The Drag-n-Drop Portal System

# Beyond the Portal

For anyone who wants to quickly set up their own portal web site, PHP-Nuke offers easy installation and powerful features. Coupled with a load of add-ons and an extensive support network, PHPNuke is the right choice for inviduals, clubs and even businesses. In this article we show you how to set up just such a system for your own use and tell you about the problems and worries of the PHPNuke system.

BY JAMES MOHR

Peter Wunderlich, Faido, visipix.com

For some, simply being on the Internet is enough. They toss a few more or less coherent pages together and that is their Internet "presence". For others like me, it has to be a lot more. We strive for a professional looking website with professional content, hoping to give a little something back to the community that has given us so much. Reaching that point is not always easy. However, it can be if you find the right tools to help you achieve that goal.

## My Search for the Holy Grail

In my department at work, I am often referred to as the “script grandpa” as I can usually come up with a script to perform almost any task. Be it a shell script or perl, I can normally figure out something that does the job.

However, given a choice between writing code and writing documentation, I actually *prefer* writing the documentation. The same also applies to the various websites I run.

On my websites, I am frequently caught between the desire to add new features, and wanting to add new content. For several years, I have been been working on the Linux Tutorial [1]. Originally, it consisted of a set of static HTML pages, the bulk of which came from my book “Linux User’s Resource”. As I continued to work on the site, I came to realize that providing static pages was not enough and I began adding features, eventually creating a very simplistic, yet modular, portal system. This included pop-up glossary terms, a Linux quiz with over 500 questions and answers, and a number of other features to help people learn Linux more easily. I eventually reached the point where I was spending more time writing code to add new features than I was spending on adding new content. The exact opposite of what I wanted.

I didn’t want to simply leave the site as it was, as some of the features I wanted would enhance the usefulness considerably. Making it an open source project and posting it to SourceForge got me quite a few volunteers, but most disappeared within a few weeks because of “other commitments”, never having written a line of code. Some did help, but progress was slow. So, I had to write most things myself. What I needed was something that gave me the features I wanted **now**, thus letting me begin adding content again. It also meant I had to find a portal system that would allow me to easily integrate my existing material (as I didn’t want to spend as much time porting my code to the portal as I would coding the new features).

Once the decision was made, I began my search. In addition to the necessary features and ease of porting my code, I needed something that had enough support to help me when I ran into trouble.

Amazingly enough, the search did not last very long as I quickly came across PHPNuke [2].

## The Good, The Bad and the Ugly

One thing that put me off at first was the fact that, as it's name implies, PHPNuke is written in PHP. With the exception of a very brief introduction a couple of years ago, I had no real previous experience. Was I going trade the extra time spent coding for extra time spent learning a new programming language? Fortunately, the answer is a resounding "No!" Although not identical to perl, the constructs are very similar to perl and porting my code was amazingly simple. In fact, within a couple of weeks I had my site ported. Well, at least on my test system.

The next step was to begin implementing the features that I had been wanting to add for several years. Still, in order to put together a useful site with PHPNuke, you do not need to be a good programmer. In fact, you don't need to be a programmer at all. PHPNuke provides so many features simply by installing it that there may not be any need to add anything else.

Just some of the features available include:

- User management (including paid subscriptions)
- Forums
- User contributed news
- Surveys
- User contributed Web Links
- Downloads
- Newsletter
- Encyclopedia
- Journal
- Reviews
- Statistics
- Feedback

Features are nice, but are often useless if you don't know how to implement them. The PHP-Nuke_HOWTO, available on the PHPNuke site [2], is a great piece of work, covers all of the important features, and is loaded with specific tips and tricks. Although it goes into the inner-workings in a few places, I feel that it is lacking real depth.

The lack of in-depth documentation is compounded by the fact that there are almost no comments in the code. The only comments I have found so far are the copyright notices at the top of each file. That means you often have to do a lot of footwork when trying to modify or use existing code.

