

Drupal for itSMF and Serbia

Ian Dickson ianjdickson@gmail.com
Main site www.likal.com

itSMF International were looking at Drupal for their new website earlier this year. They asked if I could help.

This paper looks at Drupal and then explores the build process.

Introducing Drupal - Overview

Drupal is a MySQL / PHP based CMS that has evolved to the state where people with HTML and CSS knowledge can use it to build powerful sites.

Those with good php and MySQL skills can get involved with the ongoing development as well, though you do not need these skills to use Drupal.

Functional CMS vs Stylish CMS

I was studying CMS a while ago looking for one that would suit my needs for the project that became Likal.com

It became clear that there were two types of CMS – ones that were built by hard core coders who valued function over visual design, and ones that allowed web designers to control ever pixel on the page, but that didn't really DO very much.

I opted for Functional, and it was obvious that Drupal (then in version 4) was the one.

It had a strong developer base, and, most importantly, was designed for multi editor use, and content mapping was tag based. Both of these are essential for community sites (my main interest).

A Framework, not a CMS

Drupal is a modular framework.

Drupal is NOT a CMS in the conventional sense of the word (a system designed to make the building and running of websites easy, within tightly controlled parameters) but a Framework.

By analogy with buildings – programming is an architectural skill set. A Framework is the built skeleton of a flexible building plus modules for major elements, and a conventional CMS is a completed building awaiting the attentions of the interior designer.

A CMS is the easiest option IF your desired site fits a CMS recipe.

The programming route is the most flexible, but also the hardest and most costly.

The Framework approach is a nice medium. It takes some work to get to grips with in order to develop a site, but the heavy lifting has been done by programmers, and the flexibility avoids many of the issues that affect CMS's as soon as you want to change things.

However...this approach does mean that Drupal is not suited to people wanting to build quick one-off sites that they can then forget about. For those, a CMS like Wordpress might be the way to go. Drupal is better suited to people (non programmers) building and managing multiple sites, esp where site functionality is subject to change or mission creep, (and would thus run into CMS limitations).

Because Drupal is open and php based, it is also of interest to php programmers, who can build on the work done and develop new modules, themes etc. Those new to php can view code and see how it works.

Core and Contributed Modules

Out of the box Drupal – Core Drupal (CD) - is very minimalist. At first it seems as though it can do almost nothing and many new users give up at this point.

However think back to the analogy – they have the skeleton of a building, and first need to install interior walls, fixtures and fittings before getting to the painting.

Core Drupal is deliberately kept slim. The main reason for this is that it allows different people to solve problems in different ways via Contributed Modules, and this serves to help evolve the best solutions, while not cramping those who need something specific.

All sites will need to use a few of the many hundreds of Contributed Modules (CM), and site design is very much a matter of deciding which of these to use.

For background it's worth noting that initially there were no CM. Over the years programmers using CD would need extra functions and they solved this by writing code that hooked into CD then did what they needed. Over time a library of functions has developed. These are CM.

The good news is that as you get familiar with Drupal you realise that CM fall into various categories:-

Essential Generic

These CM tend to be big projects run by key Drupallers and designed to really extend Drupal. They tend to be focussed on data handling and processing / management and most non technical people are amazed that they are not in Core Drupal.

Key examples are Views, which is essentially a way of building and running SQL queries to control data display, and Content Construction Kit which is a way of controlling data entry and management.

Both of these projects arose when it was realised that while lots of small ad hoc modules were being developed to do parts of these tasks, often reinventing the wheel, none were full solutions. So some leading Drupallers decided to create formal, full, generic solutions instead.

Essential Local

All web hosting services are different. Essential Local CM are normally utility type functions that allow you do things your host doesn't, or lets you do things easily that are possible but hard via your host.

I like a module called PoorMansCron – this allows you to tell Drupal to do a cron run even if you can't run cron on your server. I started using it because my first web host didn't let me set up cron. I use it today because it's easy and gives me one thing less to worry about re my host.

Essential for Style

A range of CM that you will need to consider according to the type of site you want to run.

A lot of these are media and social networking oriented, or designed to work with third parties and other software.

In many cases there is more than one CM for each problem, though usually there is a clear favourite.

Minor Interest

Most of the CM are of little use.

They either serve to solve a specific problem, or are legacy modules whose purpose has been taken over by a more general solution. Some are apparently silly modules – e.g. Pirate (on International Talk like a Pirate Day, turns your site into Pirate speak. Arr, that it doe's Cap'n) - can be used by trainee module designers because small modules are easier to explore and understand at a code level than big complex ones.

