A Tale of Three Templates

Lead by NYPHP members Chris Snyder, Daniel Kushner and Hans Zaunere, see how the different templating solutions compare. Chris looks at the powerful Smarty engine, Daniel examines the pure PHPLIB implementation, and Hans maintains that PHP is a templating engine itself! The panel will examine three identical tasks, and compare each solution for performance, maintainability and development lifecycle.

And now, thanks to Andrew Yochum, a fourth method for templating, PHPTAL is available.

Basic Tasks

- Process posted vars and display a form.
- Display the results of a MySQL query.
- Display a directory listing with image thumbnails and human-readable filesizes.

Topics

Chris

Why use a template engine?
What is Smarty and what does it do?
Task 1 Task 2 Task 3

Daniel

What is PHPLib and what does it do?
How does the PHPLib approach differ from Smarty's?
Three tasks

Andrew

What is PHPTAL and what does it do?
Task 1 | Task 2 | Task 3

Hans

Why not to use a template engine? (or: why PHP is a template engine)
Task 1 | Task 2 | Task 3

Display and/or process a form

The goal here is to use PHP to build a form with either default or user-submitted values.

Comment:

> Default comment.

The form:

comment	
option	✓ use option if checked.
	submit

Source of this script

```
<html>
<head>
   <title>Task 1: Display and/or process a form</title>
</head>

<body>

<h2>Display and/or process a form</h2>

<p>The goal here is to use PHP to build a form with either default or user-submitted values.</p>

<?php

// TASK: display and/or process a form
$comment= 'Default comment.';
$option= 0;

// process
if ( isset ( $_REQUEST['submit'] ) ) {
    $comment= $_REQUEST['comment'];
    $option= $_REQUEST['option'];
}

// display comment
print "<em>Comment:</em><blockquote>".htmlentities($comment)."</blockquote>";

// display option
if ($option==1) {
    print "<p>Option was selected</p>";
}

?>

<form action="" method="post">
   <h3>The form:</h3>
   <table>
      <tr>
         <td>comment</td>
         <td><input type="text" name="comment" size="56" /></td>
      </tr>
      <tr>
         <td>option</td>
         <td><input type="checkbox" name="option" value="1" /> use option if checked.</td>
      </tr>
      <tr>
         <td>&nbsp;</td>
         <td><input type="submit" name="submit" value="submit" /></td>
      </tr>
   </table>
</form>
```

```
<hr width="100%">

<h2>Source of this script</h2>

<?php

   $output= highlight_file($_SERVER['SCRIPT_FILENAME'],1);
   print $output;

?>

</body>
</html>
```

Display the results of a mysql query

The goal here is to use PHP to parse and display the results of a database query.

And here are the results, unprocessed and in a simple bulleted list:

- **Verisign is damage: route around it**

  Yesterday, Verisign (the company I'd like to see put to death) broke the Internet by redirecting all unregistered .COM and .NET addresses to a page on their site where they run a search-engine. For a lot of good technical reasons, this is a bad idea, and it makes a savage mockery of Verisign's (unbelievably lucrative) monopoly on critical pieces of the Internet's infrastructure.

  Today, the makers of the BIND DNS software responded by announcing a patch that will interpret Verisign as damage and route around them.

  INSTEAD OF GETTING "SITE NOT FOUND" MESSAGES IF YOU MAKE A TYPO OR IF THE SITE DOESN'T EXIST, YOU GET SENT TO A VERISIGN PAGE. AND E-MAILS THAT SHOULD BOUNCE ARE LIKELY HARVESTED BY VERISIGN FOR THEIR DATA.

  However, the ISC is about to undercut the Site Finder service with a patch to its BIND software.

  BIND runs on about 80 percent of the Internet's domain name servers -- the machines that translate human-readable Web addresses like www.wired.com into machine-readable Internet addresses used by the Internet's vast network of computers.

  The patch will be released by the end of Tuesday, said Paul Vixie, ISC's president.

