Dynamic webpages with php

Nathan Warner

Northern Illinois
University

Computer Science
Northern Illinois University
United States

Contents

1 Setting up apache on arch linux 2

1.1 Configure PHP with Apache . . . 2
1.2 Enable PHP in Apache . . . 2
1.3 Start and Enable Apache . . . 2
1.4 Test the PHP Configuration . . . 3
1.5 Adjust Permissions . . . 3
1.6 Allow http traffic . . . 3
1.7 Test apache config . . . 3
1.8 Pages not loading, change default php module . . . 4
1.9 Apache document root . . . 4
1.10 Permission error: 403 . . . 5

2 php in .php 6

3 Lexical structure 7

3.1 Case Sensitivity . . . 7
3.2 Statements and Semicolons . . . 7
3.3 Whitespace and Line Breaks . . . 7
3.4 Comments . . . 7
3.5 Variables . . . 8
3.6 Constants . . . 8

Setting up apache on arch linux

In order to view code written in .php files, we must set up a web server. On arch linux, we can setup apache. First, install the required packages

```
pacman -Sy apache php php-apache
```

1.1 Configure PHP with Apache

To make Apache process PHP files, you need to edit the Apache configuration.

```
sudo vi /etc/httpd/conf/httpd.conf
```

Add the following lines to the end of the file to load the PHP module and configure PHP handling:

```
# Load PHP module
LoadModule php_module modules/libphp.so
AddHandler php-script .php
Include conf/extra/php_module.conf
```

1.2 Enable PHP in Apache

The file php_module.conf should have been installed with the php-apache package. If it's missing, you may need to create it manually at /etc/httpd/conf/extra/php_module.conf.

```
/etc/httpd/conf/extra/php\_module.conf
```

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

This line ensures that Apache will serve index.php as the default file if it's available.

1.3 Start and Enable Apache

```
sudo systemctl start httpd
sudo systemctl enable httpd
```

1.4 Test the PHP Configuration

```
echo "<?php phpinfo(); ?>" | sudo tee /srv/http/index.php
```

Open your web browser and go to http://localhost/index.php. You should see the PHP info page, indicating that PHP is correctly configured.

1.5 Adjust Permissions

If you plan to edit files in /srv/http frequently, you might want to adjust the permissions:

```
sudo chown -R $USER:http /srv/http
sudo chmod -R 755 /srv/http
```

1.6 Allow http traffic

If you have a firewall enabled, allow HTTP traffic:

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Or with ufw,

```
sudo ufw allow http
```

Once you've set up Apache and PHP as described, you can view .php files by placing them in the web root directory and accessing them via a web browser.

1.7 Test apache config

Sometimes, a small syntax error in the Apache configuration file can cause the server to fail to start. Run the following command to check the configuration:

```
sudo apachectl configtest
```

If there's an error, this command will display a message that can help pinpoint the issue.

1.8 Pages not loading, change default php module

switch to the mpm_prefork module, which works well with the default PHP module on Arch Linux.

Edit the Apache configuration file /etc/httpd/conf/httpd.conf and comment out or remove the line loading the mpm_event module:

```
# LoadModule mpm_event_module modules/mod_mpm_event.so      # Comemnt
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so    # add
```

Then restart apache

```
sudo systemctl restart httpd
```

1.9 Apache document root

If you'd prefer to store your PHP files outside of /srv/http, you can change the Apache DocumentRoot to point to a different directory.

First, make the directory you want as root

```
mkdir -p ~/mywebsite
```

Open the Apache configuration file:

```
sudo vi /etc/httpd/conf/httpd.conf
```

Locate the DocumentRoot directive, which should look like this:

```
DocumentRoot "/srv/http"
```

Change this to the path of your new directory, such as:

```
DocumentRoot "/home/yourusername/mywebsite"
```

Also, update the <Directory> block for /srv/http to match your new directory:

```
<Directory "/home/yourusername/mywebsite">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Ensure that Apache has permission to read files in your chosen directory:

```
sudo chown -R $USER:http ~/mywebsite
sudo chmod -R 755 ~/mywebsite
```

After making these changes, restart Apache to apply the new configuration:

```
sudo systemctl restart httpd
```

Place your PHP files in the new directory (/mywebsite). You can then access them in your browser as before:

```
http://localhost/test.php
```

This way, you can keep your PHP files in a custom directory without needing to store everything in /srv/http.

1.10 Permission error: 403

For Apache to access files in this directory, it needs execute (x) permissions on each parent directory in the path.

Ensure that each directory in the path (/home and /home/*username*) has the appropriate execute permissions for Apache to access subdirectories. This doesn't mean Apache will have access to all files, but it will allow it to "traverse" the directories.

```
sudo chmod o+x /home
sudo chmod o+x /home/username
```

This will allow "other" users (including the Apache user) to traverse the /home and /home/*username* directories.

You should also change permissions on the documentroot directory, and possible the parent directories. Then restart httpd

```
sudo chown -R $USER:http ~/documentroot
sudo chmod -R 755 ~/documentroot
sudo systemctl restart httpd
```

- **$USER:** This is an environment variable that represents the currently logged-in user's username. By using $USER, you're setting yourself as the owner of the directory and its contents, allowing you to manage the files easily.
- **http:** This is the group associated with the Apache server on many Linux distributions (including Arch Linux). Assigning the http group to the directory allows Apache to access the files and directories within, provided the group has the necessary permissions.

php in .php

php code should be contained with php blocks

```
example.php

<?php #no space between ? and php
    ... php code
?>
```

We can have as many blocks as we want, or have everything in one block. Anything not in these blocks are sent to the server as plaintext (or html if its html code).

Lexical structure

3.1 Case Sensitivity

The names of user-defined classes and functions, as well as built-in constructs and keywords such as echo, while, class, etc., are case-insensitive.

Variables are case sensitive.

3.2 Statements and Semicolons

Use semicolons to separate statements, lines must end with a semicolon.

3.3 Whitespace and Line Breaks

In general, whitespace doesn't matter in a PHP program. You can spread a statement across any number of lines, or lump a bunch of statements together on a single line. For example, this statement:

```
raisePrices($inventory, $inflation, $costOfLiving, $greed);
```

Could just as well be written with more whitespace

```
raisePrices (
    $inventory ,
    $inflation ,
    $costOfLiving ,
    $greed
);
```

3.4 Comments

Comments can be made with #, // or */.

3.5 Variables

Variable names always begin with a dollar sign ($) and are case-sensitive

3.6 Constants

A constant is an identifier for a simple value; only scalar values—Boolean, integer, double, and string—can be constants. Once set, the value of a constant cannot change. Constants are referred to by their identifiers and are set using the define() function:

```
1 define('PUBLISHER', "O'Reilly & Associates");
2 echo PUBLISHER;
```