Turner

Photo by Steve Harvey on Unsplash

Plugins

Where does this thing go?

Bill Bohling
Sr. Systems/Software Developer
Turner

Plugins

- What are plugins?
- Drupal Plugin API
- How do I use plugins?
  - Core plugin types
- How do I create my own plugins?
- Advanced topics
  - Derivatives
  - Collections

What are plugins?

- D.O. – Small, swappable pieces of functionality.

- drupalize.me – Plugins are a design pattern.

Photo by NeONBRAND on Unsplash

Plugin API

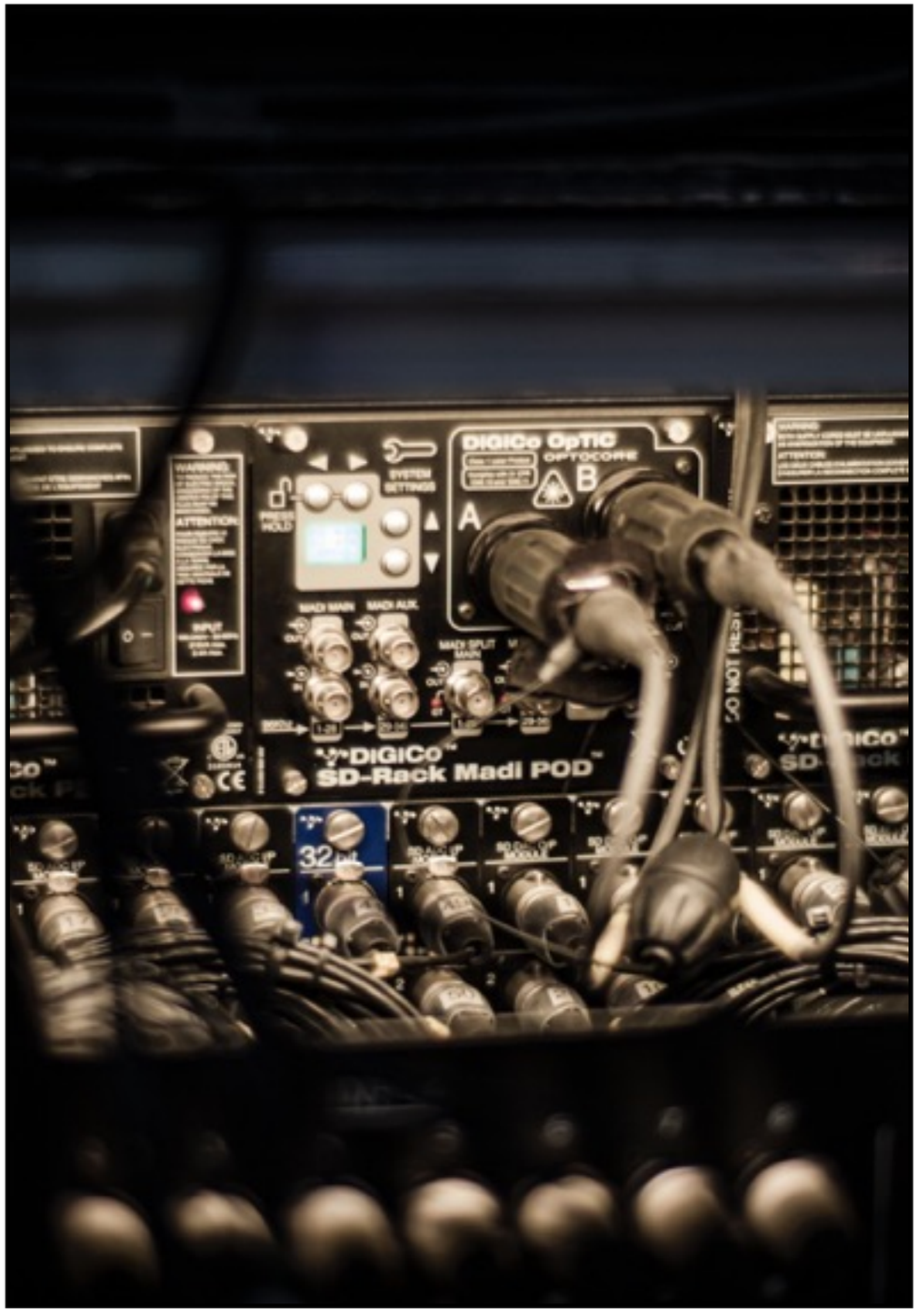

Photo by Adi Goldstein on Unsplash

Plugin API

- Plugin type – the plugin type defines the interface to implement, mechanisms for discovery and instantiation, and how the plugin is used.