Information Model

For most new users, the Modules aspect is the easy bit. Once they realise that only a few modules will apply to them they start building sites.

It's fair to say that most Drupal sites do not fully use the facilities for information management provided by Drupal. Personally I have also been surprised at the difficulty that many designers have with the information model. I think this is because programmers care mainly about doing stuff and most designers care about looks. Publishers care about readers, and the Drupal information model is about readers.

The Drupal information model is all about Nodes and Taxonomies.

Nodes

In Drupal almost any Set of information is a node.

A Story is a set of information comprising Title, Text and gubbins about who wrote it, when and what tags it might have. A Story is a Node.

There can be many types of Node – Event Nodes have information about events, and some specific fields like start date and time etc.

Types of Node are called Content Types. (Agreed, it is confusing terminology).

In general a Content Type is designed to fit a specific purpose, (not subject) – if it is an Event it will have Time/Date fields. If it is a CV it will have qualifications fields. You need a new Content Type when you need Content with specific fields, and no current Content type has them, or could be usefully extended by them.

Taxonomy

The art, or science, of classification.

Yahoo began as a taxonomic attempt to categorise the Internet.

Google totally ignores taxonomy in favour of brute force pattern matching.

The Semantic Web, when it arrives, will do so because computers have learnt how to do taxonomy via brute force pattern matching.

The key thing about taxonomy and web sites, is that they work within a site, and, for all useful purposes, each site is an island with a taxonomy designed for it, and any relationship to other sites taxonomies is pretty much co-incidence. (Which is why we don't yet have a Semantic web).

Websites, at least the type of site you'll use Drupal for, are always About Something and For A Readership.

In Drupal taxonomy helps people find what (nodes) they are looking for, by allowing those who post, or edit, to attach tags to the information (nodes).

The detail of taxonomy is a complex area, but in general you can have :-

- Free Tagging – keywords, any keywords you like. Think YouTube and Flickr.
- Hierarchical – eg Linnean taxonomy of life where the structure tells you about relationships. E.g. Homo sapiens actually means humans are apes are primates are mammals are vertebrates are animals.
- Mixed – semi structured but not formal. Most sites use this. You might require that people choose from a Country or a Nearest Town from a drop down list, but give them free reign to name “people in the image”.

The key advantage of a taxonomy is that it constrains users and thus makes finding content easier, esp. when the seeker can't be sure that any useful content exists.

For example if you have a site about domestic animals and you allow free tagging, photos of cows might be tagged by breed name. More importantly, photos of a breed might just be tagged “cow”. If someone seeks photos of Jersey Cows, and they be sure that a blank result for Jersey means that no Jersey Cows exist?

So one option is to have a required tag for species, with a drop down list, including Cow, and then another for Breed (context sensitive, so just Cattle Breeds would show, and including “unknown” for people who don't know the breed), and another free tagging for “unlisted breeds”.

The Editor (an expert in domestic animals) might check the site from time to time for “unknown” and add the breed in if he recognises it.

Now the seeker comes and sees the used tags as options – so chooses Cow then Jersey. If he draws a blank now he can be fairly sure that no images of Jersey Cows exist on the site.

The real value of taxonomy approaches over brute force search pattern matching is for web sites About something – so at least some of the terms can be defined, and where contributors are likely to know enough to tag correctly, or there are editors who will monitor and tag / correct according to requirements.

If anyone here is involved with help desks and customer service this Node and Taxonomy approach can be very powerful, both for internally (staff) and externally (customer) focussed knowledge/support systems, PROVIDED that an editor monitors and weeds out wrong material.

Community Aspects

One other very nice feature of Drupal is that as well as having the usual Forums, it also has a much more powerful comment based approach, (in CD) and naturally this has been extended by CM, to include rating systems etc.

This allows for people to comment posts, and the comments to be used by editors to amend, update or promote accordingly.

Interestingly people seem to take more care commenting a “run of site” post than they do making a forum post – I.e. Comments tend to be on topic and accurate, and don't turn into conversations the way forum posts do.

This can be used to help in quality control of sites and knowledge bases – editors can monitor comments and react accordingly.

OTOH in “unedited” environments (such as the Drupal handbook itself) the comments simply add to the material and help people find their way.

Design and Layout – aka Themes

Can Drupal sites look great? Yes, but...

