

The Panels API: Diving Down the Rabbit Hole

Sam Boyer (sdboyer)



Agenda

- ▶ A (quick) intro to how Panels actually works, and dispelling a few myths
- ▶ Implementing the API (in two parts):
 - Creating Panels plugins
 - Using Panels to output the content your module generates (e.g., og_panels)
- ▶ Then, your questions and ideas!

We're likely going to run out of time, but I want to get to as many questions about the API as possible. So if we don't get to all your ideas, write them up and send them to drupal@samboyer.org. I'll answer them in a public setting so that we can (hopefully!) all benefit =)



API Docs

This talk pretty much assumes that you're looking at the Panels API documentation. You can find it here:

<http://doxy.samboyer.org/panels2/>

Yeah, I know. Frames. Ick. Also, you can download a copy of the presentation at:

<http://doxy.samboyer.org/panels2api.odp>

Obviously that'll only work for folks who can read OpenOffice presentation files.



Terminology

- ▶ There's lots of it in Panels. A fair bit of it is confusing, and at least some of it is downright bad.
 - We know. We're working on fixing it. Talk to me afterwards if you're interested in seriously engaging in the process for fixing it.
- ▶ Some things are (at least cursorily) defined in the Panels Glossary on the API Docs. If you don't see it in the lefthand navbar, this is the direct URL:
http://doxy.samboyer.org/panels2/panels_glossary.html
- ▶ If I use a term you're not clear on, pop your hand up so we can get it taken care of.

Hide all Show all Cache settings

DISPLAY

+

PANEL

Calendar



▶ Calendar

PANE

+

PANEL

Left above

+

Right above

PANEL

Custom (sample)



▶ sample

PANE

Custom (another sample pane)



