The Guide to Craft CMS Development

A practical method to develop Craft CMS projects and version control them with Git

by

Ryan Masuga

The Guide to Craft CMS Development

A practical method to develop Craft CMS projects and version them with Git.

by Ryan Masuga

First Edition: June 2017
Revision: 201706-02

eBook ISBN: 978-0-9966336-2-8

Masuga Design, LLC
Grand Rapids, MI

https://gomasuga.com/course/craft-cms-guide

Contents

Thanks

Thanks to my family. I’m so grateful for my wife and three joyful kids.

Thanks to the team at Masuga Design (past and present). The team, the team, the team.

Thanks to you, for picking up what I’m putting down.

1 Introduction

In 2015 I published The Guide to ExpressionEngine Development, where I went into great detail how we at Masuga Design set up and version control ExpressionEngine projects. As I mentioned in that book: “If you’re not constantly learning and revising how you do things, you might want to get in the habit of challenging yourself, lest you stagnate and stop growing.”

A lot of growing can happen in a couple years.

I’ve been referred to as “the EE guy” because of my involvement with ExpressionEngine over the past 11 years, from creating devot-ee.com in 2009 to publishing the ExpressionEngine book. When I’m thinking about how to solve a client problem, I “think in EE” as far as how the templates would work or how we would store the data.

Over the past few years, my friends at Pixel & Tonic have been working on their own CMS, making massive improvements and delivering features at breakneck pace. We’d always kept our eye on Craft, and had used it on a couple projects here and there. Slowly, the scales tipped. We found that when we were planning out client builds, it made just as much sense (if not more) to use Craft as it did to use ExpressionEngine 2.

ExpressionEngine is in an interesting place in 2016–2017. Version 2 was originally slated to be retired in October 2016, but that got extended to April 2017. Yet some add-ons we used on our EE 2 builds are still not available for ExpressionEngine 3. We knew we could bypass that entire situation by using Craft, where a lot of functionality that requires add-ons in ExpressionEngine is baked right into the core.

In 2016, it happened that every new build we did used Craft. It has slowly worked its way into our projects and now we’re developing with it every day. We still have a lot to learn, but every day we’re putting in that time - learning a trick here or a caveat there - that will put our Craft knowledge on par with our ExpressionEngine knowledge (hey, it’s ten years of experience with ExpressionEngine versus three and a half years with Craft, there are some hours to put in yet!)

If you purchased the Guide to ExpressionEngine Development, you’ll see similarities in the book structure here, including some overlapping ideas (and content) that I think are important enough to share all over again. I can assure you it’s not a simple copy and paste job.

Because I have such as extensive background developing for ExpressionEngine, I added a section for others who might be looking to migrate from ExpressionEngine to Craft for one reason or another. It took me a while to make the transition from channel entries loops to Craft entries loops and beyond, and I hope to save you time and effort with examples and links to numerous resources.

I'm by no means as expert as some of the people in the Craft community, but we have a pretty good system down for getting Craft sites up and running, getting them under version control, and updating them. My hope is that: 1) by sharing our process, you'll get some key takeaways to use in your own work, and 2) over the course of writing this guide, I'll learn new things as well.

[illegible]

- Ryan Masuga, June 2017

Goals of this Guide

I’ve been using content management systems almost since I was thrown into web development around 2000. From an employer’s bespoke solution to ExpressionEngine, some other systems in between, and on to Craft, I’ve been in and around them for many years. This past year we’ve done all our new projects with Craft, so my goal is to share with you how my company uses it.

For simplicity’s sake I’ll mostly be using the pronouns “I” and “me” throughout (e.g., when referring to “what I know”) but at the same time I’m also referring to “we” and “us” as a company at Masuga Design, as we’ve learned things and refined our processes collectively. I certainly can’t take credit for everything.

Share What I(we) Know

One thing I’ve learned when writing a book is that many things that seem obvious to me are not obvious to others, so I thought it was time to put down our method of working with Craft sites, as it may save a developer like yourself many headaches in the future.

We have war stories. About taking over other developers’ work. About revisiting our own work years (or even mere months) later. These battles have led us to ask smarter questions and do better development, and have led to some of the methods we use today.

Give You Enough To Go On

I’m not the most advanced Git user in the world. But that’s OK, because I don’t need to be to effectively use Craft in a Git workflow. That means you don’t have to be an expert, either. You just need enough to go on. My goal is to shed light on our relatively simple Git workflow. We don’t use submodules and don’t have more than a few branches: master, dev and topic branches. There are plenty of people out there who know more about Git than I do, and I’m in no way claiming to be an expert.

After reading this guide you should be able to competently set up a Craft site in a way the is easy to maintain, and version control it with Git.

Achieving this goal makes you a more valuable developer because the work you do for clients will be secure and redundantly backed up, and in the event you’re contracted to work with other developers who are using a Git workflow, you would be able to work with them with minimal training.

Help You Help Yourself

More skills is never a bad thing. Whatever you learn and can apply to your career will ultimately help you. I think that if you are using (or thinking of using) Craft, learning to structure a site without over-complicating it and then versioning it with Git will make you a much more valuable developer than those who don't have these skills or knowledge.

Craft is a fraction of the overall CMS market, but it has traction. There are a ton of potential clients out there who have a site on an old version of another CMS like ExpressionEngine that may not have a clear upgrade path. There are also those who are looking to have a site built with Craft from the start. *Even a fraction of the CMS market is a nearly limitless amount of work.* After implementing the ideas in this guide, you will be capable of making and supporting a better product, and those potential clients and all that work could easily be yours.

What This Guide Is Not

We're not talking about templates here!