Plugin API

- Plugin type – the plugin type defines the interface to implement, mechanisms for discovery and instantiation, and how the plugin is use.
- **Discovery – This is the process of finding the definitions and metadata for all plugins of a given type. Methods include annotation, YAML, hooks and static discovery.**

Plugin API

- Plugin type – the plugin type defines the interface to implement, mechanisms for discovery and instantiation, and how the plugin is used.
- Discovery – This is the process of finding the definitions and metadata for all plugins of a given type. Methods include annotation, YAML, hooks and static discovery.
- **Plugin factory – Responsible for instantiating a specific plugin and returning a usable instance.**

Plugin API

- Plugin type – the plugin type defines the interface to implement, mechanisms for discovery and instantiation, and how the plugin is used.
- Discovery – This is the process of finding the definitions and metadata for all plugins of a given type. Methods include annotation, YAML, hooks and static discovery.
- Plugin factory – Responsible for instantiating a specific plugin and returning a usable instance.
- **Plugins – A plugin is an implementation of a plugin type definition.**

Using plugins

Photo by rawpixel on Unsplash

Turner

Using plugins

- Blocks
- Menus
- Fields – types, widgets and formatters
- REST resources
- Image effects and formatters
- Validation constraints
- Data types
- **Views!**

Turner

Using plugins

- `drupal debug:plugin` — get a list of all the plugin types on your Drupal instance
- `drupal debug:container` — list the services available on your Drupal instance

```
• $manager = \Drupal::service('plugin.manager.block');
```

Using plugins

```
$plugin_manager = \Drupal::service('plugin.manager.block');

$plugin_manager->getDefinitions();
```

devel_execute_php
devel_switch_user
help_block
node_syndicate_block
search_form_block
shortcuts
system_branding_block
system_breadcrumb_block
system_main_block
system_menu_block:account
system_menu_block:admin
system_menu_block:devel
system_menu_block:footer
system_menu_block:main
system_menu_block:tools
system_messages_block
system_powered_by_block
user_login_block
views_block:comments_recent-block_1
views_block:content_recent-block_1
views_block:who_s_new-block_1
views_block:who_s_online-who_s_online_block

Using plugins

```
$plugin_manager = \Drupal::service('plugin.manager.block');

$plugin_manager->getDefinitions();

$block = $plugin_manager->getDefinition('search_form_block');
```

```
[
  admin_label => Drupal\Core\StringTranslation\TranslatableMarkup {#3831},
  category => Drupal\Core\StringTranslation\TranslatableMarkup {#3838},
  id => search_form_block,
  class => Drupal\search\Plugin\Block\SearchBlock,
  provider => search,
]
```

Using plugins

```
$plugin_manager = \Drupal::service('plugin.manager.block');

$plugin_manager->getDefinitions();

$block = $manager->getDefinition('search_form_block');
```

```
[
  admin_label => Drupal\Core\StringTranslation\TranslatableMarkup {#3831},
  category => Drupal\Core\StringTranslation\TranslatableMarkup {#3838},
  id => search_form_block,
  class => Drupal\search\Plugin\Block\SearchBlock,
  provider => search,
]
```

```
$search_block = $block->build();
```

Plugin type

When you want to enable different ways of doing the same thing.

Photo by Manuel Nägeli on Unsplash

Creating a plugin type

- An interface – to ensure that all plugins of this type really are swappable.
- A base class – an abstract class that extends the PluginBase class and implements your interface.
- A plugin manager – keeps track of plugins of this type.
- An annotation file – defines the properties of your plugin.
- Make your plugin manager a service.

Creating a plugin type

```
drupal generate:plugin:type:annotation
```

modules/custom/drupalcamp_2018/src/Annotation/Musician.php

modules/custom/drupalcamp_2018/src/Plugin/MusicianBase.php

modules/custom/drupalcamp_2018/src/Plugin/MusicianInterface.php

modules/custom/drupalcamp_2018/src/Plugin/MusicianManager.php

modules/custom/drupalcamp_2018/drupalcamp_2018.services.yml

modules/custom/drupalcamp_2018/src/Plugin/Musician/

Creating a plugin type

MusicianManager

```
public function __construct(\Traversable $namespaces, CacheBackendInterface
$cache_backend, ModuleHandlerInterface $module_handler) {
  parent::__construct('Plugin/Musician', $namespaces, $module_handler, 'Drupal
\drupalcamp_2018\Plugin\MusicianInterface', 'Drupal\drupalcamp_2018\Annotation
\Musician');

  $this->alterInfo('drupalcamp_2018_musician_info');

  $this->setCacheBackend($cache_backend, 'drupalcamp_2018_musician_plugins');

}
```

Turner

Creating a plugin type

Musician Annotation

```
class Musician extends Plugin {

  public $id;

  public $name;

  public $instrument;

  // other attributes like rhythm, pitch, etc.

}
```

Creating a plugin type

Extending DefaultPluginManager

- CategorizingPluginManagerTrait – Methods for categorizing plugin definitions based on a ‘category’ key. Used in core by ActionManager, BlockManager, ConditionsManager and FieldTypePluginManager.

