

Advanced PHP

PHP Quebec

March 31, 2005. Montreal

Rasmus Lerdorf <rasmus@php.net>

<http://talks.php.net/mtladv05>

Security

Large developer teams

High-complexity applications

High-traffic Applications

I18n and L10n



Security
Meet Dewie the turtle!





Brought to us by our friends at the FTC <http://www.ftc.gov/infosecurity/>



Indirect Attacks

- * XSS - Cross-site scripting attacks
- * Spoofing

Direct Attacks

- * Buffer Overflows
- * Path tricks
- * Application logic attacks

Overflows

It is a bit difficult to talk about buffer overflows because we tend to fix them as soon as we discover them. But some overflows we have hit in the past include problems in:

- o serialize/unserialize
- o pack/unpack
- o jpg algorithm
- o exif header field overflows
- o static buffer in getMimeHeaders() in ycore++

There is also a current issue in most libc realpath() calls that is troublesome.

realpath man page

```

REALPATH(3)                FreeBSD Library Functions Manual
REALPATH(3)

NAME
    realpath - returns the canonicalized absolute pathname

LIBRARY
    Standard C Library (libc, -lc)

SYNOPSIS
    #include <sys/param.h>
    #include <stdlib.h>

    char *
    realpath(const char *pathname, char
    resolved_path[MAXPATHLEN]);

DESCRIPTION
    The realpath() function resolves all symbolic links, extra
    ``/' characters and references to ./ and ../ in pathname, and copies
    the resulting absolute pathname into the memory referenced by
    resolved_path. The resolved_path argument must refer to a buffer capable of
    storing at least MAXPATHLEN characters.

    The realpath() function will resolve both absolute and
    relative paths and return the absolute pathname corresponding to pathname.
    All but the last component of pathname must exist when realpath() is called.

RETURN VALUES
    The realpath() function returns resolved_path on success.
    If an error occurs, realpath() returns NULL, and resolved_path contains
    the pathname which caused the problem.

ERRORS
    The function realpath() may fail and set the external
    variable errno for any of the errors specified for the library functions
    chdir(2), close(2),

```

fchdir(2), lstat(2), open(2), readlink(2) and getcwd(3).

CAVEATS

This implementation of realpath() differs slightly from the Solaris implementation. The 4.4BSD version always returns absolute pathnames, whereas the Solaris implementation will, under certain circumstances, return a relative resolved_path when given a relative pathname.

SEE ALSO

getcwd(3)

HISTORY

The realpath() function call first appeared in 4.4BSD.

Imagine this C code

```
#define DOCROOT "/home/y/share/htdocs"
#define SCRIPT "my_script"

int len = sizeof(DOCROOT)
        + sizeof(SCRIPT);
        + strlen(user_input);

char path = (char )malloc(len+1);
char safe_path[MAXPATHLEN];

snprintf(path, len, "s/s/%s", DOCROOT, user_input, SCRIPT);
if(realpath(path, safe_path)) {
    DIR *dir = opendir(safe_path);
    ...
}
```

Problem code in FreeBSD realpath.c

```
...
resolved_len = strlcat(resolved, next_token, PATH_MAX);
if (resolved_len >= PATH_MAX) {
    errno = ENAMETOOLONG;
    return (NULL);
}
...
```

It makes sure we never get a string back longer than PATH_MAX, but the silent strlcat truncation of the tokenized source path is a big problem!



Dumb things

Don't do dumb things!

```
<?php
    system($user_data);
?>
```

```
<?php
    include "$path/$user_data";
?>
```

```
<?php
    eval($user_data);
?>
```

Others

preg_replace with /e option, exec(), popen(), passthru, and backticks ``

Security in a web application boils down to always checking any user-supplied input data.

Exploits

- o `readfile($filename)`
- o `system($cmd)`
- o file uploads into `document_root`
- o XSS - Cross Site Scripting hacks

Input Filter hook

```
PHP_MINIT_FUNCTION(my_input_filter)
{
    sapi_register_input_filter(my_sapi_input_filter);
    return SUCCESS;
}
```

For a complete example, see `README.input_filter` in the PHP 5 source distribution. For PHP4, you will have to patch your source with http://lerdorf.com/php/input_filter.txt



Nobody is going to use that hook! People don't seem to understand how to filter their input. We'll need to spoonfeed again.

API?

```
<?php
    $name = filter(POST, 'name'); / Default filter /
    $age = filter(POST, 'age', PFILT_INTEGER);
    $addr = filter(POST, 'addr', PFILT_TEXT, 'UTF-8');
?>
```

Config

```
input_filter.default = FILTER_TEXT
```

Strip or Escape?

```
abc;123{def}
abc 123 def
abc3B1237Bdef%7D
```

Large Development Teams

Make sure your infrastructure is solid before you start

- o 75 properties
- o 25 international portals
- o 15 Languages
- o Hundreds of millions of registered users
- o Literally billions of page views per day
- o Hundreds of engineers spread around the world