This guide is intended to instruct on how we set up our site structure and version our sites. It won't go into detail on how to build a Craft site. There is no specific site we're setting up. Other than in the "Coming from ExpressionEngine" chapter, there are no specific template examples. There are other resources out there for that aspect of Craft development, many of which I reference in Resources at the end of the guide.

This guide should convey our experience and suggestions in *structuring your site and versioning it*.

Assumptions

One can't account for everyone or every situation when writing on a subject like this. Different operating systems, various software applications, a plethora of setups, and wildly differing ideologies on how to do anything all come into play, so I need to move forward and make a few assumptions. If I don't happen to mention your favorite Git app, I'm sure you can take the seed of what I'm talking about and apply it to your situation. Besides, this book is based on how *we* work and what has worked for us, and you can take all of it or none of it!

Your Skills

I'll assume that you're a developer that cares about their process and the end product, which is easy to do, because you're reading this guide. You don't necessarily need to have an understanding of how to work with Craft sites. Maybe you've already built and launched Craft sites before (however small), but you didn't yet do so with version control. Or maybe you have used version control, but want to confirm you're doing things "correctly." Or you've used version control and want to reassure yourself that you're using it better than we are. All of those scenarios are perfectly fine. What I'm not going to do in this guide is teach Craft 101. If you're more interested in templating, see the numerous resources at the end of this guide.

Hardware

We're an all-Mac shop. If you develop with Craft on a Windows machine, more power to you, and most of this guide should still apply to you without issue. (We only fire up a Windows machine in this office when we need to do a browser test or two. Fortunately for us, that is becoming more and more rare because we can use Browserstack from the safety of our own machines, but at least testing in Internet Explorer isn't the total nightmare it was a few years ago). Any server paths you come across in this guide will have a UNIX look to them. If you're developing Craft sites on a Windows machine, use your imagination in these cases. And may the web.config be with you.

Software

Well, you absolutely have to use Craft or you're going to quickly be up a creek with this guide. The rest of the applications you use are up to you, but below I explain what we use and why.

Craft. Version 2.6 is the most current version as I write this, and because Craft is so easy to update, we find that we keep almost all of our installs very up-to-date. The updates come fast and furious with Craft, though, so I wouldn't be surprised if we are using a much newer version by the time the first version of this guide is finished.

MAMP Pro. If you're not too technically minded and just want to get down to business putting sites together locally, MAMP Pro makes this a piece of cake. We use MAMP Pro 4, which is able to run each local site on a different version of PHP to better keep it in line with what the site may be using on a production server. Either way, this app makes getting local sites up and running very painless. There are certainly other solutions out there such as Vagrant or Valet, but we've been good with MAMP Pro.

Tower. This is the premier Git client for Mac. Easy to use, and easy to see your commits and branches. There are other GUI Git apps such as Atlassian's Sourcetree that look pretty solid, but we've used Tower since it was in beta and haven't really found any reason to switch. Any examples in this book will reference Tower, but if you're using a similar app, things should still make sense.

Sublime Text This is a very robust text editor with numerous themes. There are Twig syntax highlighting bundles out there for this editor. You might also consider using Atom. Nearly any editor you use should have decent highlighting options for Twig.

Sequel Pro. You're going to need local copies of your databases. Don't use a remote DB when developing locally (I talk more about this under Syncing the Database in the Git Chapter). Unless you're just fine with phpMyAdmin, you may want to use an app that provides a friendlier interface for working with your MySQL databases. Sequel Pro is that app. I used to use Navicat, but I've found that Sequel Pro feels more nimble and it's very easy to use. If you're using MAMP Pro version 3 or higher, Sequel Pro is bundled with it, as is something called MySQLWorkbench (which I've never used). Whether you use the

Terminal. Sometimes you have to get dirty. We try to keep our trips into Terminal to a minimum, but if you want to get a Craft site going in a few minutes like we do, you're going to have to get into the command line to "yo craftinstall" at a minimum.

2 Practical Development

“There is no code faster than no code.”
–*Kevlin Henney*

This whole chapter boils down to: don’t over-complicate, and make things easy to understand. This seems like no-brainer common sense, but we’ve inherited enough sites to know that this must be easier said than done.

Comment Your Code

“A child of five could understand this. Send someone to fetch a child of five.”
– Groucho Marx

This sounds almost didactic, but I can’t emphasize enough…you really should comment your code. *Comment your freaking code*. I mean this as far as the templates go, but the advice holds true (if not more so) for any custom plugins you build (and in addition to that, add field instructions for your clients!). I guarantee that at some point you will go back in to some dark area of a site you’ve put together and wonder how that area works, and what the hell you were thinking when you built it. I know I’ve done it: got too clever by half in building a rickety house of cards and coming back six months later when the client needs a tweak and feeling like a damn fool that I didn’t outline the reasoning behind a ridiculously complex contraption.

On a basic level, we comment every template, and here’s the basic layout of how that works: At the top of every template we include a comment block that has pertinent information about the template. By having this block at the top we can see what a template is about at a glance, such as: What URLs is this template responsible for? Is there any important info we need to know? What are all the include parameters that are available for us to use?

I can’t tell you how much time having these common comments at the top of each template has saved, never mind any comments throughout the code. Here’s a couple basic samples of what you’d see at the top of a template:

3 Security

Craft takes security seriously, from image MIME type verification, to elevated user sessions. There aren't too many changes you, as the developer, have to make to ensure your site is using security best practices. Most of the tips here involve the control panel. Here are some practices and ideas you can use:

Force SSL on Your Site

These days, there is almost no reason not to use SSL.

