Accelerate with Service Autowiring!

Bring Symfony's new Innovation into your Drupal App

with your friend @weaverryan

SymfonyCasts

DrupalCon SEATTLE 2019

Yo! I'm Ryan!

- Lead of the Symfony documentation team
- Writer for SymfonyCasts.com
- Symfony evangelist… Fanboy
- Husband of the much more talented @leannapelham
- Father to my much more charming son, Beckett

SymfonyCasts

symfonycasts.com
twitter.com/weaverryan

(0)

So, Service Configuration

@weaverryan

Barista: prepares drinks for us

```php
// modules/custom/coffee_shop/src/Service/Barista.php
namespace Drupal\coffee_shop\Service;

class Barista {

  public function prepareDrink($type) {
    // ...

    return 'I just made you a '.$type;
  }

}
```

With a config factory dependency

```php
// modules/custom/coffee_shop/src/Service/Barista.php
namespace Drupal\coffee_shop\Service;

use Drupal\Core\Config\ConfigFactoryInterface;

class Barista {
  private $configFactory;

  public function __construct(ConfigFactoryInterface $configFactory) {
    $this->configFactory = $configFactory;
  }

  public function prepareDrink($type) {
    // ...

    return 'I just made you a '.$type;
  }
}
```

Registered as a service

```yaml
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
  coffee_shop.barista:
    class: Drupal\coffee_shop\Service\Barista
    arguments: ['@config.factory']
```

And used in our controller...

```php
// modules/custom/coffee_shop/src/Controller/CoffeeController.php

namespace Drupal\coffee_shop\Controller;

class CoffeeController {

  public function brewCoffee($type) {
    $barista =  \Drupal::getContainer()
      ->get('coffee_shop.barista');

    $text = $barista->prepareDrink($type);

    return [
      '#type' => 'markup',
      '#markup' => $text,
    ];
  }

}
```

Brilliant!

@weaverryan

Except...

We did WAY too much work!

(1)

Goodbye machine service ids

@weaverryan

Service id === Class name

```
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
- coffee_shop.barista:
+ Drupal\coffee_shop\Service\Barista:
    class: Drupal\coffee_shop\Service\Barista
    arguments: ['@config.factory']
```

And used in our controller...

```
// modules/custom/coffee_shop/src/Controller/CoffeeController.php

namespace Drupal\coffee_shop\Controller;

class CoffeeController {

  public function brewCoffee($type) {
    $barista =  \Drupal::getContainer()
      ->get('coffee_shop.barista');
      ->get('Drupal\coffee_shop\Service\Barista');

    // ...
  }

}
```

Use the simple ::class Syntax

```php
// modules/custom/coffee_shop/src/Controller/CoffeeController.php

namespace Drupal\coffee_shop\Controller;

use Drupal\coffee_shop\Service\Barista;

class CoffeeController {

  public function brewCoffee($type) {
    $barista =  \Drupal::getContainer()
      ->get(Barista::class);

    // ...
  }

}
```

Because... if a class will only be registered once as a service...

...why add the extra layer of abstraction?

The "class" key is optional

```yaml
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
  Drupal\coffee_shop\Service\Barista:
    arguments: ['@config.factory']
```

Let's add a second service

@weaverryan

New CoffeeMachine Service

```php
// modules/custom/coffee_shop/src/Service/CoffeeMachine.php
namespace Drupal\coffee_shop\Service;

class CoffeeMachine {
  public function brew()
  {
    return 'coffee brewed!';
  }
}
```

Use it in Barista

```php
// modules/custom/coffee_shop/src/Service/Barista.php

class Barista {
  private $configFactory;
  private $coffeeMachine;

  public function __construct($configFactory, CoffeeMachine $coffeeMachine) {
    $this->configFactory = $configFactory;
    $this->coffeeMachine = $coffeeMachine;
  }

  // ...
}
```

Add service & update arguments

```
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
  Drupal\coffee_shop\Service\Barista:
    arguments:
      - '@config.factory',
+     - '@Drupal\coffee_shop\Service\CoffeeMachine'

+ Drupal\coffee_shop\Service\CoffeeMachine:
+   arguments: []
```

(2)

Autowiring

@weaverryan

Autowire: automatically add my arguments

```
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
  Drupal\coffee_shop\Service\Barista:
+    autowire: true
     arguments:
-      - '@config.factory',
-      - '@Drupal\coffee_shop\Service\CoffeeMachine'

  Drupal\coffee_shop\Service\CoffeeMachine:
    arguments: []
```

It works by reading the type-hint

```php
// modules/custom/coffee_shop/src/Service/Barista.php

class Barista {
  private $configFactory;
  private $coffeeMachine;

  public function __construct(
    ConfigFactoryInterface $configFactory,
    CoffeeMachine $coffeeMachine) {

    $this->configFactory = $configFactory;
    $this->coffeeMachine = $coffeeMachine;
  }

  // ...
}
```

Autowire: automatically add my arguments

```yaml
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
  Drupal\coffee_shop\Service\Barista:
    autowire: true
    arguments: []

  Drupal\coffee_shop\Service\CoffeeMachine:
    arguments: []
```

(3)

Default Service Config

@weaverryan

Default options for all services?

```
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
+ # defaults for services in this file
+ _defaults:
+   autowire: true
```

```
  Drupal\coffee_shop\Service\Barista:
    arguments: []

  Drupal\coffee_shop\Service\CoffeeMachine:
    arguments: []
```

(4)

Service Auto-Registration

@weaverryan

Auto-register your services

```yaml
# modules/custom/coffee_shop/coffee_shop.services.yml

services:
  _defaults:
    autowire: true

  # auto-registers all services in this directory
  # using the class name as the service id
  Drupal\coffee_shop\Service\:
    resource: '../src/Service'
```

Amazing!