Merger of 24 Web Companies

Net Controls	Four11	Classic Games
ViaWeb	WebCal	Yoyodyne
Sportacy	Hyperparallel	Log-Me-On
Geocities	Encompass	Online Anywhere
Broadcast.com	MyQuest	Arthas.com
eGroups	Kimo	Sold.com
Launch Media	HotJobs	Inktomi
Overture	3721	Kelkoo

Make that 27 now with FareChase, MusicMatch and Flickr

- o Revision control - CVS, SVN, Perforce, Bitkeeper
- o Mailing Lists
- o Twikis, Personal Pages
- o Bug tracking - Bugzilla
- o Regression Testing - phpt, phpunit

PHPT is a simple test framework for PHP.

hello.phpt

```
--TEST--
Hello World test
--FILE--
<?php
    echo "Hello World";
?>
--EXPECT--
Hello World
```

filter.phpt

```
--TEST--
Input Filter test
--SKIPIF--
if(!extension_loaded('input_filter')) print "skip";
--POST--
a=<b>1</b>
--GET--
b=<i>2</i>
--FILE--
<?php
    echo $_POST['a'];
    echo $_GET['b'];
?>
--EXPECT--
12
```

phpt output

```
TIME START 2003-10-15 10:19:50
=====
PASS Hello World test [hello.phpt]
PASS Input Filter test [filter.phpt]

=====
TIME END 2003-10-15 10:19:50
=====
TEST RESULT SUMMARY
-----
Number of tests :      2
Tests skipped   :      0 ( 0.0%)
Tests warned    :      0 ( 0.0%)
Tests failed    :      0 ( 0.0%)
Tests passed    :      2 (100.0%)
-----
Time taken      :      0 seconds
=====
```

phpt failed test output

```
TIME START 2003-10-15 10:32:48
=====
PASS Hello World test [hello.phpt]
FAIL Input Filter test [filter.phpt]

=====
TIME END 2003-10-15 10:32:48
```

```

=====
TEST RESULT SUMMARY
-----
Number of tests :      2
Tests skipped   :      0 ( 0.0%)
Tests warned   :      0 ( 0.0%)
Tests failed    :      1 (50.0%)
Tests passed    :      1 (50.0%)
-----
Time taken      :      0 seconds
=====

```

```

=====
FAILED TEST SUMMARY
-----
YIV test [filter.phpt]
=====
Some tests failed and a complete report has
been saved to /tmp/php_test_results_20031015.txt

```

failed test detailed output

```

=====
/home/rasmus/t/filter.phpt
=====
--TEST--
YIV test
--SKIPIF--
if(!extension_loaded('input_filter')) print "skip";
--POST--
a=<b>1</b>
--GET--
b=<i>2</i>
--FILE--
<?php
    echo $_POST['a'];
    echo $_GET['b'];
?>
--EXPECT--
1 2
=====
---- EXPECTED OUTPUT
1 2
---- ACTUAL OUTPUT
12
---- FAILED

```

PHPT Sections

--TEST--	title of the test
--SKIPIF--	php code which prints "skip"
--POST--	POST variables passed to test script
--GET--	GET variables passed to test script
--INI--	each line contains an ini setting e.g. foo=bar
--FILE--	the test script
--EXPECT--	the expected output from the test script
--EXPECTF--	sscanf version of expected output

--EXPECTREGEX--

regex version of expected output

High Complexity Applications?

Only if you don't build them right

Be Lazy!

The greatest inefficiencies come from solving problems you will never have.

Outside-in and Inside-out view

- o What do outsiders (users) see looking in?
- o What do insiders (developers) see looking out?

Outside-in

- o The URL - The API to your Web App
- o User interface flow

Inside-out

- o File Layout
- o Separation of layout from business logic
- o Application API
- o Server Architecture and Load Balancing

`http://www.example.com/a/b?c=1&d=2`

`/a/b` is what we are looking for

`?c=1&d=2` are optional modifiers

\$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.

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:

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

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

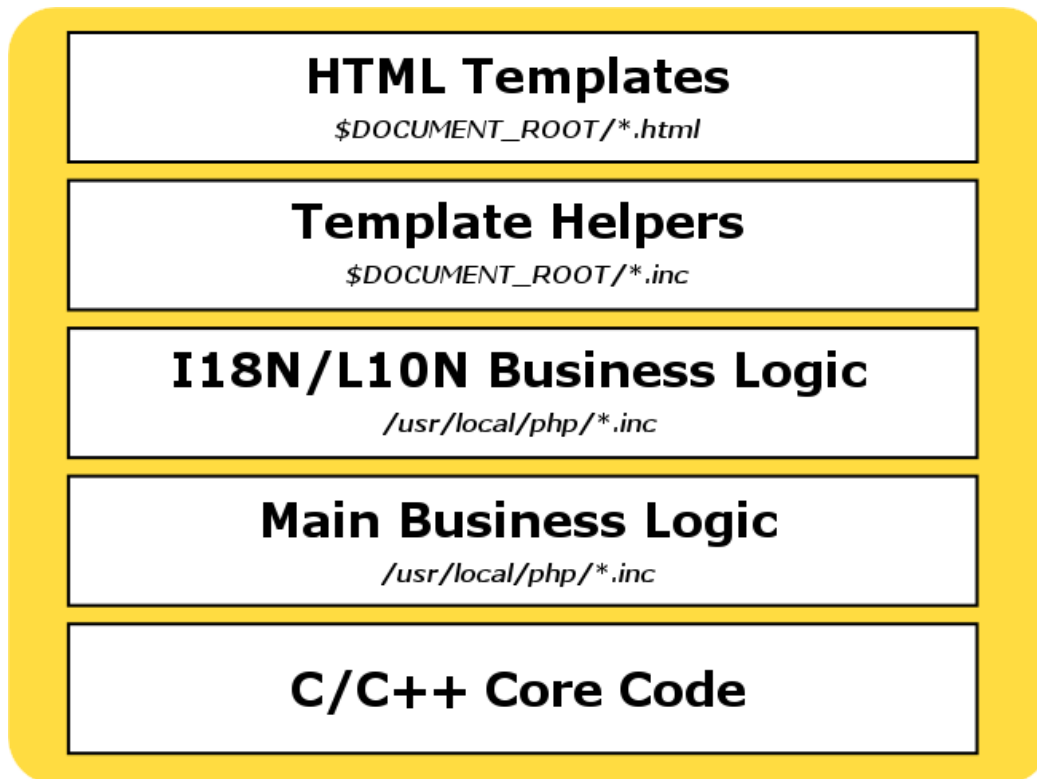
Keep it simple!