- SSL certificates are affordable, or free
- In 2014 Google started giving a ranking boost to secure HTTPS/SSL sites
- There isn't a performance impact
- Security is improved
- A little lock on your site looks great (like you pay attention to details...) and instills user trust to some degree

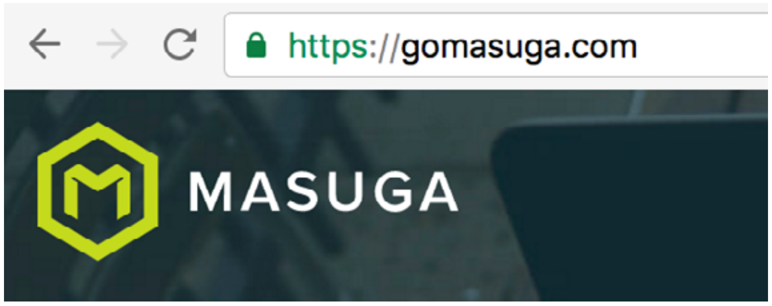


Fig. 3.1 gomasuga.com uses SSL.

To force SSL on our site, we added this rule to our .htaccess file:

```
# Force SSL.
RewriteCond %{ENV:SECURE_REDIRECT} !on [NC]
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
```

Note that the RewriteCond might be a different syntax, depending on your server. Sometimes you might see this instead:

```
RewriteCond %{HTTPS} !=on
```

We have since moved servers, so now our rewrite rule look like the following:

```
# Force SSL.
RewriteCond %{SERVER_PORT} 80
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
```

There is more info on enforcing SSL in the Craft docs here:
https://craftcms.com/support/force-ssl

Place the /craft Folder Above Webroot

This is an easy thing to do, because Craft comes this way out of the box, which you can see when you unzip a fresh download of Craft (the "`/craft`" directory is sitting right next to "`/public`"). You should keep it this way if at all possible.

Note that the robots.txt file that comes in a default install contains a rule to ignore the / craft directory from indexing as if it were in the public directory:

```
# Don't allow web crawlers to index Craft CMS
User-agent: *
Disallow: /craft/
```

You can safely remove the "Disallow" rule if you keep the /craft directory out of your public folder.

Enable CSRF Protection

Craft has built-in protection against Cross-Site Request Forgery attacks (CSRF). In Craft 3.0, this setting will be set to true by default, but on Craft 2.x sites you need to enable it. This is another setting in your general.php file

```
'enableCsrfProtection' => true,
```

There is more info in the Craft docs here: https://craftcms.com/support/csrf-protection

Change the CP Trigger

This takes about 5 seconds to do. Ideally use something difficult to guess, or something that makes sense to you or your client. This is easily done by updating one configuration variable in your general.php file:

```
'cpTrigger' => 'p13dp1p3r',
```

Per the docs, updating the cpTrigger changes “The URI segment Craft should look for when determining if the current request should route to the CP rather than the front-end website.” Yes, this is merely “security by obscurity” but why not do this if the time investment is practically nothing?

Strong Passwords for Admins

This probably goes without saying, but your site security is only as strong as your weakest Admin account credentials. You’ll be in for a real party if your cpTrigger is set to the default of “admin” and your weakest admin credential is admin/password123. Make sure your admins use strong passwords, and only give full admin access to those people that really need it.

Move the Control Panel to a Subdomain

```
‘baseCpUrl’ => ‘http://gilfoyle.yourcraftsite.com/’,
```

Theoretically, you could have the control panel at an entirely different domain this way, too (on the same server). I have yet to test this in practice, but it should make for a fun experiment. Keep in mind that if you use this method and you’re planning to run your site over SSL, you’ll likely need a wildcard (or multi-domain) SSL certificate to secure everything.

Change the phpSessionName Variable

Another setting located in general.php, this also hides the fact that your site is using Craft. The default phpSessionName is “CraftSessionId” but you can set this to something else, In our case, we went with “MasugaSession.”

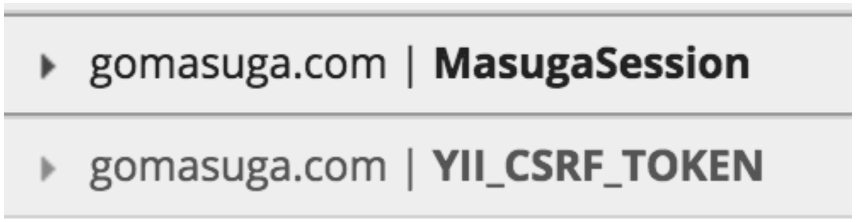

gomasuga.com | MasugaSession
gomasuga.com | YII_CSRF_TOKEN

Fig. 3.2 Changing the phpSessionName Variable

Remove the X-Powered-By Header

This is yet another config setting in general.php. It prevents Craft from identifying itself to tools like BuiltWith and Wappalyzer.

```
‘sendPoweredByHeader’     => false,
```

How does this make things more secure? For one, if you make it harder for someone to figure out what the site is built with, it takes them longer to get rolling with the common exploits for that system (if there are any “common” exploits).

Elevated User Sessions

As of Craft 2.6.2784 (circa May 2016), Craft comes with Elevated User Sessions to help prevent XSS vulnerabilities. If you write plugins for Craft where you’re allowing the user to do an action that needs an elevated session (e.g., changing a user’s email address or assigning admin status to a user), you can easily require an elevated session from your plugin before performing the action - which you should do!

