mysql_pconnect('localhost');
echo $conn;
?>
```

Output:

```
Resource id #94
```

Caveats

- Watch out for multi-credential connections
- Make sure you match up max_connections and MaxClients



Creating a Database

Create a DB

```
<?
  mysql_connect('localhost');
  if(mysql_query("CREATE DATABASE foo")) {
    echo "Database foo created";
  } else {
    echo mysql_error();
  }
?>
```

Output:

```
Database foo created
```

Create a Table

```
<?
  mysql_select_db('foo');
  $result = mysql_query("CREATE TABLE users (
    id varchar(16) binary NOT NULL default '',
    Password varchar(16) NOT NULL default '',
    Name varchar(64) default NULL,
    email varchar(64) default NULL,
    ts timestamp(14) NOT NULL,
    PRIMARY KEY (id)
  )");
  if($result) {
    echo "Table created";
  } else {
    echo mysql_error();
  }
?>
```

Output:

```
Table created
```

INSERT Query

```
<?php
function add_user($id, $pass, $name, $email) {
  $result=mysql_query("insert into users values
                      ('$id',ENCRYPT('$pass'),'$name','$email',NULL)");
  if($result) {
    echo "Row inserted<br />";
  } else {
    echo mysql_error()."<br />";
  }
}

mysql_connect('localhost');
mysql_select_db('foo');

add_user('rasmus','foobar','Rasmus Lerdorf','rasmus@php.net');
add_user('carl','carlspass','Carl AlexandeR Lerdorf','carl@lerdorf.com');
?>
```

Output:

```
Row insertedRow inserted
```

Selecting Data

SELECT Query

```
<?
  mysql_connect('localhost');
  mysql_select_db('foo');
  $result = mysql_query("select * from users");
  if(!$result) echo mysql_error();
  else {
    while($row = mysql_fetch_row($result)) {
        echo "$row[0] - $row[1] - $row[2] - $row[3] - $row[4]<br />\n";
    }
  }
?>
```

Output:

```
rasmus - EGRsIW/RRmGbU - Rasmus Lerdorf - rasmus@php.net - 20040113154208
carl - EGMHoZe7RYfNY - Carl AlexandeR Lerdorf - carl@lerdorf.com -
20040113154208
```

mysql_fetch_array()

```
<?
  $result = mysql_query("select * from users order by id");
  if(!$result) echo mysql_error();
  else {
    while($row = mysql_fetch_array($result,MYSQL_ASSOC)) {
        echo "$row[id] - $row[Password] - $row[Name] -
              $row[email] - $row[ts]<br />\n";
    }
  }
?>
```

Output:

```
carl - EGMHoZe7RYfNY - Carl AlexandeR Lerdorf -
              carl@lerdorf.com - 20040113154208
rasmus - EGRsIW/RRmGbU - Rasmus Lerdorf -
              rasmus@php.net - 20040113154208
```

Dealing with timestamps

Using DATE_FORMAT

```
<?
mysql_connect('localhost');
mysql_select_db('foo');
$result = mysql_query(
  "select id, email,
   date_format(ts,'W M D, Y %r') as d
   from users order by ts");
if($result) {
  while($row = mysql_fetch_assoc($result)) {
    echo "$row[id] - $row[email] - $row[d]<br />\n";
  }
} else {
  echo mysql_error();
}
?>
```

Output:

```
rasmus - rasmus@php.net - Tuesday January 13th, 2004 03:42:08 PM
carl - carl@lerdorf.com - Tuesday January 13th, 2004 03:42:08 PM
```

Changing Existing Rows

Using UPDATE

```
<?
  mysql_connect('localhost');
  mysql_select_db('foo');
  $result = mysql_query(
    "update users set email = 'babycarl@lerdorf.com'
     where id = 'carl'");
  if($result) {
    echo mysql_affected_rows();
  } else {
    echo mysql_error();
  }
?>
```

Output:

```
1
```

REPLACE INTO

You can also use REPLACE INTO to update a row if it exists and insert it if it doesn't.

Magic Quotes

Escaping troublesome characters

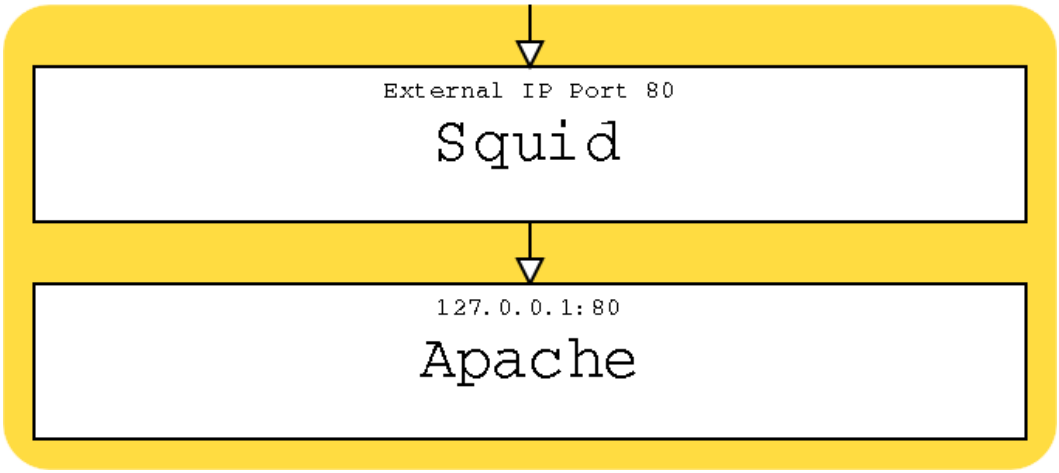
When you are inserting data into a MySQL database, certain characters have a special meaning and must therefore be escaped if you wish to insert these characters literally.

By default, PHP will escape these characters for you in any data coming from the user in GET, Post or Cookie data. This magic escaping is known as Magic Quotes and can be configured in your php.ini file by setting the magic_quotes_gpc directive.

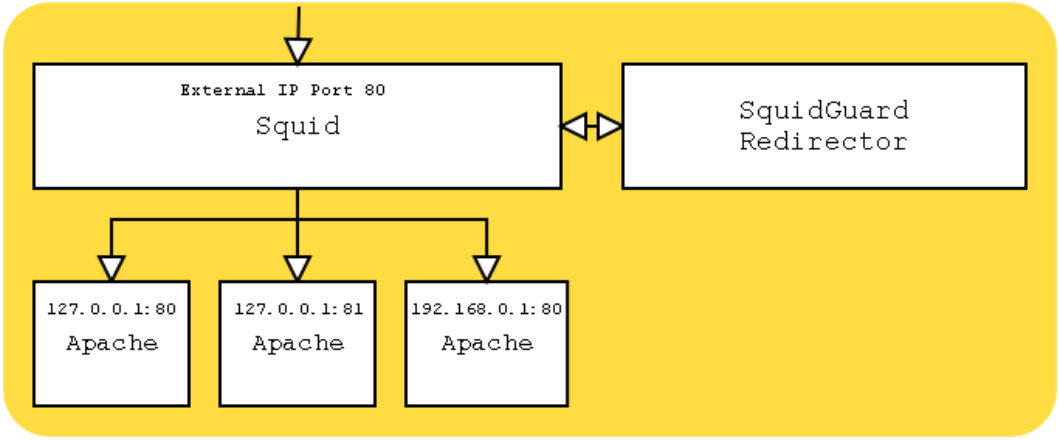
The characters affected are \ ' " and NUL (char 0). If these characters appear in user-supplied data they will be escaped with a \ (backslash).

Some people prefer to turn this feature off and handle escaping data manually using the addslashes() function. There is a converse function, stripslashes(), which removes the backslash characters in an escaped string.

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.

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
```

