



DrupalCon

NASHVILLE **2018**

APRIL 9-13



DrupalCon
NASHVILLE 2018
APRIL 9-13

Drupal 8, where did the code go? From info hook to plugin

Peter Wolanin (pwolanin)



They're *dead* Jim...

- hook_block_info()
- @hook_menu()
- hook_element_info()
- hook_entity_info()
- hook_filter_info()
- hook_field_widget_info()
- hook_image_toolkits()
- hook_search_info()
- ... *and many more*



Here's What's In the Talk

- Background and “What’s a plugin?”
- Example of a core info hook conversion
- Simple but common plugin example:
adding new tabs (a.k.a. local tasks)
- Configurable plugins (ConfigEntity based)
 - Adding a custom block
 - Adding a custom text filter
- Considerations for defining your own plugins



Who Am I?

- Drupal 5, 6, 7, 8 core contributor
- Drupal Security Team
- BioRAFT software architect
- Helped create some Drupal 8 plugins including SearchPlugin, MenuLinks, LocalTasks, LocalActions, ContextualLinks
- DrupalCamp NJ organizer

Photo by amazeelabs, by-nc-sa



<https://www.drupal.org/u/pwolandin>

Drupal 8 Background



I'll assume you know something about:

- The DIC/container/service container
 - ▶ The container is an object that contains instances of stateless “services” (the current request, the current user, URL generator, etc)
- The new routing system - names instead of paths.
 - ▶ a route name is just a machine name connected to a path and callbacks to provide, title, content, access etc. - like a D7 menu router
- Namespaced classes (PHP 5.3+) like
`\Drupal\search\Plugin\Block\SearchBlock`

Plugins



- Encapsulate some re-useable functionality inside a class that implements one or more specific interfaces
- Plugins combine what in Drupal 7 was an info hook and a number of implementation hooks and possibly configuration: e.g.
`hook_search_info()` **and** `hook_search_execute()`, **etc.**,
or `hook_block_info()` **and** `hook_block_view()`,
`_configure()`, `_save()`
- Evolved from ctools and views plugins, but have quite different mechanisms to discover them

Plugin Manager and IDs



- Every plugin type has a manager - registered as a service (available from the container) and used to find and instantiate the desired plugin instance(s)
- Each plugin has an ID, which may be in its definition, or generated as a derivative
- For a given plugin ID one single class will be used for all plugin instances using that plugin ID
- A plugin instance is specified by the combination of plugin ID and its configuration values, potentially coming from a ConfigEntity



“The Drop is
always moving”

drupal.org/node/65922

<http://flickr.com/photos/alkalinezoo/32265389/>

7.x: hook_image_toolkits()



```
/**
 * Implements hook_image_toolkits().
 */
function system_image_toolkits() {
  include_once DRUPAL_ROOT . '/' . drupal_get_path('module',
'system') . '/' . 'image.gd.inc';
  return array(
    'gd' => array(
      'title' => t('GD2 image manipulation toolkit'),
      'available' => function_exists('image_gd_check_settings') &&
        image_gd_check_settings(),
    ),
  );
}
```

8.x: GD Image Toolkit Plugin



```
/**
 * Defines the GD2 toolkit for image manipulation within Drupal.
 *
 * @ImageToolkit(
 *   id = "gd",
 *   title = @Translation("GD2 image manipulation toolkit")
 * )
 */
class GDToolkit extends ImageToolkitBase {
  //... more methods ...//

  public static function isAvailable() {
    // GD2 support is available.
    return function_exists('imagegd2');
  }

  //... more methods ...//
}
```

8.x: ImageToolkitManager



```
class ImageToolkitManager extends DefaultPluginManager {
    // ... various methods ... //

    /**
     * Gets a list of available toolkits.
     */
    public function getAvailableToolkits() {
        // Use plugin system to get list of available toolkits.
        $toolkits = $this->getDefinitions();

        $output = array();
        foreach ($toolkits as $id => $definition) {
            if (call_user_func($definition['class'] . '::isAvailable')) {
                $output[$id] = $definition;
            }
        }
        return $output;
    }
}
```