Much of this security-related info can be found online, from these sources:

https://craftcms.com/support/securing-craft

http://craftcms.stackexchange.com/a/4449/227 (this thread contains the answer of the year from Brad Bell)

4 Setting Up Craft

This chapter covers everything up to the point we get our site into version control. First the installation, then a review of the directory structure, then moving things around to be more secure and better fit our view of how a project is set up. As mentioned back in Chapter 1, we're not going to get into anything too fancy as far as templates and front-end strategy. There are other Craft training sources out there that handle that aspect of development, and I mention those in the Resources chapter. However, we will cover some Twig later in the guide. For those of you coming over from ExpressionEngine, we'll look at some ways to do things with Twig compared to ExpressionEngine templates in the "Coming From ExpressionEngine" chapter.

I will first go through a "vanilla" setup where we manually download Craft and set it up on our local machine. Then I'll go into more advanced detail about automating the installation process.

Install Craft

I'm pretty vanilla when it comes to my local setup. I just want to get in and get working. I haven't had the need for a Vagrant virtual development environment or anything remotely complicated. In setting up a site for the purposes of this guide, we'll keep it simple. (Again, I'm making an assumption that you're using a reasonably recent version of a Mac. If that's not what you use, you'll have to use your imagination.)

For getting a sample site together so we can get into versioning it, we'll go "super native" and not bother with SEOmatic or any of the other plugins we'd normally install. We just need the most basic setup: a locally installed Craft build with a "Hello World!" template.

Installing Craft is very straightforward. We're more or less copying the official docs for this section. I cover every step that I go through. Take what you need from it, and leave the rest!

I create a directory in Sites named according to the domain I'm going to use. For this guide, I'll use "craftguide.com." This is not where I install Craft, though. This is a directory to hold anything associated with the site such as reference files, symlinks to DropBox directories, Sublime Text project files, or any other miscellaneous I want to keep organized with the project. Within the main craftguide.com directory, I'll make a "_site" folder. I prefix with the underscore to keep it at the top. This is where Craft lives. In fact, to make the "_site" directory, I download a fresh copy of Craft (there is a link right at the top of craftcms.com), unzip it, and rename it "_site."

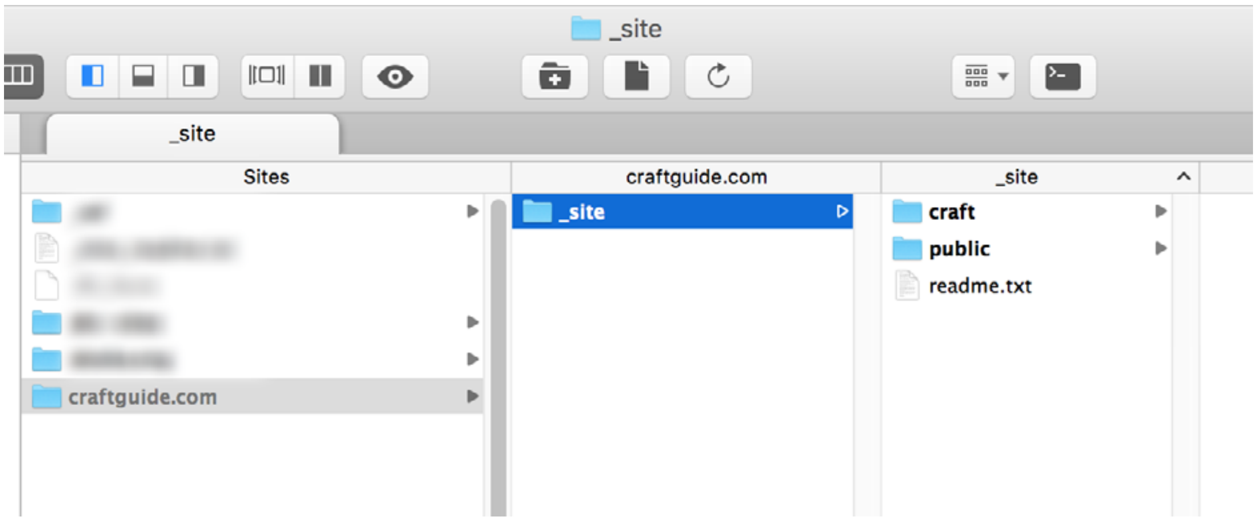


Fig. 4.1 Basic Craft Install

The top-level directory in Sites usually has the name of the actual domain (with the '.com' or the '.org' and so on), even though we use ".dev" on our sites while in development. So, for example, gomasuga.com has a local top-level directory titled "gomasuga.com" and the local dev site for it is at gomasuga.dev. Same goes for all our sites.

Note that Craft comes enabled for a secure setup out of the box. There is a `/public` directory and the `/craft` directory is not in it. "Public" is your web root, and where the site will be serving from. The Craft system is already safely out of the way.

If your eventual web host can't accommodate this setup (and we never use a host that can't, or won't allow us to alter the webroot of a site ourselves) then you may have to put the `/craft` directory into the `/public` directory, or alternatively take everything out of public and place it alongside the `/craft` directory. Finally you'd delete the `/public` directory, and everything would be living at the same level. This is less than ideal, but Craft does come with a rule in the robots.txt file by default to help search engine crawlers skip over the `/craft` directory:

```
Disallow: /craft/
```

If you're really security conscious, that's not cool either, because even though you might have taken pains to hide the fact your site is on Craft in other areas (altering default session name, or hiding the powered-by header, for example) anyone can view a robots.txt file, and now they know you're using Craft.

Let's assume you have a decent web host and it will be no problem to set up your site much like Craft comes out of the box.

5 Version Control

This guide has a goal of getting you to use Git with Craft. By now we have a basic site structure set up, so now we want to get it into a repository so that we can start keeping track of our modifications.