  Link *(2003-09-17 04:41:42)*

- **NYT cartoon: The Copyright Cops**

  Hilarious and instructive cartoon in today's *New York Times* about copyright crackdowns and the RIAA lawsuits, with guest cameos by the EFF's Fred Von Lohmann and the RIAA's Amy Weiss. Link *(registration required) (2003-09-17 05:16:40)*

- **Virtual Museum of Bacteria**

  The subject line says it all, folks. An online tribute to the glory that is, um, bacteria. Link *(via Viridian list) (2003-09-16 15:09:20)*

- **Translate gangsta to pirate**

  Nice Gangsta-Pirate translation table:

Gangstah	**Pirate**
fo'ties	bottles o' rum
bling bling	booty
Yo!	Avast!
Homey	Matey
Bee-atch	Scurvey dog

  Link (*via Making Light*) *(2003-09-17 03:18:31)*

Attribution

Content by Cory Doctorow and Xeni Jardin of BoingBoing.net, subject to Creative Commons attribution/non-commercial license.

SQL Data Available: task2-sql.txt

Source of this script

```
<html>
<head>
   <title>Task 2: Display the results of a mysql query</title>
</head>

<body>

<h2>Display the results of a mysql query</h2>

<p>The goal here is to use PHP to parse and display the results of a database query.</p>
<?php

// TASK: display results of a MySQL query

// get database connection info
include('task2-config.php');

/* Dataset:  --- with apologies to boingboing.net see task2-sql.txt */

$db= @mysql_connect(MYDBHOST, MYDBUSERNAME, MYDBPASSWORD);
if (!$db) {
    exit("Database error, could not connect to server.");
}

if (!$database= @mysql_select_db(MYDBNAME)) {
    exit("Database error, could not connect to database.");
}

?>

<p>And here are the results, unprocessed and in a simple bulleted list:</p>

<ul>

<?php

$query = "SELECT * FROM presentations.3templates ";
$result = mysql_query($query);

if ($result) {

    while ($array= mysql_fetch_assoc($result)) {
        print "<li><h4>$array[title]</h4>$array[content] <i>($array[created])</i></li>\n";
    }

} else {

    print "<li>No results.</li>";

}

?>

</ul>

<p>&nbsp;</p>

<h3>Attribution</h3>

<p>Content by Cory Doctorow and Xeni Jardin of <a href="http://boingboing.net">BoingBoing.net</a>, subject to Creative Commons <a href="http://creativecommons.org/licenses/by-
nc/1.0">attribution/non-commercial license</a>.</p>
<p>SQL Data Available: <a href="task2-sql.txt">task2-sql.txt</a></p>


<hr width="100%">

<h2>Source of this script</h2>

<?php

   $output= highlight_file($_SERVER['SCRIPT_FILENAME'],1);
   print $output;

?>

</body>
</html>
```

Display a directory listing

This is an example of how one might display a list of files in the local directory, complete with image thumbnails and human-readable filesizes.

Files in /var/www/www.nyphp.org/content/presentations/3templates:

- task1-plain.php (1KB)
- phptal (4KB)
- task3-plain.php (2KB)
- task2-config.php (177B)
- index.php (4KB)
- whyyes.php (4KB)
- whynot (4KB)
- phplib (4KB)
- task2-plain.php (1KB)
- ramp.logo.gif (1KB)
- nyphp.powered.logo.gif (4KB)
- smarty (4KB)
- task2-sql.txt (3KB)

