
# Drupal Multisite Installation

April, 2010

*Cesar Salazar, Drupal Development Lead*

523 W. 6th Street, Suite 330, Los Angeles, CA, 90014 **1.888.483.1770** **www.oshyn.com** newbusiness@oshyn.com

# Drupal Multisite Installation

# Introduction

Drupal is a great Open Source CMS. Whether you are currently evaluating CMS solutions, have already chosen Drupal or are currently using Drupal this white paper will help you understand Drupal Multisite Installation for medium or large Drupal sites.

At Oshyn, we are frequently asked if Drupal will allow management of multi-lingual or multi-regional sites. The simple answer is yes. A detailed answer is more complex because there is more than one way to implement, and the method you use will depend on the site's particular needs. The intent of this white paper is to describe the options for managing multiple affiliate sites. We will analyze the most popular design options, describing the pros and cons of each one and evaluating when to use one over another. (In a future white paper we will explain the options available for managing multiple languages sites using Drupal.)

In this white paper we will evaluate the following design options for managing multiple sites:

1. Different Drupal installations, multiple databases
2. Multiple sites in one Drupal installation, multiple databases
3. Multiple sites in one Drupal installation, multiple databases with some shared tables
4. Multiple sites in one Drupal installation, using Domain module, one database

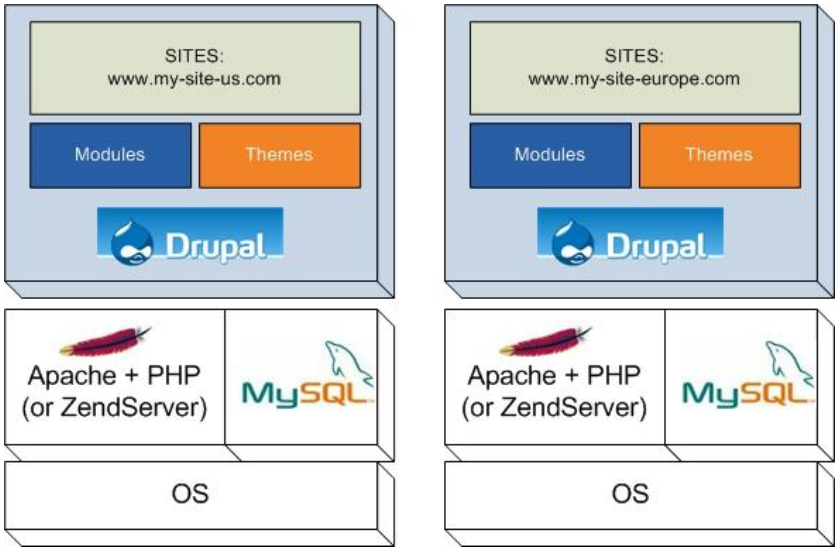
# 1.0 Multiple Drupal Installations

A Multiple Drupal Installation architecture will separate completely one site from another. Most of the time this is not the best solution, especially if the sites should be integrated one with the other,

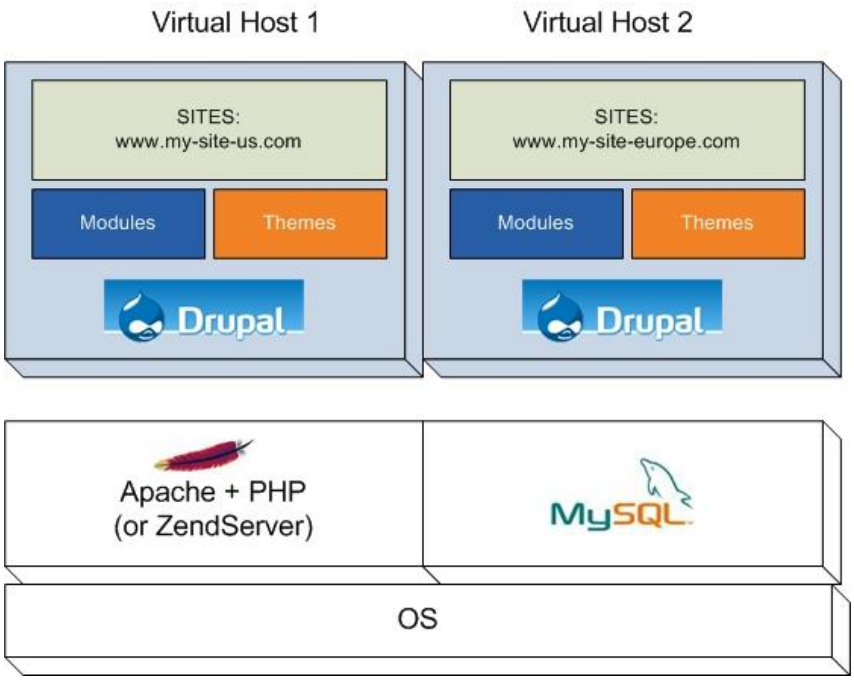
Usually a better solution is to have multiple sites in one Drupal installation, but we'll evaluate this architecture because there are situations when this could be the best way of implementing a solution for a client.