7.x: image_desaturate() Function



```
/**
 * Converts an image to grayscale.
 *
 * @param $image
 *   An image object returned by image_load().
 *
 * @return
 *   TRUE on success, FALSE on failure.
 *
 * @see image_load()
 * @see image_gd_desaturate()
 */
function image_desaturate(stdClass $image) {
    return image_toolkit_invoke('desaturate', $image);
}
```

8.x: Image::desaturate() via Plugin



```
/**
 * Defines an image object to represent an image file.
 *
 * @see \Drupal\Core\ImageToolkit\ImageToolkitInterface
 * @see \Drupal\image\ImageEffectInterface
 */
class Image implements ImageInterface {
    //... more methods ...//

    public function apply($operation, array $arguments = array()) {
        return $this->getToolkit()->apply($operation, $arguments);
    }

    public function desaturate() {
        return $this->apply('desaturate', array());
    }

    //... more methods ...//
}
```

7.x: Operation by Function Prefix



```
/**
 * Convert an image resource to grayscale.
 *
 * Note that transparent GIFs lose transparency when desaturated.
 *
 * @param $image
 *   An image object. The $image->resource value will be modified.
 * @return
 *   TRUE or FALSE, based on success.
 *
 * @see image_desaturate()
 */
function image_gd_desaturate(stdClass $image) {
    // PHP using non-bundled GD does not have imagefilter.
    if (!function_exists('imagefilter')) {
        watchdog('image', 'The image could not be desaturated.');
```

8.x: Operations Are Also Plugins



```
/**
 * Defines GD2 Desaturate operation.
 *
 * @ImageToolkitOperation(
 *   id = "gd_desaturate",
 *   toolkit = "gd",
 *   operation = "desaturate",
 *   label = @Translation("Desaturate"),
 * )
 */
class Desaturate extends GDImageToolkitOperationBase {

    protected function execute(array $arguments) {
        // PHP using non-bundled GD does not have imagefilter.
        if (!function_exists('imagefilter')) {
            $this->logger->notice("The image could not be desaturated");
            return FALSE;
        }
        return imagefilter($this->getToolkit()->getResource(), IMG_FILTER_GRAYSCALE);
    }
}
```

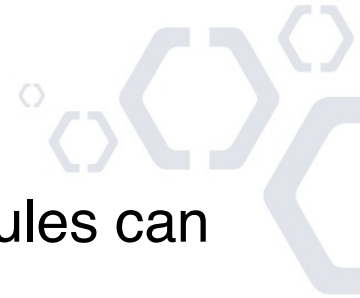

A photograph of a red buoy and a metal hook on a sandy beach. The buoy is in the background, slightly out of focus, and the hook is in the foreground, in sharp focus. The text "What About Hooks?" is overlaid in the center of the image.

What About Hooks?

http://flickr.com/photos/avelino_maestas/2532426169/

BIKRAFT

Hooks Still Have Their Place



- Most plugin managers invoke an `_alter` hook so the modules can add to or alter the plugins' definitions
- For example, `hook_block_alter()` allows you to alter the block plugin definitions
- Info hooks that simply return a data array, without associated code and logic, were not candidates to become plugins (but may have become YAML)
- Others like `hook_cron()` or `hook_entity_access()` are still present

Plugin Discovery



- The discovery of plugins is basically the same as invoking an info hook (and, in fact, it can even be implemented that way)
- Discovery gives you an array of plugin definitions, each of which is just an array of keys and values
- The discovery process fills in defaults, such a `'provider'` which is the name of the module providing the plugin
- Discovery can make derivatives (many from one) like `FieldUiLocalTask` - one tab per bundle

Plugin Discovery & Config in Core



- YAML based:
MenuLink, LocalTask, LocalAction, ContextualLink
- Annotation, some config, but no config entity:
ImageToolkit, Archiver
- Annotation and config entity (many) including:
Block, ViewsDisplay, ImageEffect, SearchPlugin
- Not a pure Plugin but uses Annotation discovery:
Entity (Node, User, etc.)

