CUNY Academic Commons - Feature #1871

CAC Featured Content Rewrite

2012-04-30 04:19 PM - Dominic Giglio

Status:	Resolved	**Start date:**	2012-04-30
Priority name:	High	**Due date:**	
Assignee:	Dominic Giglio	**% Done:**	70%
Category name:	WordPress (misc)	**Estimated time:**	40.00 hours
Target version:	1.4		

Description

I'm creating this issue to consolidate all pending issues concerning the CAC Featured Content Widget, as well as to discuss what needs to be done to properly rewrite the plugin going forward. I hope to get the rewrite done in time for the 1.4 release.

Related issues:

Related to CUNY Academic Commons - Bug #1829: 'No Image' checkbox on Featured...	**Rejected**	**2012-04-15**
Related to CUNY Academic Commons - Feature #1559: Crop Text Length Field on F...	**Rejected**	**2012-01-20**
Related to CUNY Academic Commons - Bug #1453: Multiple problems with featured...	**Rejected**	**2011-12-16**
Related to CUNY Academic Commons - Feature #1982: i18n for CAC Featured Content	**Resolved**	**2012-07-09**
Related to CUNY Academic Commons - Feature #1983: Media Library integration w...	**Assigned**	**2012-07-09**

History

#1 - 2012-04-30 04:22 PM - Matt Gold

Thanks, Dom. Check out the "related issues" feature to associate other tickets with this one.

#2 - 2012-04-30 05:44 PM - Dominic Giglio

Boone, Ray, Matt,

When I sat down to start looking at the features that needed to be fixed and the state of the code, I decided that the time I would need to spend patching could all be put into a rewrite which would leave the plugin in a much better position going forward. So that's what I've done.

I'll first give a brief overview of how the new plugin is structured and then finish up with a summary of what has been implemented and what still needs to be done going forward. Starting my plugin research from the WordPress Widgets API Codex page (http://codex.wordpress.org/Widgets_API), I followed the link to this Github gist: https://gist.github.com/1229641. It outlines three different examples of a boilerplate widget. I chose to go with the first because it had the most code and comments. I figured that would be easier for me since my only other experience writing a plugin was issue #1457 (clear status updates).

I started my new branch by renaming anything in the existing plugin that would cause a conflict with the names in my new plugin by just appending "old" to them. I needed to be able to run them both at the same time to compare features as I ported them from old to new. The design of the new plugin is actually quite simple and has taken on a kind of MVC structure. The main plugin file contains the class that extends WP_Widget as well as the code necessary to load the widget. The CAC_Featured_Content_Widget class starts with a class member variable called $widget which is a multi-dimensional array of all the settings needed to build the admin interface. The constructor creates an instance of the helper class and then calls the parent constructor after a little custom initialization provided by the helper class. The widget() method uses the 'view' element of the $widget array to decide if it should require an external view file (from the views sub-directory) otherwise it calls the html() method. The html() method is convenient for smaller plugins and allows you to keep all your plugin code in a single file. The form() method loops over the elements of the $widget class variable and outputs anything it finds. This method should never need to be touched, any features that need to be added/updated/removed should be edited in the $widget array. And finally the update() method simply updates the old values in the DB with the new ones entered on the widget admin page.

The helper class is responsible for gathering info for the plugin to display and the views sub-directory handles all the output to the front end of the site. I'm really looking forward to feedback on the helper class. It is pretty simple as of right now, but is really there for the image manipulation methods I have a feeling we'll need going forward. Because it is instantiated in the main widget constructor we can use it throughout the plugin to help keep it as DRY as possible.

I am most excited by the implementation of the views. You can create a new view for the plugin at anytime by simply creating a file in the views sub-directory and naming it cac-featured-{featured_content_type}.php and as long as there is an associated featured content type in the $widget class variable it will magically be loaded when enabled in the admin.