In general implementing multiple related sites in multiple Drupal installations could pose a big problem, but it could be a requirement in some cases. The client could for example, want one site for their office in the United States and one site for their office in Europe; with each site managed separately. In this case, having multiple Drupal installations could be the best option, even if they are hosted on the same servers.

The scenario would look something like this:



When we are talking of multiple Drupal installations we are not necessarily talking of multiple environments. Both installations could reside in the same Apache and MySQL boxes. The scenario is very similar; we would just need to configure apache's virtual hosts properly. The scenario in this case, is just a little bit different:



The benefits of having multiple Drupal installations for serving the sites are:

- Sites can escalate independently
- Sites can be managed independently
- Sites can be installed in different environments

The drawbacks of this design are obvious:

1. It won't be easy to share content and users between the sites

Since we are not sharing the database, we have to be aware that you cannot access content from the other site directly, except by implementing a sharing mechanism like RSS feeds, web services or similar.

And since users in one site are completely different than users in the other, implementing a shared one time login would be a pain. You cannot have shared users, except by implementing your own custom solution, which could be difficult to implement and also difficult to maintain.

2. More difficult to maintain

In general, having multiple installations to maintain is a drawback: when you fix bugs, run updates, change any setting, etc. ð if you want them to be applied in all the sites ð you would have to do it manually one by one.

The roles, permissions and users also have to be maintained separately.

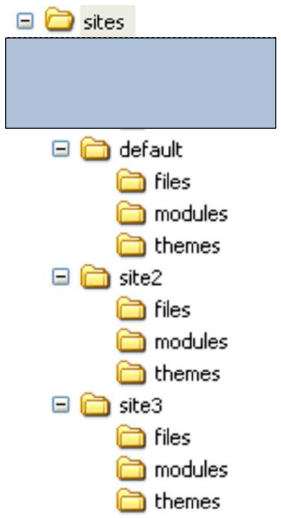
## 2.0 One Drupal Installation, Multiple Databases

Drupal provides the option to manage multiple sites within the same Drupal box. This is a very nice feature that this CMS offers.

