

Dependency Injection Containers

VilniusPHP Susitikimas #3

Povilas Balzaravičius

2013 m. vasario 7 d.

Kas aš toks?

- Povilas Balzaravičius
- @Pawka
- github.com/pawka
- linkedin.com/in/pawka
- pawka.linija.net



Kas yra Dependency Injection?

Kuo blogas šis kodas?

```
1  class FeedGenerator {  
2  
3      protected $feed;  
4  
5      public function __construct() {  
6          // Create feed every 6 hours.  
7          $this->feed = new SomeFeed('0 */6 * * *');  
8      }  
9  }  
10 //..  
11 $generator = new FeedGenerator;
```

Kuo blogas šis kodas?

```
1 class FeedGenerator {  
2  
3     protected $feed;  
4  
5     public function __construct() {  
6         // Create feed every 6 hours.  
7         $this->feed = new SomeFeed('0 */6 * * *');  
8     }  
9 }  
10 //..  
11 $generator = new FeedGenerator;
```

- 1 Sudėtinga parašyti testą.
- 2 Gali tekti keisti kodą modifikuojant Feed klasę.
- 3 Teks modifikuoti FeedGenerator klasę, norint pakeisti SomeFeed vykdymo intervalą.

Ar dabar geriau?

```
1  class FeedGenerator {  
2  
3      protected $feed;  
4  
5      public function __construct($interval) {  
6          $this->feed = new SomeFeed($interval);  
7      }  
8  }  
9  //..  
10 $generator = new FeedGenerator('0 */6 * * *');
```

Ar dabar geriau?

```
1  class FeedGenerator {  
2  
3      protected $feed;  
4  
5      public function __construct($interval) {  
6          $this->feed = new SomeFeed($interval);  
7      }  
8  }  
9  //..  
10 $generator = new FeedGenerator('0 */6 * * *');
```

- 1 Sudėtinga parašyti testą.
- 2 Gali tekti keisti kodą modifikuojant Feed klasę.
- 3 SomeFeed klasė vis dar priklauso nuo FeedGenerator.

Ką daryti?

Iškelti vidinių objektų kūrimą
už klasės ribų!

Kodėl šis kodas geresnis?

```
1 namespace Feed;
2
3 class FeedGenerator {
4
5     protected $feed;
6
7     public function __construct(FeedInterface $feed) {
8         $this->feed = $feed;
9     }
10 }
11 //..
12 $feed = new SomeFeed('0 */6 * * *');
13 $generator = new FeedGenerator($feed);
```

- 1 Patogu rašyti testus (vietoj Feed galima paduoti mock objektą).
- 2 \$feed objektas nepriklauso nuo FeedGenerator klasės.
- 3 Modifikuojant FeedInterface klases, nereikės keisti FeedGenerator kodo.

Sveiki, aš DI

Štai Jums Dependency
Injection!

Objektų kūrimas

DI naudojimui konteineris nereikalingas.

```
1 $config = new \Doctrine\DBAL\Configuration();
2 $connectionParams = array(
3     'dbname' => 'mydb',
4     'user' => 'user',
5     'password' => 'secret',
6     'host' => 'localhost',
7     'driver' => 'pdo_mysql',
8 );
9 $conn = \Doctrine\DBAL\DriverManager::getConnection(
10     $connectionParams, $config);
11 $productService = new Product($conn);
12 $interval = "0 */6 * * *";
13 $feed = new \Feed\SomeFeed($interval);
14 $feed->setProductService($productService);
15 $feedGenerator = new \Feed\FeedGenerator($feed);
16 //Maybe add objects to some registry.
17 //..
```

Kas yra DI?

- Projektavimo šablonas (*design pattern*).
- Aktualus tik objektiniame programavime.
- Atskiria komponentus (*decoupling*).
- Standartizuoja ir suteikia galimybę centralizuoti objektų kūrimą Jūsų sistemoje.

DI būdai

```
1 $feed = new Feed("0 */6 * * *");
2
3 //Konstruktorius
4 $generator = new FeedGenerator($feed);
5
6 //Metodas (setter)
7 $generator = new FeedGenerator;
8 $generator->setFeed($feed);
9
10 //Atributas
11 $generator = new FeedGenerator;
12 $generator->feed = $feed;
```

Kam kurti savo jei yra..

DIC bibliotekos

Reikalavimai DI konteineriui

- Turi veikti greitai.
- Privalo dirbti su bet koku PHP objektu.
- Objektai neturi žinoti apie DI konteinerį.
- Nekurti objekto jei jis jau sukurtas.¹

¹Kai to reikia.

Pimple

Pimple

A simple Dependency Injection Container for PHP 5.3.