▶ another sample pane

PANE

+

PANEL

+

PANEL

Left below

+

Right below

PANEL

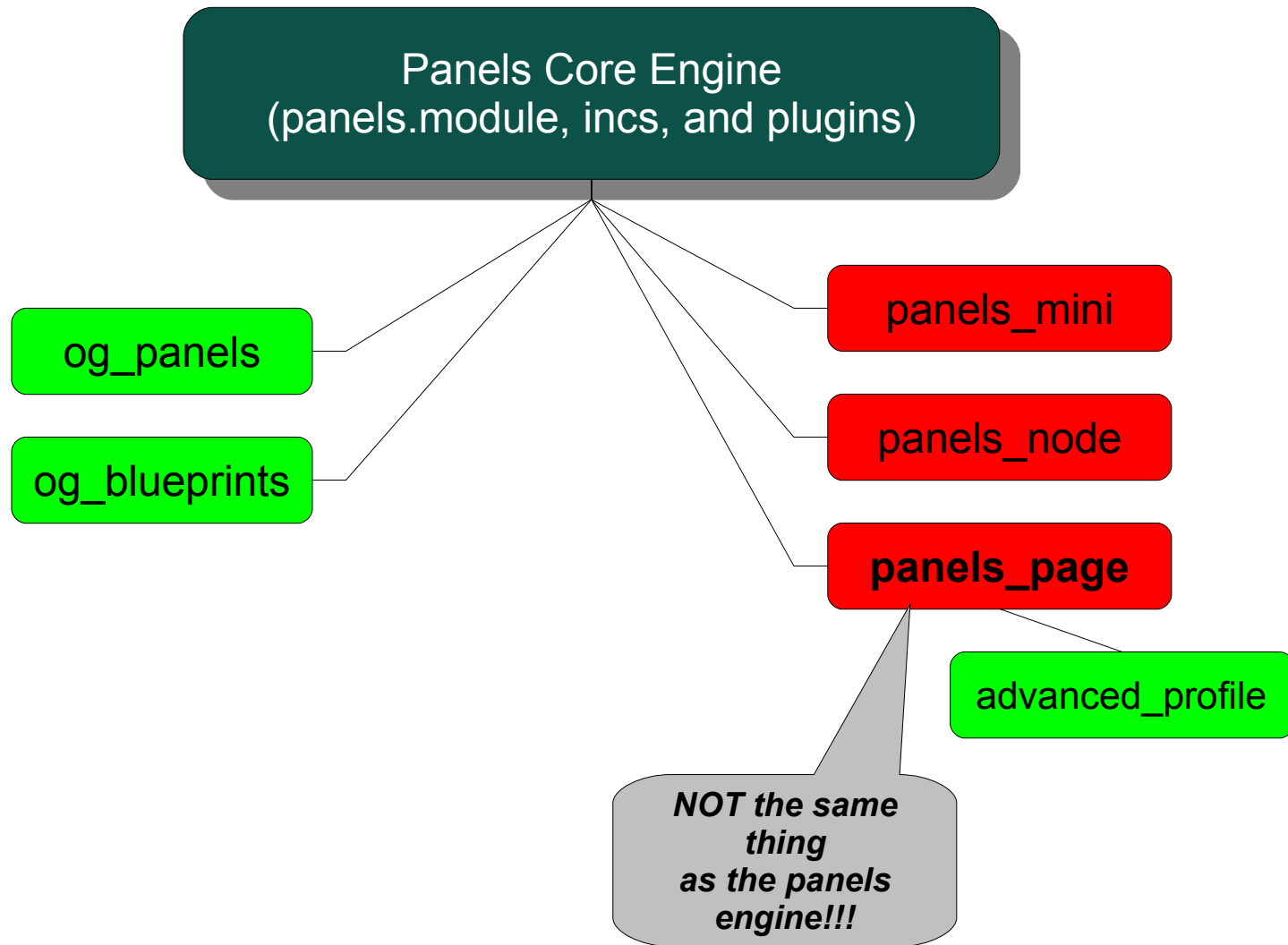
+

PANEL

Save

Cancel

Client Modules





Misconceptions

- ▶ First and most important: panels != panels_page
 - Path overrides are NOT native panels behavior; panels_page implements them as a panels 'client module.'
 - Panels works just as well with code-based Panels as with ones from the UI
 - However, the API tools for working directly from code, as in from a raw `new panels_display()`, are skimpy
 -
- ▶ More generally – my **response** to Bèr Kessels' recent post on the Planet



Render-time Flow

- ▶ Retrieve `panels_display` configuration info
 - For db-stored implementations, this involves (at minimum) a call to `panels_load_display()`
 - For implementations from code (via export or otherwise), that code directly configures the display
- ▶ Load the relevant data into Panels' context; specifically, into `$display->context`
 - Ultimately, **everything** calls `panels_context_create()`
 - `panels_page` calls it indirectly & dynamically via `arg/rel/context` plugins, which themselves call it
 - `og_panels` calls it directly, with `$type = 'group'`
- ▶ Then, proceed to `panels_render_display()`



Render-time Flow (cont)

- ▶ `panels_render_display()` has lots of moving parts, but a general summary:
 - The display iterates over each pane and asks, “What data do you want from what's tucked in context?”
 - The display hands off the requested data to the pane, which builds some structured data ready for output, then hands that data back to the display
 - The display then fires the structured data off to a style plugin, which uses `theme()` functions to wrap the output in a panels style (think rounded corners on gdo)
 - Once all the panels have been built, the individual chunks of html are placed into the appropriate sections as defined by the layout plugin



Keep in mind...

- ▶ Plugin structure is almost identical from D5 to D6. From a Panels API perspective, upgrades will be trivial.





Implementation: Plugins

- ▶ The Panels engine uses seven* plugin types:
 - **Context:** `arguments`, `relationships`, and (of course) `context`. Former two are wrappers for the latter. These plugins basically load and organize data for use later.
 - **Content Types:** basically, `content_type` plugins define panes. Presently the meatiest plugin. Devs, expect 90% of your Panels plugin work to be on these.
 - **Output/'Theming':** `layout` and `style` plugins. Layout is pretty self-explanatory; style plugins determine the `theme()` callbacks used to render panels and/or panes.
 - NOTE: In D6, layout plugins will support `tpl.php` file formats instead of the more awkward heredoc syntax
 - **Caching:** `cache` plugins define caching strategies; can be applied per-pane.
 - ***Switchers:** so...technically `switchers` are plugin type #8, but you *don't* want me to go there right now =)



General Plugin Info

- ▶ **ALL** Panels plugins adhere to the same general pattern
- ▶ Plugins are defined by their *plugin definition array*, which works via a hook(ish) system
 - Very similar to the keyed array for a menu item in `hook_menu*()`
 - The array defines *callback properties* and *setting properties*
 - Callbacks properties are strings of function names
 - Settings are mixed values that affect how the Panels engine interacts with and manages the plugin
- ▶ The Panels engine then uses the settings and fires the callbacks at certain designated times. Docs have (a little) more on that.



Hook in Your Plugins

- ▶ Panels uses two different hooks acquire the list of plugins for inclusion:
 - `hook_panels_$plugin_type()`, where `$plugin_type` is the name of the plugin type being requested
 - See `panels_load_hooks()`
 - `hook_panels_include_directory($plugin_type)`
 - See `panels_panels_include_directory()`
- ▶ Whenever possible, please use `hook_panels_include_directory()`!
- ▶ Now, let's step through some examples of these plugins...



Using the Panels Engine

- ▶ These functions are sufficient for most cases:
http://doxy.samboyer.org/panels2/group_MainAPI.html
- ▶ Four main functions you'll be calling:
 - `panels_edit()`
 - *Call the display content editor*
 - `panels_edit_layout()`
 - *Call the display layout editor*
 - `panels_edit_layout_settings()`
 - *Call the display layout settings editor*
 - `panels_render_display()`
 - *Initiate a render on your display object*
- ▶ See `og_blueprints_blueprint_edit()` for handling. (http://doxy.samboyer.org/og_blueprints)