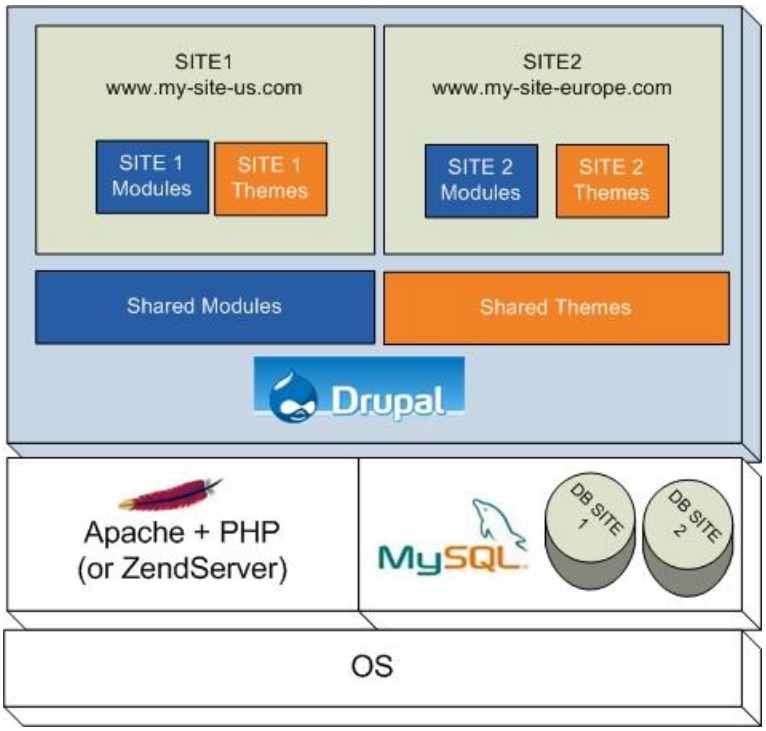
All the sites created in Drupal should be kept inside the sites folder. The sites folder can contain as many sites as we want. Drupal redirects the user to the appropriate site based on the URL entered. If no site matches, the user is redirected to the site installed in the ñdefaultòfolder.

Also, we have the option to share modules and themes between all the sites, by placing them in the ñallòfolder.

The folder structure is something like:




All the modules and themes located in the ñallòfolder are shared for all the sites. This is especially helpful when we need to have a theme applied to all the sites, and we need to perform occasional changes to the theme. Any change will be made only once and immediately it will be applied to all the sites.




The advantages over the previous approach are:

1. The modules and themes code can be reused among the sites
2. The updates can be applied to both sites

When updating a shared module, you will have to put all your sites in maintenance mode, update your code, and then run update.php in all of them.

In general, having the same code for all the sites is an advantage, but there are occasions when this could be a drawback; for example when you don't want to have any downtime in one of your sites and another one requires frequent updates.

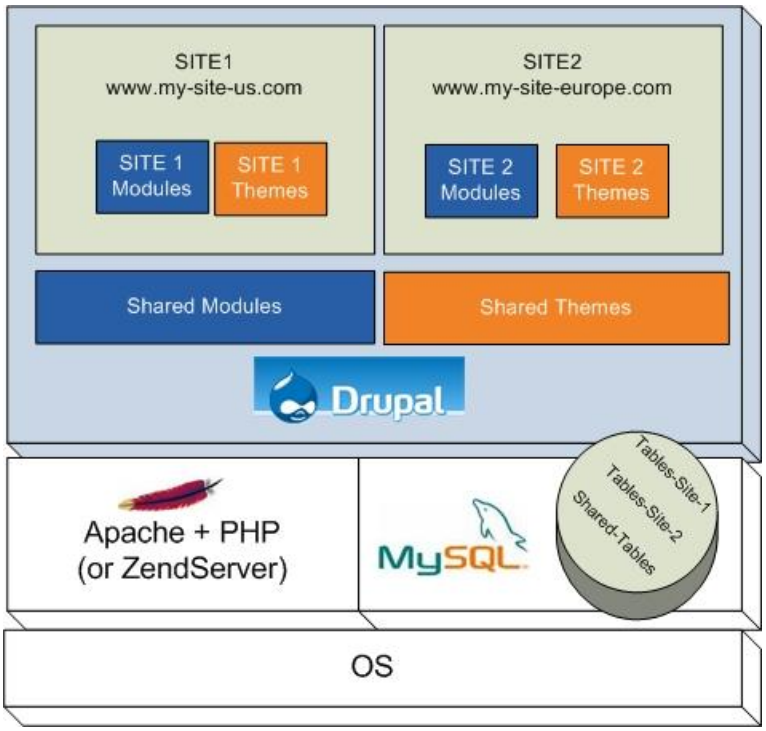
Consider this: what if you need to apply a bug fix in the code for a shared module urgently because site 1 needs it, but in the other hand you cannot have any downtime in site 2. If the module is shared among the sites, and the change you are applying requires a database change, you have no option; you will have a little downtime in all of your sites.

Regarding the data, we are not sharing it among the sites. When we talk about the data we are referring to everything that is stored in the database, which is content, settings and users.