So cac-featured-content.php is like the controller, cac-featured-content-helper.php is the model and the views are - well - the views. :-)

What's been done:
- **refactor and modularize code and support files** (this is where I spent most of the time)
- create the initial views for each featured content type
- abstract out the JavaScript that was being echoed by the old version. wp_enqueue_script() now loads the JS only on widgets.php

To Do:
- **implement image handling and re-sizing**
- update JS for admin page (it's currently commented out because it would cause conflicts if old and new are loaded at the same time)
- create resource content type and associated view
- create the view for featured blogs
- create the view for featured posts
- write new README...
- ...and anything else I've missed

I've marked this issue as 40% done because I am now at the point where I am beginning to hit a wall as far as my own skills go. I am going to need advice and help with implementing the blog and post content types because they are so image heavy. I am hesitant to just copy the existing plugin's code because we have an opportunity here to untangle all of that spaghetti, after all it is the reason for this entire rewrite I don't have a lot of experience with programmatically working with images inside of WordPress. But the plugin works, you can enable it, change its settings in the admin and it'll display on the front end. Boone: I hope you feel that this new architecture accomplishes what you expressed to me when we first discussed the plugin's history and potential future.

I look forward to ANY AND ALL feedback from all of you. Please don't hold back, I am a big boy - criticism will only make me better. :-)

I've pushed my dev branch up to the repo:

https://github.com/castiron/cac/tree/featured-content

#3 - 2012-05-01 01:13 PM - Boone Gorges

Hi Dom. Overall, the changes you've made are really solid. I'm thrilled that you've managed to reduce the number of lines of code, and make it far, far more readable. Your strategy of including views is elegant and reminiscent of the way WP loads templates.

I did a little bit of cleanup https://github.com/castiron/cac/commit/6c9d0d4306eca8fb5ef053446b1a19329afded59, but mostly just to get it up and running (there were a few places where you were referencing variables before they'd been defined - enable WP_DEBUG mode on your local installation to catch these sorts of issues early).

One of the biggest problems with the original plugin is that it used a bunch of different techniques for displaying different kinds of content. Eg, for groups, it uses a bp_has_groups() template loop, while for blog posts it does something totally different. In theory, I like the fact that Michael was trying to use native template functions to do his work. But that's exactly how we ended up with such a mess. So, instead, what I'm thinking is something like this: use the exact same template for nearly all of the markup, and populate the dynamic content with variables defined on the fly. In pseudo-code:

```
switch ( $content_type ) {
  case 'member' :
    $name = bp_core_get_user_displayname( $params['featured_member'] );
    $link = bp_core_get_user_domain( $params['featured_member'] );
    break;

  case 'group' :
    $group = groups_get_group( array( 'group_id'  => $params['featured_group'] ) );
    $name = $group->name;
    $link = bp_get_group_permalink( $group );
    break;

  // etc
}
?>

<h3><a href="<?php echo esc_attr( $link) ?>"><?php echo esc_html( $name ) ?></a></h3>
<?php /* etc */ ?>
```

At least, that was my idea. It centralizes all the markup, which means easy upkeep. On the other hand, it reduces flexibility - markup remains the same from type to type, and can't easily be changed. So, I'm not really sold either way. You might combine the two strategies defining the template variables outside the template files (say, as methods on a helper class), providing a main View file cac-featured-core.php, and then falling back on that core file when no more specific view is available. This is how WP template hierarchy works, and kinda brings the best of both worlds.

I'd be interested to hear what you and Ray think about that idea.

As for the rest of your todos/questions: The general technique used by the old plugin to get Post and Blog content (including image detection) is pretty good, so you should use it as a guide. Here's a schematic for Featured Posts:

1. Use switch_to_blog() to go to the blog in question and get the post data (just dump it into a variable like $post = get_post($post_id) and then restore_current_blog())
2. Scrape $post->post_content for an image
3. Parse $post into the proper pieces for display

For featured blogs, you'll need to switch to the blog to loop through posts until you find one that has an inline image.

Make sure that these auto-generated images (from inside of posts - and this can go for avatars too) are **fallbacks**. If the widget owner has specified a specific image for the widget, you can skip all of the image-scraping logic.

