CRAFTSMAN CODING

MODX & WORDPRESS

SIDE BY SIDE

BY EVERETT GRIFFITHS

CRAFTSMANCODING.COM

MODX and WordPress, Side by Side

Everett Griffiths

This book is for sale at http://leanpub.com/modx-vs-wordpress

This version was published on 2014-11-22

Cover design by Nick Hoag, Owner/Director at The Future Forward

Contents

Chapter 1: Introduction

Goals of this Book

This book compares MODX 2.x and Wordpress 3.x side-by-side (specifically MODX 2.3 and WordPress 4.0). Its goals are to demonstrate through examples how to accomplish the same tasks in both systems. The focus of this book is on practicality: topics that are essential to understanding each system are included, as well as challenging topics that every good site administrator should know (such as how to reset a lost password).

This book aims to be a crossover manual so that those who are familiar with one system can quickly get up to speed on the other. This book aims to identify strengths and weaknesses in each system so that the reader will see for themselves how one system can accomplish a particular task better or more thoroughly than the other.

Intended Audience

This book is valuable to any developer who wants to quickly learn the ropes of either WordPress or MODX, but it is most valuable to the people who already are familiar with one system and want to learn the other. Just like books that teach a foreign language, this book attempts to repeat each discussion in two ways. If you already "speak" WordPress, this book will teach you the most important MODX "phrases". If you are fluent in MODX and need to know how to get by in the land of WordPress, this book will show you the WordPress equivalents of your "native MODX tongue".

This book is not intended for PHP beginners or for people who are just learning to use a content management system (CMS). If you are just starting to learn the ropes of how the web works or how to begin web development, then this book is probably not for you. If you have a technical background and have worked with other web applications before however, then you should have little trouble following this book and its examples.

Organization of this Book

Each chapter in this book covers a given topic in its most common permutations. When necessary, the topics are given some background and explanation because that helps understand why the applications are structured the way they are. When a solution in one system is discussed, it is discussed from the point of view of the other system. For example, if the topic being covered is templates or uploading media, the section describing how this works in WordPress is discussed from a MODX point of view, whereas the section describing how it works in MODX is explained in a way intended to make sense to a WordPress user. The terms "MODXers" and "WordPressers" are used throughout to refer to users who are versed in the respective systems.

WordPress: Pros and Cons

Pros

WordPress came on the scene in 2003 and quickly became the go-to platform for blogging: it is easy to set up, it can be customized fairly easily, and its admin dashboard is streamlined and intuitive. According to TechCrunch, it is one of the most popular web applications of all time: it powers approximately a fifth of all new sites launched in the United States (http://techcrunch.com/2011/08/19/wordpress-now-powers-22-percent-of-new-active-websites-in-the-us/). With that kind popularity, it is easy to find developers and themes, and there is a huge repository of plugins available for downloading.

Blogging has always been one of the core competencies in WordPress, and it stands out as one of the easiest blogging systems to set up and maintain. For the end users, it represents a clean and simple platform that helps them reach their audience quickly with minimal fuss.

Beginning with version 3, WordPress is finally wandering into the content management arena instead of being just a blogging platform.

The biggest draw to WordPress is probably its manager: it is well laid out, it is easy to use, and it is pretty snappy. Its PHP code is simplistic and easy for beginners to tinker with. If you are building a site that fits a standard bill, then chances are good that WordPress will allow you to get it built quickly and easily, and you won't need to be a developer to do it. WordPress is fantastically forgiving of incompetence: even the most inept luddite can fumble their way through building and running a WordPress site, and that fact alone is a hugely powerful testament to the value of this system.

Cons

MODXers may be surprised at the incompleteness of the WordPress API in some areas, or at its baffling thorniness in others. MODXers will be daunted by the complexity of WordPress themes. A full WordPress theme involves a dozen or so PHP files, and it is not always clear how they interact with each other, and it can be a time-consuming chore to track down which files are being used to render a page. There are functions and logical flows in the template files too, and since they are PHP code, they can crash or even be hacked! The only comparison that is remotely similar in the MODX world is the PHx plugin and its kin of output filters, which allows users to put logical statements into their template files, and its overuse often leaves MODX templates in the same state as most WordPress themes: unpredictable and difficult to debug.