Source of this script

```
<html>
<head>
   <title>Task 3: Display a directory listing</title>
</head>

<body>

<h2>Display a directory listing</h2>

<p>This is an example of how one might display a list of files in the local directory, complete with image thumbnails and human-
readable filesizes.</p>

<?php

// TASK: display a directory listing with thumbnails for images and human-readable filesizes

// handy humansize function:
// input is number of bytes, output is a "human-readable" filesize string
function humansize($size) {

        // Setup some common file size measurements.
    $kb = 1024;         // Kilobyte
    $mb = 1024 * $kb;   // Megabyte
    $gb = 1024 * $mb;   // Gigabyte
    $tb = 1024 * $gb;   // Terabyte

        if($size < $kb) return $size."B";
    else if($size < $mb) return round($size/$kb,0)."KB";
    else if($size < $gb) return round($size/$mb,0)."MB";
    else if($size < $tb) return round($size/$gb,0)."GB";
    else return round($size/$tb,2)."TB";
}

// get local directory path
$path= dirname($_SERVER['SCRIPT_FILENAME']);

?>

<h3>Files in <?php print $path; ?>:</h3>

<ul>

<?php

$d = dir($path);
$icon = '';

while (false !== ($entry = $d->read())) {

    if ( substr($entry, 0, 1)=='.' ) continue;

    // get size
    $size = filesize($path.'/'.$entry);
    $humansize = humansize($size);

    // find filename extension
    $dotpos = strrpos($entry, '.');

    if ($dotpos) {
        $ext = substr($entry, $dotpos+1);
        if ($ext === 'jpeg' || $ext === 'gif' || $ext === 'png') {
            $icon = "<img src='$entry' style='width: 48px; height: auto; vertical-align: text-
top;' alt='icon' title='$entry' />";
        }
    }
    print "<li><a href='$entry'>$entry</a> ($humansize) $icon</li>\n";

    $icon= '';
}

$d->close();

?>

</ul>


<hr width="100%">

<h2>Source of this script</h2>

<?php

   $output= highlight_file($_SERVER['SCRIPT_FILENAME'],1);
   print $output;

?>
```

Why Use Templates?

A reasonable question, but the answer is usually something like "Just try it."

Every project is different. The most compelling application of templates is on a project where the programmer is not the same person as the interface designer.

But let's think that through, from the standpoint of intelligent laziness.

- The programmer wants to use templates because that way the interface designer can make all the changes she wants, without the programmer having to touch the code.
- The interface designer wants to use templates because then she doesn't need to rely on the programmer, or learn PHP, in order to get the site looking the way she wants.
- When the boss wants a new look to the site in six months, the programmer will merely have to sit back and look busy while the interface designer creates a new set of templates (or maybe just a newstylesheet if she used XHTML in the first place).
- In essence
  - Display tweaks don't involve code hacks.
  - The designer doesn't need PHP expertise; the programmer doesn't need design expertise.
  - The same code can be used under multiple interfaces.

But what if you work on a small project, where you are both the programmer and designer?

The fact is that you are rarely ever programmer and designer at the same time for the life of a project. By using templates, you make your separate lives easier.

And consider the third fundamental reason to separate application logic from display: code reuse. If the underlying code is truly brilliant, and you know it will be, you'll want to use it again and again on different sites -- either for different clients, or over time. Humans are much more fickle about how things look than they are about how things work-- in fact, they rarely get tired of something that works correctly, but they often want to freshen or customize its look.

If you are selling the same code to multiple clients, and you are intelligently lazy, your ultimate goal is to convince your clients to pay you for code that you've already written, and have their in-house art department do the design and build the templates.

How nice it would be to say, "This code uses a robust, well-documented template language. Here is a website with everything you need to know to build your templates."

This brings us to Smarty.

Introducting Smarty Templates

What Is Smarty?

Smarty is a is PHP template engine with an extensible architecture, built-in template compiling, and optional output caching.

The project was started in late 2000 by Monte Ohrt and Andrei Zmievski, and was the result of a failed attempt at writing an all-purpose template extension for PHP in C. They decided to start from scratch and rewrite it as a PHP class.

Recent development has been done by Monte and Messju Mohr, and involved adding object support, optimizing the compiler, and enhancing custom function support.

Code and notes in this presentation apply to version 2.6, at RC1 as of August 11, 2003.

So What Does It Look Like?

Here is a hello world example.

Three Tasks Shortcut:

1. Process posted vars and display a form.
2. Display the results of a MySQL query.
3. Display a directory listing with image thumbnails and human-readable filesizes.