- o All files (.html/.php/.inc) with HTML under \$DOC_ROOT
- o All non-HTML files, somewhere else

No direct access to .inc

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

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.



For larger projects you want to be very careful when designing how the layers talk to each other. I like to provide a very clean API to the template layer and start each project by thinking about what the ideal template should look like.

Poll Example

Let's design a little application that allows people to answer a set of questions in various formats. We want to make it really easy for our content people to create new polls and we want to provide them with a flexible template API.

Poll Template

```
<?php
start_poll(1);
$os = array("FreeBSD", "Linux", "OSX", "Windows", "Other");
?>
<p class="purpose">
Please answer a couple of questions for us.
</p>

<p class="question">
1. What is your name?
</p>
<?php text_answer('name',64)?>

<p class="question">
2. Which operating systems do you use on a daily basis?
</p>
<?php select_any_of($os)?>

<p class="question">
3. Which operating system do you prefer?
</p>
<?php select_one_of($os)?>

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

In our template helper layer we implement our frontend API.

Helper Layer

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <base href="http://php.net" />
    <title>Questionnaire</title>
    <style type="text/css" media="screen">@import
"/poll/style.css";</style>
  </head>
<body>
<h3>Questionnaire</h3>
<?php

require 'logic.inc';

function start_poll($poll, $qn=1) {
  global $at_least_one_vote;

  init($poll, $qn);
  $at_least_one_vote = false;
  $self = getenv('REQUEST_URI');
  // consider adding a crumb here to prevent poll spoofing

echo <<<EOT
  <form action="$self" method="POST">
  <div class="main_box">
EOT;

}

function end_poll() {
  global $at_least_one_vote;

  if(!$at_least_one_vote) {
    echo '<div align="center"><input type="submit"
      value=" Submit Answers " /></div></form>';
  } else {
    echo '<div align="center"><input type="submit"
      value=" Change Answers " /></div></form>';
  }
  echo "</div\n";
}

function select_one_of($choices) {
  global $id, $qn, $at_least_one_vote;

  if(!is_array($choices))
    trigger_error("You must pass an array to select_one_of()",
E_USER_ERROR);

  // Grab the user's recorded answer to this question if any
  $answer = voted($id,$qn);
  $total_votes_on_this_question = total_votes($id,$qn);
  $i='a';
  echo '<table class="results">';
  foreach($choices as $c) {
    $checked = $answer?in_array($i,$answer):false;
    echo "<tr><td><input type=\"radio\" name=\"q[$qn][\]"
      value=\"\$i\" ".($checked?'CHECKED':'')." /> $c</td>";
    if($answer) {
      $at_least_one_vote = true;
    }
  }
}

```

```

        $percentage = 100*(($v=(int)results($id,$qn,$i))/
            $total_votes_on_this_question);
        $img = ($percentage) ?
            ('') :
            '';
        echo "<td>&nbsp;$v of $total_votes_on_this_question (" .
            sprintf("%.1f", $percentage).")</td><td>
$img</td></tr>\n";
    } else {
        echo '</tr>';
    }
    $i++;
}
echo "</table>\n";
$qn++;
}

function select_any_of($choices) {
    global $id, $qn, $at_least_one_vote;

    if(!is_array($choices))
        trigger_error("You must pass an array to select_any_of()",
            E_USER_ERROR);
    // Grab the user's recorded answer to this question if any
    $answer = voted($id,$qn);
    $total_votes_on_this_question = total_votes($id,$qn);
    $i='a';
    echo '<table class="results">';
    foreach($choices as $c) {
        $checked = $answer?in_array($i,$answer):false;
        echo "<tr><td><input type=\"checkbox\" name=\"q[$qn][\"
            value=\"\$i\" \" .($checked?'CHECKED':'')." /> $c</td>";
        if($answer) {
            $at_least_one_vote = true;
            $percentage = 100*(($v=(int)results($id,$qn,$i))/
                $total_votes_on_this_question);
            $img = ($percentage>0.0) ?
                ('') :
                '';
            echo "<td>&nbsp;$v of $total_votes_on_this_question (" .
                sprintf("%.1f", $percentage).")</td><td>
$img</td></tr>\n";
        } else {
            echo '</tr>';
        }
        $i++;
    }
    echo "</table>\n";
    $qn++;
}

function text_answer($field_name,$len) {
    global $id, $qn;

    $answer = get_text_answer($id,$qn);
    $size = (int)($len/2);
    echo <<<EOT
    <table class="results">
    <tr><td>
    <input type="textfield" size="$size" maxlength="$len"
name="t[$qn]" value="$answer" />
    </td></tr>
    </table>
    EOT;

    $qn++;
}??>

