How to write a WordPress plugin that I'll use

Posted on **June 7, 2011**

I tend to be very fastidious about the WordPress plugins that I'll install. I'll often write my own simple version of a plugin rather than install one from someone else that does a bunch of stuff I don't need. Here is my philosophy behind writing WordPress plugins, best witnessed through the plugins I've written lately, like Markdown on Save, Login Logo, Monitor Pages, and WP Help.

FEWER FEATURES AS A FEATURE

There are diminishing returns as you add features. That is, the more you add, the more likely you're adding something that X % of your plugin's users won't ever use. Stick to the basics. I'll often release a "0.1" version of my plugin with really obvious features missing. When I get a flurry of "You should add Y!" messages, that validates my assumption that Y is necessary. Start with the smallest version that gets the core job done. Iterate as needed.

CODE THE HELL OUT OF IT

The best part of starting small is that you can code the hell out of the plugin. Do it right. Make each line of code beautiful. Make sure you're using WordPress APIs properly, and while you're at it, add i18n support (WP Help 0.2 shipped with support for Bulgarian, German, Spanish, Mexican Spanish, Macedonian, Dutch, Brazilian Portuguese, and Russian!)

REDUCE UI

If you can do without UI, don't make it. Make every bit of UI prove its necessity. As an example, look at my Login Logo plugin. It has zero UI. It looks for the presence of a file named `login-logo.png` in the `wp-content` directory. The rest is "magic." It measures the image, generates appropriate CSS, and gives you an instantly and easily customized login screen. The plugin is invisible. It's completely out of sight, and out of mind. Finally, UI screens are generally

where plugin authors make security mistakes. By skipping them, you make it much more likely that your plugin is secure.

CODE IT FOR THE FUTURE

Don't use deprecated APIs. Plan features in future-forward ways. Implement it in such a way that a site that is using the plugin doesn't break if the plugin suddenly goes away. One example of this is my Markdown on Save plugin, which offers per-post Markdown formatting. First, I decided that for performance reasons, I wanted to parse Markdown then the post was updated, not on display. The obvious place to store the generated HTML was in the `post_content_filtered` column that WordPress provides (but does not use). But then I considered what would happen if someone deactivated the plugin or deleted the plugin. The code that accessed `post_content_filtered` would not work. Their blog would spit out raw Markdown. And any exports they made would export raw Markdown. What if they were exporting to WordPress.com which doesn't support Markdown? So I decided to store the *Markdown* in `post_content_filtered`, and store the generated HTML in `post_content`. When you edit a Markdown-formatted post, it swaps in the Markdown, so you can edit that. But if you deactivated the plugin, it would fall back to the HTML. So you can feel free to use this plugin and know that if one day you wake up and you hate Markdown, all you have to do is deactivate the plugin and all of your posts are back to HTML.

SECURE IT

Writing secure WordPress plugins isn't hard. It just takes awareness. Take the time to do your research and code a plugin that will be an asset to its users, not a liability.

This entry was posted in **WordPress** and tagged **coding**, **design**, **features**, **plugins**, **security**, **WordPress** by **Mark Jaquith**. Bookmark the **permalink [http://markjaquith.wordpress.com/2011/06/07/how-to-write-a-plugin-that-ill-use/]** .

40 THOUGHTS ON "HOW TO WRITE A WORDPRESS PLUGIN THAT I'LL USE"

jacob
on **June 7, 2011 at 2:09 pm** said:

Fantastic! I'm reading Wrox's "Professional WordPres Plugin Development" right now and

it reiterates all of the points you just made. I highly recommend it to anyone interested in Plugin Development.

Dan
on **June 7, 2011 at 2:37 pm** said:

Mark, I agree with you up to a point. The point being that you don't mention usable documentation and code commenting. I'd much rather have well documented code or code commenting than a fancy GUI any day.

Emil
on **June 7, 2011 at 3:13 pm** said:

Great stuff Mark and thank you for the post.

Cheers,
Emil

Neo
on **June 7, 2011 at 3:44 pm** said:

Nice post. I am building a blog myself and i think your post will help me a lot.