Let's Start Learning The Drupal 8.x Plugin Toolkit

<http://flickr.com/photos/booleansplit/2376359338/>

BICRAFT

Pre-requisite: a Drupal 8 Module



- As in Drupal 7, code is provided by modules - so you need a module. You need a .info.yml file - like a .info file for Drupal 7
- You no longer need even an empty .module file
- The only remaining code in .module files are a few hook implementations: most code lives in classes

modules/mymodule/mymodule.info.yml

```
name: 'My test module'  
type: module  
description: 'DrupalCon Nashville demo.'  
core: 8.x
```

Adding Two Tabs:



For most uses, just add a YAML file listing your tabs:
mymodule/mymodule.links.tasks.yml

```
mymodule.list_tab:  
  route_name: mymodule.list  
  title: 'List'  
  base_route: mymodule.list
```

← Plugin ID

```
mymodule.settings_tab:  
  route_name: mymodule.settings  
  title: 'Settings'  
  base_route: mymodule.list
```

← Local tasks reference a base route which groups them

Adding Two Tabs:

Unlike Drupal 7 you don't need to jump through the hoops of a default local task, or making the paths align in a certain hierarchy



The image displays two browser screenshots side-by-side, illustrating the structure of two tabs in a Drupal 8 configuration page. The left screenshot shows the 'Mymodule settings' page, with the URL `drupal-8.dd:8083/admin/config/also-mymodule/settings` in the address bar. The right screenshot shows the 'Mymodule list' page, with the URL `drupal-8.dd:8083/admin/config/mymodule/list` in the address bar. Arrows point from the 'Settings' tab in the left screenshot to the 'Settings' tab in the right screenshot, and from the 'List' tab in the right screenshot to the 'List' tab in the left screenshot, indicating that both tabs are active and accessible from the same configuration page.

Left Screenshot: Mymodule settings

- URL: `drupal-8.dd:8083/admin/config/also-mymodule/settings`
- Page Title: Mymodule settings ☆
- Navigation: Back to site, Manage, Shortcuts, peter
- Active Tab: Settings
- Other Tab: List
- Breadcrumbs: Home » Administration » Configuration
- Path: `Drupal\mymodule\Controller\MyController::settings`

Right Screenshot: Mymodule list

- URL: `drupal-8.dd:8083/admin/config/mymodule/list`
- Page Title: Mymodule list ☆
- Navigation: Back to site, Manage, Shortcuts, peter
- Active Tab: List
- Other Tab: Settings
- Breadcrumbs: Home » Administration » Configuration
- Path: `Drupal\mymodule\Controller\MyController::dolist`

LocalTask Plugins Need Routes:

Of course, those routes need to be defined or pre-existing:
mymodule/mymodule.routing.yml

```
mymodule.list:  
  path: '/admin/config/mymodule/list'  
  defaults:  
    _controller: '\Drupal\mymodule\Controller\MyController::dolist'  
    _title: 'Mymodule list'  
  requirements:  
    _access: 'TRUE'  
  
mymodule.settings:  
  path: '/admin/config/also-mymodule/settings'  
  defaults:  
    _controller: '\Drupal\mymodule\Controller\MyController::settings'  
    _title: 'Mymodule settings'  
  requirements:  
    _access: 'TRUE'
```



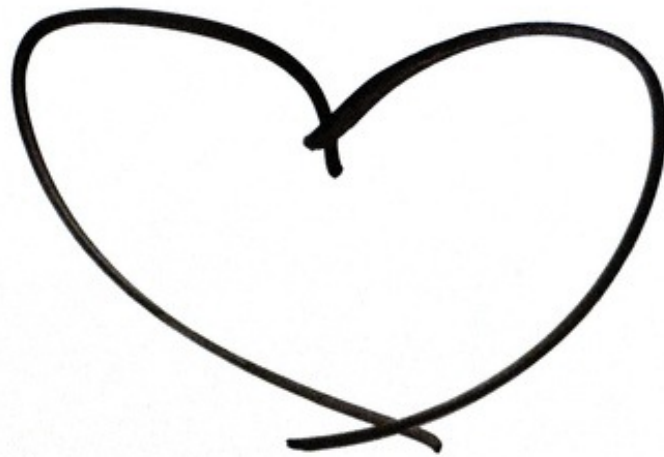
LocalTask Plugin Keys:

The plugin configuration options and defaults are on the LocalTaskManager class

```
class LocalTaskManager extends DefaultPluginManager implements
LocalTaskManagerInterface {
  protected $defaults = array(
    // (required) The name of the route this task links to.
    'route_name' => '',
    // Parameters for route variables when generating a link.
    'route_parameters' => array(),
    // The static title for the local task.
    'title' => '',
    // The route name where the root tab appears.
    'base_route' => '',
    // The plugin ID of the parent tab (or NULL for the top-level tab).
    'parent_id' => NULL,
    // The weight of the tab.
    'weight' => NULL,
    // The default link options.
    'options' => array(),
    // Default class for local task implementations.
    'class' => 'Drupal\Core\Menu\LocalTaskDefault',
    // The plugin id. Set by the plugin system based on the top-level YAML key.
    'id' => '',
  );
}
```



your
g
me



The New Block

Blocks as Plugins



- Each custom block is defined in code as a class
- When the admin places the block into a region in a theme, a configuration object is created to track that setting
- The config object is a ConfigEntity - it's an abstraction on top of CMI (storing your Drupal configuration in YAML files) - it makes it convenient to list, load, etc. using entity functions.
- Drupal can easily list the active block instances
- Note - you don't need to worry about the config!

Blocks Implementation



- **Blocks implement the** `\Drupal\Core\Block\BlockPluginInterface`
- **If you extend the abstract** `\Drupal\Core\Block\BlockBase` **class**
then all you need to implement is the `build()` **method**
- `build()` **is basically the same as** `hook_block_view()` **in Drupal 7**
- **For example, I added to my module** `\Drupal\mymodule\Plugin\Block\MyBlock`

Side Note - PSR-4



- When I add the Block to my module:
`\Drupal\mymodule\Plugin\Block\MyBlock`
- This is at the corresponding file path (under the Drupal root directory):
`modules/mymodule/src/Plugin/Block/MyBlock.php`
- Note that the class name and the path match up under `src/`
- This mapping of class name to path is how the class can be automatically found and loaded

```
/**
 * Provides a block with 'Mymodule' links.
 *
 * @Block(
 *   id = "mymodule_my_block",
 *   admin_label = @Translation("Mymodule block")
 * )
 */
class MyBlock extends BlockBase {
  public function build() {
    return [
      'first_link' => [
        '#type' => 'link',
        '#title' => $this->t('Mymodule List'),
        '#url' => Url::fromRoute('mymodule.list'),
      ],
      'second_link' => [
        '#type' => 'link',
        '#title' => $this->t('Mymodule Settings'),
        '#url' => Url::fromRoute('mymodule.settings'),
      ],
    ];
  }
}
```



Block layout

Custom block library

Bartik

Seven

[Home](#) » [Administration](#) » [Structure](#)

Block placement is specific to each theme on your site. Changes will not be saved until you click *Save blocks* at the bottom of the page.

[Demonstrate block regions \(Bartik\)](#)

[Show row weights](#)

BLOCK	CATEGORY	REGION	OPERATIONS
Header			Place block
✚ Site branding	System	Header	Configure
Primary menu			Place block
✚ Main navigation	Menus	Primary menu	Configure
Secondary menu			Place block
✚ User account menu	Menus	Secondary menu	Configure
Highlighted			Place block
✚ Status messages	System	Highlighted	Configure
Featured top			Place block

Blocks Admin Page Has a New Function: Place block



Blocks Admin Page Has a New Function: **Place block**

The screenshot shows the Drupal 8 Blocks Admin interface. A modal dialog titled "Place block" is open, allowing a user to place a block in a specific region. The dialog includes a search input field with the text "my" and a table of available blocks. The "Place block" button in the table is highlighted with an orange box.