```


SQL Schema

```

CREATE TABLE poll_votes (
  user_name varchar(32) binary NOT NULL default '',
  poll_id smallint(6) NOT NULL default '0',
  question smallint(6) NOT NULL default '0',
  answer
set("","a","b","c","d","e","f","g","h","i","j","k","l","m","n","o",
    "p","q","r","s","t","u","v","w","x","y","z") NOT
NULL default "",
  text_answer varchar(255) binary NOT NULL default '',
  PRIMARY KEY (user_name,poll_id,question)
) TYPE=MyISAM;

CREATE TABLE poll_results (
  poll_id smallint(6) NOT NULL default '0',
  question smallint(6) NOT NULL default '0',
  answer char(1) NOT NULL default '',
  text_answer varchar(255) binary NOT NULL default '',
  votes int(11) NOT NULL default '0',
  INDEX (poll_id)
) TYPE=MyISAM;

```

Logic Layer

```

<?php

function db_connect() {
  mysql_pconnect("localhost","poll_user","password");
  mysql_select_db("polls");
}

function db_query($q) {
  $res = mysql_query($q);
  if(!$res) {
    trigger_error("db_query($q): ".mysql_error(), E_USER_ERROR);
  }
  return $res;
}

function voted($poll, $question) {
  global $user;

  $res = db_query("select answer from poll_votes
                  where poll_id=$poll
                  and user_name='$user'
                  and question=$question");

  return mysql_num_rows($res) ? mysql_result($res,0):false;
}

function total_votes($poll, $question) {
  $res = db_query("select sum(votes) as total from poll_results
                  where poll_id=$poll and question=$question");

  return mysql_num_rows($res) ? mysql_result($res,0):false;
}

function results($poll, $question, $answer) {
  $res = db_query("select votes from poll_results
                  where poll_id=$poll
                  and question=$question
                  and answer=$answer");

  return mysql_num_rows($res) ? mysql_result($res,0):false;
}

```

```

}

function init($poll, $start_qn = 1) {
    $GLOBALS['user'] = $user = $_SERVER['PHP_AUTH_USER'];
    $GLOBALS['qn']    = $start_qn;
    $GLOBALS['id']    = $poll;

    db_connect();

    if(getenv('REQUEST_METHOD')=="POST") {
        // check bread crumb here
        foreach($_POST['q'] as $qn=>$answer) {
            if(($oanswer=voted($poll,$qn))!==false) {
                // if user already voted on this question, decrement old
vote
                $where = "where poll_id=$poll and question=$qn and
answer=$oanswer";
                db_query("update poll_results set votes=votes-1
$where");
            }

            // MySQL 4.1 specific query
            db_query("insert into poll_results
                values ($poll,$qn,$answer,1)
                on duplicate key update set votes=votes+1");

            // record user's [new] answer in poll_votes
            db_query("replace into poll_votes values
('$user',$poll,$qn,$answer)");
        }
    }
}
?>

```

Agenda

High Traffic Applications
Profile, Optimize and Scale it

Some simple guidelines

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

./configure

```
./configure --with-apxs=/usr/sbin/apxs --without-pic \  
--with-config-file-scan-dir=/etc/php
```

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 need it.

variables_order

```
variables_order = "GPC"\  
  
<?php\  
    echo $_SERVER['DOCUMENT_ROOT'];\  
    echo getenv('DOCUMENT_ROOT');\  
?>
```

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-0.4pl.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-0.4pl
running: /tmp/tmpRfLAqf/apd-0.4pl/configure
running: make
apd.so copied to /tmp/tmpRfLAqf/apd-0.4pl/apd.so
install ok: apd 0.4pl
```

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 pprof:

```
www: ~> pprof
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.