So let's get into why you'd want to use Git, what it is, basic commands, tips and tricks, and software. Then let's get the Craft site we set up into version control!

Why Use Git?

I've used Subversion in the past, and there are other version control systems out there such as Mercurial, but Git really is one of the easiest to use, and it has had widespread adoption. I wouldn't have taken the time to learn it and use it if it wasn't going to be relatively easy to deal with and ultimately benefit me - and this was years ago as a freelancer. Now that we're a small team working on numerous projects simultaneously, it makes even more sense. Earlier I outlined the general points that explain why using version control is a great idea, and here are some of points specific to Git:

- **Git is relatively easy to learn and use.** Some claim it's harder to learn because there are more commands than other version control systems, but for everyday use, the number of commands you need is totally manageable. Version control isn't rocket science.
- **Work at your pace - even offline.** You can commit work, view history, and make branches all without having to be connected. You have a full repository with you all the time. If you have a part of a project that is taking a long time to finish, you could commit to it for weeks before anyone else ever even knows you're working on it. As a perfect example, when I wrote this chapter I needed to make a couple small changes to gomasuga.com - but the repo where our site's code is hosted (Bitbucket) is having (unusual) connectivity issues due to a database upgrade. I can still commit my changes locally. When Bitbucket's service is back to normal, I'll "push" my work out to the main repo at that time.
- **Git is cleaner than Subversion (SVN).** Subversion litters all directories with .svn folders, but Git only uses a single .git directory at the top level of your repo.
- **Many Craft developers use Git.** By using Git you're diving into the version control system that many Craft devs already use. Many keep their add-on code in repositories at GitHub or Bitbucket or Beanstalk.

- **Branching and merging are easy.** It's easy to make a new "branch" to embark on site updates, and this helps make your development safer, because you're not working on your master branch. You're basically siloing your work into topic branches and only introducing your changes to your main codebase if and when you decide to merge it in.

There are other sources that do a great job explaining the advantages of Git and how to use it. For example, see Git Tower's Learn Version Control with Git article.

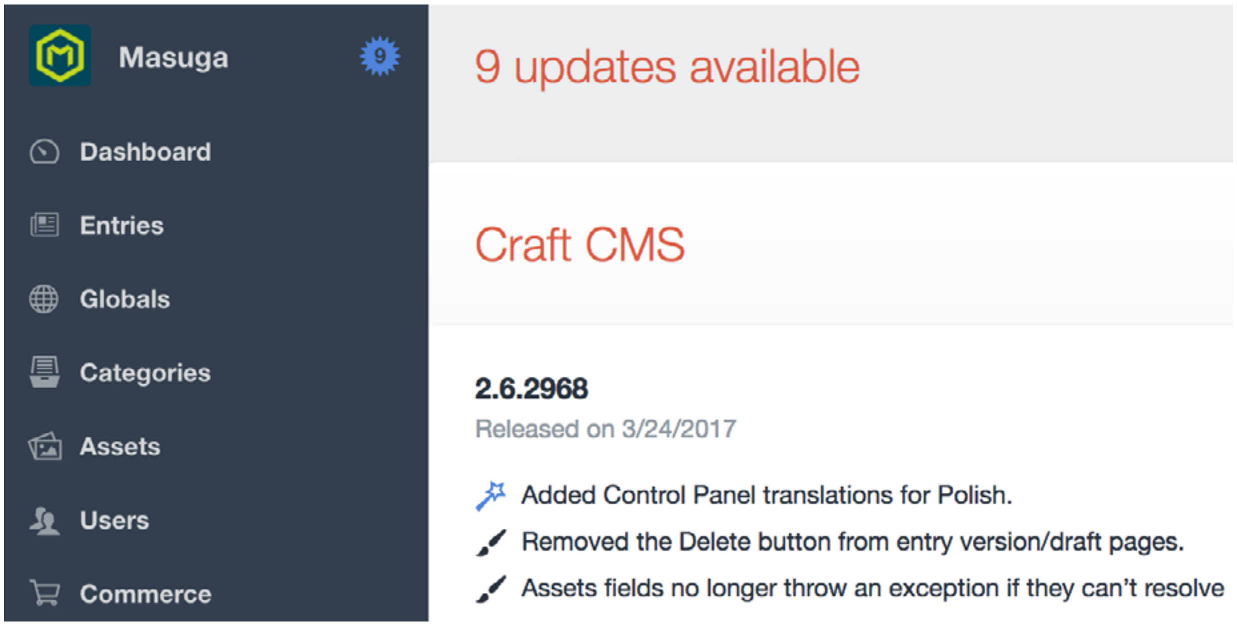
Consider this basic scenario: you inherit a Craft site from another developer (or string of developers) and your new client is counting on you to fix some bugs that they just haven't been able to solve. "Well, our first developer had feature X working. It was the second developer who added this other functionality. I think that was when the functionality we need stopped working. Or maybe it was the other way around." If there is no commit history for a site...you'll never know.

That's fun stuff, and really happened to us when we took over development of a site. Nothing was versioned, so there was no history. One of our first steps was to get that site under version control so we could document the changes we were going to make going forward.

Another example of why Git makes development better: you don't have failed experiments and other junk lying around. On another site we inherited (different CMS, but the point still stands), we were (again) tasked with optimizing it. It was obvious that the previous developer did not use version control. Entire template groups were duplicated: the "search.group" that contained 13 templates was duplicated as "seach_new.group" - with what appeared to be the same 13 templates in it, including 4 new templates. Which group is in use? Are both? Can we trust that all of the templates in either of these groups are even being used? What about the curiously titled "wtf.html" (actual template!) - was that an experiment or some sort of Easter-egg?