Index | Next»

Questions? csnyder@chxo.com

Task 1: Process a form

Comment was:

> Default comment.

The form:

comment	Default comment.
option	☐ use option if checked.
	submit

Index | Next»

Questions? csnyder@chxo.com

Source of this script

```
<?php
// initialize template
include_once('libs/Smarty.class.php');
$template= new Smarty;
$template->assign('title', 'Task 1: Process a form');
$template->assign('next', 'task1-alt-smarty.php');


    // TASK: display and/or process a form
    $comment= 'Default comment.';
    $option= 0;

    // process
    if ( isset ( $_REQUEST['submit'] ) ) {
        $comment= $_REQUEST['comment'];
        $option= $_REQUEST['option'];
    }

// assign the values to the template
$template->assign('comment', $comment);
$template->assign('option', $option);

// display the template
$template->display('task1.tpl');
?>

<h3>Source of this script</h3>
<?php
$output= highlight_file($_SERVER['SCRIPT_FILENAME'],1);
print $output;
?>

<h3>Source of the template</h3>
<?php
$output= highlight_file(dirname($_SERVER['SCRIPT_FILENAME']).'/templates/task1.tpl',1);
print $output;
?>
```

Source of the template

```
{* Smarty: task 1 template *}
{include file="header.tpl"}

<h3>Comment was:</h3>
<blockquote>{$comment|escape}</blockquote>
{if $option eq 1}<p>Option was checked.</p>{/if}

<form action="" method="post">
<h3>The form:</h3>
<table class="edit">
    <tr>
        <td class="label">comment</td>
        <td><input type="text" name="comment" size="56" value="{$comment|escape}" style="padding: 3px;" /></td>
    </tr>
    <tr>
        <td class="label">option</td>
        <td><input type="checkbox" name="option" value="1" {if $option eq 1}checked="checked"{/if} /> use option if checked.</td>

    </tr>
    <tr>
        <td>&nbsp;</td>
        <td><input type="submit" name="submit" value="submit" /></td>
    </tr>
</table>
</form>

{include file="footer.tpl"}
```

Task 2: Results of a MySQL query

The Articles from the Database:

title	date
Verisign is damage: route around it	Wednesday, September 17, 2003 at 04:41

Yesterday, Verisign (the company I'd like to see put to death) broke the Internet by redirecting all unregistered .COM and .NET addresses to a page on their site where they run a search-engine. For a lot of good technical reasons, this is a bad idea, and it makes a savage mockery of Verisign's (unbelievably lucrative) monopoly on critical pieces of the Internet's infrastructure.

Today, the makers of the BIND DNS software responded by announcing a patch that will interpret Verisign as damage and route around them.

INSTEAD OF GETTING "SITE NOT FOUND" MESSAGES IF YOU MAKE A TYPO OR IF THE SITE DOESN'T EXIST, YOU GET SENT TO A VERISIGN PAGE. AND E-MAILS THAT SHOULD BOUNCE ARE LIKELY HARVESTED BY VERISIGN FOR THEIR DATA.

However, the ISC is about to undercut the Site Finder service with a patch to its BIND software.

BIND runs on about 80 percent of the Internet's domain name servers -- the machines that translate human-readable Web addresses like www.wired.com into machine-readable Internet addresses used by the Internet's vast network of computers.

The patch will be released by the end of Tuesday, said Paul Vixie, ISC's president.