- Autorius: Fabien Potencier
- `pimple.sensiolabs.org`
- Paprastas.
- Greitas.
- Integravimas - kelios eilutės.

Darbas su objektais

```
1  $container = new Pimple();
2  // ..
3
4  $container['feed.somefeed.interval'] = '0 */6 * * *';
5  $container['feed.somefeed.class'] = '\Feed\SomeFeed';
6
7  $container['feed.somefeed'] = function($c) {
8      return new $c['feed.somefeed.class'](
9          $c['feed.somefeed.interval']
10     );
11 };
12
13 //Lazy loading
14 $container['feed.generator.class'] = '\Feed\FeedGenerator';
15 $container['feed.generator'] = $container->share(function($c) {
16     return new $c['feed.generator.class']($c['feed.somefeed']);
17 });
18
19 // ..
20 $manager = $container['feed.generator'];
```

Išskaidymas į modulius

```
1 class FeedContainer extends Pimple {  
2  
3     public function __construct() {  
4         $this['feed.somefeed.interval'] = '0 */6 * * *';  
5         $this['feed.somefeed.class'] = '\Feed\SomeFeed';  
6  
7         $this['feed.somefeed'] = function($c) {  
8             return new $c['feed.somefeed.class'](  
9                 $c['feed.somefeed.interval']  
10            );  
11        };  
12  
13        $this['feed.generator.class'] = '\Feed\FeedGenerator';  
14        $this['feed.generator'] = $this->share(function($c) {  
15            return new $c['feed.generator.class']($c['feed.somefeed']);  
16        });  
17    }  
18 }
```

Išskaidymas į modulius

```
1  class GlobalContainer extends Pimple {
2
3      public function __construct() {
4          $this['feeds'] = $this->share(function($c) {
5              return new FeedContainer;
6          });
7      }
8  }
9
10 //..
11 $container = new GlobalContainer;
12 //Change value
13 $container['feeds']['feed.somefeed.interval'] = '0 */12 * * *';
14 $generator = $container['feeds']['feed.generator'];
```

Sparta

Konteinerio kūrimas

```
./run-tests.sh  
Pimple  
265K  
579K  
0.00086s  
DependencyInjection  
267K  
931K  
0.006124s  
DependencyInjection (dumped)  
688K  
738K  
0.000495s
```

Šaltinis: <https://gist.github.com/igorw/3833123>

Symfony Dependency Injection Component

Symfony DIC

- Symfony komponentas.
- <http://symfony.com/components>
- Daug galimybių.
- Greitas (teisingai naudojant).
- Pagal nutylėjimą naudojamas Symfony2 karkase.

Palaikomi formatai

Prikalusomybes galima aprašyti šiais formatais:

- PHP
- XML
- YAML
- INI (Palaiko tik parametrų aprašymus).

Naudojimas: PHP

```
1 use Symfony\Component\DependencyInjection\ContainerBuilder;
2 use Symfony\Component\DependencyInjection\Reference;
3
4 $container = new ContainerBuilder();
5
6 $container->setParameter('feed.somefeed.interval', '0 */6 * * *');
7 $container
8     ->register('feed.somefeed', 'SomeFeed')
9     ->addArgument('%feed.somefeed.interval%');
10
11 $container
12     ->register('feed.generator', 'FeedGenerator')
13     ->addArgument(new Reference('feed.somefeed'));
```

Naudojimas: XML

```
1 <parameters>
2   <parameter key="feed.somefeed.interval">0 * / 6 * * * </parameter>
3 </parameters>
4
5 <services>
6   <service id="feed.somefeed" class="SomeFeed">
7     <argument>%feed.somefeed.interval%</argument>
8   </service>
9
10  <service id="feed.generator" class="FeedGenerator">
11    <argument type="service" id="feed.somefeed" />
12  </service>
13 </services>
```

Naudojimas: YAML

```
1 parameters:
2     # ...
3     feed.somefeed.interval: 0 */6 * * *
4
5 services:
6     feed.somefeed:
7         class:      SomeFeed
8         arguments:  [% feed.somefeed.interval%]
9     feed.generator:
10        class:      FeedGenerator
11        arguments:  [@ feed.somefeed]
```

DI perdengimas

- Palaiko YAML ir XML formatai.
- Leidžia perdengti anksčiau aprašytas DI priklausomybes (pvz. dirbant su Symfony2).
- Patogu naudoti skirtingas konfigūracijas: test, dev, live.

```
1 use Symfony\Component\DependencyInjection\ContainerBuilder;  
2 use Symfony\Component\Config\FileLocator;  
3 use Symfony\Component\DependencyInjection\Loader\XmlFileLoader;  
4  
5 $container = new ContainerBuilder();  
6 $loader = new XmlFileLoader($container, new FileLocator(__DIR__));  
7 $loader->load('services.xml');
```

Kompiliavimas

- Konteineris **gali** būti kompiliuojamas.

```
1 $container->compile();
```

- Aptinkamos klaidos:

- Neegzistuojančios priklausomybės.
- Ciklinės priklausomybės.