Danny van Kooten
on **June 7, 2011 at 5:53 pm** said:

Good points Mark. Couldn't agree more on "Fewer features as a feature", less is usually more. KISS. I don't want all kinds of heavy plugins loading if I only need a tiny piece of functionality from each of them.

Pingback: How to write a WordPress plugin that I'll use « Mark on WordPress

Shirley
on **June 7, 2011 at 6:58 pm** said:

Great tips. I'm currently writing a plugin and my main problem is holding back. There are so many features that I want to add to it… My current dilemma is whether to create one plugin with just the basic functionality and then add additional plugins for added features OR just jam-pack the plugin with all features and allow them to be enabled/disabled based on settings.

Dave
on **June 8, 2011 at 2:38 am** said:

Markdown… finally!

This could potentially solve a huge problem I'm having trying to write original sources in markdown, then sending to latex and html. Copying the html into a blog post just doesn't seem to work as well as it might.

I'm using kramdown, btw, which supersets Gruber's with (more or less) the php markdown extra.

Initial test looks looks good.

While we're on the subject of markup… sure would be nice to just go straight to haml…

j/k. Sort of.

Ryan Hellyer
on **June 8, 2011 at 5:45 am** said:

Thanks for the post Mark.

I fell into the more UI is better approach when I first started out with plugin development. People kept requesting more options, so I kept adding them!

Eventually I realised how bad an idea that is. I'm now stuck with a couple of annoyingly overly complex plugins requiring some hideous legacy compatibility to make sure they don't break on upgrades.

KISS … Keep It Simple Stupid, definitely applies for WordPress plugins (and themes).

Doug Stewart
on **June 8, 2011 at 6:30 am** said:

Shorter Jaquith: The UNIX Philosophy[1] will never go out of style.

I like it. *grin*

[1] http://en.wikipedia.org/wiki/Unix_philosophy

Scott Basgaard
on **June 8, 2011 at 6:36 am** said:

Thanks again for another great post Mark!

Your points of “Fewer features as a feature” and “Reduce UI” hit home as it’s not always something I’m best at but really admire when I am installing and using various plugins.

Like Ryan said above, these definitely apply to plugins, themes and core as well. I think we can all agree that WordPress itself has done a great job at simplifying what we really need.

Azhar Kamar
on **June 8, 2011 at 6:49 am** said:

Awesome tips!

Nikolay Bachiyski
on **June 8, 2011 at 7:08 am** said:

If I would have to write a post with the same title, I would write exactly the same.

Jennifer Avventura
on **June 8, 2011 at 7:49 am** said:

This is all gibberish to me. But thanks for the super informative post, Ill be back to learn more.

Jason Lefkowitz
on **June 8, 2011 at 1:36 pm** said:

I agree with all of these points, except for the example given in the “Reduce UI” one. Less UI is indeed better, but relying on magic filenames takes a good idea a step too far, IMHO. What happens if two plugins pick the same name for their magic file, for instance? You can minimize the risk of a collision like that by increasing the length of the filename, but it seems better to me to just let the user pick the file they want to use, regardless of what it’s named, and eliminate the risk completely…

> 
>
> **Mark Jaquith**
> on **June 8, 2011 at 3:28 pm** said:
>
> Well, that plugin has a specific audience. It’s meant for site implementors more than users. There are already plugins that let you upload a login logo image (and tweak colors and a bunch of other stuff). I wanted to make something that WordPress professionals could use for their sites and their client sites. It simplifies it for them.
>
> So no, I wouldn’t recommend that every plugin work based on magic file names. But the general principle here is that you should make all of your options earn their ticket. Here’s a better example. I wrote code in WordPress that enforces your trailing slashes preference in WordPress. i.e. http://example.com/about instead of http://example.com/about/ But there’s no additional UI for that. So how do we do it? We look at your permalink settings. If the format you’ve chosen has a trailing slash, we give you trailing slashes. If it doesn’t, we don’t. That could have been its own option, but it didn’t need to be, because we can infer it from the permalink format you give us.

Trisha Cupra
on **June 8, 2011 at 7:45 pm** said:

Totally agree. The same rules should be applied to themes – fewer features and less UI.

Themes these days are so bloated, soon they'll all explode and die.