We are not talking necessarily about different databases. Multiple sites can use the same database if they use a different prefix on the tables for each site.

Since we are not sharing data, this approach still has some of the drawbacks as the previous ones. The main drawback is that the content and users are not shared among the sites.

## 3.0 Shared Tables




In this approach, multiple sites can run in the same Drupal box, with some tables shared among the sites.

Drupal provides the option to define a default prefix for the tables, and/or a prefix for specific tables. By using this option, we could make multiple sites work over the same shared tables.

This sounds like a very useful and interesting option, but itôs actually not recommended by Drupal. From their website: ñ***This procedure could result in unexpected results, depending on which tables you choose to share, including broken version updates and/or security holes"***

However, this approach is useful when we want to have different sites, that share the same codebase (modules and themes), shared users and need a shared sign on.

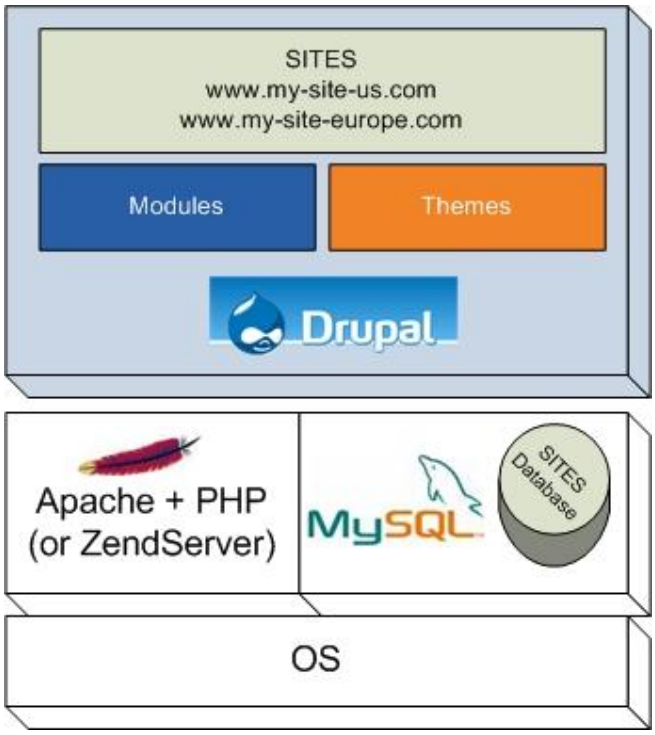
The design of the system needs to be done carefully. You must consider for each module and table, if you want it to be shared or if you want it to be managed separately.

There may be some gray areas when doing such design. For example: if you want to share users but not content among the sites, you won't be able to share the tables that contain both references to the users and to the content.

If you share the users, you should also consider the roles, permissions, etc. Depending on what modules you are using, the list can become quite long.

Another problem arises when you want to share only some nodes or some users, but not all of them. This is not possible using this approach.

## 4.0 Domain Access Modules




The last option is to use the Domain Access modules. This set of modules allows us to have multiple sites, in the same drupal box, using the same database and the same tables.

Instead of using different tables, as in the previous approaches, Domain Access modules rely on Drupal's Node Access system to determine what content is available on each site.

Generally this is my recommended approach for affiliated sites that want to share content and/or themes and/or users and/or settings.

Most of the settings in the site will be shared among the sites; some of them are configurable per site.

Also the content and the users are shared among the sites. Domain Access modules will allow each node to be displayed only in one site, in some, or in all of them. The same goes for users.

Each site can be configured to use a different theme, but we cannot control which modules are enabled for one or for another site. All the modules will be available for all the sites.