% pprof -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					
cumm								
%Time	(excl/cumm)	(excl/cumm)	(excl/cumm)	Calls				
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					
cumm								
%Time	(excl/cumm)	(excl/cumm)	(excl/cumm)	Calls				
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	strchr					
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					

KCacheGrind Output

```

www:~> cd /tmp
www:/tmp> ls -lat pprof.*
-rw-r--r--  1 nobody 4294967295  62771 Apr 17 07:36
pprof.02466.0
-rw-r--r--  1 nobody 4294967295 269311 Apr 17 07:36
pprof.02465.0
-rw-r--r--  1 nobody 4294967295  62779 Apr 17 07:34
pprof.02464.0
www:/tmp> pprof2calltree -f pprof.02465.0
Writing kcachegrind compatible output to
cachegrind.out.pprof.02465.0
www:/tmp> kcachegrind cachegrind.out.pprof.02465.0

```

Function Summary Profile (sorted by total execution time)			
Total Time Taken	Avg. Time Taken	Number of Calls	Function Name
0.0012639508	0.0012639508	1	{include}
0.0009410545	0.0009410545	1	{include}
0.0003150582	0.0003150582	1	*hits
0.0002689241	0.0000179283	15	*url
0.0001970418	0.0000985209	2	mysql_query
0.0001859924	0.0001859924	1	mysql_pconnect
0.0001290126	0.0001290126	1	{include}
0.0001019851	0.0001019851	1	mysql_select_db
0.0000070533	0.0000070533	1	{include}
0.0000059746	0.0000059746	1	{include}
0.0000049799	0.0000049799	1	mysql_num_rows
Opcode Compiling		0.0709690896	
Function Execution		0.0025330913	
Ambient Code Execution		0.0031329459	
Total Execution		0.0056660372	
Total Processing		0.0766351268	

- o Basic profiling.
- o xdebug_start_profile()
- o xdebug_dump_function_profile(mode)

PECL_Gen replaces ext_skel in PHP5

```

<?xml version="1.0"?>
<extension name="myext">
  <summary>My Cool Extension</summary>
  <logo src="myext.png" mimetype="image/png" />

  <maintainers>
    <maintainer>
      <user>rasmus</user>
      <name>Rasmus Lerdorf</name>
      <email>rasmus@php.net</email>
      <role>lead</role>
    </maintainer>
  </maintainers>

  <release>
    <version>0.1</version>
    <date>2005-04-01</date>
    <state>alpha</state>
    <license>php</license>
    <notes>
      - This is cool
    </notes>
  </release>

  <deps language="c" platform="unix">
    <header name="math.h"/>
    <lib name="m" function="floor"/>
  </deps>

  <constant name="MYMODE" type="int" value="1" />
  <global name="foo" type="long" value="42"/>

  <function role="internal" name="MINIT">
    <code>init();</code>
  </function>

  <function name="get_foo">
    <proto>int get_foo(void)</proto>
    <code>
      RETURN_LONG(MYEXT_G(foo));
    </code>
  </function>

  <function name="foobar">
    <proto>string foobar(void)</proto>
    <summary>A dummy function</summary>
    <description>
      This function returns a static string.
    </description>
    <code>
      RETURN_STRING("foobar", 1);
    </code>
  </function>
</extension>

$ pecl-gen myext.xml

```

```

Creating 'myext' extension in '/home/rasmus/myext'
Your extension has been created in directory myext.
See myext/README for further instructions.

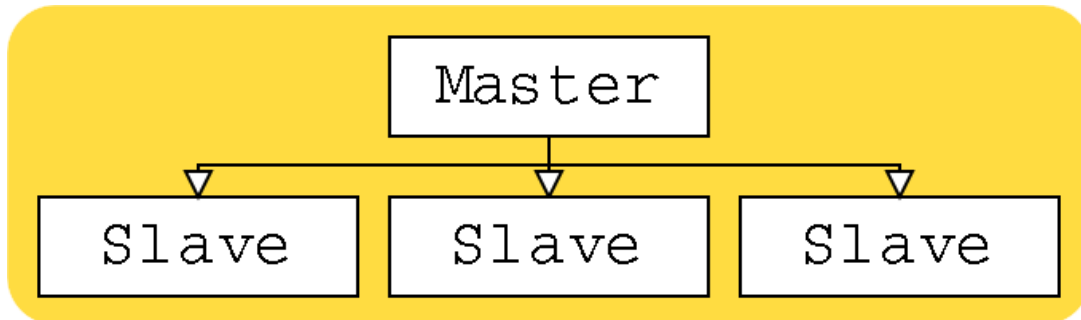
```

This will create the following directory:

```
myext
|-- EXPERIMENTAL
|-- README
|-- config.m4
|-- manual
|   |--myext
|       |-- configure.xml
|       |-- functions
|       |-- reference.xml
|-- package.xml
|-- php_myext.h
|-- tests
|-- myext.c
|-- myext.dsp
```

```
$ cd myext
$ phpize
$ ./configure
$ make
$ sudo make install
```

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:

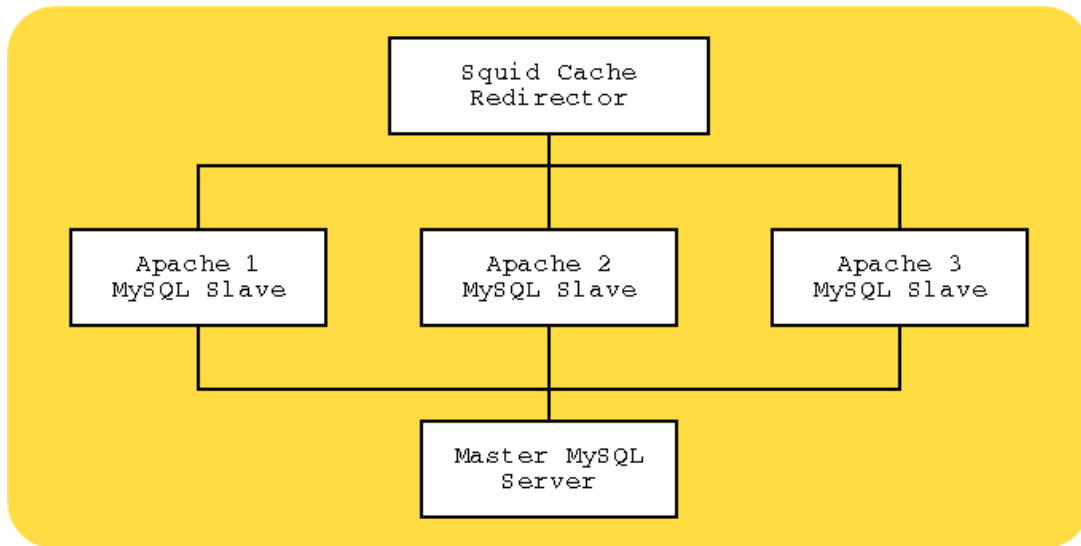
```
GRANT REPLICATION SLAVE ON . TO repl@"%" IDENTIFIED BY 'foobar';
```

```
[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
```

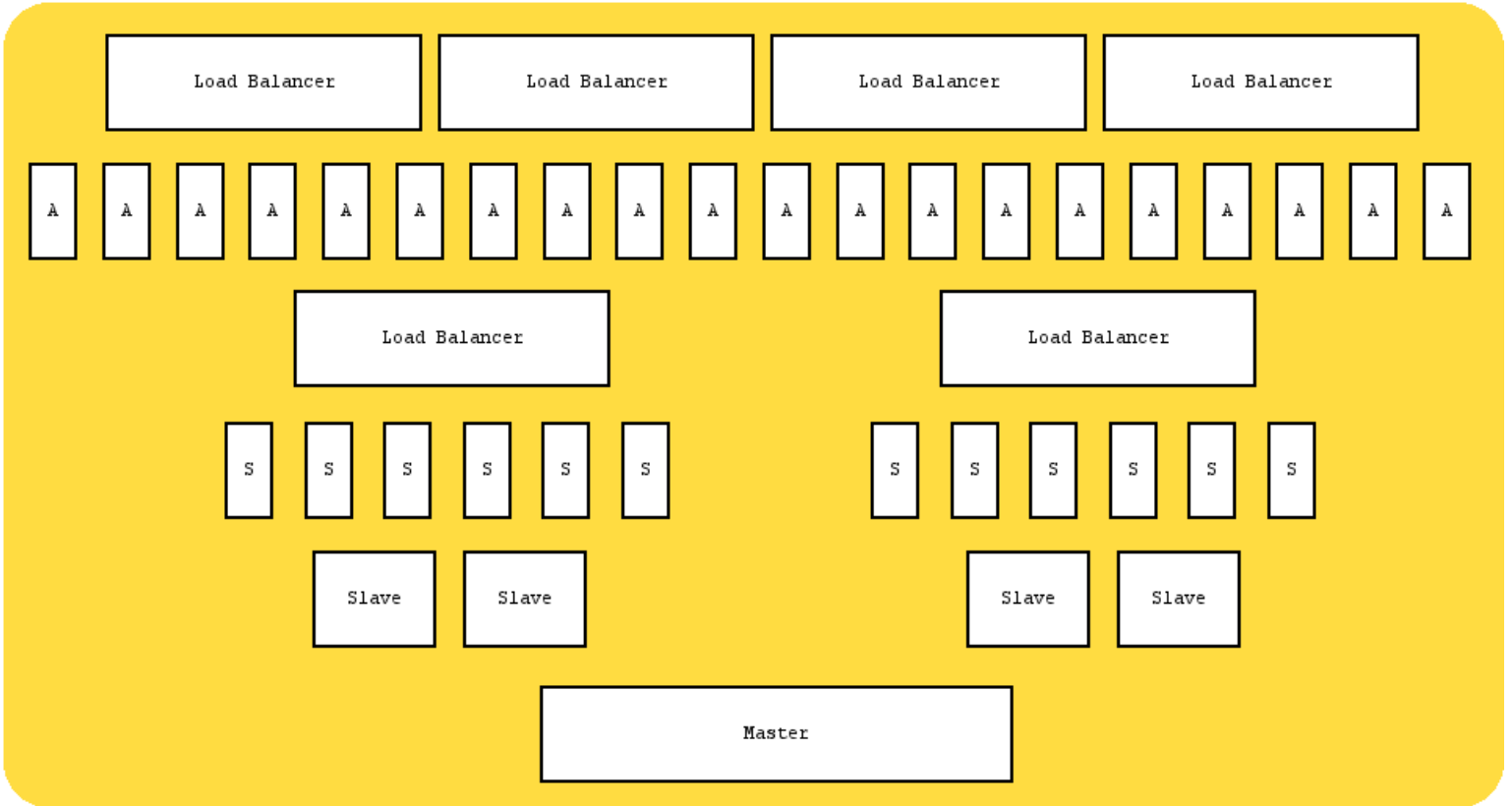
Make sure each slave has its own unique server-id. And since these will be read-only slaves, you can start them with these options to speed them up a bit:

```
--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.



I18n and L10n

Why can't we all just speak a single 7-bit ASCII Language anyway?

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

Runtime Templating

vs.

Offline Templating

PHP's strings are binary safe and mostly oblivious to character encodings and everything is fine if you just pass the data through. If you need to manipulate it you will need to do some work.

- o setlocale
- o utf8_decode
- o iconv
- o mbstring
- o preg_* functions support UTF-8
- o pecl/translit
- o Full ICU-based Unicode Support eventually

php.ini

