Auto Wiring

2023

Spryker

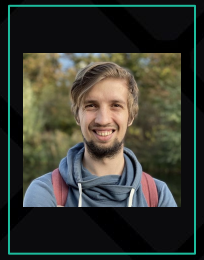
Team!

Chris Zepernick

Axel Beckert

Lukas Gabsi

Vasily Rodin

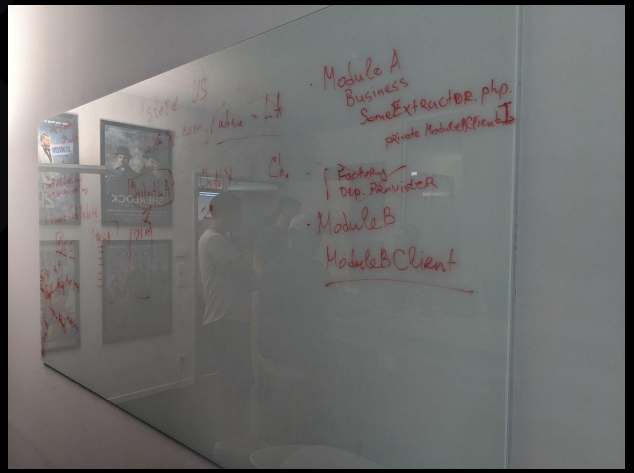


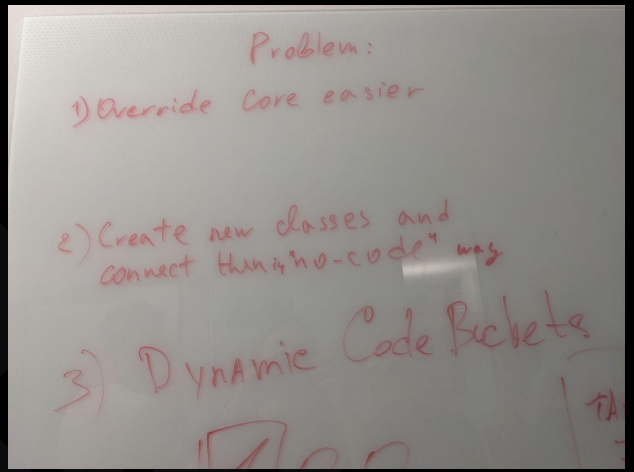
Volkan Akbulut

Volodymyr Lunov

Helen (Olena) Laktionova

Problem statement (1/2)

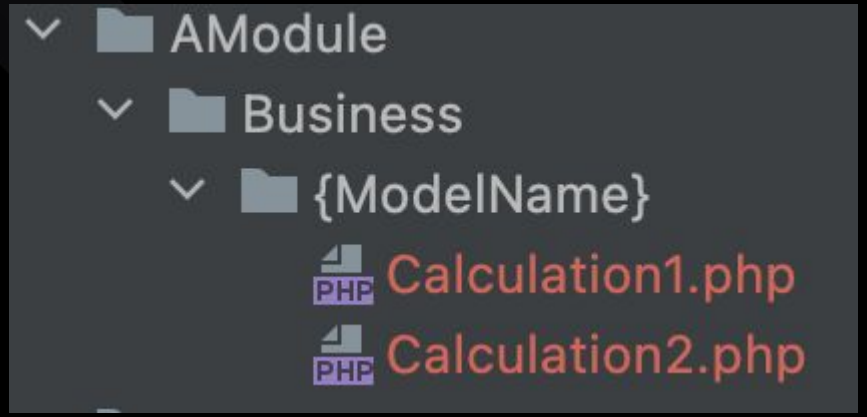




Problem statement (2/2)