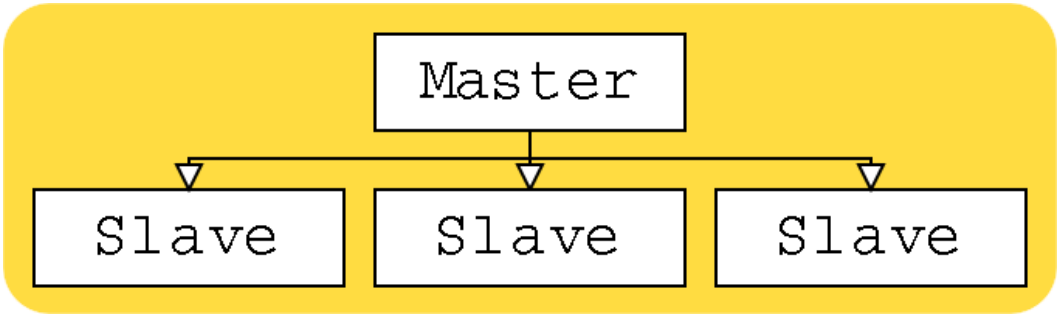
Single-server reverse proxy setup (set up Apache to listen to port 80 on the loopback):

```
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';
```

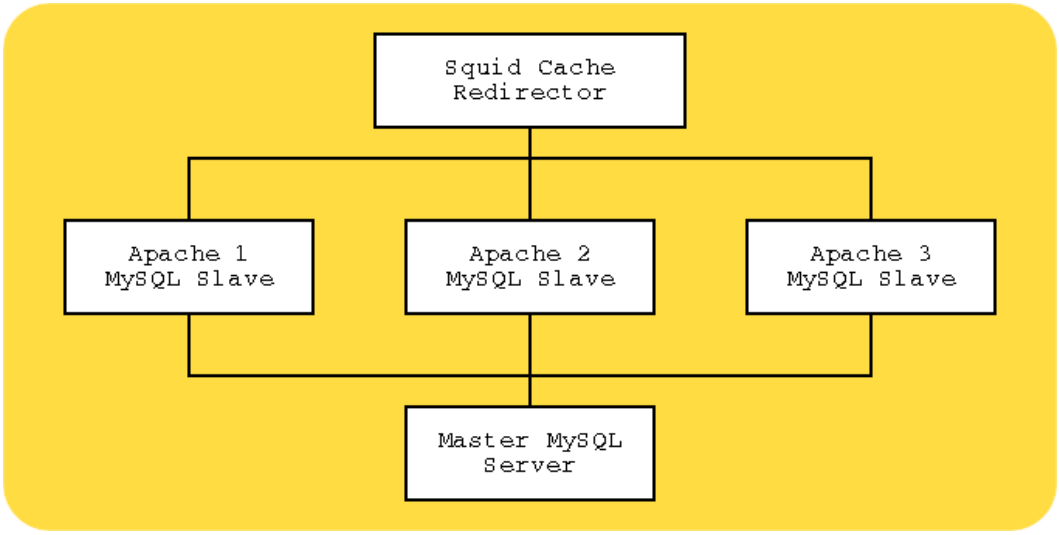
If you are using MySQL 4.0.2 or later, replace FILE with REPLICATION SLAVE in the above. Then on your slave servers:

```
[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.

$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

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:

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

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

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

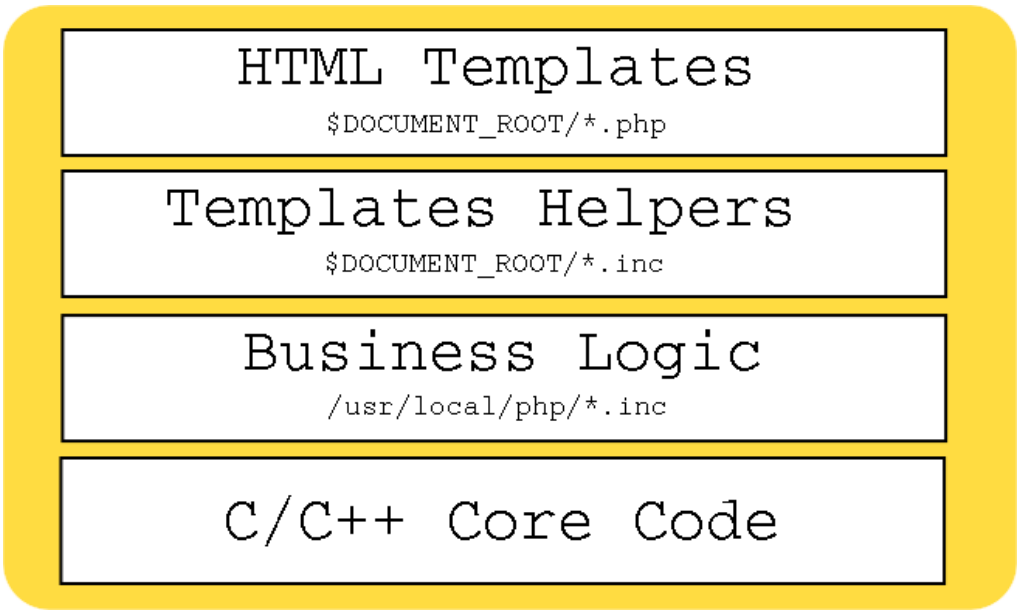
Then in your generate.php script use the contents of $REDIRECT_URI to determine which URL the person was trying to get to. In your database you would then have fields linking content to the URL they affect and from that you should be able to generate the page. Then in your generate.php script do something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

App Architecture

A suggested architecture for a PHP application. The template layer should have as little business logic as possible. As you go down you have less presentation and more business logic.



Simplistic Example

In terms of a real-world example of what goes in these 4 different layers, assume we are writing a database-backed application that needs to fetch a user record containing various fields. This is a very common thing to do in a web app. Our template layer might look something like this:

php.ini