- FilteredPluginManagerTrait – Methods for plugin managers that allow filtering plugin definitions. Used by BlockManager, ConditionManager and LayoutPluginManager.

- PluginDependencyTrait – For calculating dependencies of a plugin. Views and config entities use this one.

What is a plugin?

A plugin is an instance of a plugin type.

Photo by Brandon Wilson on Unsplash

Creating a plugin

MY_MODULE/**src/Plugin/Musician**/Guitarist.php

```
namespace Drupal\MY_MODULE\Plugin\Musician;

use Drupal\drupalcamp_2018\Plugin\MusicianBase;

/**
 * Provides a guitarist
 *
 * @Musician (
 *   id = guitarist,
 *   name = Guitar,
 *   instrument = guitar,
 *   . . .
 * )
 */
class Guitarist extends MusicianBase {

}
```

Turner

Creating a plugin

```
$union_rep = \Drupal::service('plugin.manager.musician');

$picker = $union_rep->getDefinition(‘guitarist');

$picker->play('Stairway to Heaven');
```

Plugin derivatives

Derivatives are plugins that are derived from other data.

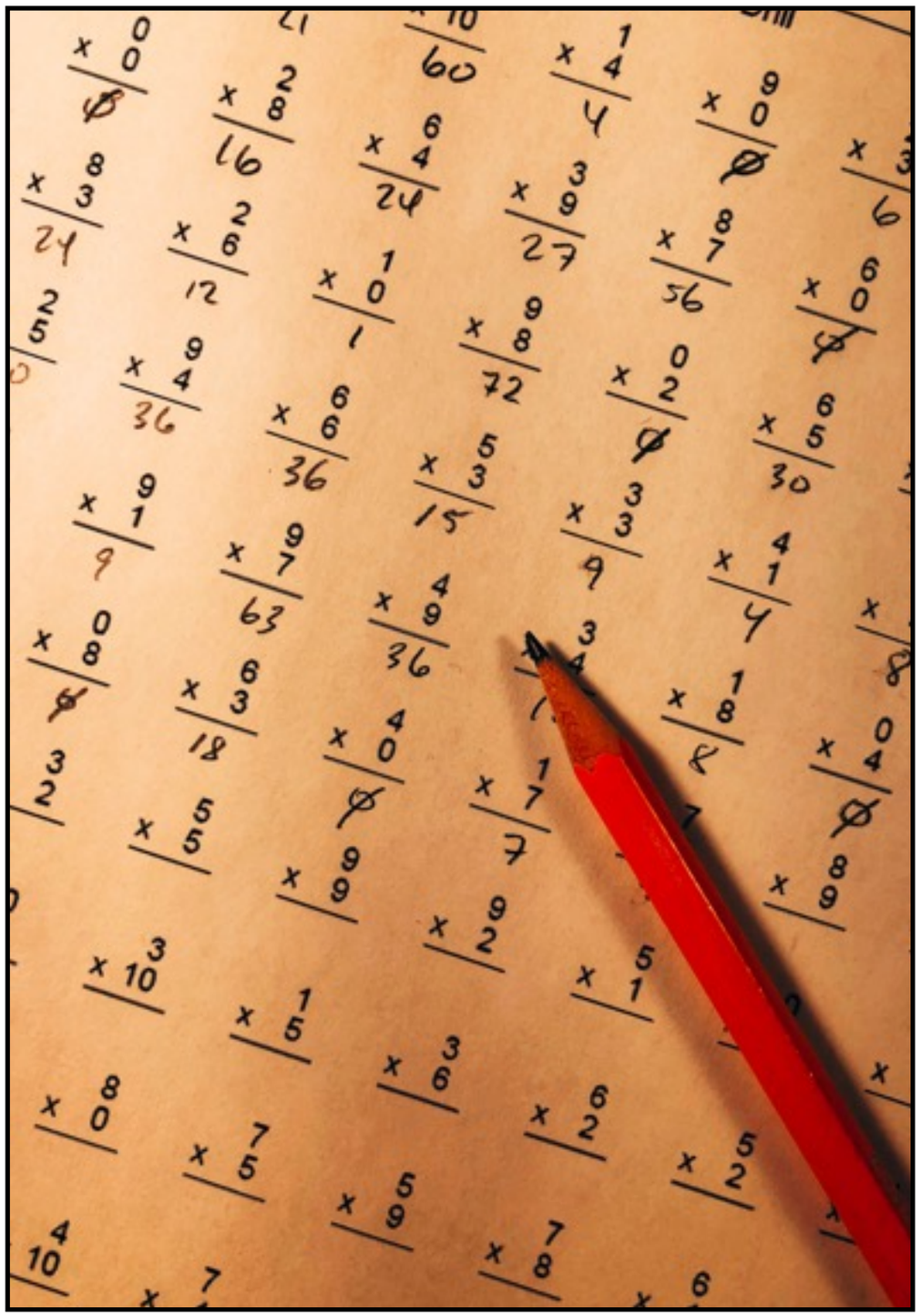

Photo by Chris Liverani on Unsplash

Plugin derivatives

```
namespace Drupal\MY_MODULE\Plugin\Musician;

use Drupal\drupalcamp_2018\Plugin\MusicianBase;

/**
 * Provides a guitarist
 *
 * @Musician (
 *   id = guitarist,
 *   name = Guitar,
 *   instrument = guitar,
 *   deriver = Drupal\drupalcamp_2018\Plugin\Derivative\GuitarDeriver
 * )
 */
class Guitarist extends MusicianBase {

}
```

Turner

Plugin derivatives

```
class GuitarDeriver extends DeriverBase implements ContainerDeriverInterface {

[boring stuff here]

  public function getDerivativeDefinitions($base_plugin_definition) {
    $this->derivatives = [];

    $node_storage = $this->entityTypeManager->getStorage('node');
    $node_ids = $node_storage->getQuery()
      ->condition('type', 'musician')
