Web Dev primarily HTML and PHP

I'm going to give an overview of web development for users of Slackware.It's going to be from the perspective of approaches for a budding web developer who uses Slackware, the problems you might face due to using Slackware Linux and an eclectic look at what tools are available to help you, starting simplest first.

There will be some embedded information on HTML and PHP, what they do and what they do not do. I will probably throw in some anecdotal info from my experience.

Structure of HTML & PHP

HTML evolved from a need to easily share digital documents between researchers working at CERN (Conseil Européen pour la Recherche Nucléaire). Lets have a look at a HTML document as it used today 2020:

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8" />
<title>  </title>
</head>
<body>
<h1>Hello World  </h1>
</body>
</html>
```

If you're good at geometric progressions, you will straight away see common re-occurring elements <> and </>; HTML is not programming code but merely about how things should be presented or markup. Its a web browser that does the interpretation of the tags and then presents the content according to the markup.

<> designates the start of markup and </> the end of the markup. The file is conveniently organized into head and body areas and the whole text wrapped by `which indicates start of a HTML document and by` which indicates a HTML document has come to an end.

So as a developer, you have two perspectives working with HTML; the text if the file is opened with a text editor and how it looks after a web browser has digested what's written and then presented it (also called has rendered document).

HTML is not the only text document that has associated with it instructions of how it should look. When you use a word processor like LibreOffice you see the text as you type, but behind the scenes the text has associated with it instructions such as font type, font size, and so on. For this reason, never use a word processor when constructing HTML documents, since you will have to edit out the accessory info.

There are many alternatives to working with HTML documents using a text editor, but as good as any

in my opinion is geany available from slackbuilds.org.It has basic text highlighting and gets the job done.

Using Built in PHP dev Server to render HTML

Now, if we were working with geany and saved the above text as a file say index.html to your Desktop, you then have two choices; open and edit with a text editor or open it as intended with a web browser. If you do that, all you will see is “hello world”.

You could, if you want, have several thousand files on your Desktop and right click each one with the mouse to look at them. That's rather inconvenient and making poor use of a web browser. Inside let's organize things and put files together in a directory on the Desktop; let's call it *webPlay*.

So, this is what we now have on our Desktop:

```
webPlay
└── index.html
```

We only have one file at the moment in our directory ; we could put 1000's in that single directory and right click to open each with a web browser. When Tim Berners-Lee came up with the idea of HTML (hypertext markup language), he also had a vision of an architecture of how researchers at one institute would request a document “get” it and view it.

Software that can be leveraged to make things convenient involves being able to “listen” to requests made by a browser and to respond to that request or, to put it another way, deliver the document.

We can make use of the basics of the system utilizing what's referred to as the *built in PHP development server*. PHP, which we will discuss later, are other files that are processed by web architecture. The important point is that it is more than capable of working with HTML as well.

The key thing about web development is that if you have the means to get the basics up and running then you can progress.

So, a few things to mention; PHP is included with a standard Slackware installation; like everything else nothing stays still and the version of PHP is associated with the version of Slackware you have. We can find that out by opening a Terminal Emulator. I use XfCE, so all I have to do is left click the terminal Emulator icon bottom of screen.

At the prompt, I type as follows:

```
bash-5.0$ php -v
PHP 7.4.1 (cli) (built: Dec 19 2019 00:29:31) ( ZTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
with Zend OPcache v7.4.1, Copyright (c), by Zend Technologies
bash-5.0$
```

So, I have PHP version 7.4.1 available, which is very useful since most modern web software like

CodeIgniter require PHP 7 and above as stated on the documentation of CodeIgniter4. That version of PHP came with Slackware -current which I upgraded to early last year (5.4.12) - a very good vintage!

Let's fire up the dev server and “point it” to where our HTML files are:

```
bash-5.0$ cd Desktop
bash-5.0$ pwd
/home/andrew/Desktop
bash-5.0$ php -S localhost:8080 -t webPlay
[Sun Nov 15 14:23:18 2020] PHP 7.4.1 Development Server
(http://localhost:8080) started
```

When I open up the Terminal Emulator, it opens with the “context” that it is at ~ i.e home

So, I change directory to Desktop, because that's where my directory containing my HTML file is. I also tell PHP, via the terminal, that I would like to use localhost and port 8080.

If I now open up any web browser and type into the “address bar” (the address bar is NOT the text box where you search for stuff, a la Google but the thin top text space) http://localhost:8080, I wonder what we will see?

Well it's 2 words: hello world

How did that work? I didn't stipulate any file name. Well, we called our HTML document index.html; the suffix designates that it is an HTML file and index is significant as well. Web server software is configured to look for files to open as a starting point that are called index.php or index.html.

To put it another way, index.html is the default file that the PHP dev server will look for and render, so we didn't have to state the filename specifically. You would get the same result with:

```
http://localhost:8080/index.html
```

A single index.html is about as simple as you can get; if you uploaded it say via cPanel on hosting, and then you pointed your website domain to your hosting, as instructed by your hosting company then you will have a simple, but working one page web site.

More sophisticated web systems such as WordPress still have somewhere within them one or more index.php (with WP its located at the web root). In these cases, the index.php file is often referred to as the “bootstrap” not to be confused with front-end Twitter Bootstrap.

Before we move on to PHP let's have a last look at HTML and how to link one page to another. Go to geany →file→new with template→file_html5.html. That produces a ready made html5 template. All you have to do is name it and save it somewhere. Save it to /Desktop/webPlay as about_me.html.

Open it up and in the <body> </body> write a few words. Now let's see how we can link our index.html to it.

In index.html I have added a line starting <a href

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
<meta charset="utf-8" />
<title>  </title>
</head>
<body>
<h1>hello World</h1>
<a href ="about_me.html">link to about me  </a>
</body
</html>
```

Now, in your web browser, re-fresh to http://localhost:8080/

Click on the line that says *about_me* and you will see that the HTML document about_me.html is now rendered. You now have the basis for a few pages on your web site. Some will pooh-pooh this, but it's better than the option of no web site, if you are a small business. Without a web site in today's world, you can't be found.

Making use of HTML presentation but with PHP functionality

HTML documents are fine. Their purpose is to display content as you intend. Nothing changes unless you edit them live via your hosting cPanel, or you edit them local and upload via cPanel with “overwrite all” clicked. Thus, HTML pages are often referred to as “static” pages. Now lets have a look at PHP still using dev inbuilt server, for now, and still using directory webPlay.

Open a terminal Emulator and change directory (cd) to webPlay.

Type this code:

```
mv index.html index.php
```

What the code does is rename index.html to index.php Refresh the browser and what do you see? Exactly the same. So what's the difference? Well, PHP documents use HTML layout, but the .php suffix tells web servers and our built in dev server that, due to the suffix, look out for, and process any PHP code within the documents.

Let's do a quick example:

type <?php echo phpinfo(); ?> somewhere within the body tags eg:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>  </title>
</head>
<body>
<h1>hello World</h1>
```

```
<a href ="about_me.html">link to about me  </a>
<?php echo phpinfo();?>
</body
</html>
```

Refresh your browser and now what do you see? Quite a lot of information concerning PHP! When I first did this it demonstrated the potential of PHP.

Introducing Databases into the mix

Next we will have a simple look at using a database.When you mention anything about databases most people immediately think MySQl or Maria;so you can't do anything with a database on your Desktop since you need a daemon. Well actually you can since sqlite3 is server-less. Assuming our Terminal Emulator is in webPlay lets create a database from the command line. Actually lets first check we have the tools in place:

```
bash-5.0$ ls /var/log/packages | grep sqlite
sqlite-3.30.1-x86_64-1
//yes we are in luck !
```

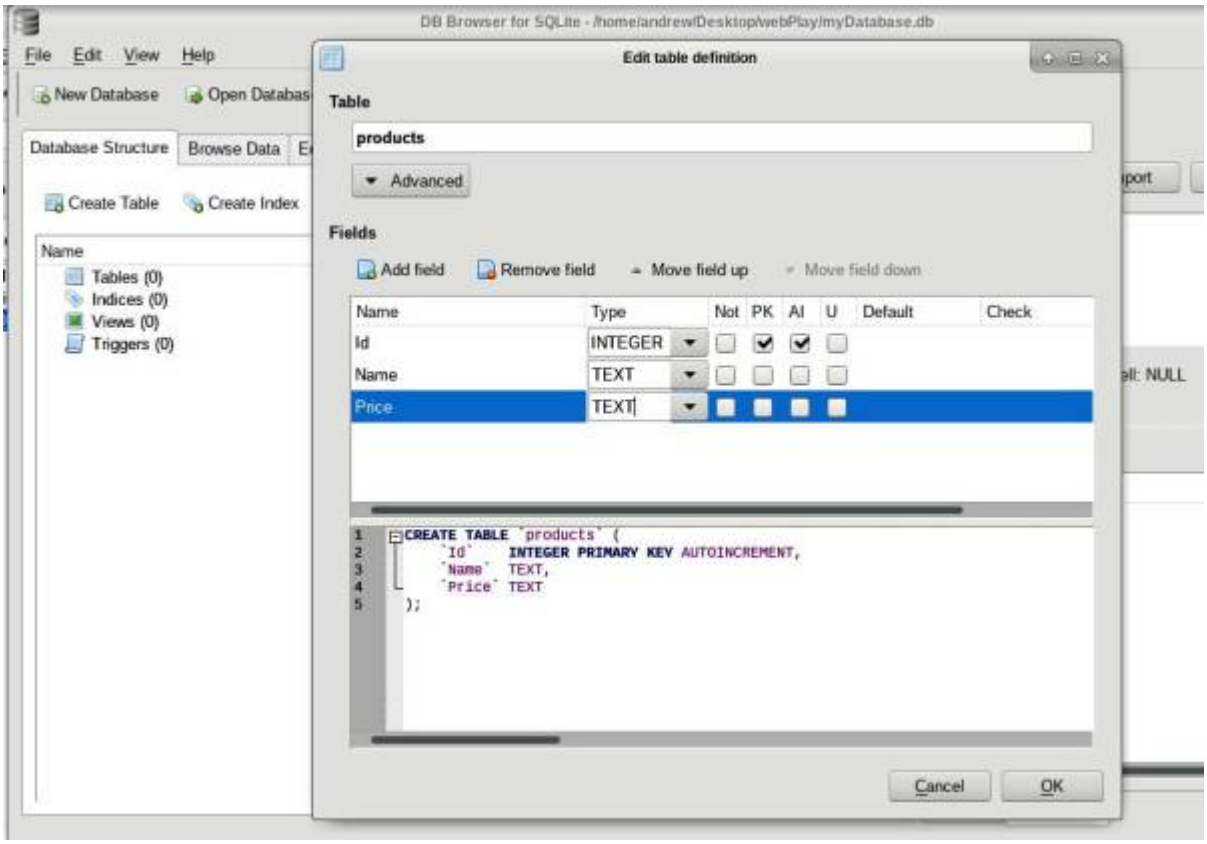
```
 bash-5.0$ sqlite3 myDatabase.db
```

The above line will create an sqlite3 database file and then it will go to the sqlite prompt i.e

```
sqlite>
//using the sqlite prompt lets confirm our database was created:
sqlite> .databases
main: /home/andrew/Desktop/webPlay/myDatabase.db
sqlite> .exit
```

The command line is all well and good but sometimes a gui is a better approach.We could install a table, then populate but instead why not make use of sqlitebrowser in my case i have sqlitebrowser-3.9.1-x86_64-2_SBo

To fire that up from a command line just type “sqlitebrowser”.



With sqlitebrowser its just a case of using file open and navigating to where the database is, its quite intuitive and from the image you can see that a table can be created, with fields and even data typed into the fields. We will just populate with a couple of entries for demonstration. Probably the only two things you have to remember when typing in is “apply” and then finally “write changes”. Also notice i have a table with a field called “Id” which is primary key and auto increment.

Now back to out index.php file.

Take out the php function phpinfo() and add the following :