Place block

+ Add custom block

my

BLOCK	CATEGORY	OPERATIONS
Mymodule block	My test module	Place block

Menus

No blocks in this region

← → ↻ ⓘ Not Secure | drupal-8.dd:8083/admin/structure/block ☆ 🔔 0

← Back to site ☰ Manage ☆ Shortcuts 👤 peter

Configure block

Block description: Mymodule block

Title *

Mymodule block Machine name:
mymoduleblock [Edit]

Display title

Visibility


Content types	Content types
Not restricted	<input type="checkbox"/> Article
Pages Not restricted	<input type="checkbox"/> Basic page
Roles Not restricted	

Region *

Sidebar first ▼

Select the region where this block should be displayed.

Save block



Save the Block to Create a New Plugin Instance



Drupal 8 local

Home

Mymodule block

[Mymodule List](#)

[Mymodule Settings](#)

Welcome to Drupal 8 local

No front page content has been created yet.
Follow the [User Guide](#) to start building your site.
[Add content](#)



Search

Block Hook to Plugin Comparison



Drupal 7.x

Drupal 8.x

`hook_block_info()`

BlockManagerInterface::
getDefinitions()

`hook_block_view($delta)`

BlockPluginInterface::
build()

?

BlockPluginInterface::
access(AccountInterface \$account)

`hook_block_configure($delta)`

BlockPluginInterface::
blockForm(\$form, \$form_state)

?

BlockPluginInterface::
blockValidate(\$form, \$form_state)

`hook_block_save($delta, $edit)`

BlockPluginInterface::
blockSubmit(\$form, \$form_state)

Block Discovery and Annotations




- Each Plugin type must be in the expected class namespace for your module - for blocks:
`namespace Drupal\mymodule\Plugin\Block;`
- All core plugins have a custom annotation class - you have to use the right one for your plugin
- The annotation class provides both a documentation of the possible keys in the plugin definition and default values
- There is also a generic Plugin annotation class

```
/**
 * Defines a Block annotation object.
 *
 * @Annotation
 */
class Block extends Plugin {
  /**
   * The plugin ID.
   *
   * @var string
   */
  public $id;

  /**
   * The administrative label of the block.
   *
   * @var \Drupal\Core\Annotation\Translation
   *
   * @ingroup plugin_translatable
   */
  public $admin_label;

  //... more properties ...//
}
```





Filtering Text

<http://flickr.com/photos/pinksherbet/253412963/>

BIKRAFT

7.x Info Hook



```
/**
 * Implements hook_filter_info().
 *
 * @ingroup project_issue_filter
 */
function project_issue_filter_info() {
  $filters['filter_project_issue_link'] = array(
    'title' => t('Project issue to link filter'),
    'description' => t('Converts references to project issues
                      (in the form of [#12345]) into links.'),
    'process callback' => '_project_issue_filter',
    'tips callback' => '_project_issue_filter_tips',
    'cache' => TRUE,
  );
  return $filters;
}
```


8.x Plugin With Annotation

```
/**
 * Provides a filter to format a node ID as a link.
 *
 * @Filter(
 *   id = "mymodule_node_link",
 *   module = "mymodule",
 *   title = @Translation("Format a node ID as a link")
 * )
 */
class NodeLinkFilter extends FilterBase {
  public function process($text, $langcode) {
    $regex = '(?:(!\w)\[(#\d+(?:-[\d\.]*)?)\](?!w))|...';
    $callback = function($matches) { ... }
    $text = preg_replace_callback("/$regex/", $callback, $text);
    return new FilterProcessResult($text);
  }

  // ... more code ...//

  public function tips($long = FALSE) {
    return $this->t('Node ID numbers (ex. [#12345]) turn into links
automatically.');
```





Drupal 8 local

Home

[Home](#)



Search



First test post

- [View](#)
- [Edit](#)
- [Delete](#)

Here lies a nice node.

Text formats and editors ☆

[Home](#) » [Administration](#) » [Configuration](#) » [Content authoring](#)

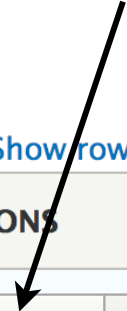
Text formats define how text is filtered for output and how HTML tags and other text is displayed, replaced, or removed. **Improper text format configuration is a security risk.** Learn more on the [Filter module help page](#).