A couple small things:

- CAC_Featured_Content_Helper::get_all_groups(), you should be using $bp->groups->table_name instead of the literal wp_bp_groups. I tried swapping it, but the problem is that the widget init happens to early, so that $bp->groups->table_name is empty. This can probably be fixed by not doing the groups query that early in the load order (ie at widgets_init). I'll leave that as an exercise :)
- Loading all members + all groups + all blogs into the widget at runtime works OK on the Commons, but it won't scale. It's low priority, but in the long term, it might make sense to explore other options (maybe ajax autosuggestion or something like that).
- Whenever you output user-provided content to the page (either as the 'value' of form elements or when building a view's markup), always use WP's escaping functions, to reduce XSS vulnerabilities. esc_html(), esc Check out https://codex.wordpress.org/Data_Validation

Thanks for your hard work on this so far!

#4 - 2012-05-01 11:10 PM - Dominic Giglio

Awesome! this is exactly the kind of feedback I was hoping for.

Give me a day or so to go over all these points while I have the code open in front of me.

Will be back ASAP with questions and new commits!! :-)

#5 - 2012-05-06 03:10 PM - Dominic Giglio

So I think the first thing I need to cover is your comments about the views. I see the potential benefits in centralizing the views into a single file so we can define variables once and then output the appropriate markup for the content type chosen by the admin. However, I think this will complicate things going forward. The content types are sufficiently different to result in an incredibly long and complex view file (IMO).I'm curious about why you think the template variables should be defined separately from the view markup? There aren't many variables and I know some of them are redundant, but keeping each view self contained allows us to develop each view separately and always be sure that we aren't affecting other views. Do you see a potential problem with the current setup?

Second, I see how the pre-populated select lists for featured blog, post, group and member would be a problem on a Multisite install with like tens of thousands of blogs/groups or hundreds of thousands of members. Should I just go back to the text inputs for now and once we have an initial release of a working rewrite we can discuss the AJAX auto-suggestions?

#6 - 2012-05-08 09:55 AM - Boone Gorges

> The content types are sufficiently different to result in an incredibly long and complex view file

Maybe. Seems to me that they all have the form

```
<div>
  <? avatar ?>
  <? headline link to content ?>
  <? description text ?>
  <? readmore link ?>
</div>
```

How those items get populated will, indeed, be very different in each case. But the semantic markup structure is the same in each case. So you would create some template functions to populate the content (or methods on a class), which is where the logic would reside.

That said, I don't feel strongly about this. I like your Views infrastructure, and I'm happy to move forward with it. To the extent that there's some redundancy built in, at least the structure is clear, so that when it comes time to change the markup, it'll be obvious where and how to change it.

> Should I just go back to the text inputs for now and once we have an initial release of a working rewrite we can discuss the AJAX auto-suggestions

Since you've already written the dropdowns, etc, I suggest that you show them conditionally.

```
  if ( function_exists( 'wp_is_large_network' ) && wp_is_large_network() ) {
    // show textboxes
  } else {
    // show dropdowns
  }
```

We can then add autocomplete later, and maybe deprecate the dropdowns at that point.

#7 - 2012-05-24 11:18 PM - Dominic Giglio

Boone,

Since we'll be seeing each other tomorrow, I'm not gonna write up a huge list of everything I've done since school ended. I just wanted you to know that I've pushed a bunch of changes to origin/featured-content if you'd like to take a look tonight. If not I will go over all of my latest changes with you

tomorrow.

Hope all is well.

D

#8 - 2012-05-25 08:04 AM - Boone Gorges

Thanks, Dom! I've just had a glance at the changesets and I have a couple of comments (and may have more after actually running it :)). Looks great overall though. Let's talk about it later today.

#9 - 2012-06-03 10:07 PM - Dominic Giglio

- % Done changed from 40 to 60

Boone, Ray, Matt,

I've pushed my latest plugin code up to the featured-content branch. Boone, the plugin has now gotten to a point where I think it's ready for Matt to try it out on cdev. If you find anything wrong while looking through the code, or while running it in your local environment, let me know and I'll fix it before you send it to cdev. Here's a quick refresher on how the redesign is structured:

1.) cac-featured-content.php loads the plugin, handles everything in the admin section and starts the rendering process for the front end.
2.) cac-featured-controller.php is responsible for gathering all the info needed in the views. I renamed it to better express its job as a "controller" operating between cac-featured-content.php (the "model") and the various "view" files: views/cac-featured-*.php. cac-featured-controller.php uses the static class in cac-featured-helper.php and a small view class (defined at the beginning of cac-featured-content.php) to setup all the view variables. This is the class we discussed at the last meeting used by the controller to help keep the plugin's variables out of the global namespace.
3.) the view files all use a similar structure to output the information gathered in cac-feaured-controller.php and output them to the page. I've re-factored almost all logic out of the view files.