@weaverryan

1) Create & use services and **touch zero config**

2) Add new constructor args to a service and **touch zero config**

It doesn't work :(

@weaverryan

_defaults & service auto-registration are not (yet) supported by Drupal :(

(5) & (6)

Let's build it ourselves!

A Service Provider

```php
// modules/custom/coffee_shop/src/CoffeeShopServiceProvider.php
namespace Drupal\coffee_shop;

use Drupal\Core\DependencyInjection\ContainerBuilder;
use Drupal\Core\DependencyInjection\ServiceProviderInterface;

class CoffeeShopServiceProvider implements ServiceProviderInterface {

  public function register(ContainerBuilder $container) {
    // called when the services are being built
  }

}
```

Find all files in the Service directory

```php
// modules/custom/custom/coffee_shop/src/CoffeeShopServiceProvider.php

public function register(ContainerBuilder $container)
{
  $finder = new Finder();
  $finder->in(__DIR__.'/Service')
    ->files()
    ->name('*.php');

  foreach ($finder as $fileInfo) {
    $class = 'Drupal\coffee_shop\Service\\'
        .substr($fileInfo->getFilename(), 0, -4);

    // ...
  }
}
```

Register them as autowired services!

```php
// modules/custom/custom/coffee_shop/src/CoffeeShopServiceProvider.php

public function register(ContainerBuilder $container) {
  // ...

  foreach ($finder as $fileInfo) {
    $class = 'Drupal\coffee_shop\Service\\'
      .substr($fileInfo->getFilename(), 0, -4);

    // don't override any existing service
    if ($container->hasDefinition($class)) {
      continue;
    }

    $definition = new Definition($class);
    $definition->setAutowired(true);
    $container->setDefinition($class, $definition);
  }
}
```

Smallest File in your App

```
# modules/custom/coffee_shop/coffee_shop.services.yml

# just crickets here!
```

(7)

Named Arguments

Add a scalar argument

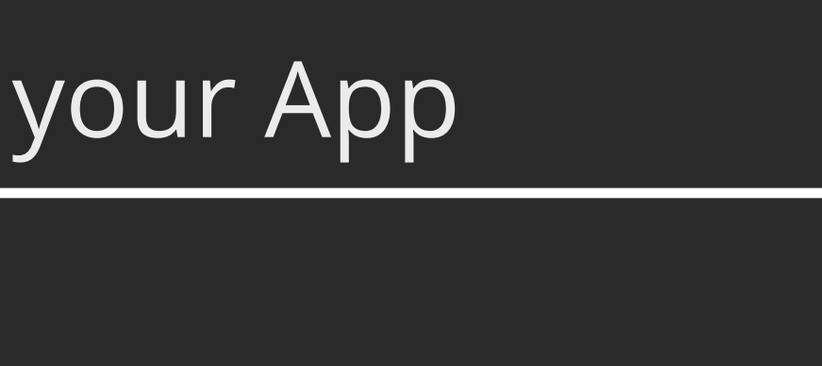
```php
// modules/custom/custom/coffee_shop/src/Service/CoffeeMachine.php
namespace Drupal\coffee_shop\Service;

class CoffeeMachine {
  private $numberOfScoops;

  public function __construct(int $numberOfScoops) {
    $this->numberOfScoops = $numberOfScoops;
  }

  public function brew()
  {
    return sprintf('using %d scoops!', $this->numberOfScoops);
  }
}
```

Autowiring Fails!!

```
[ERROR] Cannot autowire service
        "Drupal\coffee_shop\Service\CoffeeMachine": argument
        "$numberOfScoops" of method "__construct()" is type-hinted
        "int", you should configure its value explicitly.
```

Smallest File in your App

```yaml
# modules/custom/custom/coffee_shop/coffee_shop.services.yml
services:
  Drupal\coffee_shop\Service\CoffeeMachine:
    autowire: true
    arguments:
      $numberOfScoops: 3
```




```php
public function __construct(int $numberOfScoops) {
  $this->numberOfScoops = $numberOfScoops;
}
```

(8)

Only use the "New" Autowiring

@weaverryan

Smallest File in your App

```yaml
# modules/custom/custom/coffee_shop/coffee_shop.services.yml
parameters:
  container.autowiring.strict_mode: true

services:
  # ...
```

Autowiring Fails!!

```
[ERROR] Cannot autowire service "Drupal\coffee_shop\Service\Barista":
        argument "$configFactory" of method "__construct()"
        references interface
        "Drupal\Core\Config\ConfigFactoryInterface" but no such
        service exists. You should maybe alias this interface to the
        existing "config.factory" service.
```

Create "aliases" for your type-hints

```yaml
# modules/custom/custom/coffee_shop/coffee_shop.services.yml
services:
  # ...

  Drupal\Core\Config\ConfigFactoryInterface:
    alias: 'config.factory'
```

Ideally, core would add these aliases and empower autowiring

Wait, so, do what?

(1)

Use class names as your service ids and remove the *class* key.

@weaverryan

(2)

Use *autowire: true* and omit arguments that Drupal can guess.

(3) & (4)

Leverage a service provider to auto-register services & default them to autowire.

(5)

Specify any non-autowireable arguments with $named arguments

(6)

Turn off "magic" autowiring and add service aliases for the type-hints you want to autowire

THANK YOU!

Ryan Weaver
@weaverryan

Free Symfony Tutorial
https://symfonycasts.com/symfony

http://bit.ly/drupal-autowiring-code

SymfonyCasts

DrupalCon SEATTLE 2019