You too, will find that non version-controlled sites can potentially contain tons of junk like this. This is not an issue on version controlled sites, because experiments would take place on topic branches, and be nowhere near the production codebase, unless they had been merged in and the code in them was being used.

6 Updating Craft



9 updates available

Craft CMS

2.6.2968

Released on 3/24/2017

Added Control Panel translations for Polish.

Removed the Delete button from entry version/draft pages.

Assets fields no longer throw an exception if they can't resolve

Fig. 6.1 Nine updates available? Oh my! How quickly those releases come out. Fortunately it's relatively easy to update Craft when you can carve a few minutes out of your day.

This could be the shortest chapter in the history of chapters on updating content management systems. One of Craft's big selling points is the 1-click update. It's even pitched as "magical." So, you could just 1-click update whenever the control panel says you're behind, and be done with things.

One thing I personally don't like is the frequency of the Craft updates. I know that sounds ridiculous, but it seems like whenever I log into any of our Craft control panels, the site is out of date. It's almost a rule: every Craft site is always out of date! If it isn't Craft itself, it's a plugin or two that has an update every few days (looking at you, SEOmatic). I enjoy Craft quite a bit, but don't like feeling behind every two seconds.

We're actually not into the 1-click thing. Because we're using source control, it's not the biggest selling point for us. That said, the semi-manual update process we use is still easy, and we find that we're keeping our Craft sites more current and updated than we do with sites on other content management systems, where the update process is more tedious.

If you are updating Craft completely manually, note that only the '/app' folder matters. Craft has instructions for manual updates in their official docs, too. Even Craft's manual update process is easier than updating some other content management systems.

So how do you try to leverage the benefit of a 1-click update while keeping your site in version control?

Jeremy Gimbel's answer to the "What's the best practice for upgrading a versioned Craft site?" question on Stack Exchange is very close to how we roll with updates. However, we update Craft in almost the exact same way as Jérôme Coupé's later answer in that thread. Here is our five step update process:

1. Backup live database and import into our local development area
2. Ensure our repo is clean: no unstaged or uncommitted files
3. Update development site by clicking Craft's "Update" button, and test
4. Update production in the same way (Craft's "Update" button)
5. Commit and deploy

Here is more detail on each of the above steps:

Backup live database and import into our local development area

We start by backing up databases ourselves rather than relying on Craft to do it, so we don't need to have database backups enabled for the update. We ran into an issue on one site where we had a larger database, and the upgrade was hanging (this was one of the only issues we've ever run into with Craft's upgrade process so far). We found out this was due to Craft stalling on the database backup step, but fortunately it's easy to turn that off. So, if you want to skip the database backup when upgrading, make sure to set

```
'backupDbOnUpdate'        => false,
```

in your general.php file. This is one of the settings we always add to our default general.php file, as you know by now. Grab a copy of the live DB (which I usually do via Sequel Pro) for our local site so the local DB is as up to date as the live site.

7 Customization

There isn't a ton you can do here out of the box, but there are a number of third-party plugins that can help with customization of the control panel. Any plugins I mention here will likely get more detail in the Resources section under "Generally Useful Plugins." (I won't go so far as to say "must-have" plugins because that is a very subjective and project-driven subject.)

General Control Panel Customization

I think the default color scheme of the control panel is general enough to fit nearly any site. There are no garish colors that I feel the need to override, and the layout and typography is clean and makes sense. However, if you feel the need to tweak what's there, a plugin or two can help.

Control Panel CSS plugin for Craft CMS

https://github.com/lindseydiloreto/craft-cpcss

The Control Panel CSS plugin allows you to overwrite the default Craft Control Panel styles.

Control Panel JS plugin for Craft CMS

https://github.com/lindseydiloreto/craft-cpjs

A related plugin to cpcss, the cpjs plugin lets you insert additional JavaScript into the Craft Control Panel. We have not had a need for anything like this yet, but I imagine this would certainly let you customize things in ways that cpcss may not let you.

Modifying the Control Panel Sidebar

Should you feel the need to improve upon or reorganize the links in the control panel sidebar, there are a couple plugins for that:

Control Panel Nav

https://github.com/engram-design/CPNav

This plugin allows you to rename, reorder, or toggle visibility on menu items for the Control Panel. You can create your own items as well - internal or external links.