The view files are far from done. I am still learning a lot about the internals of BuddyPress; if you see areas where I can improve that code please point me in the right direction. While trying to get the appropriate info in the controller and views I look through the BuddyPress core and try to find functions/methods that provide the values I need. But I'm not always sure they are the most appropriate for the job.

I realize that I still need to put the escaping functions into the views, I haven't gotten to that yet cause I wanted to make sure I've got the logic down before I get into escaping the values that the logic generates. One thing that I think is really missing is more robust error checking. The code right now kind of assumes the admin is going to put in everything exactly the way it expects it to be.

So going forward from here I will be working on error checking and fallbacks as well as starting the autocomplete code. I've noticed a few things in the admin JS that I need to tweak anyways. Some of the logic based around checking for the dummy "__i__" widget doesn't appear to be working the way it's supposed to. I'll write back when I dig into that a little further. I figure if the autocomplete is in JS then I'll work on all JS at the same time.

Boone, I also managed to figure out a way to reinitialize the widget JS when a new widget gets dropped in the admin section (we talked about this at the meeting) you can see how I did it here: https://github.com/castiron/cac/commit/3f2e0b4de64325ed2f5eced78abf314d201edbb2

Let me know what you think of the latest code.

Latest commits: https://github.com/castiron/cac/commits/featured-content

D

#10 - 2012-06-06 11:30 AM - Boone Gorges

DOM

I've had a chance to spend some time with the latest changes, and generally things are looking great. The new approach with cac-featured-controller.php is exactly what I had in mind when we chatted a few weeks ago. The admin is looking clean and pretty - far more consistent and less confusing than what was there before.

I've pushed a couple changes to the featured-content branch. Briefly:
- Fixed a couple of BP template functions to get URLs and avatars right
- Added rudimentary autocomplete for groups in the admin. This should give you an idea of how to implement. It's pretty messy (cobbled together from some other stuff I had for a different project) but it should give you an idea of how to do it.

Some issues and suggestions:
- the init() JS method isn't being properly called on page load. You can tell because the autocomplete is not getting set up until you change content type.
- As you note, there's no error checking. I'd suggest doing some checks in the widget's update() method. Probably the most important is to check to see whether the post/group/blog/member data entered actually points to an existing object. Note: This will be less important when you have autocomplete in place, and even now I don't think it's super important, so move it to the bottom of the priority list.
- For the sake of consistency and readability, I suggest moving all template functions out of the views - the only place where Views should get their data is out of the $cfcw_view object. I'm thinking eg of cac-featured-member.php, where you call bp_core_get_userlink(). This is perhaps a matter of taste, but it seems to me like it would be best to do all setting of values in the controller file.

MATT

The latest version of the plugin is available for your testing on cdev. Please have a look at it, and review whether it's capturing the functionality you want. Known issue: autocomplete is only in place for groups, and it will only work after you've switched the content type on pageload.

#11 - 2012-06-06 11:43 AM - Dominic Giglio

I'm so happy that you like the work I've done so far.

Thank you for adding some autocomplete examples to get me going. That's gonna help tremendously. The JS is my current focus. I noticed there were some issues when I went in to add some code that hides image_url, image_width and image_height inputs if the user un-checks the "show images" checkbox. For some reason things are getting executed twice. I like the idea of error checking in update(), will put that at the bottom of my list. I agree with your template suggestion, I will move those calls to the controller.

I'm about to head up to the dentist, will be working on all this great stuff when I get home.

#12 - 2012-06-06 11:48 AM - Boone Gorges

> Thank you for adding some autocomplete examples to get me going.

No problem. I meant to mention that I dropped into a separate class, in a different file, but you are welcome to arrange things however you'd prefer to have them organized.