Link

title	date
NYT cartoon: The Copyright Cops	Wednesday, September 17, 2003 at 05:16

Hilarious and instructive cartoon in today's *New York Times* about copyright crackdowns and the RIAA lawsuits, with guest cameos by the EFF's Fred Von Lohmann and the RIAA's Amy Weiss. Link *(registration required)*

title	date
Virtual Museum of Bacteria	Tuesday, September 16, 2003 at 15:09

The subject line says it all, folks. An online tribute to the glory that is, um, bacteria. Link *(via Viridian list)*

title	date
Translate gangsta to pirate	Wednesday, September 17, 2003 at 03:18

Nice Gangsta-Pirate translation table:

Gangstah	Pirate
fo'ties	bottles o' rum
bling bling	booty
Yo!	Avast!
Homey	Matey
Bee-atch	Scurvey dog

Link (*via Making Light*)

Attribution

Index | Next»

Questions? csnyder@chxo.com

Source of this script

```
<?php
// initialize template
include_once('libs/Smarty.class.php');
$template= new Smarty;
$template->assign('title', 'Task 2: Results of a MySQL query');
$template->assign('next', 'task3-smarty.php');

// TASK: display results of a MySQL query
// get database connection info
include('../task2-config.inc');

$db= @mysql_connect($ORGHOST,$ORGUSER,$ORGPASS);
if (!$db) {
    exit("Database error, could not connect to server.");
}
if (!$database= @mysql_select_db($PRESENTATION_DB)) {
    exit("Database error, could not connect to database.");
}

// query database and dump results into $articles array
$query= "SELECT * FROM presentations.3templates ";
$result= mysql_query($query);
$articles= array();
if ($result) {
    while ($array= mysql_fetch_assoc($result)) {
        $articles[]= $array;
    }
}

// assign the articles array to the template
$template->assign('articles', $articles);

// display the template
$template->display('task2.tpl');
?>

<h3>Source of this script</h3>
<?php
$output= highlight_file($_SERVER['SCRIPT_FILENAME'],1);
print $output;
?>

<h3>Source of the template</h3>
<?php
$output= highlight_file(dirname($_SERVER['SCRIPT_FILENAME']).'/templates/task2.tpl',1);
print $output;
?>
```

Source of the template

```
{* Smarty: task 2 template *}
{include file="header.tpl"}

<h3>The Articles from the Database:</h3>
<table style="width: 600px;">
    <tr class="header">
        <td>title</td><td>date</td>
    </tr>
{section name="a" loop=$articles}
    <tr class="{cycle values="odd,even" advance=0}">
        <td><h4>{$articles[a].title|escape}</h4></td>
        <td>{$articles[a].created|date_format:"%A, %B %e, %Y at %H:%M"}</td>
    </tr>
    <tr class="{cycle values="odd,even" advance=1}">
        <td colspan="2">{$articles[a].content}</td>
    </tr>
{sectionelse}
    <tr class="odd">
        <td colspan="2">There are no articles yet.</td>
    </tr>
{/section}
    <tr class="header">
        <td colspan="2">&nbsp;</td>
    </tr>
</table>

<h3>Attribution</h3>
<p>Content by Cory Doctorow and Xeni Jardin of <a href="http://boingboing.net">BoingBoing.net</a>, subject to Creative Commons <a href="http://creativecommons.org/licenses/by-
nc/1.0">attribution/non-commercial license</a>.</p>

{include file="footer.tpl"}
```

Task 3: Print a directory listing

Contents of /var/www/...:

icon	name	size
	index.php	4KB
	nyphp.powered.logo.gif	4KB
	phplib	dir
	phptal	dir
	ramp.logo.gif	1KB
	smarty	dir
	task1-plain.php	1KB
	task2-config.php	177B
	task2-plain.php	1KB
	task2-sql.txt	3KB
	task3-plain.php	2KB
	whynot	dir
	whyyes.php	4KB

Index | Next»