One thing I that feel borders on "ugly" is the attitude toward security. PHPNuke has a reputation for not being secure. Almost every software product has security bugs and PHPNuke has gone a long way in fixings, plus there are a number of add-ons that really *enhance* the security. However, it does seem that the developers are more interested in features and not security.

The one thing that really bothers me is a comment by Francisco Burzi (developer of PHPNuke) in the article "Clarifications on a possible rewrite of PHP-Nuke" available on the *phpnuke.org* website:

"The new code will be closed. This means that script kiddies should get a computer science master or PhD before hacking it. This will reduce the security issues."

To me that sounds like a very Microsoft attitude toward security. Security by obscurity has never been a good thing. Further, the "closed" code of which he speaks is described just the way it sounds, and future versions may not be open source.

## A Quick Look Inside

For me, one of the most significant aspects of PHPNuke is it's user management capabilities. On many sites, certain features are only available to registered users. For example, in order to post to their forums, you need to be registered. PHPNuke implements this in a very logical and efficient manner. Users simply fill out a forum with their username, email and password, and PHPNuke immediately sends them a confirmation email containing a link back to the site. Clicking on the link activates your account.

This method is important for two reasons. First, it prevents a person from signing up someone else without their knowledge. Second, if the user really wants to sign-up, he or she has to do it in two steps. This is useful for the site owner as a verification that the user *really* wants to join. (often referred to as double opt-in) This becomes important when using the newsletter feature so that you don't get accused of spamming your users.

Officially, PHPNuke can be installed on Windows or Linux, but is also being run on FreeBSD, OS/2, and even MacOS. The key conditions are that you have have an Apache web server with PHP version 4.2.x or better, and have the mod_php Apache module installed. In PHP-Nuke version 5.3, an SQL abstraction layer was added, which means that PHPNuke *should* work with MySQL, mSQL, PostgreSQL, ODBC, ODBC_Adabas, Sybase and Interbase servers.

In a nutshell, using this SQL abstraction layer, all queries are standardized as they are passed to the SQL abstraction layer, which then calls the appropriate function for the database you are using. Since MySQL is what is officially tested and what I have used in all of my installations, I am going to stick with that.

A PHPNuke page is broken into five areas: header, left column, contents, right column and footer. What actually appears in each of these is defined either in the system administration module or the various module, theme or block files. In the left and right columns, you see various "blocks" that the administrator can define either as a separate PHP file or as HTML code. The content of the header is typically defined within each theme, and the footer is defined in the system preferences.

The content is created by the modules themselves, which are the primary components of PHPNuke. Each of the features mentioned above is managed by a module, for which there is a sub-directory underneath the *module* directory. The actual work is done by the *index.php* file within the respective module sub-directory. With this standardized structure, all you need to do is create a new sub-directory and an *index.php* file in order to add your own module.

When activated, modules will be visible by default in the "Modules" block on the left hand side of the page. However, being active does not necessarily mean they are visible. On my site, I have a number of modules that, although active, are not visible in the Modules block. However, I still access them in the same way, like this:

```
modules.php?name=MODULE_NAME
```

where *MODULE_NAME* is the module name as well as the name of the directory where the module resides.

I use this characteristic for many features on my site that I don't want to include in the menu. For example, I have a number of "special pages" that are only accessible from certain other pages. Creating these pages is done using a separate module, and thus I need a module that is invisible to the user, but still accessible.

Also, inactive modules are still accessible by the administrator. This allows you to test the module before you put in online.

Where the modules are the primary components of PHPNuke, you might have figured out that the *modules.php* script is the key to all of this. Although the *module.php* does not usually create the framework of the page or add the actual contents, it is responsible for ensuring that the correct module is called. If the module can not be found, is inactive or inaccessible for another reason, it is the job of the *module.php* file to display the appropriate message.

## Getting Ready to Launch

The first step is getting a copy of PHPNuke. As of this writing, the current version is 7.3. However, this version is not yet available to the general public. In order to download it, you need to pay a *small* fee. For this article, I purchased a copy to make sure I was as current as possible.