#13 - 2012-06-06 09:51 PM - Dominic Giglio

Boone,

I've pushed up a single commit that fixes the variables between controller and view files. I've removed all function calls from the members view, and moved them to the controller as you suggested. I also slightly altered the blog description, group permalink, member user link and last activity class variables. I think they make a little more sense when looking at a view file if all "properties" are consolidated under their respective "types."

What I mean is, instead of $cfcw_view->permalink I'm now using $cfcw_view->group->permalink

You can see all those changes here:

https://github.com/castiron/cac/commit/8055bf55874f7df67ac0a6d1195d8798ab5696ef

I am now moving on to get the JavaScript to do what it's supposed to; I don't want to start studying your autocomplete code until I know the JS works. But from what I've seen so far in your code I think autocomplete will be a lot of fun to implement for the rest of the content types. Thanks again for getting me started.

And **happy anniversary**!!! (your tweet came through right as I was sending this) :-)

#14 - 2012-06-07 07:09 AM - Boone Gorges

The changes look good, Dom - thanks. I'll look forward to seeing what you've come up with as you iron out some of the JS wrinkles.

And thanks for the anniversary wishes!

#15 - 2012-06-07 11:41 PM - Dominic Giglio

Boone,

OK, so I almost completely rewrote the admin JS implementation. There were quite a few problems with the old code. These mostly stemmed from the inherent difficulty in working with multiple widgets that interact with WordPress primarily through ajax. I spent some time in the jQuery docs re-learning how to properly work with events. I settled on the new .on() function to attach (delegate) events to the widgets sidebar that will react to ajax and type change events. This is important since we don't know how many widgets a user will create and the CRUD actions performed on widgets all take place through ajax, so delegating events allows the plugin to handle any number of widgets in almost any configuration. Basically, the old code was doing everything in the init() function, including attaching events to various DOM objects. Since the code was calling init() all over the place these events were stacking up on top of each other. I added some console.log()'s and started clicking submit in one of the widgets over and over again, and web inspector got to the point where it just said "there are 1500 messages that have not been displayed!" LOL

So I've moved all that event based code into an expanded document.ready(). The cacFeaturedContent object is now only responsible for housing the functions that control hiding and showing inputs. As far as the bug where your autocomplete didn't initialize until a select box change goes, that was because it was in the typeChange() function, so it literally couldn't be initialized until a user changed types. That's been fixed. I want to do a little more with hiding and showing image related inputs when the "display images" checkbox is checked or unchecked but I just haven't gotten to it yet. I started the code but it was acting really weird. My next commit should have that functionality. A nice benefit of this rewrite is that there is no longer a need to echo out the <script> element in the form() method to call cacFeaturedContent.init() after a submit.

Here's the commit: https://github.com/castiron/cac/commit/a8dd3718fbc4b7a64f94fda7ba696e653cdc8fbd

And the actual code: https://github.com/castiron/cac/blob/featured-content/wp-content/plugins/cac-featured-content/js/cac_featured_admin.js

Let me know if you find any problems or errors after you get a chance to look at it, and after you take care of the 16 to-do's from Tues and Wed!! :-)

I'll be working on the checkbox and autocomplete stuff for the rest of this week.

#16 - 2012-06-11 12:04 PM - Boone Gorges

These changes look great, Dom. I'm going to merge what you've got so far into the master branch, as it's now clear that this is working sufficiently well to make the next release, and because it'll make it easier to test on cdev alongside of some other new features. You can continue to work on your own branch if you want to and we'll just merge as you work.

Matt, whenever you get a chance to check it out on cdev, that'd be great. Thanks.

#17 - 2012-06-18 03:44 PM - Dominic Giglio

Boone,

I think I've got a handle on how the PHP and JS work together to implement the autocomplete functionality. I've started the featured member code and pushed up some rough stuff to Github:

https://github.com/castiron/cac/commit/6efa8ce0d2ef205ba5b5ed59c3fa1b4bae282692

Could you take a look at that and tell me if that's how you'd like me to keep adding in the other content types? I'm really wondering if you want it all in the autocomplete php file or if you think it should be separated into individual files. That code was written on a plane so it's far from complete, I just needed to get something showing up in the dropdown autocomplete list. Will be refactoring and cleaning it up tomorrow.

