WRITING DRUPAL 7 PLUGINS …

READY FOR DRUPAL 8 !

- Who am I?
- What I do?
- Why ?

Pol Dellaiera

Consultant @ Trasys (NRB Group)

Twitter: @drupol
Drupal: http://drupal.org/u/pol

Having fun at the European Commission

- Developing and maintaining an Intranet,
- Creating automatic deployment tools using Git, Gerrit, Jenkins and Aegir.

While developing Openlayers, I wanted something sustainable, reliable and nice for people to get into the code.

And as I'm lazy, I wanted something that I also can use in Drupal 8, as is.

This is how discovered Service Container.

CTools

- Written for Drupal 7 by Merlinofchaos (views, panels,... !!!)
- It works
- Documentation is not up to date
- Mix between procedural and OOP.

Service Container

- Written for Drupal 7 by Fabian Franz (fabianx), Daniel Wehner (dawehner) and me,
- Code coverage and tests,
- Only OOP.

Manage plugins with CTools in D7

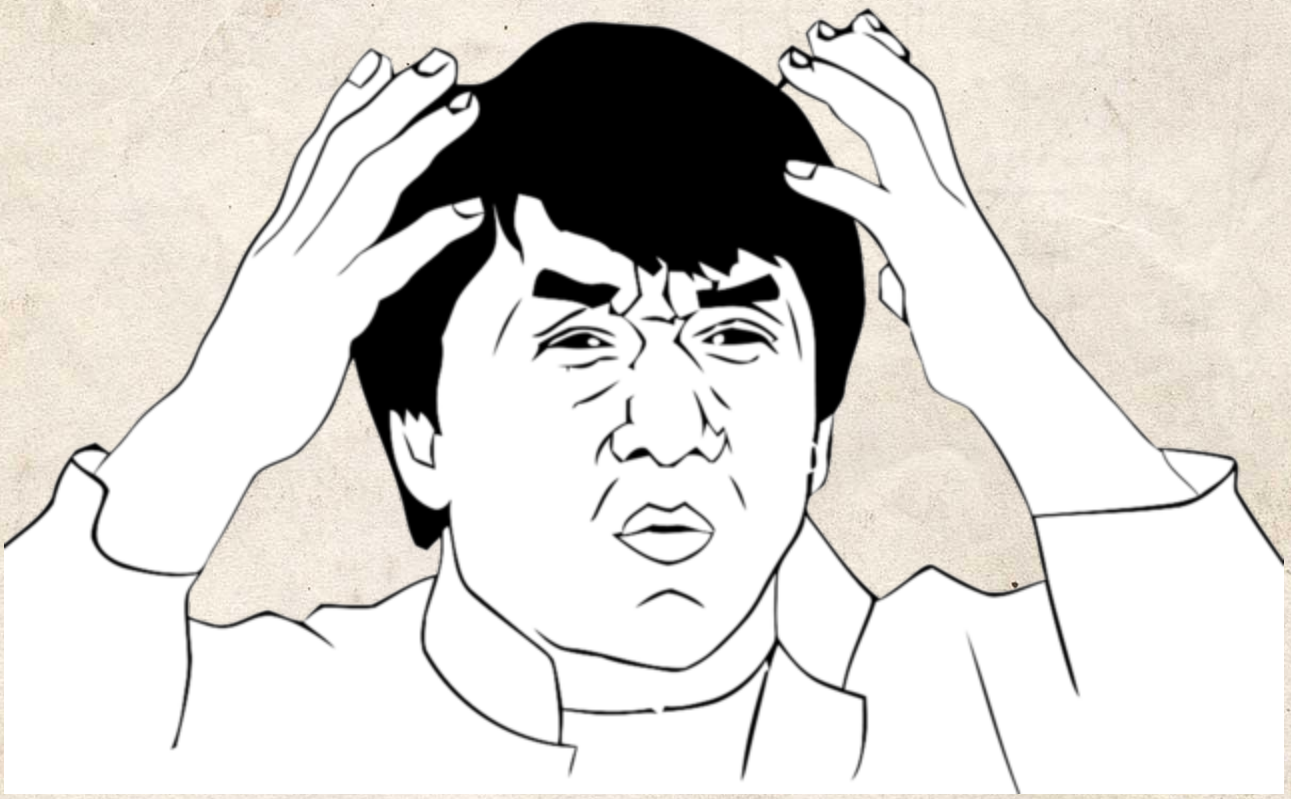
```
dependencies[] = ctools
```

Add the right dependencies to your module.

Manage plugins with CTools in D7

```php
/**
* Implements hook_ctools_plugin_type().
*/
function example_ctools_plugin_type() {
 return array(
   'operation' => array(
     'use hooks' => TRUE,
   ),
 );
}
```

Provide plugins with CTools in D7



Step 1: editing the info file

```
files[] = plugins/operation/example_sum_operation.inc
```

If the plugins are based on a class, you must define them.