```
auto_prepend_file = "./helpers.inc"
include_path = "/usr/local/lib/php"
```

Template Layer

```
<?title('Example Page')?>
<?greeting()?>
<h1>Heading 1</h1>
  <p>
    Page content
  </p>
<h1>Heading 2</h1>
  <p>
    Yada Yada
  </p>
<h1>Heading 3</h1>
  <p>
    Page content
  </p>
<?footer()?>
```

Template Helpers

```
<?php
  include "logic.inc";
  echo '<?xml version="1.0" encoding="UTF-8"?>';
?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="en" lang="en">
<?php
  $user = get_user_record($_COOKIE['user_id']);
  function greeting() {
    global $user;
    echo "Hi " .$user['first_name'] ."!<br />\n";
    if($age=birthday($user)) {
      echo "Congratulations, today is your ";
      echo "$age birthday!<br />\n";
    }
  }
  function title($title) {
    echo "<head><title>$title</title></head>\n";
    echo "<body>\n";
  }
  function footer() {
    echo "</body>\n</html>";
  }
?>
```

Business Logic

```
<?php
  function get_user_record($id) {
    mysql_connect('localhost');
    mysql_select_db('users');
    $res = mysql_query("select * from users where id='$id'");
    if(!$res) echo mysql_error();
```

```
    else $row = mysql_fetch_array($res);
    return $row;
  }
  function birthday($user) {
    if(strftime('m d')==strftime('m d',$user['bday']))
      $age = strftime('Y') - strftime('Y',$user['bday']);
      if(($age100)>10 && ($age100)<20) $ap='th';
      else switch($age%10) {
        case 1: $ap = 'st'; break;
        case 2: $ap = 'nd'; break;
        case 3: $ap = 'rd'; break;
        default:  $ap = 'th'; break;
      }
      return $age.$ap;
    else
      return false;
  }
?>
```

In this case the final layer written in C contains the mysql_* functions, and the date() function. These happen to be standard PHP functions. If birthday() is called many times on every single request and since how you figure out if it is someone's birthday is unlikely to change very often, this may be a good candidate to translate into C. Although, in this example, the birthday function is probably too simple to see much of a performance improvement. On the other hand, other than a little bit of added parameter parsing, if you compare the C version of birthday() to the PHP version, it isn't that much harder to write it in C.

C Layer

```
PHP_FUNCTION(birthday)
{
  time_t timestamp, now;
  struct tm ta1, tmbuf1, ta2, tmbuf2;
  int age;
  char ret_age[8];

  if (zend_parse_parameters(1 TSRMLS_CC, "l", &timestamp) == FAILURE)
      return;

  ta1 = php_localtime_r(&timestamp, &tmbuf1);
  time(&now);
  ta2 = php_localtime_r(&now, &tmbuf2);

  if(tmbuf1.tm_mday==tmbuf2.tm_mday && tmbuf1.tm_mon==tmbuf2.tm_mon) {
      age = tmbuf2.tm_year - tmbuf1.tm_year;
      if((age100)>10 && (age100)<19) sprintf(ret_age,"%dth",age);
      else switch(age % 10) {
          case 1: sprintf(ret_age,"%dst",age); break;
          case 2: sprintf(ret_age,"%dnd",age); break;
          case 3: sprintf(ret_age,"%drd",age); break;
          default:sprintf(ret_age,"%dth",age); break;
      }
  } else {
      RETURN_FALSE;
  }
  RETURN_STRING(ret_age,1);
}
```

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.

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");
?>
```

In PHP4

```
<?php
    class OS {
        var $name;
        function OS($name) {
            $this->name = $name;
        }
    }

    function changeName(&$obj, $name) {
        $obj->name = $name;
    }

    $linux = new OS('linux');
    $win = $linux;
    changeName($win, 'windows');
    echo $linux->name. "<br />\n";
    echo $win->name;
?>
```

Output:

```
linux
windows
```

In PHP5

```
<?php
    class OS {
        var $name;
        function OS($name) {
            $this->name = $name;
        }
    }

    function changeName($obj, $name) {
        $obj->name = $name;
    }

    $linux = new OS('linux');
    $win = $linux->__clone();
    changeName($win, 'windows');
    echo $linux->name. "\n";
    echo $win->name;
?>
```