#18 - 2012-06-18 04:38 PM - Boone Gorges

Hi Dom - Yes, this looks like the right approach. Make sure that you actually feed some search terms into get_users(), of course :)

#19 - 2012-06-25 01:18 AM - Dominic Giglio

I just pushed up some code for the blogs autocomplete. I'll be getting posts in tomorrow and then moving on to "tightening things up." Working on attr escaping and error checking. Let me know what you think about the blog implementation, I'm know $wpdb should be used sparingly, but I think in this case it was the right way to go. If you know of another easier/safer way point me in that direction and I'll change it.

Commit: https://github.com/castiron/cac/commit/fa58a75de50c5835815baea2e69604ee36eb1457

#20 - 2012-06-28 08:09 PM - Dominic Giglio

- % Done changed from 60 to 70

Boone,

Autocomplete done
error checking added
escaping == added (but I need to talk with you again about proper escaping) Take a look and let me know if I've done it "correctly."

I moved the requiring of the autocomplete class to the constructor, but now that I've done that I'm having second thoughts. I think those classes will now be loaded when they're not always needed. Let me know if you think that was a mistake and I'll move the requiring back to 'admin_init'.

At this point I think the plugin is really solid. It's definitely ready for testing so we can seek out any hidden bugs or missing functionality. I'm going to start looking into the other issues that have been stuck in my Redmine list since I started this re-write. I also think it's important for me to stop looking at this code for a little bit so I can come back to it with fresh eyes after it's been put through some real world testing.

Commits: https://github.com/castiron/cac/commits/featured-content

#21 - 2012-06-29 09:29 AM - Boone Gorges

Hi Dom.

> At this point I think the plugin is really solid

Agreed. I think this is pretty much ready to go for now. There are a few small issues that are not blockers for this first release, but that I want to get into this ticket so they aren't forgotten about.

- Blogs autocomplete, as you're currently written it, will only work for subdomain installs. The handler should probably have a conditional in it: if (is_subdomain_install()) { do the stuff that you've already written } else { do some other stuff for subdirectory installs }. Not crucial at the moment. Related: $wpdb is fine here. There's no better way to do it in WP core.
- Escaping all looks good.
- You're correct that you're loading cac-featured-autocomplete.php too generously. At a minimum, only load when is_admin(). Better: load when is_admin() and current_user_can('manage_options'). Better still: all of the above + only when looking at widgets.php. (Actually, you only really need to check the last point; if you're viewing widgets.php, it means that is_admin() && current_user_can('manage_options'), by definition.)

I'm going to merge these latest changes into the master branch again, so that folks will have a chance to look on cdev.

#22 - 2012-06-29 02:25 PM - Dominic Giglio

I didn't even think about the differences between sub-domain/directory installs. That definitely needs to be addressed.

I'm going to take a run at Issue #1401 and Issue #1249, they seem fairly easy and I'm tired of staring at them on "My Page" here on Redmine. :-)

In addition to the features you mentioned, I still want to work on the JS a little more. I think the image related inputs should toggle (show/hide) when the Display Images checkbox changes state. I would also like to disable the post slug input until the user enters a valid blog name. But these things can definitely wait. I really want to see what (and how many) bugs show up once our users start poking at it in the wild.

I'm also gonna work on a new Readme so when this new version gets pushed to Github, visitors get a nice intro to how it works and how to use it.

#23 - 2012-07-02 04:10 PM - Matt Gold

Great work on this, Dom. I've tested on CDev with a featured post, member, blog, and group, and all is well. Some nice, slick stuff in here -- autocomplete, ability to choose a specific photo, etc.

Just a few small suggestions:

-- If we're going to be packaging and releasing this (as I think we should), you might include instructions under the widget fields for blog URL and post slug that explain that no http:// is needed (if that's still the case).

-- When trying to feature a post, one of the autocomplete fields asks for "Blog Name," but if I type "CUNY Pie," nothing comes up from the autocomplete. I think that this is because what's actually being completed is the blog URL, not the name.