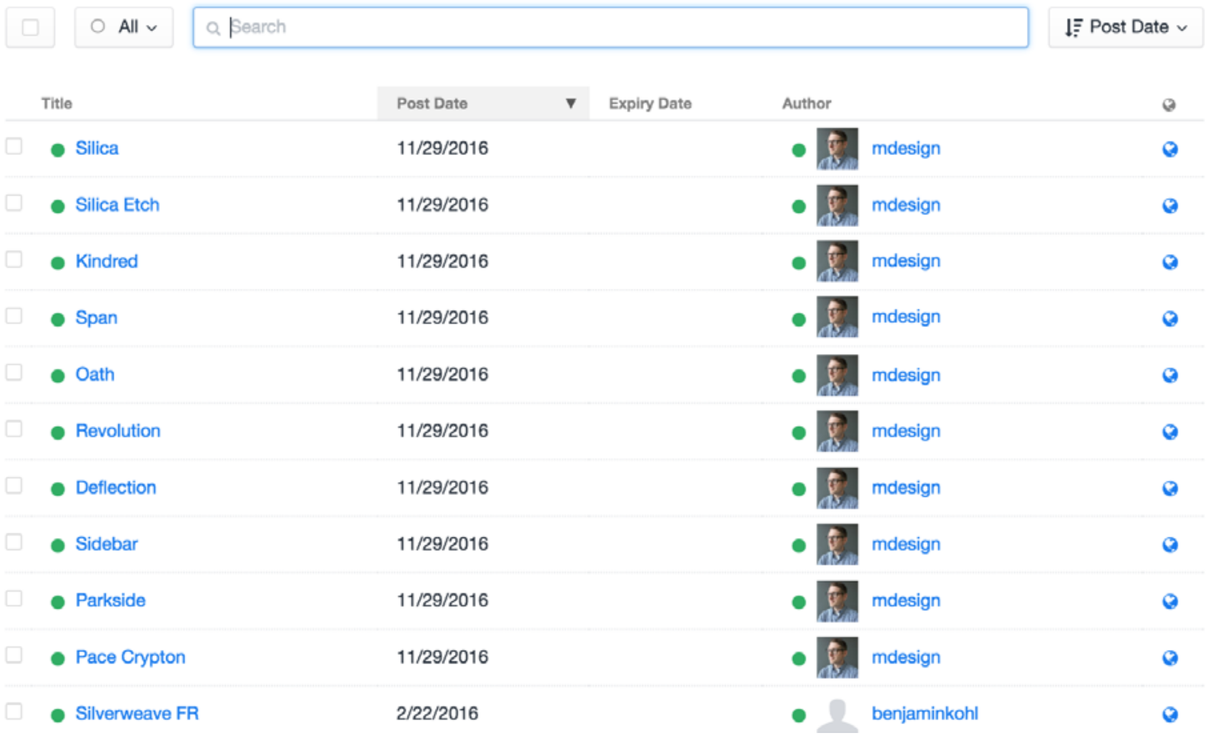
Craft Sidebar Enhancer

https://github.com/picdorsey/craft-sidebarenhancer

This is a similar option to Control Panel Nav, but doesn't allow for layouts for different groups.

Customizing Edit Lists

You can do this natively. Make sure to adjust columns for your channels and singles. If you have a channel that doesn't require Expiry Date, there's no reason your client should have to look at that empty column when they view that channel in a list. Elect to show fields that a client might need to see based on the channel.



Title	Post Date ▼	Expiry Date	Author
Silica	11/29/2016		mdesign
Silica Etch	11/29/2016		mdesign
Kindred	11/29/2016		mdesign
Span	11/29/2016		mdesign
Oath	11/29/2016		mdesign
Revolution	11/29/2016		mdesign
Deflection	11/29/2016		mdesign
Sidebar	11/29/2016		mdesign
Parkside	11/29/2016		mdesign
Pace Crypton	11/29/2016		mdesign
Silverweave FR	2/22/2016		benjaminkohl

Fig. 7.1 Our sample "Fabrics" channel with the default edit columns. Not terribly useful for anyone.

You can edit the table columns to show or hide by clicking the gear icon below the entries list. You can also rearrange their order so they make maximum sense. Once we've done that for the Fabrics channel, our edit list might look like the next Figure.

8 Optimizing Craft

This chapter deals primarily with optimization on the front end, and some of the quicker things you can implement. We're not going into detail on things like Memcache, Redis, Varnish, or services like Cloudflare - but only those things that are easy to do with Craft itself and perhaps a plugin or two.

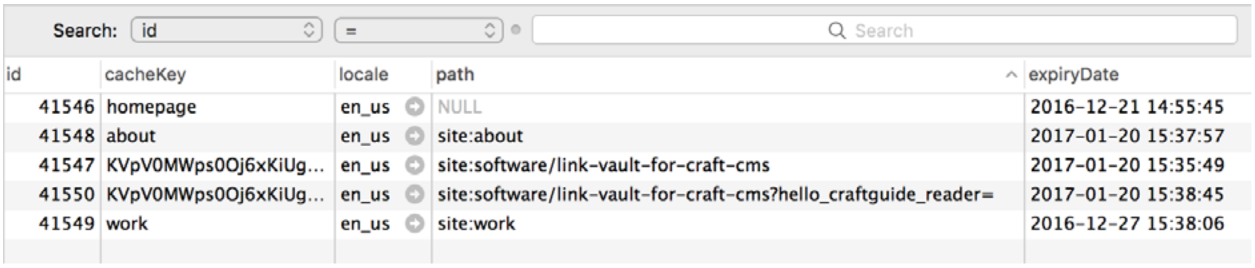
Page Caching

One of the most obvious ways to get started optimizing your templates is with Craft's built-in {% cache %} tag. Using it, you can cache whole templates, or parts of templates (fragment caching). For many sites, this built-in cache tag is enough to really help speed things up.

At the most basic, you'll use the cache tag to wrap some content so your template doesn't need to do more work than necessary to retrieve all (or part) of a template. Here's a basic (markup simplified) example:

```
{% cache %}
  {% set relatedProducts = entry.relatedProducts %}
  {% if relatedProducts|length %}
    {% for product in relatedProducts %}
      <div>
        <a href="{{ product.url }}">
        {% for image in product.productPhoto %}
          {% set transImage = craft.imager.transformImage(image, { width: 320,
             jpegQuality: 60, interlace: true }) %}
          <img src="{{ transImage.url }}" alt="{{ product.title }}">
        {% endfor %}
        <span>{{ product.title }}</span>
        </a>
      </div>
    {% endfor %}
  {% endif %}
{% endcache %}
```

That is caching a chunk of a product entry page that displays related products. We don't want to query for them and loop through them every single time someone visits the page, so we cache it. That cache will go into the template caches table with a hashed cacheKey made up by Craft (see the Figure to see what a default cacheKey looks like).