- Atliekama optimizacija.

- Sukompiliuotas rezultatas gali būti saugomas į diską ir naudojamas kaip *cache'as*.²

P.S. Symfony2 tuo rūpinasi pagal nutylėjimą :-)

²Pamenat spartos palyginimo rezultatus?

Sukompiliuoto rezultato saugojimas į diską

- Naudojamos Dumper klasės.
- Galima konvertuoti iš vieno formato į kitą.
- Palaikomi formatai:
 - PHP
 - XML
 - YAML
 - GraphWiz (www.graphwiz.org)

Sukompiliuoto rezultato saugojimas į diską

```
1 use Symfony\Component\DependencyInjection\ContainerBuilder;  
2 use Symfony\Component\DependencyInjection\Dumper\PhpDumper;  
3  
4 $file = __DIR__ . '/cache/container.php';  
5  
6 if (file_exists($file)) {  
7     require_once $file;  
8     $container = new ProjectServiceContainer();  
9 } else {  
10     $container = new ContainerBuilder();  
11     // ...  
12     $container->compile();  
13  
14     $dumper = new PhpDumper($container);  
15     file_put_contents($file, $dumper->dump());  
16 }
```

ProjectServiceContainer - pavadinimas pagal nutylėjimą.

Tagged servisai

- *Tag* - atributas, žymintis panašius objektus.
- Leidžia vykdyti analogiškus veiksmus visiems pažymėtiems objektams.

Tagged servisai

Modifikuojam FeedGenerator klasę, pridėdami kelių Feed objektų palaikymą:

```
1  class FeedGenerator {  
2  
3      protected $feeds = array();  
4  
5      public function __construct() {  
6      }  
7  
8      public function addFeed(FeedInterface $feed) {  
9          $this->feeds[] = $feed;  
10     }  
11 }
```

Tagged servisai

Turim aprašytas kelias skirtingas Feed klases.

```
1 <parameters>
2   <parameter key="feed.somefeed.interval">0 */6 * * * </parameter>
3   <parameter key="feed.otherfeed.interval">0 */12 * * * </parameter>
4 </parameters>
5
6 <services>
7   <service id="feed.somefeed" class="SomeFeed">
8     <argument>%feed.somefeed.interval%</argument>
9     <tag name="feed" />
10  </service>
11  <service id="feed.otherfeed" class="OtherFeed">
12    <argument>%feed.otherfeed.interval%</argument>
13    <tag name="feed" />
14  </service>
15
16  <service id="feed.generator" class="FeedGenerator" />
17 </services>
```

Tagged servisai

```
1 use Symfony\Component\DependencyInjection\ContainerBuilder;
2 use Symfony\Component\DependencyInjection\Compiler\CompilerPassInterface;
3 use Symfony\Component\DependencyInjection\Reference;
4
5 class FeedCompilerPass implements CompilerPassInterface
6 {
7     public function process(ContainerBuilder $container)
8     {
9         if (!$container->hasDefinition('feed.generator')) {
10             return;
11         }
12
13         $definition = $container->getDefinition('feed.generator');
14         $taggedServices = $container->findTaggedServiceIds('feed');
15         foreach ($taggedServices as $id => $attributes) {
16             $definition->addMethodCall('addFeed',
17                 array(new Reference($id)));
18         }
19     }
20 }
```

Išvados

- Norint pradėti naudoti DI, neprivaloma naudoti tam skirtos bibliotekos.
- DI verčia kodą rašyti teisingai.
- Paprasčiau atlikti pakeitimus.
- Paprasčiau (įmanoma?) rašyti testus.
- Ar verta DI naudoti egzistuojančiuose projektuose? TAIP (žr. aukščiau esančius punktus).

Resursai

- Symfony Dependency Injection component docs:
http://symfony.com/doc/current/components/dependency_injection/
- Fabien Potencier blog: What is dependency injection?
<http://fabien.potencier.org/article/11/what-is-dependency-injection>
- Inversion of Control Containers and the Dependency Injection pattern: <http://www.martinfowler.com/articles/injection.html>
- Zend DI: <https://packages.zendframework.com/docs/latest/manual/en/modules/zend.di.introduction.html>

Ačiū

Atsiliepimai: <https://joind.in/8105>