Constructors and Destructors

```
<?php
    class OS {
        function __construct($name) {
            echo "begin";
        }

        function __destruct() {
            echo "done";
        }
    }

    $linux = new OS('linux');
    $linux = NULL;
?>
```

PHP5 OO - Private

- 'Strongest' access level
- Can only be accessed/called from the same class
- Force use of get()/set() functions to access properties

```
<?php
class Bedroom {
    private $action;

    function __construct() {
        $this->action = 'fun';
    }
}

$br = new Bedroom();
echo $br->action. "\n";
?>

Fatal error: Cannot access private property bedroom::$action in foo.php on line
12
```

PHP5 OO - Protected

Only accessable/callable by the same, or an inherited class.

```
<?php
class Bedroom {
    protected $action;

    function __construct() {
        $this->action = 'fun';
    }

    protected function peek() {
        echo $this->action. "\n";
    }
}

class Keyhole extends Bedroom {
    function peek() {
        echo $this->action. "\n";
    }
}

$kh = new Keyhole();
$kh->peek();

$br = new Bedroom();
$br->peek();           / Doesn't work /
?>
```

PHP5 OO - set/get

Property access handlers combined with 'private' can be used as a replacement for dynamic properties.

```
<?php
class OSses {
    private $names = array();

    public function __set($name, $value) {
        if (!in_array($name, array('windows', 'dos')))
            $this->names[$name] = $value;
    }

    public function getNames() {
        return $this->names;
    }
}

$osses = new OSses();
$osses->linux = 'rocks';
$osses->windows = 'blows';

var_dump($osses->getNames());
?>

array(1) {
  ["linux"]=>
  string(5) "rocks"
}
```

__call() is used to catch method calls the same way __get() and __set() is used to catch property accesses

```
<?php
class hello {
    function __call($name, $args) {
        echo "Hello ".ucfirst($name)."!\n";
    }
}

$h = new hello;
$h->rusty();
$h->anton();
?>

Hello Rusty!
Hello Anton!
```

PHP4:

```
<?php
if (mysql_connect('localhost')) {
   if (mysql_select_db('game')) {
      $r = mysql_query('SELECT * FROM news');

      ...
```

PHP5:

```
<?php
try {
    mysql_connect('localhost');
    mysql_select_db('game');
    $r = mysql_query('SELECT * FROM news');
} catch (Exception $e) {
    / Do recovery /
    exit();
}
```

```
<?php
    echo "<pre>";
    print_r(stream_get_wrappers());
    print_r(stream_get_filters());
    echo "</pre>";
?>

Array
(
    [0] => php
    [1] => file
    [2] => http
    [3] => ftp
    [4] => compress.zlib
)

Array
(
    [0] => string.rot13
    [1] => string.toupper
    [2] => string.tolower
    [3] => string.strip_tags
    [4] => convert.*
)
```

Examples

```
<?php
readfile("php://filter/read=string.toupper/resource=http://www.example.com");

$fp = fopen("compress.zlib://foo-bar.txt.gz", "wb");

$httpsfile = file_get_contents("https://www.example.com/foo.txt");

$sock = fsockopen("ssl://secure.example.com", 443, $errno, $errstr, 30);

?>
```

stream_copy_to_stream

```
<?php
$src = fopen('http://www.example.com','r');
$dest1 = fopen('first1k.txt','w');
$dest2 = fopen('remainder.txt','w');

stream_copy_to_stream($src, $dest1, 1024);
stream_copy_to_stream($src, $dest2);
?>
```

```
-B <begin_code>  Run PHP <begin_code> before processing input lines
-R <code>        Run PHP <code> for every input line
-F <file>        Parse and execute <file> for every input line
-E <end_code>    Run PHP <end_code> after processing all input lines

$ find . | \
    php -B '$l = 0;' \
        -R '$l+=count(file($argn));' \
        -E 'echo "Lines: $l\n";'

Lines: 102643
```

```
<?php
$doc = domxml_open_file('book.xml');
$root = $doc->child_nodes();
$root = $root[0];
$books = $root->child_nodes();

foreach ($books as $book) {
  if (@$book->tagname == 'book') {
    $content = $book->child_nodes();
    foreach ($content as $elem) {
      if (@$elem->tagname == 'author') {
        $author = $elem->get_content();
      }
      if (@$elem->tagname == 'title') {
        $title = $elem->get_content();
      }
    }
    echo "{$title} was written by {$author}\n";
  }
}
?>
```

No more SAX parsing and we get DTD validation and XPath queries

Book XML File

```
<books>
  <book>
    <title>The Grapes of Wrath</title>
    <author>John Steinbeck</author>
  </book>
  <book>
    <title>The Pearl</title>
    <author>John Steinbeck</author>
  </book>
</books>
```

Book PHP File

```
<?php
$books = simplexml_load_file('book.xml');

foreach ($books->book as $book) {
  echo "{$book->title} was written
        by {$book->author}<br>\n";
}
?>
```

World's Easiest RDF parser?

```
<?php
$rdf = simplexml_load_file('rss_feed.rdf');
echo $rdf->channel->title;
foreach ($rdf->channel->item as $item) {
    echo '<a href="'.$item->link.'">' .
         $item->title .
         '</a>';
}
?>
```

SQL Interface for flat files!