```
default_charset = "utf-8"
```

There isn't much localization support directly in PHP. A few things can help out.

- o Generated templates
- o setlocale
- o gettext

PHP5

Some examples and a sneak preview

All XML handling based on libxml2

```
<?php
$dom = new domDocument;
$dom->load('data.xml');
?>
```

XSL

```
<?php
$domxsl = domDocument::load('stylesheet.xsl');
$proc = new xsltProcessor;
$proc->importStyleSheet($domxsl);
echo $proc->transformToXML($dom);
?>
```

XPath

```
<?php
$ctx = new domXPath($dom);
$result = $ctx->query(
    '/breakfast_menu/food[@itemno > 3]/price/text()');

foreach($result as $node) {
    echo $node->nodeValue."<br />\n";
}
?>
```

SimpleXML

```
<?php
$menu = simplexml_load_file('menu.xml');
foreach ($menu->food as $item) {
    echo "{$item->price}\t{$item->name}<br />\n";
}
?>
```

With PHP5's solid XML support and improved internal OOP support, Web Services are now a natural fit for PHP. Most people seem to think of SOAP when we say Web Services, so here is the obligatory SOAP example:

```
<?php
$amazon_index = array(
    'DVD', 'Photo', 'Electronics', 'OfficeProducts',
    'HealthPersonalCare',
    'Toys', 'Baby', 'VideoGames', 'MusicTracks', 'OutdoorLiving',
    'Blended', 'MusicalInstruments', 'Magazines', 'DigitalMusic',
    'Jewelry', 'Video', 'Tools', 'PCHardware', 'SportingGoods',
    'Classical', 'Software', 'Books', 'VHS', 'Wireless',
    'Restaurants',
    'Music', 'GourmetFood', 'Miscellaneous', 'Kitchen',
    'WirelessAccessories',
    'Merchants', 'Beauty', 'Apparel'
);

function amazon($index, $keywords, $timeout=7200) {
    $dest_file = "/tmp/aws_{$index}_".md5($keywords);
    if(file_exists($dest_file) && filemtime($dest_file) >
(time()-$timeout)) {
        $result = unserialize(file_get_contents($dest_file));
    } else {
        $aws = new SoapClient('http://webservices.amazon.com/'.

'AWSECommerceService/US/AWSECommerceService.wsdl',
        array("trace" => 1));
        $result = $aws->ItemSearch(array(
            'SubscriptionId'=>'XXXXXXXXXXXXXXXXX',
            'AssociateTag'=>'lerdorf-20',
            'Request'=>array(array('SearchIndex'=>$index,
                'Keywords'=>$keywords)
            )
        ));
        $tmpf = tempnam('/tmp','YWS');
        file_put_contents($tmpf, serialize($result));
        rename($tmpf, $dest_file);
    }
    return $result;
}
?>
```

```
<?php
function Add($x,$y) {
    return $x+$y;
}

$server = new SoapServer(null,array('uri'=>"http://test-uri/"));
$server->addFunction("Add");
$server->handle();
?>
```

Or using a WSDL:

```
<?php
function Add($x,$y) {
    return $x+$y;
}

$server = new SoapServer("./add.wsdl");
$server->addFunction("Add");
$server->handle();
?>
```

add.wsdl

```
<?xml version="1.0" ?>
<definitions
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd"
  xmlns:tns="http://localhost/~rasmus/pecl/soap/test.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://localhost/~rasmus/pecl/soap/test.wsdl">

  <types>
    <xsd:schema
targetNamespace="http://localhost/~rasmus/pecl/soap/test.wsdl">
      <xsd:import
namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
    </xsd:schema>
  </types>

  <message name="AddRequest">
    <part name="x" type="xsd:double" />
    <part name="y" type="xsd:double" />
  </message>
  <message name="AddResponse">
    <part name="result" type="xsd:double" />
  </message>

  <portType name="TestServicePortType">
    <operation name="Add">
      <input message="tns:AddRequest" />
      <output message="tns:AddResponse" />
    </operation>
  </portType>

  <binding name="TestServiceBinding"
type="tns:TestServicePortType">
    <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
```

```
<operation name="Add">
  <soap:operation soapAction="Add" style="rpc" />
  <input>
    <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </input>
  <output>
    <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </output>
</operation>
</binding>