Pingback: DPXradar 001 – Link Roundup for Online Business Owners

Muhammad Rizwan Shahzad
on **June 9, 2011 at 4:08 am** said:

good tips..

Brent Shepherd
on **June 9, 2011 at 10:05 pm** said:

And another simple one that is often overlooked: develop the plugin with 'WP_DEBUG' set to true.

Many plugins are developed with error reporting unchanged in the wp-config file.

It acts as an aid to "Code the hell out of it" and is kinder to those of us that have error reporting set and get all notices & warnings in our error logs.

Vakance
on **June 10, 2011 at 11:12 am** said:

Good points, Mark. The only thing I disagree is about UI. Sometimes UI is really necessary for plugins.

Azad
on **June 10, 2011 at 2:53 pm** said:

really nice , i might actually try to learn how to write a plugin for wordpress in summer , we have to see…

Tata Sanjaya
on **June 11, 2011 at 12:55 am** said:

really nice, i wanna to make wp plugin..

sara
on **June 11, 2011 at 1:29 pm** said:

its ok!

Pingback: WordPress Markdown on Save Plugin Review

Scot Manaher
on **June 17, 2011 at 2:57 pm** said:

I would like to see more WordPress plugin developers concentrate on efficiency of their plugins. Sure, this or that plugin might have some great features and is dazzling but if the plugin slows down page load times for the reader and admin, then are we really benefiting from plugin? I install plugins to maximize efficiency, but if they are counter productive then what is the use?

mike
on **June 21, 2011 at 11:14 am** said:

Pretty much the same path I take when developing plugins especially the KISS part. People will just shout (as you said) if they think the plugin is missing a key feature.

As for the UI part, not sure really if I'd agree or not but I do see the point. The more simple the better but then there are some users who have no idea how to use FTP software so I'd rather take that factor out and let them use a web-based file-upload utility which they're most likely more familiar with – but then, that's just me. If there's some sort of configuring needed, then create an Admin UI but again KISS.

Code for the future – After coding my first plugins I've learned from my mistakes. I know better now.

Security – nuff said.

Cheers mate! Love this post!

Maria
on **June 23, 2011 at 8:40 pm** said:

Merci pour tout ces partages, c'est vraiment super sympas de ta part :p

yuswadi
on **June 24, 2011 at 4:12 pm** said:

good read sothing of your idea..good success

edogawaconan
on **July 19, 2011 at 11:10 am** said:

for god knows why reason I can't do create blockquoted paragraph on latest update (3.2.1) – the character always gets converted to > whenever I saved it.

> 
>
> **edogawaconan**
> on **July 19, 2011 at 11:10 am** said:
>
> I mean, >

Pingback: How-To Write a Killer WordPress Plugin | ChurchMag

Benefit Brow Bar
on **August 8, 2011 at 4:39 am** said:

Smooth this silky balm on your bare skin, and your makeup will glide on like a dream. You can also wear it alone to minimize pores, smooth out texture, add moisture, and soak in nourishing vitamins C and D.

Pingback: 20 Really Handy Tutorials to help you build WordPress Plugins | Customize WordPress Blog

Pingback: A Free wordpress newsletter » 20 Really Handy Tutorials to help you build WordPress Plugins

dr dre beats studio ferrari
on **September 26, 2011 at 11:00 pm** said:

Thanks for this wonderful blog. It really aroused my interest to go over and read it again. Your article conveys to the knowledgeable minds of the readers.

integritytreecare
on **December 2, 2011 at 6:08 pm** said:

I really enjoyed this post. I am building my more advanced plugin skills, and what you say makes sense. I am also trying to find a couple of hellacious wordpress developers to teach advanced WP development. There are so many people that are pretty good with WordPress that want to master the advanced stuff, so I am getting the students together, if anyone could teach it…..
Thanks again for the post!

gazeteler
on **December 13, 2011 at 7:48 pm** said:

Awesome and good tips

ferrets as pets
on **April 3, 2012 at 1:38 pm** said:

It can be really tempting to make things more complicated than they have to be. But I'll

try to be more aware when I'm doing that and stop myself. Great advice.

symptoms of kidney infection
on **April 10, 2012 at 6:06 am** said:

Is there any preference to which programming language to use?

Comments are closed.