-- Have we lost the ability to feature a wiki page? Apologies if I've missed discussion about that.

-- I'd reorder the elements slightly so that Content type appears closer to the top. Maybe something like this:

Widget Title:

Featured Content Type:

Enter featured member username/blog title/etc.:

Crop Length (characters):

Read More Label:

`Display Images`

Image URL:

Image Width:

Image Height:

-- Maybe for the next iteration of the plugin, I'd love to have the ability to enter a custom text description. For instance, if a group describes itself in ways such that the first ten words of the description aren't very focused, it would be great to replace them with a better description. This would also be good for groups that don't have descriptions -- the description we see for the CUNY Pie blog is "building communities since 2009." So, an optional text box would be good if possible.

I'd suggest that we think about which of these suggestions are doable immediately, and implementing them, and then saving the rest for a second iteration.

Thanks for your work on this, Dom. I'm excited to have a working widget almost ready!

#24 - 2012-07-02 05:21 PM - Dominic Giglio

-- If we're going to be packaging and releasing this (as I think we should), you might include instructions under the widget fields for blog URL and post slug that explain that no http:// is needed (if that's still the case).

I planned on putting short instructions under important inputs but wanted to get the plugin tested first. Will probably add them as a final (or close to final step)

-- When trying to feature a post, one of the autocomplete fields asks for "Blog Name," but if I type "CUNY Pie," nothing comes up from the autocomplete. I think that this is because what's actually being completed is the blog URL, not the name.

I thought I saw a little bug like this, but when I tried to reproduce I couldn't. I will investigate.

-- Have we lost the ability to feature a wiki page? Apologies if I've missed discussion about that.

Are wiki pages a "resource?" I left the resource featured content type out so I could focus on the four core types. I figured once the new plugin structure was in place it would be trivial to add resources back in. That was the reason for the re-write after all. Could you (or Boone) give me a little guidance here? What is a "resource?"

-- I'd reorder the elements slightly so that Content type appears closer to the top

I like your suggested layout. I'll reorder them and probably add some help instructions below relevant inputs at the same time. Should be in the next release for testing.

I also like your idea about a custom text description to override the one provided by the chosen content type. This might be a little tricky, but I'll try and get something into the next version for testing.

I'm so glad that the first testing version is already providing the features you want and need!

#25 - 2012-07-03 02:59 PM - Boone Gorges

> Could you (or Boone) give me a little guidance here? What is a "resource?"

For our purposes, "resource" == "wiki page". It was named "resource" for purposes of the release plugin. Basically, Resource is a pared down version of the other content types: image fields, title field, URL field. No autocomplete or validation necessary.

IMO it's not necessary to have Resource available for 1.4. I definitely do not want to hold up the release for it. Dom, if it's easy to add in (say, an hour or two), go ahead and do it now; otherwise stick it somewhere in the middle of your priority list (and maybe open a separate ticket for it, since the tasks described by this ticket are just about done).

#26 - 2012-07-04 12:28 AM - Matt Gold

I agree that this shouldn't hold up 1.4, but I would like us to add wiki/resource featuring to the widget. Many thanks.

#27 - 2012-07-04 05:56 PM - Dominic Giglio

I agree, I always planned on putting resources back in, just wanted to focus on the other featured content first.

#28 - 2012-07-09 10:34 AM - Dominic Giglio

Boone, Matt,

Here's a run down of the major commits I added to the Featured Content Plugin this past week:

1.) Reordered the inputs in the admin widget per Matt's suggestion
https://github.com/castiron/cac/commit/b9088130a244ad35151d61b0c8113bfac0492b0c

2.) Because the post input field looks for posts from the blog entered into the blog input field, I beefed up the auto-complete handler to make sure a nice message is returned to the user when a blog was not found.
https://github.com/castiron/cac/commit/540aa5e8faa210882626068c7140c1b4850a2004

3.) I added new JS that disables/enables the post input field based on the value in the blog input field. This should help cut down on user errors if they try to enter a post name before entering a valid blog name (becaue it prevents it).
https://github.com/castiron/cac/commit/c9ffdd9859fe4e2076fff0097cbac481a4c7a9ed