Text formats are presented on content editing pages in the order defined on this page. The first format available to a user will be selected by default.

[+ Add text format](#)

[Show row weights](#)

NAME	TEXT EDITOR	ROLES	OPERATIONS
↕ Basic HTML	—	Authenticated user, Administrator	Configure ▼



Enabled filters

- Limit allowed HTML tags and correct faulty HTML
- Display any HTML as plain text
- Convert line breaks into HTML (i.e.
 and <p>)
- Convert URLs into links
- Format a node ID as a link
- Align images
 - Uses a data-align attribute on tags to align images.
- Caption images
 - Uses a data-caption attribute on tags to caption images.
- Restrict images to this site
 - Disallows usage of tag sources that are not hosted on this site by replacing them with a placeholder image.
- Correct faulty and chopped off HTML
- Track images uploaded via a Text Editor
 - Ensures that the latest versions of images uploaded via a Text Editor are displayed.

Filter processing order

Body (Edit summary)

<p>Let me tell you about [#2058]

Text format Basic HTML ▼

[About text formats](#) ?

- Allowed HTML tags: <a href hreflang> <cite> <blockquote cite> <code> <ul type> <ol start type> <dl> <dt> <dd> <h2 id> <h3 id> <h4 id> <h5 id> <h6 id> <p>

- Node ID numbers (ex. [#12345]) turn into links automatically.
- You can align images (data-align="center"), but also videos, blockquotes, and so on.

Author: peter

Create new revision

Revision log message

[Empty text area for revision log message]

Briefly describe the changes you have made.

▶ MENU SETTINGS

▶ URL PATH SETTINGS

▶ AUTHORING INFORMATION

▶ PROMOTION OPTIONS



Drupal 8 local

Home

Home

Search



A second post

View

Edit

Delete

Let me tell you about [First test post](#)



Creating Your Own Plugins



- You want to upgrade your module to Drupal 8 and it defined an info hook or defined a ctools plugin type
- Use annotation based discovery by default
- It keeps the meta-data together with the class and is suited for typical plugin types where the actual class (code) is different for each plugins
- YAML discovery is good for a case like local tasks where the vast majority use a common class, but a few will be implemented with a different class (for example, to provide a dynamic title)

Plugin and General 8.x Resources



- ◆ Demo code used for this presentation:
drupal.org/sandbox/pwolanin/2087657
- ◆ *Converting 7.x modules to 8.x*
drupal.org/update/modules/7/8
- ◆ *Plugin API in Drupal 8*
drupal.org/developing/api/8/plugins
- ◆ *Why Plugins?* drupal.org/node/1637614
- ◆ *Unraveling the Drupal 8 Plugin System*
drupalize.me/blog/201409/unravelling-drupal-8-plugin-system

Drupal 8 Features I Mentioned



- Widespread use of interfaces makes it easier to replace almost any implementation
- Tabs are grouped regardless of path hierarchy
- Route name rather than system path is unique
- Multiple routes can serve the same path (HTML vs. JSON or GET vs. POST)
- YAML as a standard for config and data files
- Multiple instances of the same block
- Each block instances has a config entity

To Sum It Up

- Plugins are a way to combine the discovery of available functionality with the actual implementation of the functionality
- In Drupal 7, the combination of an info hook and multiple other hook functions (potentially in different files) served the same purpose
- When defining your own plugins, use Annotation-based discovery unless you have a very clear reason for a different type





DrupalCon
NASHVILLE 2018
APRIL 9-13

Join us for
contribution sprints
Friday, April 13, 2018

Mentored
Core sprint

9:00-18:00

Room: Stolz 2

First time
sprinter workshop

9:00-12:00

Room: Stolz 2

General
sprint

9:00-18:00

Room: Stolz 2

[#drupalsprint](https://twitter.com/drupalsprint)



What did you think?

Locate this session at the DrupalCon Nashville website:

<https://events.drupal.org/node/21025>

Take the Survey!

<https://www.surveymonkey.com/r/DrupalConNashville>

Thank you!