Example

```
<?php
@exec('rm /tmp/lb.db');
$db = sqlite_open('/tmp/lb.db', 0666, $sqliteerror);
if ($db) {
  sqlite_query($db, 'CREATE TABLE foo (bar varchar(10))');
  sqlite_query($db, "INSERT INTO foo VALUES ('fnord')");

  $result = sqlite_query($db,'select bar from foo');
  $record = sqlite_fetch_array($result);
  var_dump($record);
} else {
  die ($sqliteerror);
}
?>
```

Output:

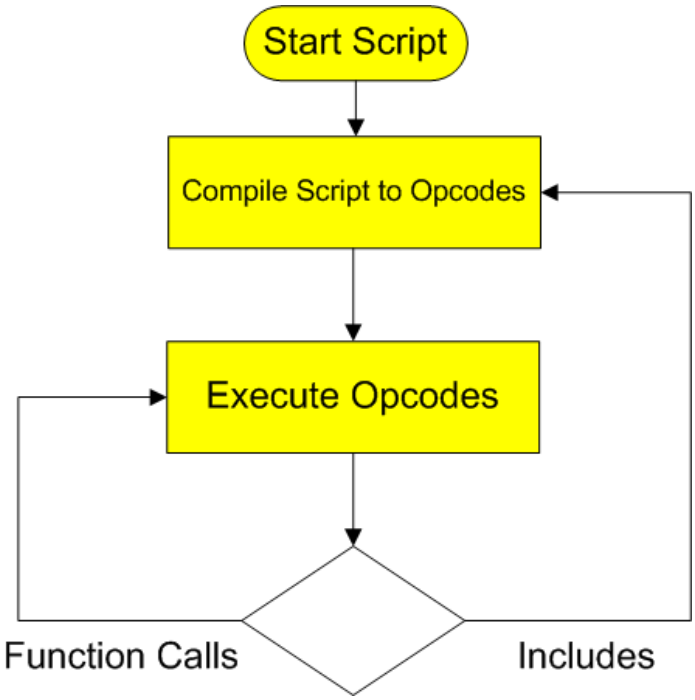
```
array
  0 =&gt; 'fnord'
  'bar' =&gt; 'fnord'
```

Same thing with the OO interface

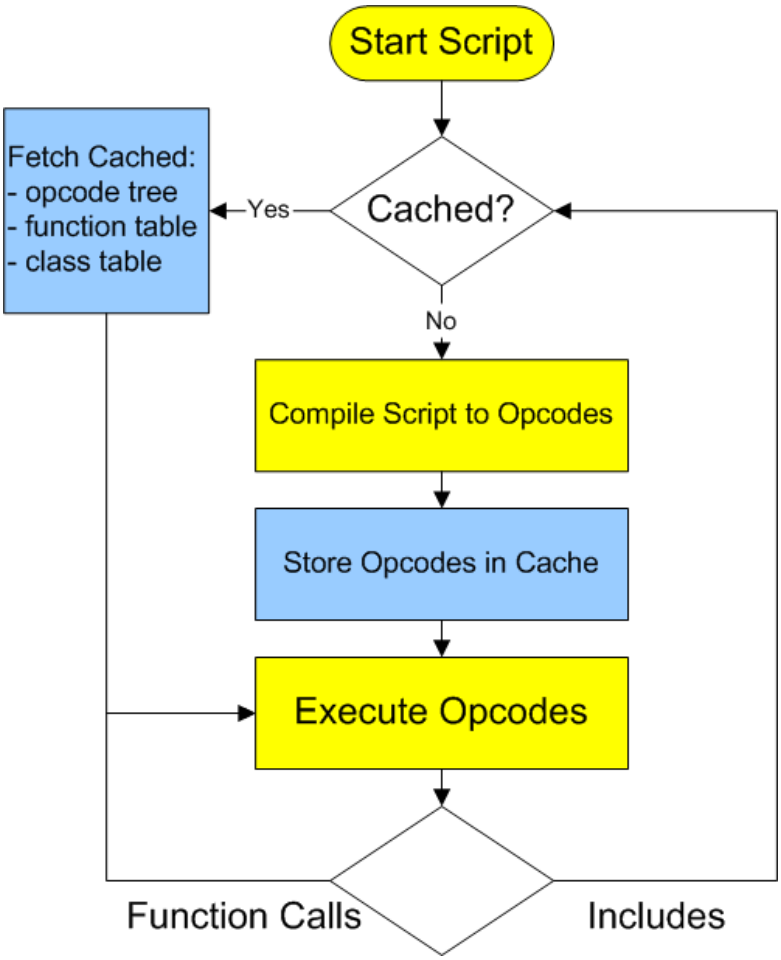
Example

```
<?php
$db = new sqlite_db(':memory:');

$db->query("CREATE TABLE foo(c1 date, c2 time, c3 varchar(64))");
$db->query("INSERT INTO foo VALUES ('2002-01-02', '12:49:00', NULL)");

$r = $db->query("SELECT * from foo");

$result = $r->fetch_array();
?>
```

Standard PHP



PHP with an Opcode Cache



There are a number of them out there.

- APC - open source
- IonCube Accelerator - free, but closed source
- Zend Cache - commercial

Installation

Typically very trivial. For IonCube:

```
yinst install yapache_libphp_phpa
```

All it does is stick a phpa.so somewhere and then tells PHP about it with this in a .ini file:

```
zend_extension = "/path/to/phpa.so"
```

When To Optimize?

Premature Optimization Is The Root of All Evil.

- Donald E. Knuth

Some simple guidelines

- Try to limit yourself to 5 or less includes per request
- Don't go overboard on OOP. Use where appropriate.
- Same goes for Layers. Abstraction, Indirection, abstract classes.
- Everything has a cost
- Use an opcode cache
- Watch your regular expressions!
- Cache! Cache! Cache!
- If you have plenty of CPU but limited bandwidth, turn on output compression

Benchmarking

Let's have a look at how we might benchmark and subsequently tune a PHP server. We will use two machines. A client machine to run our HTTP load program (http_load from acme.com) and a P4 1.7G server with 256M of RAM. We will also be using 3 freely available opcode caches:

http://cvs.php.net/cvs.php/pear/PECL/apc

http://www.turckware.ru/en/e_mmc.htm

http://www.ioncube.com

Don't blow your io buffers!

```
#  To change this on Linux, cat a larger number
#  into /proc/sys/net/core/wmem_max in your
#  httpd startup script
SendBufferSize 65535
```