```
<?php
$db = new SQLite3('myDatabase.db');
$res = $db->query('SELECT * FROM products');
while ($row = $res->fetchArray())
  {
     echo "{$row['Name']} {$row['Price']} <br>";
  }

$db->close();
?>
```

So your index.php should now look like:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
<meta charset="utf-8" />
<title> connect to db example </title>
</head>
<body>
<h1>hello World</h1>
<br>
<a href ="about_me.html">link to about me  </a>
<br>
<?php
$db = new SQLite3('myDatabase.db');
$res = $db->query('SELECT * FROM products');
while ($row = $res->fetchArray())
     {
          echo "{$row['Name']} {$row['Price']} <br>";
     }
$db->close();
?>
</body
</html>
```

if you refresh your browser this is what you should see.

Now lets have a look at the code :

```
$db = new SQLite3('myDatabase.db'); //this gives us a connection to our
database file
$res = $db->query('SELECT * FROM products'); //this is a simple sql query of
selecting all entries in the table products
while ($row = $res->fetchArray() //this is a run through of an array
```

So here we just did a retrieval of data; its one small step from there to insert, update all part of what is referred to as “crud” operations. HTML files are just static files that have markup on how the static content is presented. php on the other hand is a rich fully functional programming language that you

can either use in an OOP way. In other words create classes which have methods or in a procedural way.

A couple of other things to mention PHP, is a scripting language; an “engine” such as Zend processes the code, then renders output to the browser.

So what i have demonstrated here is that you can go a long way developing PHP on your Desktop.You may have noticed that no mention was made of the word chmod nor permissions. Everything we did was in user space, so you don't have to worry about that. With files located in our user space we have both read and write permissions to them. I mention it here, since later that's going to be a problem when we look at xampp and apache, which is another approach for working with web development.

PHP framework

Before we look at xampp and apache i'm going to mention composer composer

When your working with single HTML or PHP files, you will find you the need to update them that's not too much of a problem. When you work with more complicated web applications say CodeIgniter, new releases come out and then you have 3rd party software installed as well which have dependencies.Things can soon get out of hand. The use of composer helps updating and managing dependencies.

I've used it in 2 ways.

1)Download composer.phar to Desktop. Give it 777 permissions via su→# chmod 777 composer.php Then move it to /usr/local/bin and rename it via: # mv composer.phar /usr/local/bin/composer

I found when you at a terminal Emulator run command “composer” its up and running

2)I noticed its available on slackbuilds, so you can run the slackbuild and install the _SBo;looking at the slackbuild it seems to work in a similar way, putting composer in /usr/bin and also renames composer.phar to composer.

Now that we have composer working lets use it, to download a php framework to our Desktop.Later we will move it.

```
$ cd Desktop
bash-5.0$ composer create-project codeigniter4/appstarter CI4 --no-dev
```

In the above,the first word “composer” composer invokes the binary; “codeigniter4/appstarter” is a key pair of words that you will see on Packagist and CI4 is the name i want of the directory to be called which will contain CodeIgniter. This is the structure i get after issuing the command:

```
CI4
├── README.md
├── app
├── builds
├── composer.json
├── composer.lock
```

```
├── env
├── license.txt
├── phpunit.xml.dist
├── public
├── spark
├── tests
├── vendor
└── writable
```

Its quite compact and by using –no-dev I told composer not to download extras like Kint. Now codeigniter inside it spark which we can use to fire up the framework:

```
bash-5.0$ php spark serve
CodeIgniter CLI Tool - Version 4.0.4 - Server-Time: 2020-11-17 08:45:01am

CodeIgniter development server started on http://localhost:8080
Press Control-C to stop.
[Tue Nov 17 14:45:01 2020] PHP 7.4.1 Development Server
(http://localhost:8080) started
```

A little bit about PHP frameworks. A few HTML or PHP files is ok for a small website but PHP frameworks come in handy where you are going to do a lot of work with data. Frameworks have a design of logic that splits what you see, what does the processing, and what connects to database into whats called MVC (model,view,controller).