Questions? csnyder@chxo.com

Source of this script

```
<?php
// initialize template
include_once('libs/Smarty.class.php');
$template= new Smarty;
$template->assign('title', 'Task 3: Print a directory listing');
$template->assign('next', 'index.php');

// TASK: display a directory listing with thumbnails for images and human-readable filesizes

// handy humansize function:
// input is number of bytes, output is a "human-readable" filesize string
function humansize($size) {
    // only take numbers
    if ( !is_numeric($size) ) return $size;

        // Setup some common file size measurements.
    $kb = 1024;         // Kilobyte
    $mb = 1024 * $kb;   // Megabyte
    $gb = 1024 * $mb;   // Gigabyte
    $tb = 1024 * $gb;   // Terabyte

        if($size < $kb) return $size."B";
    else if($size < $mb) return round($size/$kb,0)."KB";
    else if($size < $gb) return round($size/$mb,0)."MB";
    else if($size < $tb) return round($size/$gb,0)."GB";
    else return round($size/$tb,2)."TB";
}
// register humansize as a custom modifier
$template->register_modifier('humansize','humansize');

// get local directory path
$path= dirname(dirname($_SERVER['SCRIPT_FILENAME']));
$template->assign('path', $path);

// get the directory contents into $files array
$d = dir($path);
$files= array();
$index= 0;
while (false !== ($entry = $d->read())) {
    if ( substr($entry, 0, 1)=='.' ) continue;
    $files[]= $entry;
}
$d->close();

// natsort the entries
usort($files, 'strnatcmp');

// get sizes and extensions
$extensions= array();
$sizes= array();
foreach( $files AS $entry ) {
    // get size
    if ( is_dir($path.'/'.$entry) ) {
        $sizes[]= 'dir';
    }
    else {
        $sizes[]= filesize($path.'/'.$entry);
    }

    // find filename extension
    $dotpos= strrpos($entry, '.');
    if ($dotpos) {
        $extensions[]= substr($entry, $dotpos+1);
    }
    else {
        $extensions[]= '';
    }
}

// assign the files, extensions, and sizes arrays to the template
$template->assign('files', $files);
$template->assign('extensions', $extensions);
$template->assign('sizes', $sizes);

// display the template
$template->display('task3.tpl');
?>

<h3>Source of this script</h3>
<?php
$output= highlight_file($_SERVER['SCRIPT_FILENAME'],1);
print $output;
?>

<h3>Source of the template</h3>
<?php
$output= highlight_file(dirname($_SERVER['SCRIPT_FILENAME']).'/templates/task3.tpl',1);
print $output;
?>
```

Source of the template

```
{* Smarty: task 3 template *}
{include file="header.tpl"}

<h3>Contents of /var/www/...:</h3>
<table>
    <tr class="header">
        <td>icon</td><td>name</td><td>size &nbsp;</td>
    </tr>
{section name="f" loop=$files}
    <tr class="{cycle values="odd,even"}">
        <td>{if $extensions[f] eq "gif" or $extensions[f] eq "gif" or $extensions[f] eq "gif"}<img src="../{$files[f]}" alt="thumbnail" style="width: 48px; height: auto;" />{else}&nbsp;{/if}</td>

        <td><a href="../{$files[f]}">{$files[f]}</a></td>
        <td>{$sizes[f]|humansize}</td>
    </tr>
{sectionelse}
    <tr class="odd">
        <td colspan="3">There are no files here.</td>
    </tr>
{/section}
</table>

{include file="footer.tpl"}
```

Index Task 1 Task 2 Task 3

Why PHP is a template engine?

(or: why an additional engine is superfluous)

- PHP was originally developed to address the need for dynamic content generation and management. This basic syntax and architecture remains.
  - Short open tag **<?=** dynamically process values in text.
  - Alternative control structure syntax (using the colon).
  - The requirement to have opening/closing tags, thus making the default behavior to pass through text while parsing instructions within it (ie, a template!).
- Web pages have become so complex that a certain level of logic is required within HTML, which is underscored by the fact that the external templating engines have this functionality.
  - Templating engines were created to reduce the bad practice of putting business logic, database queries, and other complex operations, directly inline with HTML.
  - Proper page architecture, and the separation of **presentation logic** from **complex logic** make external engines superfluous.
- "Designers don't have to learn PHP" is a myth. With the complexity of web pages, to the point of being applications themselves, **interface designers** need to have some notion of program operation and flow.
  - External template engines have their own syntax that needs to be learned by interface designers. Preparing a small library of native PHP functions for your project's front-end needs, and then teaching designers this basic syntax, is no different from learning an external engine's syntax.
  - Complex web application development will always require someone "in the middle" to integrate front and back

technologies. Even the premier commercial development platforms require someone to mesh all the piece.