This is our benchmark script

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Simple PHP Benchmark</title></head>
<body>
<h1>String Manipulation</h1>
<p>
<?php
$str = 'This is just a silly benchmark that doesn\'t do anything useful.';
$str .= 'Here we are just uppercasing the first two characters of every word ';
$str .= 'in this long string';
$parts = explode(' ',$str);
foreach($parts as $part) {
    $new[] = strtoupper(substr($part,0,2)).substr($part,2);
}
echo implode(' ',$new);
?>
</p>
<h1>Including 2 files</h1>
<p>
<?php
include './bench_include1.inc';
include './bench_include2.inc';
?>
</p>
<h1>for-loop and calling a function many times</h1>
<p>
<?php
$a = range(1,200);
$b = range(200,1);
for($i=0; $i<200; $i++) {
    echo foo($a[$i],$b[$i]);
}
?>
</p>
<h1>Define and Instantiate an object and call some methods</h2>
<p>
<?php
class test_class {
    var $test_var1;
    var $test_var2;
    var $test_var3;
    var $test_var4;


    function test_class($arg1, $arg2) {
        echo "Constructor args: $arg1, $arg2<br />\n";
    }
```

```
    function set_var1($value) {
        $this->test_var1 = $value;
        echo "test_var1 property set to $value<br />\n";
        return true;
    }
    function set_var2($value) {
        $this->test_var2 = $value;
        echo "test_var2 property set to $value<br />\n";
        return true;
    }
    function set_var3($value) {
        $this->test_var3 = $value;
        echo "test_var3 property set to $value<br />\n";
        return true;
    }
    function set_var4($value) {
        $this->test_var4 = $value;
        echo "test_var4 property set to $value<br />\n";
        return true;
    }
    function disp() {
        echo "<pre>";
        var_dump($this);
        echo "</pre>";
    }
}
$obj = new test_class(1,'arg2');
$obj->set_var1('test1');
$obj->set_var2(123);
$obj->set_var3($a);  / Array from previous test /
$obj->set_var4(array(1,2,3,4,5,6,7,8,9));
$obj->disp();
?>
</p>
</body>
</html>
```

bench_include1.inc

```
<?php
function foo($arg1, $arg2) {
    if($arg1>$arg2) return $arg1;
    elseif($arg1<$arg2) return $arg2;
    else return 'xxx';
}
?>
```

bench_include2.inc

```
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
This is just a bunch of plain text in an include file.<br />
```

```
This is just a bunch of plain text in an include file.<br />
```

An http_load run

```
2500 fetches, 5 max parallel, 1.962e+07 bytes, in 15.9486 seconds
7848 mean bytes/connection
156.753 fetches/sec, 1.2302e+06 bytes/sec
msecs/connect: 1.40331 mean, 2999.44 max, 0.171 min
msecs/first-response: 29.4603 mean, 648.731 max, 6.005 min
HTTP response codes:
  code 200 -- 2500
```

Whenever you do any sort of load testing, you need look beyond just the raw numbers and have a look at what your server is actually doing. Use vmstat:

Base PHP Load

```
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad0 ac0   in   sy  cs us sy id
25 0 0  149472  17572   26   0   0   0   0   0   0   0 1663 12390 4391 80 20  0
 5 0 0  149472  17532   27   0   0   0   0   0   0   0 1665 12322 4444 77 23  0
 7 0 0  149472  17492   24   0   0   0   0   0   2   0 1657 12409 4771 74 26  0
 5 0 0  149472  17452   28   0   0   0   0   0   0   0 1687 12520 4856 82 18  0
25 0 0  149472  17412   28   0   0   0   0   0   0   0 1649 12413 4756 78 22  0
 5 0 0  149472  17372   23   0   0   0   0   0   0   0 1645 12199 4417 77 23  0
 5 0 0  149084  17332   27   0   0   0   0   0   1   0 1679 12564 4385 76 24  0
 5 0 0  149464  17304   26   0   0   0   3   0   1   0 1663 12336 4551 79 21  0
 5 0 0  149464  17264   27   0   0   0   0   0   0   0 1662 12480 4663 82 18  0
```

Load with APC cache

```
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad0 ac0   in   sy  cs us sy id
 5 0 0  150140  11936   29   0   0   0   0   0   0   0 1629 12203 4687 75 25  0
 5 0 0  150140  11888   28   0   0   0   0   0   0   0 1619 12007 4579 79 21  0
 5 0 0  150140  11848   28   0   0   0   0   0   0   0 1640 12305 4252 76 24  0
 5 0 0  150140  11808   25   0   0   0   0   0   1   0 1630 12006 4628 84 16  0
25 0 0  150140  11776   27   0   0   0   0   0   0   0 1635 12304 4118 88 12  0
 5 0 0  150140  11736   26   0   0   0   0   0   0   0 1627 12059 4381 80 20  0
 5 0 0  150528  11696   22   0   0   0   0   0  11   0 1635 12178 4685 83 17  0
 5 0 0  150528  11656   26   0   0   0   0   0   0   0 1624 12027 4651 84 16  0
```

Load with IonCube cache

```
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad0 ac0   in   sy  cs us sy id
 5 0 0  127996  14516   28   0   0   0   0   0   0   0 1608 19343 4382 77 23  0
 5 0 0  127996  14476   25   0   0   0   0   0   0   0 1601 19416 4628 82 18  0
 8 0 0  127616  14436   23   0   0   0   0   0   1   0 1600 19181 4604 76 24  0
26 0 0  127616  14400   24   0   0   0   1   0   1   0 1605 19439 3940 84 16  0
 5 0 0  127616  14360   26   0   0   0   0   0   0   0 1612 19416 4582 79 21  0
 5 0 0  128956  14328   25   0   0   0   0   0   1   0 1599 19336 4559 72 28  0
 6 0 0  128956  14288   26   0   0   0   0   0   0   0 1598 19390 4444 79 21  0
```

Load with Turck MMCache

```
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad0 ac0   in   sy  cs us sy id
 3 0 0  136696  13284   29   0   0   0   0   0   0   0 1694 14846 4748 79 21  0
 5 0 0  136696  13244   33   0   0   0   0   0   1   0 1711 14815 4472 74 26  0
 5 0 0  136308  13204   28   0   0   0   0   0   0   0 1696 14824 4771 83 17  1
 5 0 0  136688  13164   27   0   0   0   0   0   0   0 1692 14590 4880 83 17  0
```