4.) Added new JS that shows/hides the image related inputs when the Display Images checkbox changes state
https://github.com/castiron/cac/commit/0d72987a869ff61bdb6ceb078c8ceb98ea5fa958

5.) Added featured resource type and custom description functionality
https://github.com/castiron/cac/commit/293a27bf6e340b0e02f214567d40e23e331b3332

This now covers *almost* all of the functionality provided by the original plugin. I still need to write a new readme but that should be done this week. Boone, I haven't forgotten about adding the code to check for subdomain/subdirectory installs, will do that while working on the readme.

Matt, I also changed the label for the blog input. You are correct, it searches the DB by "domain name" not "blog name." That was misleading.

One more question from me, the original plugin offered the ability to add an image from the media library. Was this used very often? Should I work on re-implementing that functionality?

All commits:
https://github.com/castiron/cac/commits/featured-content

#29 - 2012-07-09 12:56 PM - Matt Gold

Great work, Dom -- I'm looking forward to checking this out.

At this point, I don't think it's worth reimplementing the media library.

#30 - 2012-07-09 03:45 PM - Boone Gorges

Hi Dom,

This looks great. I made a few changes:

- Rewrote the "override" text for a few of your items: https://github.com/castiron/cac/commit/b1e2c9cd9d4d44ab3abd787b5209b551a24b6fd2
- Fixed a JS issue that was preventing the Display Images toggle from working correctly (in my browser, anyway): https://github.com/castiron/cac/commit/ab661cc7420c55bef71bbaec8361d9be62990d4c

The changes have been merged into the master branch. cdev appears to be down at the moment (http://redmine.gc.cuny.edu/issues/1984), but when it's up, I'll move it over for testing.

As for your questions:

> I haven't forgotten about adding the code to check for subdomain/subdirectory installs

That sounds fine. As I said earlier, it's not a huge deal if subdirectory autocomplete doesn't work in the first revision.

> the original plugin offered the ability to add an image from the media library. Was this used very often? Should I work on re-implementing that functionality?

As Matt said, let's do that sometime in the future when we've got more resources to spare. http://redmine.gc.cuny.edu/issues/1983

#31 - 2012-07-10 10:07 AM - Dominic Giglio

I'd love to try the plugin out on cdev for myself, but I can't seem to find my cdev user/pass. Can you email that to me please?

#32 - 2012-07-18 01:00 PM - Boone Gorges

Hey Dom. I gave the plugin over to the folks at City Tech OpenLab, and they seemed to really like it, but found an issue with Featured Members: http://openlab.citytech.cuny.edu/redmine/issues/391#note-7 Not urgent or anything - and I haven't verified that this isn't user error - but it may be something to look at when you get a chance.

On an unrelated note, does anyone have an objection to marking this ticket Resolved? The rewrite is, in my opinion, done; what's left to implement has already been put into different tickets. The next step would be to release a new version of the plugin in the wordpress.org repositor - Dom, I will open a separate ticket to talk about that.

#33 - 2012-07-18 01:15 PM - Dominic Giglio

Will take a look at this as soon as I can. Working on the profile re-ordering right now.

I agree, closing this ticket makes sense. If anything else pops up, it should go into a new issue anyways.

I'd like to get a new README written up before going to the plugin repo but that shouldn't hold up this release. It can just as easily be rolled into the next one.

#34 - 2012-07-18 01:21 PM - Boone Gorges

- Status changed from Assigned to Resolved

Awesome, thanks.

Please open separate tickets for new issues. Issues related directly to public release should go in #1998.

Thanks for your hard work on this, Dom!

#35 - 2012-07-18 06:15 PM - Matt Gold

> Thanks for your hard work on this, Dom!

Agreed. Many thanks, Dom.

#36 - 2012-07-18 07:25 PM - Dominic Giglio

My pleasure! I really had a lot of fun and learned SO much on this re-write. It wasn't even like work. :-)

Boone, I read thru that openlab issue you linked to and I think I know what that issue is. I'm not gonna open a new ticket for it, i'll just fix it while

working on the others.