

What is the WWW?

In essence, HTTP is nothing more than a few additions to telnet, and the WWW is nothing more than a large grouping of read-only file servers. Together, this generates an environment where users can easily place requests for documents and have those request fulfilled. HTML enhances this by providing a mechanism to connect and organize the information in the documents.

Dynamic web pages?

- CGI allows any program to interact with the web server so that its output is returned as if it were a web page.
- Apache provides a module mechanism that allows extensions to be created to add functionality to it.
- `mod_cgi`, `mod_perl`, `mod_php`, `mod_ssl`

Two ways to make dynamic pages

1. Have a process independent of the web server regenerate pages as needed. (good for pages that are updated on a regular schedule, or change independently from user input, more load-efficient than #2)
2. Have a CGI, perl, php or other script generate the page on demand. (good for pages that change based on user input, or change very often, requires more processing power on the server.)

Case Study: weather

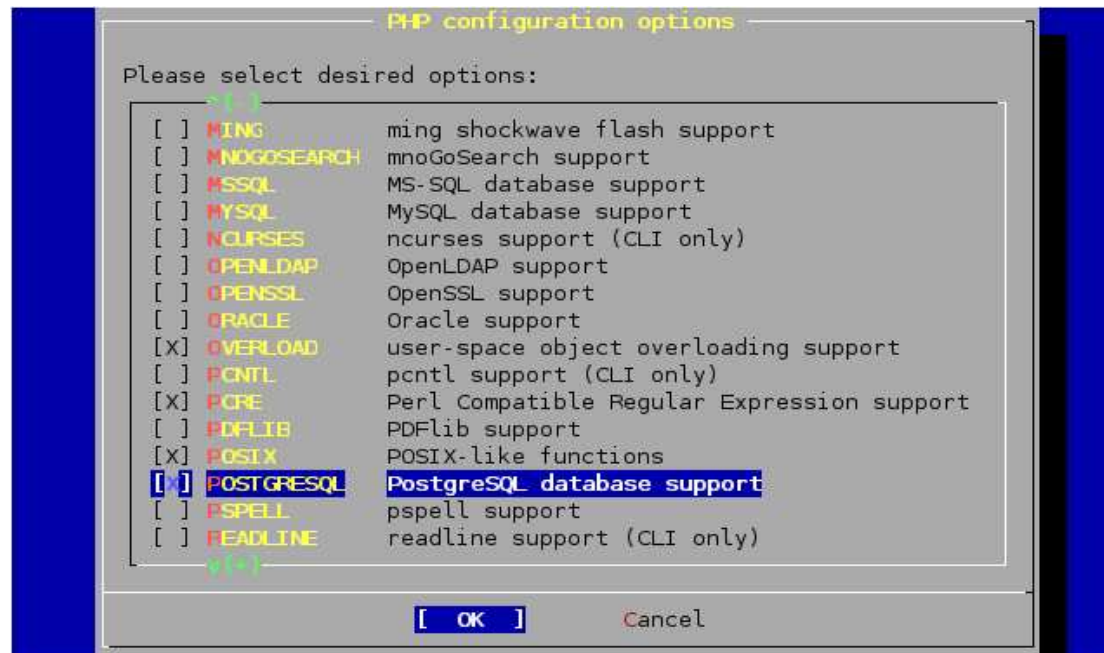
- Assuming we have 50 weather reports (one for each state) and a weather report page takes .01 seconds to generate.
- Generating these pages on demand means your server tops out at 100 hits/sec – after that you need a redundant server, which is not only expensive, but considerably more difficult to keep organized.
- Generating static pages once an hour takes $\frac{1}{2}$ a second, and the fact that the server is now serving static pages means it will easily be able to handle greater than 100 hits/sec. Additionally, if you need more hardware, it's easier to divide because the generation process can be moved to a dedicated machine.

Where are on-demand pages better

- Any type of search feature: each search is unique, to make a static page for every possible search is pretty much impossible.
- Most other situations where the content of the page is dependent on user input.

Getting php the way you want

Easiest way: Use FreeBSD and the port



Problem: Many precompiled PHP packages don't have Postgres support

Solution: compile PHP from source

1) Download the and unpack the PHP source code into a work directory.:

```
tar xyvf php-4.3.5.tar.bz2
```

2) Make sure Apache and Postgres are already installed

3) Configure and build PHP

```
./configure --with-pgsql -with-apxs
```

```
make
```

```
make install
```

4) Copy the php.ini file to /etc and review/tweak.