      ->condition('field_instrument', 'guitar')
      ->execute();

    if (!empty($node_ids)) {
      $nodes = $node_storage->loadMultiple($node_ids);
      foreach ($nodes as $node) {
        $this->derivatives[$node->id()] = $base_plugin_definition;
        $this->derivatives[$node->id()]['name'] = $node->field_name->getValue();
        $this->derivatives[$node->id()]['instrument'] = $node->field_axe->getValue();
      }
    }

    return $this->derivatives;
  }

}
```

Plugin derivatives

```
$pickers = $union_rep->getDefinitions('guitarist');
```

```
    "guitarist:3" => [
      "id" => "guitarist",
      "name" => [
        [
          "value" => "Leo Kottke",
        ],
      ],
      "instrument" => [
        [
          "value" => "12-string Acoustic",
        ],
      ],
      "deriver" => "Drupal\drupalcamp_2018\Plugin\Derivative\GuitarDeriver",
      "class" => "Drupal\drupalcamp_2018\Plugin\Musician\Guitar",
      "provider" => "drupalcamp_2018",
    ],
    "guitarist:4" => [
      "id" => "guitarist",
      "name" => [
        [
          "value" => "Mark Knopfler",
        ],
      ],
      "instrument" => [
        [
          "value" => "Stratocaster",
        ],
      ],
      "deriver" => "Drupal\drupalcamp_2018\Plugin\Derivative\GuitarDeriver",
      "class" => "Drupal\drupalcamp_2018\Plugin\Musician\Guitar",
      "provider" => "drupalcamp_2018",
    ],
```

```
$mark_knopfler = $musician_manager->getDefinition('guitarist:4');
$mark_knopfler->play('Money For Nothing');
```

Plugin collections

Because we really don’t want everybody playing all the time.

Photo by Manuel Nägeli on Unsplash

Turner

Plugin collections

Collections store plugin configurations.

Best used when plugin instances are configurable.

Usually associated with config entities, like Blocks and Views.