```
 6 0 0  136688  13116   26   0   0   0   0   0   1   0 1687 14762 4010 83 17  0
 5 0 0  135696  13004   25   0   0   0   0   0   0   0 1693 14759 4840 79 21  0
 7 0 0  135696  12956   26   0   0   0   0   0   1   0 1676 14477 4643 74 25  1
```

Our benchmark test was deliberately designed to have quite a bit of PHP processing and not a whole lot of output. 7k is somewhat small for a web page. If instead we have a whole lot of output, chances are we will be io-bound instead of cpu-bound. If you are io-bound, there is little sense in optimizing at the PHP level.

Evidence of an io-bound test

```
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad0 ac0   in   sy  cs us sy id
 4 0 0  132860  15724 1033   0   0   0   0   0   0   0 4457  954 3704  2 25 74
 5 0 0  132860  15724 1009   0   0   0   0   0   0   0 4436  714 3597  3 24 73
 6 0 0  132860  15716  980   0   0   0   0   0   0   0 4446  925 3603  5 23 72
 2 0 0  132860  15716 1028   0   0   0   0   0   6   0 4514  720 3696  2 24 73
 3 0 0  132472  15716 1018   0   0   0   0   0   2   0 4501  946 3673  2 22 76
 4 0 0  132472  15716 1039   0   0   0   0   0   0   0 4565  737 3718  2 26 73
 3 0 0  132472  15708 1010   0   0   0   0   0   2   0 4498  938 3639  2 24 75
 2 0 0  132472  15708 1012   0   0   0   0   0   0   0 4543  730 3665  5 25 70
```

Things to try if you are io-bound

```
[php.ini]
output_handler = ob_gzhandler


[httpd.conf]
LoadModule gzip_module lib/apache/mod_gzip.so
```

Benchmarking Results

So, we have determined we are cpu-bound and we need to go faster. What can we do? Some low-hanging fruit:

include_path

```
include_path = "/usr/share/pear:."
```

```
<?php
    include './template_helpers.inc';
    include 'business_logic.inc';
?>
```

open_basedir

```
open_basedir = "/usr/share/htdocs/:/usr/share/pear/"
```

open_basedir checking adds some extra syscalls to every file operation. It can be useful, but it is rather expensive, so turn it off if you don't think you need it.

variables_order

```
variables_order = "GPC"
```

```
<?php
    echo $_SERVER['DOCUMENT_ROOT'];
    echo getenv('DOCUMENT_ROOT');
?>
```

If you never use cookies, turn those off too

Add an Opcode Cache

```
zend_extension=/usr/local/lib/php/20020429/apc.so
apc.enabled=1
apc.shm_segments=1
apc.optimization=0
apc.shm_size=30
apc.num_files_hint=10
apc.gc_ttl=10
apc.mmap_file_mask=/tmp/apc.XXXXXX
apc.filters=
```

Results

156 request/sec	Standard PHP 4.3
161 requests/sec	Fix include_path, and only populate GP
191 requests/sec	Add IonCube Accelerator on top
196 requests/sec	APC no optimizer, IPC shared mem + semaphore locking
198 requests/sec	Turck MMCache no optimizer with spinlocks
200 requests/sec	APC no optimizer, mmap mem + user space sleep locks
202 requests/sec	Turck MMCache with optimizer and spinlocks

Why Profile?

Because your assumptions of how things work behind the scenes are not always correct. By profiling your code you can identify where the bottlenecks are quantitatively.

How?

PECL to the rescue!

```
www:~> pear install apd
downloading apd-0.4p1.tgz ...
...done: 39,605 bytes
16 source files, building
running: phpize
PHP Api Version        : 20020918
Zend Module Api No     : 20020429
Zend Extension Api No  : 20021010
building in /var/tmp/pear-build-root/apd-0.4p1
running: /tmp/tmprFlAqf/apd-0.4p1/configure
running: make
apd.so copied to /tmp/tmprFlAqf/apd-0.4p1/apd.so
install ok: apd 0.4p1
```

Then in your php.ini file:

```
zend_extension = "/usr/local/lib/php/apd.so"
apd.dumpdir = /tmp
```

It isn't completely transparent. You need to tell the profiler when to start profiling. At the top of a script you want to profile, add this call:

```
<?php
apd_set_pprof_trace();
?>
```

The use the command-line tool called pprofp:

```
wwww: ~> pprofp
pprofp <flags> <trace file>
    Sort options
    -a          Sort by alphabetic names of subroutines.
    -l          Sort by number of calls to subroutines
    -m          Sort by memory used in a function call.
    -r          Sort by real time spent in subroutines.
    -R          Sort by real time spent in subroutines (inclusive of child
calls).
    -s          Sort by system time spent in subroutines.
    -S          Sort by system time spent in subroutines (inclusive of child
calls).
    -u          Sort by user time spent in subroutines.
    -U          Sort by user time spent in subroutines (inclusive of child
calls).
    -v          Sort by average amount of time spent in subroutines.
    -z          Sort by user+system time spent in subroutines. (default)

    Display options
    -c          Display Real time elapsed alongside call tree.
    -i          Suppress reporting for php builtin functions
    -O <cnt>    Specifies maximum number of subroutines to display. (default 15)
    -t          Display compressed call tree.
    -T          Display uncompressed call tree.
```

```
% pprofp -z /tmp/pprof.48478

Trace for /home/y/share/htdocs/bench_main.php
Total Elapsed Time =    0.01
Total System Time  =    0.01
```

```
Total User Time    =    0.01


         Real         User        System              secs/    cumm
%Time (excl/cumm)  (excl/cumm)  (excl/cumm) Calls    call    s/call  Memory
Usage Name
--------------------------------------------------------------------------------
100.0  0.00  0.00   0.01  0.01   0.01  0.01   200   0.0001    0.0001
0 foo
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 test_class->set_var2
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 test_class->set_var1
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 test_class->set_var3
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 test_class->set_var4
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 var_dump
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 test_class->disp
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 test_class->test_class
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 main
  0.0  0.00  0.00   0.00  0.00   0.00  0.00    26   0.0000    0.0000
0 strtoupper
  0.0  0.00  0.00   0.00  0.00   0.00  0.00    52   0.0000    0.0000
0 substr
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 implode
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     2   0.0000    0.0000
0 include
  0.0  0.01  0.01   0.00  0.00   0.00  0.00     2   0.0000    0.0000
0 range
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
0 explode


