Wandu DI

전창완

Who am I

- 전창완 (im@wani.kr)
- XpressEngine / 대학생

Index

1. 왜 만들었을까?
2. 기능소개
    1. 의존성 주입(DI)
    2. Auto Resolve
    3. Auto Wiring
3. 어디에 활용할 수 있을까?

1 왜 만들었을까?

왜 만들었을까?

- Laravel의 Container는 너무나도 편리합니다.

- Packagist에는 너무나 좋은 컴포넌트들이 많습니다.

- 그 내용들을 특정 프레임워크에 종속되지 않고 사용하고 싶었습니다.
  즉, 요구사항에 맞춘 나만의 프레임워크를 만들 수 있다는 것!

는.. 그냥 대외적인 이야기고..
공부하면서 만들어 봤습니다...

2 기능소개

의존성주입

- “프로그래밍에서 구성요소간의 종속성을 소스코드에서 설정하지 않고 외부의 설정파일 등을 통해 컴파일 시점이나 실행 시점에 주입하도록 하는 디자인 패턴 중의 하나이다.”

- modernpug.github.io/php-the-right-way/#dependency_injection

```php
namespace Your\Service;

class MyService
{
    public function __construct()
    {
        $this->auth = new Authentication();
        $this->connection = new Database();
    }
}
```

```php
namespace Your\Service;

class MyService
{
    public function __construct(AuthInterface $auth, Connectable $connection)
    {
        $this->auth = $auth;
        $this->connection = $connection;
    }
}

$service = new MyService(new Authentication(), new Database());
```

```php
<?php
namespace Your\Service;

use ArrayObject;
use Wandu\Di\Container;
use Wandu\DI\ContainerInterface;

$app = new Container(new ArrayObject());

$app->closure(AuthInterface::class, function () {
    return new Authentication();
});
$app->instance(Connectable::class, new Database());

$app->closure(MyService::class, function (ContainerInterface $app) {
    return new MyService($app[AuthInterface::class], $app[Connectable::class]);
});

$app[MyService::class]; // MyService Class!
```

장점

- closure를 사용하면 객체를 불러오기 전까지 객체 생성을 늦출 수 있습니다. 즉, 등록된 클래스들 중에 사용할 클래스만 객체화가 이루어 집니다.
- DI로 인해 복잡한 의존관계를 쉽게 풀어줍니다.
- Mocking을 이용하여 테스트 하기 쉬워집니다.

사실 이 내용은 Pimple이라는
라이브러리에서도 제공합니다..

Auto Resolve

- Laravel에서 한번도 사용안한 사람은 있어도,
  한번만 사용한 사람은 없다는 기능!!

- 자동으로 의존성을 찾아서 넣어줍니다.

```php
<?php
namespace Your\Service;

use ArrayObject;
use Wandu\Di\Container;
use Wandu\DI\ContainerInterface;

$app = new Container(new ArrayObject());

$app->closure(AuthInterface::class, function () {
    return new Authentication();
});
$app->instance(Connectable::class, new Database());

$app->closure(MyService::class, function (ContainerInterface $app) {
    return new MyService($app[AuthInterface::class], $app[Connectable::class]);
});

$app[MyService::class]; // MyService Class!
```

```php
<?php
namespace Your\Service;

use ArrayObject;
use Wandu\Di\Container;

$app = new Container(new ArrayObject());

$app->bind(AuthInterface::class, Authentication::class);
$app->bind(Connectable::class, Database::class);

$app->bind(MyService::class);

$app[MyService::class]; // MyService Class!
```

알아서 의존성을 찾아서 넣어줌.

```php
namespace Your\Service;

class MyService
{
    public function __construct(AuthInterface $auth, Connectable $connection)
    {
        /* do something */
    }
}
```

```php
namespace Your\Service;

class MyService
{
    public function __construct(
        AuthInterface $auth,
        Connectable $connection,
        Others $other
    ) {
        /* do something */
    }
}
```

새로운 의존성은 그냥 추가만 하면 됨. (순서도 상관없음)

조금 더 사용하기 편하시라고..

Auto Wiring

- 스프링에서 제공하는 그것!

- Annotation을 통해 의존성을 주입합니다.

```php
namespace Wandu\DI\Stub;

class ClientWithAutoWired
{
    /**
     * @Autowired
     * @var \Wandu\DI\Stub\DepInterface
     */
    private $dep;

    /**
     * @return \Wandu\DI\Stub\DepInterface
     */
    public function getDep()
    {
        return $this->dep;
    }
}
```

@Autowired 만 추가하면 끝.
(@var도 지정해야함, PHPDoc)

```php
$app->bind(DepInterface::class, DepFoo::class);
$app->wire(ClientWithAutoWired::class);

$app[ClientWithAutoWired::class];
```

wire매서드를 통해서 의존성을 주입.

그외에..

- Service Provider
- call (callable에서 auto resolve를 사용하고 싶다면..)
- create (등록되지 않은 클래스를 만들고 싶다면..)
- github.com/wandu/di

3 어디에 활용할 수 있을까?

나만의 프레임워크 만들기

- 개발 요구사항에 맞춰서 필요한 컴포넌트를 검색합니다.

- 각 컴포넌트 별로 Service Provider를 제작합니다.

- 예를들면..
  phly/http + fast-route + twig + phormium + whoops

오래된 소스 리팩토링

- 한번에 모던한 프레임워크로 도약하면 안되는 서비스일 경우,
  조금씩 바꿔서 사용하자!

- blog.wani.kr/dev/php/wandu/wandu-di

- gist.github.com/wan2land/87e13581956faca90033

참고자료

- Clean Code
- PHP: the right way (한글판)
  modernpug.github.io/php-the-right-way
- Laravel Manual (한글판)
  xpressengine.github.io/laravel-korean-docs