Perhaps the biggest shortcoming of WordPress is its underlying code: most of its API "methods" are procedural functions declared in the main namespace. You might look through thousands of lines of source code and not find a single PHP object or class. Naming collisions are a big concern, and overriding default behavior is sometimes impossible. Some of this can be attributed to the fact that WordPress supported a dwindling number of PHP 4 users for a very long time, and to be fair, it is hard to change momentum and rewrite the core code when you have such a huge user base, but regardless, some of the core code is downright sophomoric, poorly conceived, and difficult to work with.

The other major consideration when dealing with WordPress is its near exclusive reliance on an event-driven architecture, which ends up being both a blessing and a curse. Every plugin "hooks" into either action- or filter-events and registers a callback function. The process is as simple as it is flexible, but the "event-space" can quickly get polluted with competing callback functions and there is zero visibility into which functions are modifying the output of any given event. By far, most development problems arise from these conflicts. Given the architecture, these conflicts are hard to avoid, so one of the first steps to debugging one plugin is to disable all the others. Whereas namespace pollution is easy to see because PHP will generate visible errors or warnings, the event-space pollution is a silent killer: there is rarely any warning. Even though most plugins rely on only a handful of common events, other events are almost mythical in their obscurity and many are poorly documented, so sometimes finding the right action or filter can be a demoralizing research task.

Because so many would-be "developers" have made what appear to be poorly conceived pork barrel contributions to the WordPress core, there are often multiple functions that accomplish *almost* the same thing, e.g. `get_posts()`, `query_posts()`, and `WP_Query()`. The WordPress coding standards are lax, and some parts of the core are inefficient and bloated and most of the code relies heavily on global variables. It must be said that poorly architected code is quite difficult to follow. As one core contributor groused in frustration: "WordPress is 100% open-source, 0% open-minded… [it's] one big patchwork of bad ideas and unchecked opinions" – this is a common sentiment in any open-source project, but the problem seems quite acute in WordPress. The code that is easy to implement when you start building your site is the same code that becomes a tangled, unforgiving mess when your project needs to scale or be customized.

WordPress is shorthanded when it comes to certain features that are common in more mature systems: **there is no built-in logging**, the control of permissions is simplistic, and official support is practically nonexistent. It may be king of the blogosphere, but in other arenas WordPress is a deadbeat or worse: users can be ruthlessly flamed for even suggesting that the architecture ought to be changed (full disclosure: I've been flamed for that). If you work in both systems, you may find that sometimes MODX's complexity and flexibility is matched by WordPress' primitiveness and streamlining. Tit for tat.

No Logs

The WordPress core does not log information, so debugging can be extremely challenging.

A practical note for web professionals: WordPress users are accustomed to getting things for free, so negotiating fees is a challenge. In my experience, WordPress clients tend to rank closer to hobbyists, so the project budgets follow accordingly. You have fewer tools at your disposal because the core API is sometimes maddeningly limited, and do not underestimate the pain that results from having no application logs to help you troubleshoot the inevitable problems. This means that many WordPress projects require extra work to implement and they often pay less because the competition is absolutely massive. It can be a bad place for a developer to work. Conversely, this is sometimes exactly why site owners choose WordPress: they don't want any lack of developers who can help maintain their sites.

MODX: Pros and Cons

Pros

MODX 2 (Revolution) was officially released in 2010 as the successor to MODX Evolution. It has a small and loyal following, and it offers features that are hard to find anywhere in the open-source world.

Historically, one of the strengths of MODX has been its implementation of custom fields (known as "Template Variables" in MODX parlance). It has always been easy to define and associate a variety of custom fields with a type of document, making MODX ideal for content-driven sites where the standard fields may not fit the bill.