Trace for /home/rasmus/phpweb/index.php
Total Elapsed Time =    0.69
Total System Time  =    0.01
Total User Time    =    0.08

         Real         User        System              secs/    cumm
%Time (excl/cumm)  (excl/cumm)  (excl/cumm) Calls    call    s/call  Memory
Usage Name
--------------------------------------------------------------------------------
 33.3  0.11  0.13   0.02  0.03   0.01  0.01     7   0.0043    0.0057
298336 require_once
 22.2  0.02  0.02   0.02  0.02   0.00  0.00   183   0.0001    0.0001
-33944 feof
 11.1  0.01  0.01   0.01  0.01   0.00  0.00     3   0.0033    0.0033
-14808 define
 11.1  0.04  0.04   0.01  0.01   0.00  0.00   182   0.0001    0.0001
112040 fgetcsv
 11.1  0.25  0.25   0.01  0.01   0.00  0.00     6   0.0017    0.0017
3768 getimagesize
 11.1  0.01  0.01   0.01  0.01   0.00  0.00    55   0.0002    0.0002
2568 sprintf
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     7   0.0000    0.0000
-136 printf
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
136 htmlspecialchars
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
-16 mirror_provider_url
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     7   0.0000    0.0000
112 spacer
  0.0  0.00  0.00   0.00  0.00   0.00  0.00    10   0.0000    0.0000
-552 delim
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
112 mirror_provider
  0.0  0.00  0.00   0.00  0.00   0.00  0.00    20   0.0000    0.0000
-624 print_link
```

```
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
24 have_stats
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
-72 make_submit
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     2   0.0000    0.0000
112 strrchr
  0.0  0.08  0.08   0.00  0.00   0.00  0.00     2   0.0000    0.0000
168 filesize
  0.0  0.00  0.00   0.00  0.00   0.00  0.00     1   0.0000    0.0000
-16 commonfooter
  0.0  0.00  0.11   0.00  0.00   0.00  0.00     2   0.0000    0.0000
0 download_link
  0.0  0.00  0.25   0.00  0.01   0.00  0.00     6   0.0000    0.0017
208 make_image
```

Xdebug can profile as well: http://www.xdebug.org/docs-profiling.php

Installing Xdebug

```
pear -v install http://files.derickrethans.nl/xdebug-1.3.0.tgz
+ create dir /usr/local/lib/php/docs/xdebug
52 source files, building
building in /tmp/tmpH4fmdM/xdebug-1.3.0
running: phpize
Configuring for:
  PHP Api Version:   20020918
  Zend Module Api No:   20020429
  Zend Extension Api No:   20021010
building in /var/tmp/pear-build-root/xdebug-1.3.0
running: /tmp/tmpH4fmdM/xdebug-1.3.0/configure
checking build system type... i686-redhat-linux-gnu
checking host system type... i686-redhat-linux-gnu
checking for gcc... gcc
checking for C compiler default output... a.out
...
Build process completed successfully
Installing 'xdebug.so' at ext_dir
(/usr/local/lib/php/extensions/no-debug-non-zts-20020429/xdebug.so)
about to commit 16 file operations
successfully committed 16 file operations
install ok: xdebug 1.3.0
```

Stack trace

```
<?php
    xdebug_start_trace();
    function a() {
        b();
    }
    function b() {
        xdebug_dump_function_trace();
    }
    a();
?>
```

Output:

```
Function trace
Time#FunctionLocationMemory
7.925359            ->a()/home/rasmus/phpweb/pres2/display.php(1539) : eval()'d
code:92978560
7.925380              ->b()/home/rasmus/phpweb/pres2/display.php(1539) :
eval()'d code:42978560
```

Modified error handler

```
<?php
    echo $a[1];
?>
```

Output:

```
Notice: Undefined variable:  a in /home/rasmus/phpweb/pres2/display.php(1539) :
eval()'d code on line 2
Call Stack
#FunctionLocation
1{main}()/home/rasmus/phpweb/pres2/show.php:0
2_presentation->display()/home/rasmus/phpweb/pres2/show.php:104
3pdfa4->_presentation()/home/rasmus/phpweb/pres2/objects.php:124
4_example->display()/home/rasmus/phpweb/pres2/display.php:1124
```

```
5pdfa4->_example()/home/rasmus/phpweb/pres2/objects.php:124
6eval()/home/rasmus/phpweb/pres2/display.php:1539
```

Modified var_dump()

```
<?php
    var_dump($_COOKIE);
?>
```

Output:

```
array
  'dims' =&gt; '1024_705'
  'LAST_SEARCH' =&gt;
'quickref,pres2/presentations/slides/php5intro/private.xml'
  'LAST_LANG' =&gt; 'en'
  'PHPSESSID' =&gt; '5bf70a24dc6723b683c3c212fd2b0600'
  'COUNTRY' =&gt; 'NA,127.0.0.1'
```

Stop!

One of the most important aspects of performance tuning and even programming in general is knowing when to stop. Getting your web site or product into the hands of the customer is after all the goal of all this. If you sit around and fiddle with it for months, nobody is using it.

Once you have located all the low-hanging fruit, further optimization tends to get exponentially harder for less and less gain. The time you spend on it compared to simply buying another server or two is usually not worth it. Stop and move onto the next project.

Or better yet, go home, spend some time away from these beastly computers and spend some time with your wife and/or kids

Resources

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

Manual: http://php.net/manual

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

Bugs: http://bugs.php.net

PEAR: http://pear.php.net

PECL: http://pecl.php.net

Xdebug: http://xdebug.org

Index