PHP built in server or using spark have their limitations; if you insist on wanting to use MySQl or Maria then you have to consider something like xampp or apache.

Now xampp from apachefriends xampp is basically a bundled unit of a web server and supporting files.

It comes with an installer and I have previously tried it on Slackware. Although it comes with an installer it is not installed on your system but is put in /opt and the way i view it is that it is an “embedded system” .

There are pro's and cons. Its doesn't use your system PHP as far as i understand it, so your own PHP might be more up to date than xampp, then there are issues like updating. From my limited understanding a web server does not come as default with other distros, so for them they have the choice: should I install apache or give xampp a go, maybe thinking at least I won't bork my system- its quite understandable.

Setting up a Development Environment using Apache

In slackware however, as default you should find a directory:

```
 htdocs at   /var/www/htdocs/
```

Thats where we are now going to move our CI4 to. I must at this point put in a disclaimer that I'm writing from my experience of getting things going. I have for a few months been writing a light C.M.S system on top of CodeIgniter 4.0.4; I've had no glitches, no crashes or anything else detrimental- so

I'm assuming from empirical observation I must have half a clue- others might point out how it could be bettered.

Ok this is one way I use CodeIgniter in apache:

First we are going to our Desktop to zip up CI4 and them move it to /var/www/htdocs.

```
cd Desktop
bash-5.0$ zip -r CI4.zip CI4
//now we move it to /var/www/htdocs
su->password->#
# mv CI4.zip /var/www/htdocs
```

Approach To permission problems

Now i will talk generally about problems with permissions for developers. For those coming from Windows,who have done web development they get a bit of a shock when they move to Linux. As a “normal user” you don't have any authority to do any editing of code of a web app directory located within htdocs located at /var/www/htdocs . That's because they are within your root system.

Now this is where perhaps the bone of contention sets in;you want to get on editing and developing but you don't want to mess your system up either.You could do su→password→# i.e shift to root but that wouldn't be a good idea, when all sorts of strange things could happen. With root permission you could delete or edit something you didn't want to do. So whats the answer? Well there are a few approaches.

First lets have a look at httpd.conf

```
bash-5.0$ cat --number /etc/httpd/httpd.conf
// on line 192 and 193 i see:
192 User apache
193 Group apache


535 # Uncomment the following line to enable PHP:
536 #
537 Include /etc/httpd/mod_php.conf

//so on line 537 remove the # in front of Include /etc/httpd/mod_php.conf
like I have
//you will have to fire up a editor to do that - nano is simple to use.
//just remember with nano ^ means ctrl key, so usually its change then ctrl
+ o,  ctrl + x
```

So we have seen thats its stating there is a owner called apache. You can use that and add yourself to the group.

```
// I'm going to add myself to the apache group (my user name is andrew)
```

```
# usermod -a -G apache andrew

//next cd to /var/www/htdocs
# cd /var/www/htdocs
# unzip CI4.zip
# chown apache:apache CI4  -R
# chmod 775 CI4  -R
# cd CI4
# chmod 777 writable -R

// writable directory needs   777 permissions.
```

A note on file and directory permission: on a live server directories normally have 755 permissions and files 644. The “execute bit is needed on a directory in order for something to enter a directory.

We are on our own system so we don't have to rigidly stick to that. Here is another approach: As a normal user I am part of users. You can see that when you run the code:

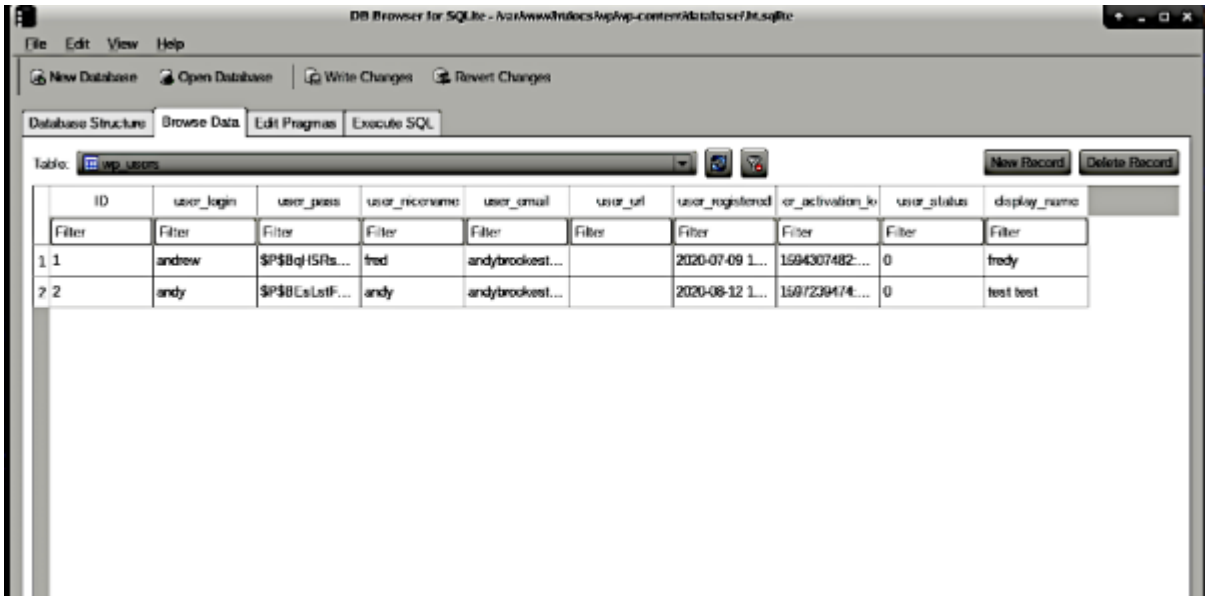
```
$ groups
```

So I will use the approach of leaving apache the owner but having the group set to “users”. Then I will endeavor to set directories to 755 and file permissions to 664 which should be Ok for our web dev purposes.

```
cd /var/www/htdocs
su->password->#
# chown apache:users CI4 -R
# cd CI4

find . -type d  | xargs chmod 775
find . -type f  | xargs chmod 664
```

Before i forget,a quick mention on databases.Most people or so it seems, just jump to MySQl and might quote its secure, since in the code to connect to the database it uses a database user name and password; grant all permissions and all that stuff. Well WordPress uses MySQl and has a table where it stores users, called wp_users; i have WP on local host and so not giving away any secrets when I show you my local WP ;it looks like this:



Its an extract of how WP uses Sqlite (yes you can opt for sqlite instead with WP) WordPress seems to be poorly written and a word of warning; in a default install its possible to display user login credentials via a simple get request to the domain!So the security of connecting to a database adds nothing to the security of your user credentials with WP

With CodeIgniter4 for security, everything accessible via the web is inside the public directory. Sqlite has a capacity of 281 Terra Bytes which should be enough for most web sites. So I will not be talking about setting up MySQl next, only how to get the httpd daemon going and set up virtual host so that we can view our app from an Ip of choice

We are going to edit /etc/httpd/extra/httpd-vhosts.conf , nano editor is as good as any for this purpose I have nano-4.7-x86_64-1 .You can check yours via:

```
bash-5.0$ ls /var/log/packages | grep nano
```

WE need root permissions so its :

```
# nano /etc/httpd/extra/httpd-vhosts.conf
```

I typed into :

```
<VirtualHost 127.0.0.9:80>
CustomLog "/var/log/httpd/CI4-error_log" common
<Directory "/var/www/htdocs/CI4">
Order allow,deny
Allow from All
AllowOverride All
Require all granted
</Directory>
ServerName CI4.org
ServerAlias CI4.0rg
DocumentRoot "/var/www/htdocs/CI4/public"
</VirtualHost>
```

you can check with :

```
# apachectl
httpd: Syntax error on line 506 of /etc/httpd/httpd.conf: Syntax error on
line 137 of /etc/httpd/extra/httpd-vhosts.conf: /etc/httpd/extra/httpd-
vhosts.conf:137: <VirtualHost> was not closed.
//i fixed that
```