Yes, each of them. Don’t believe me ? Look at the Views info file ;-)

```
name = Views
description = Create customized lists and queries from your database.
package = Views
core = 7.x
php = 5.2

; Always available CSS
stylesheets[all][] = css/views.css

dependencies[] = ctools
; Handlers
files[] = handlers/views_handler_area.inc
files[] = handlers/views_handler_area_messages.inc
files[] = handlers/views_handler_area_result.inc
files[] = handlers/views_handler_area_text.inc
files[] = handlers/views_handler_area_text_custom.inc
files[] = handlers/views_handler_area_view.inc
files[] = handlers/views_handler_argument.inc
files[] = handlers/views_handler_argument_date.inc
files[] = handlers/views_handler_argument_formula.inc
files[] = handlers/views_handler_argument_many_to_one.inc
files[] = handlers/views_handler_argument_null.inc
files[] = handlers/views_handler_argument_numeric.inc
files[] = handlers/views_handler_argument_string.inc
files[] = handlers/views_handler_argument_group_by_numeric.inc
files[] = handlers/views_handler_field.inc
files[] = handlers/views_handler_field_counter.inc
files[] = handlers/views_handler_field_boolean.inc
files[] = handlers/views_handler_field_contextual_links.inc
files[] = handlers/views_handler_field_custom.inc
files[] = handlers/views_handler_field_date.inc
```

Step 2: hook_ctools_plugin_directory

```php
/**
* Implements hook_ctools_plugin_directory().
*/
function sum_ctools_plugin_directory($module, $plugin) {
 if (($module == 'example') && ($plugin == 'operation')) {
   return 'plugins/operation';
 }
}
```

Tell Drupal to look in a particular directory for plugins.

Step 3: Define the plugin(s)

```php
<?php

/**
* Operation plugin for Example module.
*
* Calculate sum of two numbers.
*/

$plugin = array(
 'label' => t('Sum'),
 'handler' => array(
   'class' => 'example_sum_operation',
 ),
);

class example_sum_operation extends example_operation {
 public function calculate() {
   return $this->a + $this->b;
 }
}
```

Plugins are important if you want to let users extend your own module in a nice and clean way.

Writing plugins should be easy, fast and fun.

Manage plugins with SC in D7

```
dependencies[] = service_container:service_container_symfony
dependencies[] = service_container:service_container_annotation_discovery
dependencies[] = registry_autoload
registry_autoload[] = PSR-4
```

Add the right dependencies to your module.

Manage plugins with SC in D7

```
parameters:
 annotated_plugins_auto_discovery.sc_example:
   - { owner: 'sc_example', type: 'Operation', directory: 'Plugin/Operation'}
```

Declare the plugins in a .services.yml file.

Yes, a .services.yml in YAML just like in Drupal 8.

Provide plugins with SC in D7

Step 1: editing the info file

```
registry_autoload[] = PSR-4
```

No need to define the plugin class, they will be automatically loaded on demand.

Note: Your classes needs to be in the src directory.
Yes, just like in Drupal 8.

Step 2: create your plugin

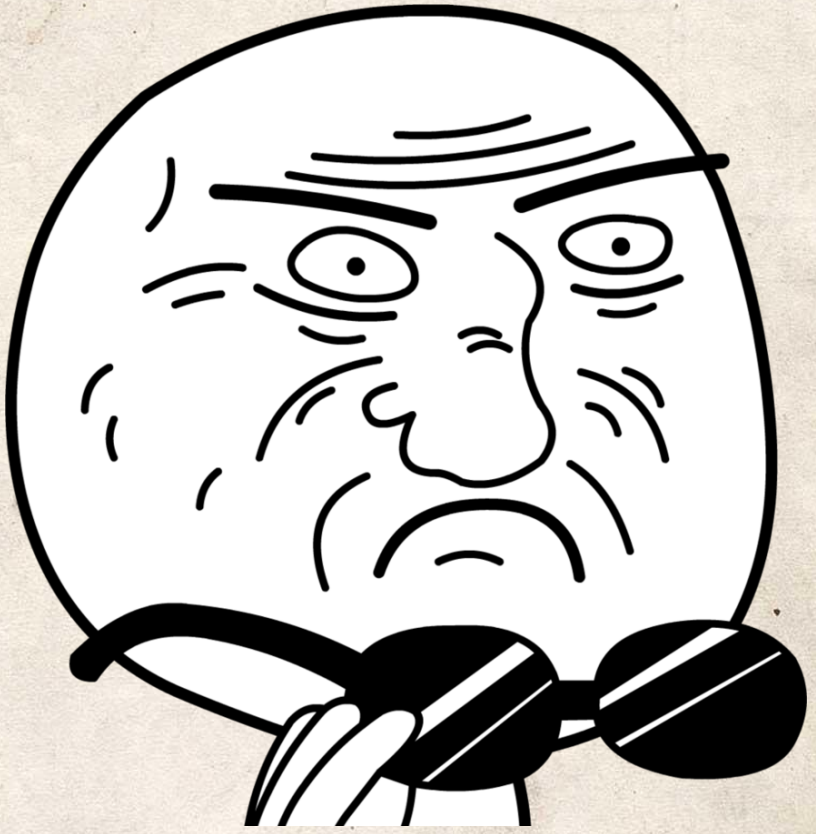
```php
<?php

/**
* Operation plugin for Example module.
*
* Calculate sum of two numbers.
*/

namespace Drupal\multiple_divide_sum\Plugin\Operation;
use Drupal\sc_example\sc_example_operation;

/**
* Class Sum.
*
* @Plugin(
*  id = "sum",
*  label = "Sum"
* )
*/
class Sum extends sc_example_operation {
 public function calculate() {
   return $this->a + $this->b;
 }
}
```

It’s done, profit.



Live demonstration

Questions? Ask me all the things!

2dotstwice
connecting the dots

AMPLEXOR
Pure Content Management

UC Leuven Limburg
MOVING MINDS

uniway
Business Critical Internet Solutions

ONE AGENCY

TOBANIA

Dropsolid
Makkelijk Digitaal Ondernemen

dazzle
the best the web can get

Combell
Your host on the internet

Acquia
THINK AHEAD