Challenges

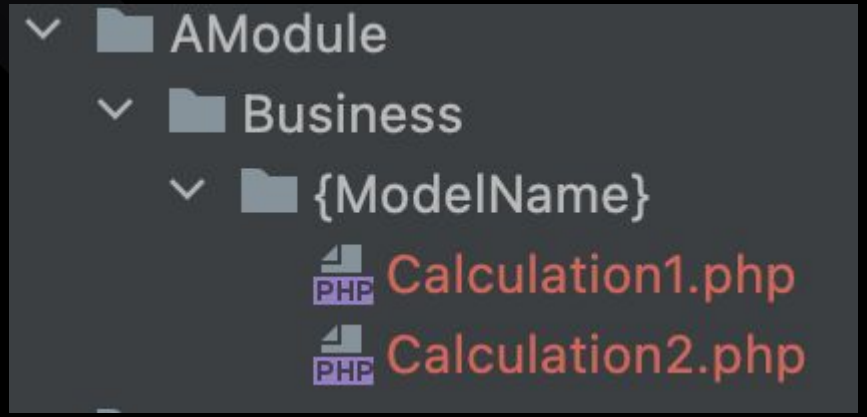
1. Override core easier.



Problem statement (2/2)

Challenges

1. Override core easier.
2. Create new classes and connect them in a “no code” way.



My last task in the project (just a single module)

Factory: 120 lines
DependencyProvider: 190 lines

What if we could get rid of most of this code?

Problem statement (2/2)

Challenges

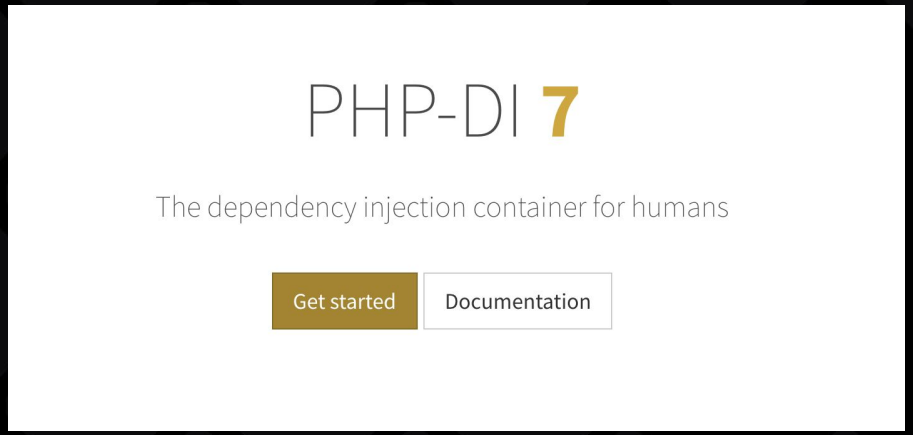
1. Override core easier.
2. Create new classes and connect them in a “no code” way.
3. “Dynamic code buckets”

Solution!

Auto Wiring!

Solution explanation

PHP-DI



+

Spryker

https://php-di.org/

How to use it

1. Install the package https://github.com/RodinVasily/spryker-autowiring
2. Include the trait (Yves, Zed and Client are supported)

```
use ClassResolverAwareTrait;
```

3. Use resolveClass method to autowire any class:

```
public function specialAddToCart($sku)
{
    $this->resolveClass(AmazingCartOperationInterface::class)
         ->specialAddToCart($sku);
}
```

You can always use the “classic” way if autowiring doesn’t work

```
$this->createAmazingCartOperation()
     ->specialAddToCart($sku);
```

Under the hood

There are 4 main cases:

- Injection of classes inside of one module (easy)
- Injection of spryker resolvable classes (not so easy)
- Injection of spryker core classes (not so easy)
- Extending of spryker core classes with new dependencies (hard and no supported!)

Customizations

\AW\Shared\Kernel\DependencyInjection\ConfigurableDependencyInjectionInterface

```
public function getDependencyInjectionConfig(): array
{
    return [
        AmazingCartOperationInterface::class =>
            DI\autowire(AmazingCartOperation::class)
                ->constructorParameter(
                    'postOperationPlugins',
                    $this->getPostOperationPlugins()
                )
    ];
}
```

More examples on https://php-di.org/doc/php-definitions.html#definition-types

Limitations & Open points

Not supported:

- Bridges
- Extending core classes

Open points:

- Compiling of containers for better performance

Result

Thank you!