Prior to the public (read: free) release, the developers typically release a version which is only available by paying this small fee. It costs about $10(US) plus tax (£5.40, 9.92 Euros). Since I think this is a great product, I have no problem supporting development like this, especially, since I get a copy of the latest version before other people. It did take about an hour for the order to go through its channels before I was actually able to download the file, but that was an acceptable wait.

If you don't want to pay the fee, you can download the prior version directly from the PHPNuke homepage: *www.phpnuke.org*. This is done by clicking Download from the main menu.

Despite its complexity, PHPNuke is extremely easy to install. If you are running on a system on which you have full root privileges, are not too worried about security, and simply want the default configuration, you can install PHPNuke and get it running, literally, in less than 5 minutes. However, even on my local machine I was not happy with the default, so it took a little longer.

## Gotcha

Depending on what version you get of the PHPNuke package, the *Install.txt* file has a little "gotcha". It says "Unzip the package into the directory you want to use on your web server." To me (as well as many other people) this implies that you unpack this in your server's document root. Unfortunately, that is not what it does. In the case of the version 7.0 that I originally downloaded from SourceForge, the package was actually unpacked into a sub-directory named *PHP-Nuke-7.0*. This meant having to move everything into my web server's document root.

Version 7.3 the files are unpacked directly, without a *PHP-Nuke-7.3* subdirectory. In addition to various .txt files, there are three directories: *sql*, which contains the sql script to create the necessary tables (*nuke.sql*); *upgrades*, which contain the scripts necessary to upgrade between versions; and *html*, which is the primary PHPNuke directory.

Once the file is unpacked, you need to edit the file *config.php* found in the top-level directory (usually your document root). Here you make the necessary changes to fit the configuration to your environment. As an extra security precaution you might want to consider moving your *config.php* script from the server's document root. Keep in mind that the system still needs to have the information in that file, so you will need to create a config.php in the document root that includes the real *config.php*. This file needs only one line:

```
<?php include("../config.php
"); ?>
```

I would also suggest creating a user and password whose only purpose is to access the database. By default the *$dbuname* variable is set to root. This is not a good idea, as root just has too much power. Note that you will need to give that user the necessary rights to manage the tables.

On my local machine, I have several web servers, including the test server for my Linux Tutorial website. Since I wanted to have PHPNuke on multiple servers, and at the same time needed to keep the instances separate, I had to create multiple databases. This meant changing the *config.php* file and the *nuke.sql* file. In the *config.php* I set the values as the following:

```
$dbname = "jimmonuke";
$prefix = "jimmonuke";
$user_prefix = "jimmonuke";
```

### Box 1: Parameters in *config.php*

***$dbhost*** – This is the name of the host where the database is running. Leave it as "localhost" if it is on the same machine as your PHPNuke installation.

***$dbuname*** – This is the name of the user used to access the database. If you configure this on a Web hoster, you will likely have a user other than root. Check with your hoster for details.

***$dbpass*** – This is the password for the user above.

***$dbname*** – The default is "nuke", but with many web hosters you may have to add a prefix like "USERNAME_nuke". For security reasons, I would always change this.

***$prefix*** – This is the prefix for the database tables. Here too, I would always change this for security reasons.

***$user_prefix*** – This should be the same as *$prefix*. However, you could have multiple PHPNuke servers sharing a common user database. In which case the *$user_prefix* would be the same for each of the PHPNuke servers.

***$dbtype*** – Sets the database type. For MySQL it must be "MySQL".

***$sitekey*** – This is the site key and used to generate the security images when logging in. It can be as long as you want.

***$gfx_chk*** – This defines when the security code image is displayed. If 0, it means the code is never displayed, 7 means always. Having the code basically prevents automated logins, so I always set mine to 7 on live systems.

See Box 1 for a brief explanation of each field.

Then, in the *nuke.sql* script I had to change all of the references from nuke_ to jimmonuke_. (Watch out for the underscore!)