5) Add the following line to httpd.conf (tells apache what to do with php files)

```
AddType application/x-httpd-php .php
```

6) You may want to teach Apache that index.php is a valid directory index. To do so, create the following in httpd.conf (a `DirectoryIndex` line most likely already exists):

```
DirectoryIndex index.php index.html
```

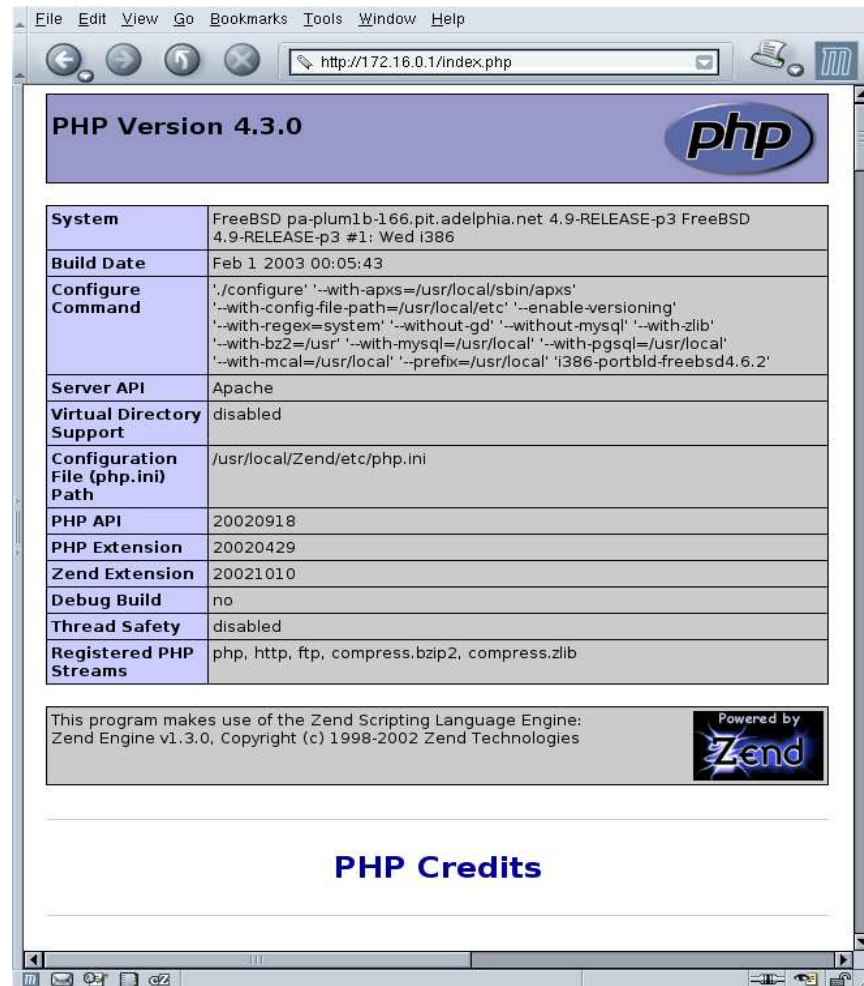
(This tells Apache what file to display when a directory is provided without a filename)

Your first PHP webpage

Look in `/etc/httpd.conf` for the line that reads:

`DocumentRoot /some/path`
This tells you where to put your documents.

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



The screenshot shows a web browser window displaying the output of the PHP `phpinfo()` function. The page title is "PHP Version 4.3.0" and features the PHP logo. The main content is a table with the following data:

System	FreeBSD pa-plum1b-166.pit.adelphia.net 4.9-RELEASE-p3 FreeBSD 4.9-RELEASE-p3 #1: Wed i386
Build Date	Feb 1 2003 00:05:43
Configure Command	'./configure' '--with-apxs=/usr/local/sbin/apxs' '--with-config-file-path=/usr/local/etc' '--enable-versioning' '--with-regex=system' '--without-gd' '--without-mysql' '--with-zlib' '--with-bz2=/usr' '--with-mysql=/usr/local' '--with-pgsql=/usr/local' '--with-mcal=/usr/local' '--prefix=/usr/local' 'i386-portbld-freebsd4.6.2'
Server API	Apache
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/Zend/etc/php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20021010
Debug Build	no
Thread Safety	disabled
Registered PHP Streams	php, http, ftp, compress.bzip2, compress.zlib

Below the table, it states: "This program makes use of the Zend Scripting Language Engine: Zend Engine v1.3.0, Copyright (c) 1998-2002 Zend Technologies" and includes the "Powered by Zend" logo. At the bottom of the page, there is a link for "PHP Credits".