Two things really to look at in the virtual host config:

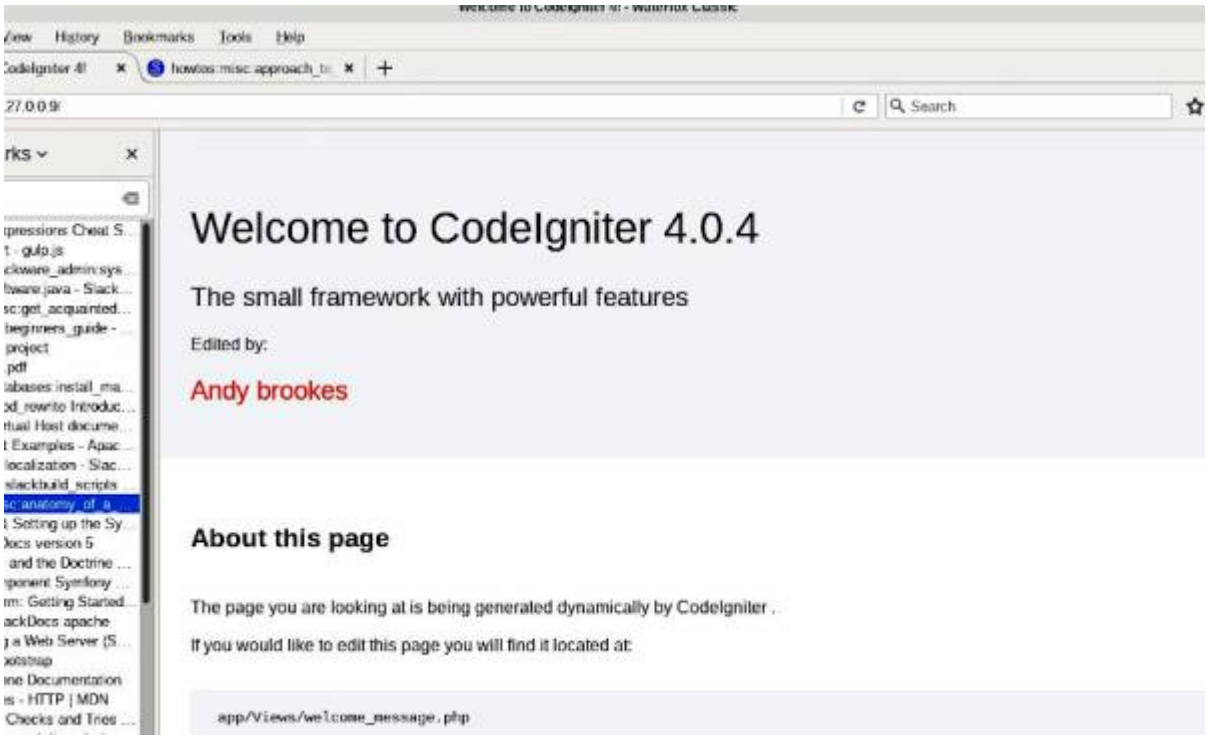
```
<VirtualHost 127.0.0.9:80>
//what im doing here is that because i'm very lazy i just want to type
127.0.0.9 in web browser address bar to see  framework
DocumentRoot "/var/www/htdocs/CI4/public"
//the web root of codeigniter is the public directory so we set it up for
apache to listen for requests at that directory
```

Next we will fire up apache daemon

```
# chmod a+x /etc/rc.d/rc.httpd
# /etc/rc.d/rc.httpd start
```

next lets check we can edit framework as normal user:

```
$ cd /var/www/htdocs/CI4/app/Views/
// next right click on welcome_message.php open with geany and edit
```



Note the 127.0.0.9 in address bar and that I can now work with framework and edit as user andrew , without risk of messing up system as root. If you follow this through and you get a problem its probably either your php version is < than 7.3 or I did something somewhere forgot about it and

forgot to mention .

Sources

Originally written by captain_sensible

howtos