Plugin API in Drupal 8

BOJAN ŽIVKOV, PETAR GNJIDIĆ
CIRCLE WEB FOUNDRY

CIRCLE WEB
FOUNDRY

Overview

- What Are Plugins?
- What is a plugin manager?
- Plugin discovery
- Plugin factories
- Demonstration

CIRCLE WEB FOUNDRY

What are plugins?

The basic idea of plugins is to allow a particular module to provide functionality in an extensible, object-oriented way.

CIRCLE WEB FOUNDRY

What are plugins?

Plugins are an OO replacement for info hooks and any hook associated with an info hook. At the same time, they provide a much more robust mechanism for replacement of logic, which is something we could not do previously. With plugins, you can actually swap a class for a particular plugin and run completely different code than what core or a contrib module provided, which is incredibly useful.

What are plugins?

- Blocks
- Image manipulation effects
- Field types, field widgets, and field formatters
- Items in a navigation menu

What are plugins?

- Plugins are reusable bits of functionality that are configurable, re-usable, and do exactly one thing.
- Plugins are PHP classes that implement a defined interface.
- Creating new plugins requires knowledge of PSR-4, Annotations, and sometimes Dependency Injection and Service Containers.
- Plugins types are defined and managed by a plugin manager.

What are plugins?

There are several things a module developer may need to do with plugins:

- **Define a completely new plugin type**
- **Create a plugin of an existing plugin type**
- **Perform tasks that involve plugins**

What is a plugin manager?

Each plugin type is managed by a plugin manager service, which uses a plugin discovery method to discover provided plugins of that type and instantiate them using a plugin factory.

What is a plugin manager?

A plugin manager describes how plugins of a given type will be located, instantiated, and generally what they'll do.

Plugin Discovery

Plugin Discovery is the process of finding plugins within the available code base that qualify for use within this particular plugin type's use case.

Plugin Discovery

- **Annotation**: Plugin classes are annotated and placed in a defined namespace subdirectory. Most Drupal Core plugins use this method of discovery.
- **Hook**: Plugin modules need to implement a hook to tell the manager about their plugins.
- **YAML**: Plugins are listed in YAML files. Drupal Core uses this method for discovering local tasks and local actions. This is mainly useful if all plugins use the same class, so it is kind of like a global derivative.
- **Static**: Plugin classes are registered within the plugin manager class itself. Static discovery is only useful if modules cannot define new plugins of this type (if the list of available plugins is static).

Plugin Factories

The Factory is responsible for instantiating the specific plugin

```php
namespace Drupal\drupalzoo;
use Drupal\Core\Plugin\DefaultPluginManager;
use Drupal\Core\Cache\CacheBackendInterface;
use Drupal\Core\Extension\ModuleHandlerInterface;

class DrupalZooManager extends DefaultPluginManager {
    public function __construct(\Traversable $namespaces, CacheBackendInterface $cache_backend, ModuleHandlerInterface
$module_handler) {

    parent::__construct('Plugin/Cat', $namespaces, $module_handler, 'Drupal\drupalzoo\CatInterface',
'Drupal\drupalzoo\Annotation\Cat');

     $this->alterInfo('drupalzoo_cats_info');
     $this->setCacheBackend($cache_backend, 'drupalzoo_cats');
    }
}
```

```php
namespace Drupal\drupalzoo\Annotation;

use Drupal\Component\Annotation\Plugin;

/**
 * Defines a cat item annotation object.
 *
 * Plugin Namespace: Plugin\drupalzoo\Cat
 *
 * @see \Drupal\drupalzoo\Plugin\DrupalZooManager
 * @see plugin_api
 *
 * @Annotation
 */
```

```php
class Cat extends Plugin {

    /**
     * The plugin ID.
     *
     * @var string
     */
    public $id;

    /**
     * The name of the cat.
     *
     * @var \Drupal\Core\Annotation\Translation
     *
     * @ingroup plugin_translatable
     */
    public $name;

    /**
     * The color of the cat.
     *
     * @var string
     */
    public $color;
}
```

CIRCLE WEB
FOUNDRY

```
namespace Drupal\drupalzoo\Plugin\Cat;
use Drupal\drupalzoo\CatBase;

/**

 * Provides a 'Tom' cat.

 *

 * @Cat(

 *   id = "tom",

 *   name = "Tom",

 *   weight = 4.02,

 *   color = @Translation("Blue")

 * )

 */

class Tom extends CatBase {}
```

```php
namespace Drupal\drupalzoo;

use Drupal\Component\Plugin\PluginInspectionInterface;
interface CatInterface extends PluginInspectionInterface {

    /**
     * Return the name of the cat.
     *
     * @return string
     */
    public function getName();
    /**
     * Return the weight of the cat.
     *
     * @return float
     */
    public function getWeight();
    /**
     * Return the color of the cat.
     *
     * @return string
     */
    public function getColor();

}
```

CIRCLE WEB
FOUNDRY

```php
namespace Drupal\drupalzoo;
use Drupal\Component\Plugin\PluginBase;

class CatBase extends PluginBase implements CatInterface {

    public function getName() {
        return $this->pluginDefinition['name'];
    }

    public function getWeight() {
        return $this->pluginDefinition['weight'];
    }

    public function getColor() {
        return $this->pluginDefinition['color'];
    }
}
```

Links

DRUPAL	https://www.drupal.org/docs/8/api/plugin-api
JOE SHINDELAR	https://drupalize.me/blog/201409/unravelling-drupal-8-plugin-system
DEMO PLUGIN MANAGER	https://github.com/pgnjidic/drupalzoo