- External template engines are superfluous and add their own layer of complexity, performance overhead, and development overhead.
  - Why learn **another** syntax that has it's own quirks and limitations?
  - Processing and parsing templates require additional resources.
  - Maintaining another set of files, for instance .tpl, requires additional development and tracking.
  - An external engine is an additional set of software to patch, maintain and track bugs and changes in.
- Templating is good; PHP has known that from it's inception.
  - Templating is easily accomplished without an external engine.
  - Templating is a development style and architectural decision; not an additional piece of software.

Task 1: Process a form

Comment was:

> Default comment.

The form:

comment	Default comment.
option	☑ use option if checked.
	submit

Source of this script

```
<?php

require_once('php-templates.inc');

if( empty($_POST['comment']) )
   $_POST['comment'] = 'Default comment.';

?>

<?php htmlheader('Task 1: Process a form'); ?>

<div style="margin-bottom: 35px; text-align: center;">
   <a href="..">Index</a>
   <a href="task1.php">Task 1</a>
   <a href="task2.php">Task 2</a>
   <a href="task3.php">Task 3</a>
</div>

<h2>Task 1: Process a form</h2>

<form action="<?=$_SERVER['PHP_SELF']?>" method="post">
<div id="demo">

   <h3>Comment was:</h3>
   <blockquote><?=htmlentities($_POST['comment'])?></blockquote>

   <h3>The form:</h3>

   <table class="edit">
       <tr>
           <td class="label">comment</td>
           <td><input type="text" name="comment" size="56" value="<?
=htmlentities($_POST['comment'])?>" style="padding: 3px;" /></td>
       </tr>
       <tr>
           <td class="label">option</td>
           <td><input type="checkbox" name="option" <?
=is_value(@$_POST['option'],'iamchecked')?
> value="iamchecked" /> use option if checked.</td>
       </tr>
       <tr>
           <td>&nbsp;</td>
           <td><input type="submit" name="submit" value="submit" /></td>
       </tr>
```

```
      </table>

   </div>
   </form>

   <h3>Source of this script</h3>
   <?php highlight_file($_SERVER['SCRIPT_FILENAME']); ?>

   </body>
   </html>
```

Task 2: Results of a MySQL query

The Articles from the Database:

title	date
Verisign is damage: route around it	2003-09-17 04:41:42

Yesterday, Verisign (the company I'd like to see put to death) broke the Internet by redirecting all unregistered .COM and .NET addresses to a page on their site where they run a search-engine. For a lot of good technical reasons, this is a bad idea, and it makes a savage mockery of Verisign's (unbelievably lucrative) monopoly on critical pieces of the Internet's infrastructure.

Today, the makers of the BIND DNS software responded by announcing a patch that will interpret Verisign as damage and route around them.

INSTEAD OF GETTING "SITE NOT FOUND" MESSAGES IF YOU MAKE A TYPO OR IF THE SITE DOESN'T EXIST, YOU GET SENT TO A VERISIGN PAGE. AND E-MAILS THAT SHOULD BOUNCE ARE LIKELY HARVESTED BY VERISIGN FOR THEIR DATA.

However, the ISC is about to undercut the Site Finder service with a patch to its BIND software.

BIND runs on about 80 percent of the Internet's domain name servers -- the machines that translate human-readable Web addresses like www.wired.com into machine-readable Internet addresses used by the Internet's vast network of computers.

The patch will be released by the end of Tuesday, said Paul Vixie, ISC's president.