By default (and standard layout options, called Themes) Drupal delivers nice information centric sites. Ideal for the likes of societies, blogs, newspapers etc.

What it does not do – without applying a lot of CSS knowledge and perhaps some Php – is allow pixel perfect control of the screen and all singing all dancing award winning design.

This does serve to foster a Keep it Clear style over Arty, but if Arty is what you want, just be aware that Drupal will make you work to get it. There are some great arty designed Drupal sites out there, but they tend to have decent budgets behind them – they used Drupal for power, and having to put work in at the front end was just a cost they decided to bear.

If you want a Blog or other simple generic site, but with a snazzy design, you should probably look at CMS's, not Drupal.

Hosting – issues

Drupal likes a nice standard hosting environment.

In my experience a cPanel system with full FTP, PhpMyAdmin, ability to backup and restore etc seems to be the best. Hosts running their own homemade software, or with limited facilities, should be avoided.

Shared Hosts are fine, provided the architecture is as above. That said, some Drupal modules are resource hungry and memory limits can be an issue.

When you install Drupal you will probably get errors, but these can usually be fixed in cPanel File Manager. (Most will be to do with Drupal installer lacking permissions to create some Directories and/or control permissions, so you'll need to do it by hand. The messages are usually clear and easy to understand).

Upgrade Path

Drupal supports a current and previous version.

Sites should be upgraded when all the modules they use are upgraded.

Upgrading should be simple, but needs to be approached with caution and care, and on any live site, it's good to practice in a sandbox duplicate.

WARNINGS

- If you have edited Drupal Core Modules, you will probably be unable to upgrade.
- If you have edited Contributed Modules, you will be unable to upgrade until you have understood how to edit the new version as well.
- If you have written your own modules, you cannot upgrade until you have rewritten it. This task will be hard if you didn't follow the best practice module writing rules.

Given the above, unless your site needs and can support the long term commitment to programming costs, DO NOT edit Drupal Core, or Contributed Modules, and don't write your own Modules either.

itSMFi and Serbia Chapter Site

We met and drew up a desired functional spec. (It was pretty clear that Drupal would be a good solution, and itSMFi had already reached that conclusion).

Much of this initial conversation was about deciding what was really important, vs what might be nice, vs what was on a wish list.

I don't think I ever said “you can't have that” to anything, but I did ask “is that important?” quite a lot. Every function, option or choice means a button or text on the screen, and, in general, the more of them there are, the more off-putting the site is to users. Most people using an organisation site just want to READ, and from time to time, say “I'd like to attend that”. Only a few people want to be more involved.

A good design aim is to for the client to realise that in running the site, anything they want doing, they will have to do themselves. Members stepping up to take on roles is a bonus, but should not be an expectation. By all means start limited and then grow, rather than try and start big and end up with an unmanaged mess.

Phase 1 - itSMFi

The initial site was built and since then has evolved slightly. At present we are exploring approaches to “private areas for small groups”, but want to do this after upgrading to Drupal 6.

A MySQL script was provided to itSMFi to allow some direct queries not covered by standard modules. A programmer might have written a module to do this. I didn't, because I am not a programmer, but also because an unsupported module would have created upgrade issues. If itSMFI had been an unsophisticated client and needed a module, this would have impacted on the spec and quote.

Phase 2 – Chapter Template and Serbia

This involved more discussion re what module set would make sense for Chapters own sites, and an install guide. Language issues needed to be addressed.

During this process the need by Serbia for a Chapter Site cropped up and this was built by way of a working example.

Post set up it became clear that additional tweaks were required, and also that, in the event of a problem, the guy who built it got the call. Even when it was a hosting issue.

The lesson learned is that my recommendation to itSMFi is that if they decide to offer a Chapter Site service, it should be a hosted, supported service. Quite probably a contracted out one. This will ensure that hosting issues either don't arise, or can at least be dealt with. Minor requests can be serviced, upgrades dealt with and major work – if needed - provided at a reasonable rate.

General Lessons

Like battle plans when the fighting starts, specifications become outdated as soon as you start building.

The key to a successful Drupal project (assuming limited resources and desire to avoid programming) has to be :-

- Be open minded about evolving ideas re functionality. Try and avoid locking yourself down.
- WORK HARD on developing a good initial taxonomy.
- Think carefully about Content Types (fewer is good).
- NEVER EDIT DRUPAL CORE CODE
- DO NOT WRITE YOUR OWN MODULES