Another area where MODX shines is in its templates: they more or less follow the Model-View-Controller (MVC) architecture pattern. MODX has always used a template system that is static: this aspect may be counterintuitive to WordPressers who are used to thinking of the templates as a dynamic part of the application (or perhaps more commonly, they don't think about the template internals at all). The MVC style approach is flexible. No matter what kind of HTML or XML you want to use to when displaying content, MODX has been one of the simplest template systems to work with, making it an easy task to adapt existing HTML/CSS wireframes into dynamic templates that can power an entire site. Each MODX document can have its own template, and each template can use whatever HTML, CSS, or JavaScript you desire, resulting in virtually unparalleled flexibility for designers. It is never a mystery figuring out which template a given page is skinned with.

Cons

Because MODX is built on a framework, it can be much more difficult for the junior PHP developer to understand its advanced programming structures.

The MODX manager is frustratingly inefficient (due to its use of ExtJS), and a MODX site may require a beefier server than its WordPress counterpart. We'll look into this in detail later on, but MODX's caching internals make its performance on the front-end quite attractive.

Perhaps the biggest drawback with MODX is its complexity, especially inside the manager. The manager does not do much handholding, and some degree of competence is required for the admins because its degree of streamlining is nowhere close to the drag-and-drop dreamland of WordPress' manager. MODX as a platform caters more to web professionals who are comfortable with building HTML or PHP sites by hand or who at least understand the underlying components of a web page.

WordPressers may be bewildered that MODX stores its templates and PHP code in the database, including the HTML for your templates. You can create MODX templates and PHP Snippets without ever adding files to the file system. This setup has left more than one noob wandering lost through the file system trying to find the "magical" directory that MODX scans for these elements, and this architecture can cause problems during migrations.

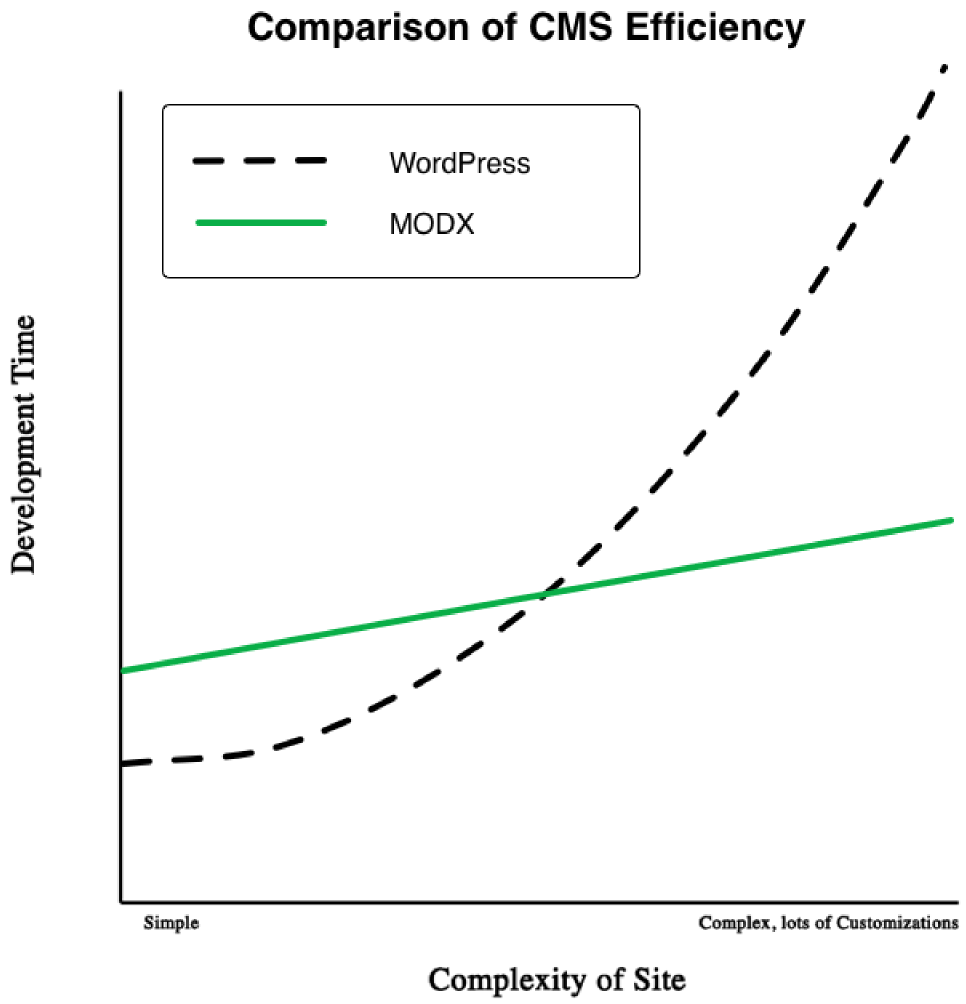
WordPress users are paradoxically confused by the simplicity of the MODX templates: instead of relying on a series of interconnected PHP files, a MODX template is self-contained, and it contains no PHP. Instead, MODX templates use simple `[[*placeholder]]` tags, which get replaced with data when a page is rendered. A MODX template cannot be the attack vector for a hack (since no PHP tags are parsed), but the downside is there is no functionality there unless you explicitly add it.