<service name="TestService">
  <port name="TestServicePort"
binding="tns:TestServiceBinding">
    <soap:address
location="http://localhost/~rasmus/pecl/soap/soap_server.php" />
  </port>
</service>

</definitions>
```

SOAP makes my head hurt. Let's look at some stuff we can all actually understand instead.

Take a pinch of RSS

```
<?php
$url = 'http://buzz.yahoo.com/feeds/buzzoverl.xml';
$xml = simplexml_load_file($url);
foreach($xml->channel->item as $item) {
    $ret[(string)$item->title] = (string)$item->link;
}
echo "<pre>"; print_r($ret); echo "</pre>";
?>
```

Output:

```
Array
(
    [1. The Masters] =>
    http://search.yahoo.com/search?p=the+masters&cs=bz
    [2. Britney Spears] =>
    http://search.yahoo.com/search?p=britney+spears&cs=bz
    [3. 50 Cent] =>
    http://search.yahoo.com/search?p=50+cent&cs=bz
    [4. Internal Revenue Service] =>
    http://search.yahoo.com/search?p=internal+revenue+service&cs=bz
    [5. Green Day] =>
    http://search.yahoo.com/search?p=green+day&cs=bz
    [6. Jennifer Lopez] =>
    http://search.yahoo.com/search?p=jennifer+lopez&cs=bz
    [7. Ciara] => http://search.yahoo.com/search?p=ciara&cs=bz
    [8. NASCAR] => http://search.yahoo.com/search?p=nascar&cs=bz
    [9. Eminem] => http://search.yahoo.com/search?p=eminem&cs=bz
    [10. Jessica Simpson] =>
    http://search.yahoo.com/search?p=jessica+simpson&cs=bz
    [11. Usher] => http://search.yahoo.com/search?p=usher&cs=bz
    [12. Paris Hilton] =>
    http://search.yahoo.com/search?p=paris+hilton&cs=bz
    [13. Lindsay Lohan] =>
    http://search.yahoo.com/search?p=lindsay+lohan&cs=bz
    [14. Mariah Carey] =>
    http://search.yahoo.com/search?p=mariah+carey&cs=bz
    [15. Gwen Stefani] =>
    http://search.yahoo.com/search?p=gwen+stefani&cs=bz
)
```

Add a spoonful of REST

```
<?php
$srv =
'http://api.search.yahoo.com/ImageSearchService/V1/imageSearch';
foreach($ret as $key=>$link) {
    $url = $srv . "?query=$key&appid=RESTDemo";
    $obj = simplexml_load_file($url);
}
?>
```

A thimble of gradeschool math

$$x^2/a^2 + y^2/b^2 = 1$$

And we end up with something like this

<http://buzz.progphp.com>

By the way, throwing a cache layer between you and whatever remote service you are accessing tends to be a good idea. In our case we can do it like this:

```
<?php
$tmp = '/tmp/'.md5($q);
if(!file_exists($tmp) || filemtime($tmp) < (time()-7200)) {
    $stream = fopen($url,'r');
    $tmpf = tempnam('/tmp','YWS');
    file_put_contents($tmpf, $stream);
    fclose($stream);
    rename($tmpf, $tmp);
}
$obj = simplexml_load_file(file_get_contents($tmp));
?>
```

- o Web Search
- o Web Search with Context
- o Creative Commons Web Search
- o News Search
- o Image Search
- o Local Search
- o Video Search

And a new one that isn't public yet.

- o Context Extraction Service

Parameter	Value	Description
appid	string (required)	The application ID. See Application IDs for more information.
context	string (required)	The context to extract terms from (UTF-8 encoded).
query	string	An optional query to help with the extraction process.

Example

```
<?php
$context = "Italian sculptors and painters of the renaissance
favored the Virgin Mary for inspiration.";
$url =
'http://api.search.yahoo.com/ContentAnalysisService/V1/termExtraction';
$post =
"query=madonna&appid=RESTDemo&context=".rawurlencode($context);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
$xml = simplexml_load_string(curl_exec($ch));
curl_close($ch);
echo $xml->asXML();
?>
```

Output:

```
italian sculptorsvirgin marypaintersrenaissanceinspiration
```

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>

Talks: <http://talks.php.net>

Xdebug: <http://xdebug.org>

Index

Large-Scale PHP	2
Agenda	3
Worries	5
Direct Attacks	6
Direct Attacks	8
Input Filtering - Current	9
Input Filtering - Future	10
Agenda	11
What is Large?	12
Development Tools	13
PHPT	14
Agenda	17
Maximize Dev Resources	18
Design	19
URL API	20
\$PATH_INFO	21
ErrorDocument	22
File Layout	23
App Architecture	24
Template API	25
Template Helpers	26
Application Logic	29
Agenda	31
Tuning	32
Tuning	33
Profiling PHP	34
Profiling with XDebug	37
PECL_Gen	38
MySQL Replication	40
Bigger?	42
Agenda	43
Template System	44
Templating	45
i18n	46
l10n	47
Agenda	48
PHP5 XML	49
Web Services	50
SOAP Server	51
No more SOAP!	53
REST services from Yahoo!	55
Resources	56