Link

title	date
NYT cartoon: The Copyright Cops	2003-09-17 05:16:40

Hilarious and instructive cartoon in today's *New York Times* about copyright crackdowns and the RIAA lawsuits, with guest cameos by the EFF's Fred Von Lohmann and the RIAA's Amy Weiss. Link *(registration required)*

title	date
Virtual Museum of Bacteria	2003-09-16 15:09:20

The subject line says it all, folks. An online tribute to the glory that is, um, bacteria. Link *(via Viridian list)*

title	date
Translate gangsta to pirate	2003-09-17 03:18:31

Nice Gangsta-Pirate translation table:

Gangstah	Pirate
fo'ties	bottles o' rum
bling bling	booty
Yo!	Avast!
Homey	Matey
Bee-atch	Scurvey dog

Link (*via Making Light*)

Attribution

Content by Cory Doctorow and Xeni Jardin of BoingBoing.net, subject to Creative Commons attribution/non-commercial license.

Source of this script

```
<?php

require_once('task2.inc');

$data = fetchdata();

?>

<?php htmlheader('Task 2: Results of a MySQL query'); ?>

<div style="margin-bottom: 35px; text-align: center;">
   <a href="..">Index</a>
   <a href="task1.php">Task 1</a>
   <a href="task2.php">Task 2</a>
   <a href="task3.php">Task 3</a>
</div>

<h2>Task 2: Results of a MySQL query</h2>

<div id="demo">

<h3>The Articles from the Database:</h3>

<table style="width: 600px;">
    <tr class="header">
        <td>title</td>
        <td>date</td>
    </tr>

  <?php if( !$data ): ?>

   <tr class="header">
      <td>No Results!</td>
   </tr>

  <?php else: ?>

     <?php foreach( $data as $key => $story ): ?>

       <tr <?=hiLine($key)?> >
           <td><h4><?=$story['title']?></h4></td>
           <td><?=$story['created']?></td>
       </tr>

       <tr <?=hiLine($key)?> >
         <td colspan="2"><?=$story['content']?></td>
      </tr>
     <?php endforeach; ?>

   <?php endif; ?>

</table>

<h3>Attribution</h3>
<p>Content by Cory Doctorow and Xeni Jardin of <a href="http://boingboing.net">BoingBoing.net</a>, subject to Creative Commons <a href="http://creativecommons.org/licenses/by-
nc/1.0">attribution/non-commercial license</a>.</p>

</div>

<h3>Source of this script</h3>
<?php highlight_file($_SERVER['SCRIPT_FILENAME']); ?>

</body>
</html>
```

Index Task 1 Task 2 Task 3

Task 3: Print a directory listing

Contents of /var/www/...:

icon	name	size
	task1-plain.php	1KB
	.svn	0B
	phptal	0B
	task3-plain.php	2KB
	task2-config.php	177B
	index.php	4KB
	whyyes.php	4KB
	whynot	0B
	phplib	0B
	task2-plain.php	1KB
	ramp.logo.gif	1KB
	nyphp.powered.logo.gif	4KB
	smarty	0B
	task2-sql.txt	3KB

Source of this script

```
<?php

require_once('task3.inc');

list($realpath,$listing) = fetchlisting('..');

?>

<?php htmlheader('Task 3: Print a directory listing'); ?>

<div style="margin-bottom: 35px; text-align: center;">
   <a href="..">Index</a>
   <a href="task1.php">Task 1</a>
   <a href="task2.php">Task 2</a>
   <a href="task3.php">Task 3</a>
</div>

<h2>Task 3: Print a directory listing</h2>

<div id="demo">

<h3>Contents of /var/www/...:</h3>

<table>
```

```
            <tr class="header">
                <td>icon</td><td>name</td><td>size &nbsp;</td>
            </tr>

          <?php foreach( $listing as $key => $file ): ?>

            <tr <?=hiLine($key)?> >
                <td><?=formicon($file['path'])?></td>
                <td><a href="<?=$file['path']?>"><?=$file['filename']?></a></td>
                <td><?=size2human($file['size'])?></td>
            </tr>

          <?php endforeach; ?>

        </table>

        </div>

        <h3>Source of this script</h3>
        <?php highlight_file($_SERVER['SCRIPT_FILENAME']); ?>

        </body>
        </html>
```