Blogging in MODX can be confusing: it's not the easy setup you get on a WordPress site. You can set up a blog in MODX, but it takes time to configure. You'll hear this a lot in this book, but it holds true in many regards, including blogging: MODX is more flexible, but it takes longer to set up.

Every open-source project suffers from a lack of documentation, and MODX is no exception: even on the official web site (http://rtfm.modx.com/) examples are sometimes difficult to find. Because MODX has a relatively small following, there are precious few books and sites that detail its usage. This is changing as MODX grows (you're holding evidence of that right now), but it can be frustrating as a developer when you are unable to find documentation. To be fair, the MODX forums are vibrant with many helpful individuals contributing to discussions (sometimes even members of the core team), but the MODX documentation is still catching up to the product, and that may leave some developers groping for solutions in the dark.

Comparing Both Systems

This chapter would not be complete if I didn't at least attempt to summarize both systems. The following informal graph represents my take on what it's like to work in both systems (based off personal experience and recorded hours).



MODX vs. WordPress : Development Time

The general lesson here is that WordPress is easier to set up, but the more customizations that are required, the harder it gets to develop and maintain. Given people's optimistic tendencies to underestimate complexity and their natural inclination towards scope creep, I have often ended up in the nasty area of that graph, burning hour after un-billable hour fixing WordPress issues that simply would not have arisen had the site been built with MODX. The flip side is that for simple sites, WordPress is quicker to work with.

There are tradeoffs to working with either system, and sometimes it's hard to determine the best course of action. I try to realistically determine how complex a site will be so I can choose the most efficient way of getting there.

Recommendations to Developers

Become a polyglot! Learn more than one system, and you will be that much wiser about application design. It will make you a better coder and more valuable to clients looking to hire someone. Each system has its own advantages, and it really behooves the developer to have experience in multiple systems. Certain problems may be more easily solved using one system over another. When drawing up specifications for a project, it can be invaluable to know off the top of your head whether a certain system is capable of performing a certain task.

If possible, I recommend that you find a couple projects (hopefully ones with some room for experimentation) and then devote yourself to finishing them using the new system. Just like living abroad and speaking a foreign language, the task requires patience, especially when the thought keeps gnawing at you: “If I were using the other system I’d be done by now.” If that thought creeps into your head (and it will), take a deep breath. It’s not the point that you know the other system better. The point is that you are learning something new, and you have the patience to wander out of your comfort zone to acquire that new knowledge and experience. Take the time to immerse yourself and stumble through the problems, even when (or especially when) it is frustrating and feels awkward. This frustration is to be expected and it is an entirely normal part of the learning process. So have patience and be confident that eventually, you will achieve a degree of fluency and competence.

Ready? Pack your bags with patience and tenacity, stamp your developer passport for foreign lands and let the adventure begin!

Complete Table of Contents

MODX and WordPress, Side by Side[1] includes 21 chapters (nearly 200 pages) of material. Here's what is included in the full version of the book:

[1]https://leanpub.com/modx-vs-wordpress