Plugin collections

Core

- Condition
- Entity
- Filter
- Image
- Tours
- Views – https://www.drupal.org/project/views/issues/1817582

Plugin collections

```
namespace Drupal\drupalcamp_2018;

use Drupal\Core\Plugin\DefaultLazyPluginCollection;

/**
 * An experimental collection of musicians.
 */
class MusicianPluginCollection extends DefaultLazyPluginCollection {


  /**
   * The key within the plugin configuration that contains the plugin ID
   */
   protected $pluginKey = 'id';


}
```

Plugin collections

```
$guitarists = new MusicianPluginCollection('guitarists');

$mark_knopfler = $guitarists->get('guitarist:4');

$mark_knopfler->play('Sultans of Swing');
```

One more thing...

https://www.drupal.org/u/sunset_bill

https://www.linkedin.com/in/bgbohling/

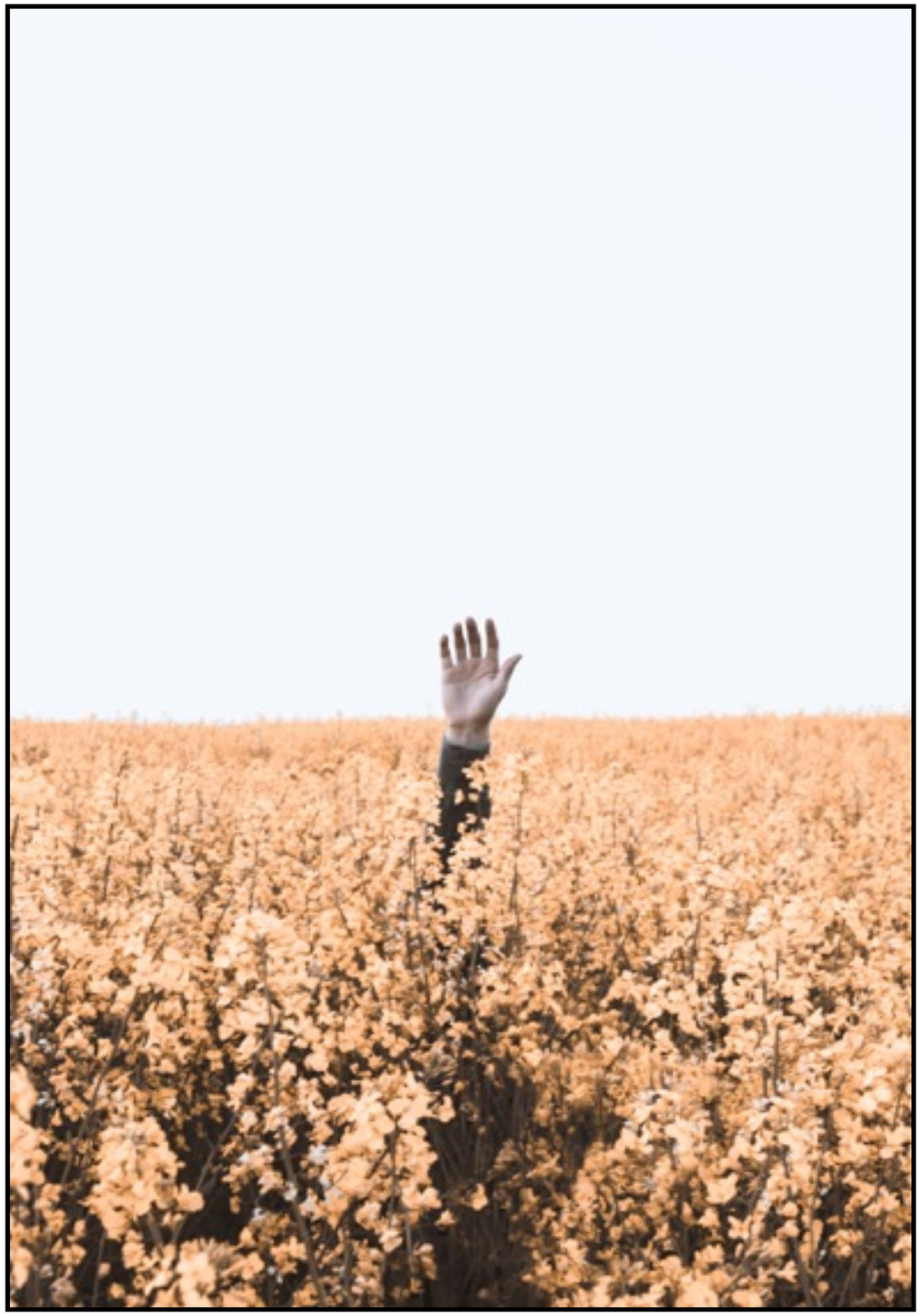

Photo by Daniel Jensen on Unsplash

Unplugging

Resources:

https://www.drupal.org/docs/8/api/plugin-api

https://api.drupal.org/api/drupal/core!core.api.php/group/plugin_api/8.6.x

Daniel Sipos
- sitepoint.com
- webomelette.com

drupalize.me

https://www.drupalcampatlanta.com/2018/sessions/plugins-or-where-does-thing-go

Photo by Adi Goldstein on Unsplash