Search: id = Search

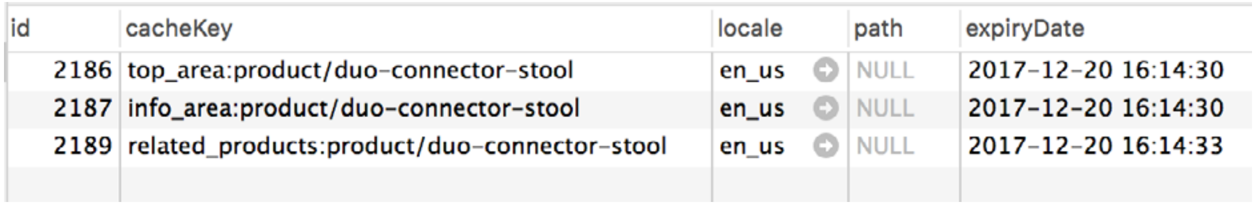
id	cacheKey	locale	path	expiryDate
41546	homepage	en_us	NULL	2016-12-21 14:55:45
41548	about	en_us	site:about	2017-01-20 15:37:57
41547	KVpV0MWps0Oj6xKiUg...	en_us	site:software/link-vault-for-craft-cms	2017-01-20 15:35:49
41550	KVpV0MWps0Oj6xKiUg...	en_us	site:software/link-vault-for-craft-cms?hello_craftguide_reader=	2017-01-20 15:38:45
41549	work	en_us	site:work	2016-12-27 15:38:06

Fig. 8.1 Hello, Craft Guide reader. In this example, note that our cache tag on the software page isn't using a key, so it's creating a cache for every full path, including those with query strings. So there are two rows for the exact same content.

Without a key, Craft will add a cache for every full path, including those with query strings. Someone could hit a page like this with nearly infinite query strings and our cache table will blow up with caches for the same page. This means we always use keys for our cache tags, so that they're easy to identify in the caches table, and so our templatecaches table doesn't get too large.

In the case of our products template example from above, we're going to have more than one cache on this page, because there are other areas of this template that would benefit from caching, too. So we'll get more descriptive with the cache tag, like this:

```
{% cache globally using key 'related_products:' ~ craft.request.path %}
  ...all the same markup as above...
{% endcache %}
{% cache globally using key 'top_area:' ~ craft.request.path %}
  ...top area stuff...
{% endcache %}
{% cache globally using key 'info_area:' ~ craft.request.path %}
  ...info stuff...
{% endcache %}
```



id	cacheKey	locale	path	expiryDate
2186	top_area:product/duo-connector-stool	en_us	NULL	2017-12-20 16:14:30
2187	info_area:product/duo-connector-stool	en_us	NULL	2017-12-20 16:14:30
2189	related_products:product/duo-connector-stool	en_us	NULL	2017-12-20 16:14:33

Fig. 8.2 See that now we have 3 different caches on a product page. They're prefixed with descriptive text, so they're easily identified.

Conclusion

In the few years it's been around, Craft has become a solid choice for discerning developers. And I do feel it is more focused on developers. If you're a designer-developer (or a straight-up designer who dabbles in CMS websites) there might be easier options for you out there than Craft, until you wrap your head around Twig for the templating.

For us, being able to whip up a new Craft site in only a couple minutes and building our library of re-usable content blocks and development patterns means that in addition to the larger, more complex projects for which we use Craft, we can use it on smaller projects as well.

I believe that introducing version control to your workflow and utilizing any (or all!) of the methods in this guide will help you be a more successful Craft developer. It has, and continues, to work for us. We're doing more Craft sites than ever, of all sizes.

Craft is still a young CMS, and the are certainly many areas in which it can and will improve. In looking at how far the developers have come in only a couple years, I think Craft CMS has a very bright future ahead, and will be a solid CMS choice for years to come.

Get Crafting

If you haven't yet downloaded Craft, go do so and give it a shot. One of the brilliant things about it is that you can test the full Pro license on your local machine (dev environment) without spending a dime on a license. I think you'll be glad you're able to add this tool to your digital toolbelt.

Get In Touch

I'd love to know how this Craft guide helped you. For that matter, I'd love to know where I might have dropped the ball, too. Did I miss an opportunity to explain something in more detail? Was something confusing? Is there a method you've found for doing something that's better than what I've written about here? I'd love to hear it. Any revisions I make to this guide that incorporate your suggestions will be credited - particularly if I test something out and decide to go with it! Email any questions or comments to me at ryan@gomasuga.com.

If this guide helped you in any measurable way, and you'd like to write a quick blurb or testimonial, that would be much appreciated and I will make sure to post it where others can see it and become aware of you and your business in the process. Of course, reviews, social shares (#craftcmsguide on Twitter), or blog posts about the Guide would be appreciated too. Make sure to let me know if you've written something up.

Thank you so much for your support! I wish you much success!

- Ryan Masuga

About the Author

Ryan Masuga owns a web development studio (gomasuga.com) and has used Craft for both client and personal projects since 2013. This is his second book about content management systems.

He studied fine art in college, receiving a BFA in painting from the University of Michigan in 1995. After living the “starving artist lifestyle” in New York City for a couple years in the late 90’s he moved back to Michigan where, by chance, he was thrown into web development. He then self-enrolled at the University of Barnes & Noble, studying web design and development books for hours, taking photos of the pages with the most useful info for later reference because he couldn’t afford to buy the books. (True story!)

Ryan enjoys reading, spending significant amounts of time with his wife and three kids, exercising, playing guitar and piano, and watching Michigan football.

He lives in West Michigan with his wife and three children.