Quick explanation of using PHP

```
<?php
$a = 5;
echo "<p>This will be executed as program code</p>";
echo "<p>The variable \"a\" is $a</p>";
?>
<p>This is considered straight HTML</p>
```

Basic procedure for extracting data from a database

```
01 <?php
02 $conn = pg_connect('dbname=test user=pgsql');
03 $result = pg_query($conn,
04                 'select * from testtable');
05 echo '<pre>';
06 $rownum = 0;
07 while ($row = pg_fetch_assoc($result)) {
08     foreach ($row as $colname => $colvalue) {
09         echo "row $rownum: $colname = $colvalue\n";
10     }
11     $rownum++;
12 }
13 echo '</pre>';
14 pg_free_result($result);
15 pg_close($conn);
16 ?>
```

Using make & php to generate static pages

- Make is a program for automating processes.
- php was originally designed ONLY as a web scripting language, but has evolved into a general-use scripting language.

An example using make to automate website maintenance (not using php)

```
all: index.html about.html products.html
```

```
index.html: header.s links.s home.s footer.s
```

```
cat header.s > index.html && \  
cat links.s >> index.html && \  
cat home.s >> index.html && \  
cat footer.s >> index.html
```

```
about.html: header.s links.s about.s footer.s
```

```
cat header.s > about.html && \  
cat links.s >> about.html && \  
cat about.s >> about.html && \  
cat footer.s >> about.html
```

```
products.html: header.s links.s products.s footer.s
```

```
cat header.s > products.html && \  
cat links.s >> products.html && \  
cat products.s >> products.html && \  
cat footer.s >> products.html
```

The same site using PHP to build pages on demand

```
<?php  
include( 'header.s' );  
include( 'links.s' );  
include( 'home.s' );  
include( 'footer.s' );  
?>
```

Bit more complicated example

...

```
<form method=post action=search.php>  
<input type=text name=searchterm>  
</form>
```

...

...

```
<?php  
// This is the file search.php  
$result = pg_query($conn,  
    "SELECT title, shortdesc "  
    "FROM docs WHERE title LIKE '%" .  
    $_POST['searchterm'] . "%'");
```

...

Responsive: decreasing page size

```
<a href='p1.htm'><img src='i/n1.png'></a><br>  
<a href='p2.htm'><img src='i/n2.png'></a><br>  
<a href='p3.htm'><img src='i/n3.png'></a><br>  
<a href='p4.htm'><img src='i/n4.png'></a><br>  
<a href='p5.htm'><img src='i/n5.png'></a><br>  
<a href='p6.htm'><img src='i/n6.png'></a><br>  
<a href='p7.htm'><img src='i/n7.png'></a><br>
```

```
<a href='home.htm'><img src='images/home.png'></a><br>  
<a href='aboutus.htm'><img src='images/aboutus.png'></a><br>  
<a href='jobs.htm'><img src='images/jobs.png'></a><br>  
<a href='investing.htm'><img src='images/investing.png'></a><br>  
<a href='contact.htm'><img src='images/contact.png'></a><br>  
<a href='products.htm'><img src='images/products.png'></a><br>  
<a href='services.htm'><img src='images/services.png'></a><br>
```

Database/PHP grabbag

```
echo "Today is " . date("l dS of F Y h:i:s A");
```

Will print something like:

Today is Wednesday 15th of January 2003 05:51:38 AM

```
echo "Welcome user from computer " . $_SERVER['REMOTE_ADDR'];
```

Will print something like:

Welcome user from computer 172.16.0.57

```
$u = "UPDATE last_visit SET visit_date = NOW() WHERE ip = " .  
    $_SERVER['REMOTE_ADDR'];
```

```
pg_query($conn, $u);
```

Allows you to track the last time a particular IP hit the site

```
$r = pg_query("SELECT visit_date FROM last_visit WHERE ip = " .  
    $_SERVER['REMOTE_ADDR']);
```

```
$row = pg_fetch_assoc($r);
```

```
echo "The last time you were here was " . $row['last_visit'];
```

Resources

<http://www.apache.org>

<http://www.php.net>

<http://pear.php.net>

<http://www.sitepoint.com/article/effective-website-acceleration/1>