Next we create the database. Since I defined my database with a name other than the default, I can not use the example in the Install.txt file. Instead, I needed to create it like this:

```
mysqladmin create jimmonuke
```

Depending on how your system is configured, you may need to use the -p option so that you are queried for your password. You can run this command as any user that has the permission rights to create new databases. I actually created the database as root, and then gave privileges to my PHPNuke user for just this one database. The SQL query looked like this:

```
GRANT ALL ON jimmonuke.* to ➥
jimmohome identified ➥
by "PASSWORD"
```

Next you need to create the tables in the database using the nuke.sql script as follows:

```
mysql jimmonuke < nuke.sql
```

When you connect to a site running PHPNuke, you typically load the index.php file in the top-level PHP directory. You will need to set the DirectoryIndex on your web server to index.php, so it is loaded instead of the standard index.html. Many web servers have both as the default, so it may not be a problem. Check your server beforehand to avoid any surprises.

## Taking Off

If you have done everything right, your system is ready to go at this point. You can now start administering your PHPNuke system by pointing your browser to:

```
http://your.site.name/admin.php
```

As I mentioned, on my personal site I did not put PHPNuke into the web server's document root. Instead, I renamed the default directory *html* to *nuke* and copied this into my document root. Therefore, I access the admin page like this:

```
http://your.site.name➥
/nuke/admin.php
```

When you call up the admin page for the first time, you are asked to create a user to act as the administrator. You also have the option of creating a "normal" user with the same data as for the admin user. My personal preference is to create a different user for a couple of important reasons.

First, it helps me keep track of who is doing what. I can log in as normal user to see how the system behaves as a normal user, which avoids issues with having more privileges. Second, more than likely I will want to post messages, news, or whatever and the username shows up for that user. If you also have an administrator with the same name, people immediately know the name of the administrator user. Not that this is an immediate problem, but the less others know about the inside of your site the better.

After creating an administrative user, you are taken to the administration page. Here, you configure not only the basic behavior of your PHPNuke system, but also which blocks and modules are displayed.

I would suggest that the first thing you do is to configure the basic system by clicking on the button labeled "Preferences". Here you define things like the site name, various defaults, footers and so forth.

After you have made your changes, press the "Save Changes" button at the bottom of the page. Next, select the "Modules" button. Here, you manage the modules that have been configured for your system, such as specifying which modules are active, who can see them, which one is the default, and so on. You'll notice that about half of the modules are set to "inactive" by default, which means this module will not appear in the list of modules in the left-hand column.

On live systems, I suggest removing all modules that you are not using. If you are not using them, you may not be as aware of security holes and may not correct them, thus perhaps giving someone access to your system. The default modules cannot be called directly, and the modules.php script prevents inactive modules from being started. However, removing them completely is a little more secure. Also I feel that you should practice using any of the modules on a test system just to be sure, before you make them live.

You will see that one of the modules is highlighted; this is typically the news module. The highlighting defines which module you see on the homepage. By default, this is the news module.

## Locking it down

As I mentioned before, PHPNuke does not have a reputation for being secure. This is not to say that the system is full of holes. However, there are number of things that you should do.

First check out the security forums on NukeCops [3]. Here you will find discussions of specific problems and how to fix them, as well as forums for several security addons. Next, check out the NukeFixes [4] site. They have patched versions of every version since 6.0. We'll be talking more about security issues in follow-up articles, but this will give you a good start. ■

### INFO


[1] Author's PHPNuke system: *http://www.linux-tutorial.info*

[2] PHPNuke: *http://www.phpnuke.org*

[3] NukeCops: *http://www.nukecops.com*

[4] NukeFixes: *http://www.nukefixes.com*

[5] PHPNuke forums: *http://www.nukeforums.com*

[6] Resources for PHPNuke: *http://www.nukeresources.com*



THE AUTHOR

*James Mohr is responsible for the monitoring of several datacenters for a business solutions provider in Coburg, Germany. In addition to running the Linux Tutorial web site (http://www.linux-tutorial.info), Jim is the author of several books and dozens of articles on a wide range of topics.*