From domain project page[1], the core Domain Access modules are:

- **Domain Alias**: Allows you to specify subdomain aliases and domain name patterns for domain entries so that multiple hostnames are matched on a single domain entry.
- **Domain Configuration:** Allows you to change select system variables for each subdomain, such as offline status, footer message and default home page.
- **Domain Content**: Provides a content administration page for each subdomain, so that affiliate editors can administer content for their section of the site.
- **Domain Navigation:** Supplies a navigation block. Creates menu items for each subdomain, suitable for using as primary or secondary links.
- **Domain Prefix**: Provides a user interface for enabling select database table prefixing for use with subdomains.
- **Domain Settings**: Exposes domain-sensitive variable settings to most configuration forms.
- **Domain Source**: Allows editors to set a primary source domain when links are written to content from other domains.
- **Domain Strict**: Forces users to be members of domains in order to view content. This is a sample extension module that uses the API.
- **Domain Theme:** Allows separate themes, theme settings and colors for each subdomain.
- **Domain User**: Allows for the automatic creation of subdomains for registered users.
- **Domain Views**: Provides Views integration for Views 5.x.1 and 6.x.2 or higher.

[1] Information obtained from http://drupal.org/project/domain

Also available in the Drupal community are several add-on modules that work with Domain Access modules in order to make each individual site very customizable. Some of these modules are[2]:

- **Domain Locale**: Provides custom language sets per domain for Drupal sites using Domain Access and core Locale module.
- **Domain Access Advanced**: Enables multiple node access modules to work together by moving Domain Access rules into db_rewrite_sql() instead of using the node access API.
- **Domain Actions**: Provides integration for Domain Access to allow actions to be taken based upon the domain. Works with core actions and the Rules module.
- **Domain Blocks:** Enables domain specific visibility settings for blocks, eliminating the need to prefix the blocks table.
- **Domain Geolocalization:** Allows domains to have associated locative information. When combined with the Domain User Default module, this module uses the Google Maps API to geocode users zipcodes for domain proximity searches.
- **Domain GMap:** Multi-domain support for GMap module.
- **Domain Login Restrict:** Restricts user login by domain affiliation.
- **Domain Menu:** Allows each subdomain to use separate primary and secondary links
- **Domain Node Type:** Gives privileged users the ability to set default domains for content types.
- **Domain Nodequeue Tab**: Provides each node with a tab where users can assign that node to a domain-specific queue.
- **Domain Relationships**: Adds relationships/grouping capability for domains.
- **Domain Taxonomy**: Extends Domain Access functionality to taxonomy objects, by filtering terms by their assigned domain.
- **Domain Toggle**: Introduces open and private domains. In private domains users are forced to be assigned to a domain in order to view content on that domain. In open domains everyone can view the content. Replaces Domain Strict for some use-cases.
- **Domain User Default:** Provides methods for users (and unauthenticated site visitors using the Session API module) to select a default domain.

[2] idem

- **User Import Domain Access:** Allows users to be assigned domains when using the user import module.
- **Ubercart Domain Access**: This module provides a basic integration of Domain Access for Ubercart, allowing an Ubercart store to span multiple domains, storing the originating domain when an order is made and displaying the correct store information on invoices.

## About Cesar Salazar

During his 8 years in enterprise technology, Cesar Salazar has performed work for clients in the automotive, banking and other industries.

One of the areas of his major expertise is related to secure web banking systems and cutting edge web technologies. Other important experience is related to automotive expert systems, including hardware control, tracking systems and management of secure contact-less credit cards.

During the last year, Cesar has been performing as a Drupal Developer and Technical Team Leader, working on sites with awesome functionalities

## About Oshyn

Oshyn, Inc. is an Enterprise Technology Agency that has earned a reputation for delivering innovative business solutions for the web, mobile devices and enterprise technology platforms. Headquartered in Los Angeles, Oshyn's growing client list includes Best Buy/Geek Squad, Coca-Cola, Electronic Arts, Epson, Graduate Prospects, Fordham University, Harbor Capital, Lexus, Mars, Miramax, National Education Association, Oliver Wyman, Sapient, Scripps, Southern California Edison and Volkswagen.

Oshyn, Inc. is partnered with the some of the most respected agencies and technology providers such as Crispin Porter + Bogusky, Jahia, Microsoft, Ogilvy & Mather, Open Text, Oracle, Sitecore, Saatchi & Saatchi and Team One.

For more information visit us at www.oshyn.com.
Follow us on Twitter @Oshyn_Inc.
Read Oshyn's Web Content Management blog at Drupal blog at:
http://www.oshyn.com/_blog/Web_Content_Management