新創公司與軟體架構

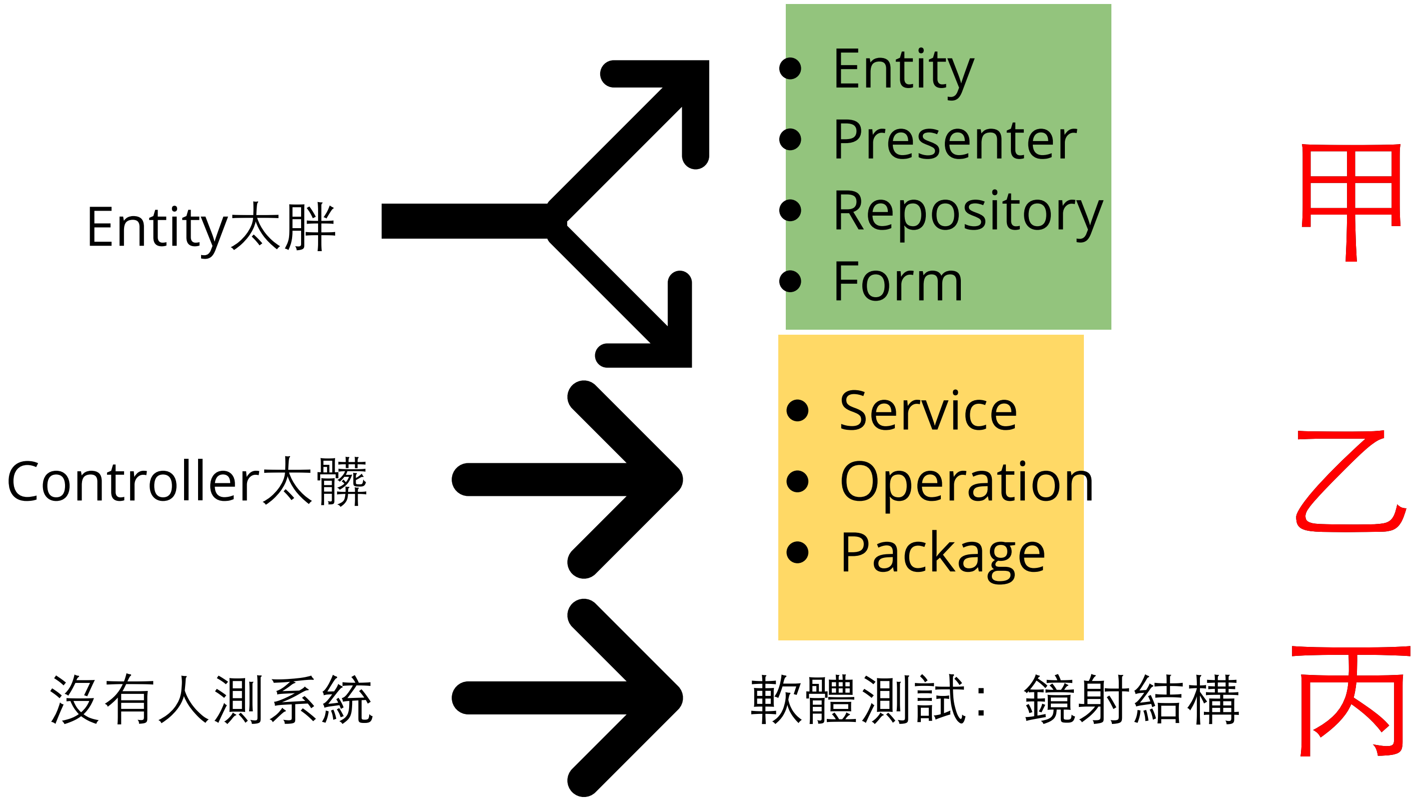
by 尤川豪

新創公司的挑戰?

一元翻譯

程式碼很難改的原因

- Entity（Model）太胖
- Controller太髒
- 沒有人測系統



Entity太胖

- Entity
- Presenter
- Repository
- Form

甲

Controller太髒

- Service
- Operation
- Package

乙

沒有人測系統

軟體測試：鏡射結構

丙

甲

Entity太胖

- Entity
- Presenter
- Repository
- Form

Entity

Presenter

把日期、金額、名稱之類的呈現（presentation）邏輯抽離出來！

問題：presentation易讓entity過胖

```
class Article
{
    public function getDate(){/*...*/}

    public function getTaiwaneseDateTime(){/*...*/}

    public function getWesternDateTime(){/*...*/}

    public function getTaiwaneseDate(){/*...*/}

    public function getWesternDate(){/*...*/}
}
```

step 1. 抽出presentation logic

```php
class ArticlePresenter
{
    protected $article;

    public function __construct(Article $article)
    {
        $this->article = $article;
    }

    public function getTaiwaneseDateTime(){
        return date('Y-m-d', $this->article->created_
    }

    public function getWesternDateTime(){/*...*/}
    public function getTaiwaneseDate(){/*...*/}
    public function getWesternDate(){/*...*/}
}
```

step 2. 從entity連到presenter

```
class Article
{
    public function present()
    {
        return new ArticlePresenter($this);
    }
}
```

step 3.

```
$article->present()->getTaiwaneseDate();
```

Repository

把查詢（query）的邏輯，也就是取得entity
的各種方式抽離出來！

step 1. 抽出query logic

```php
class UserRepository
{
    public function getPopularWomen()
    {
        return User::where('votes', '>', 100)
            ->whereGender('W')
            ->orderBy('created_at')
            ->get();
    }

    public function getActiveUsers(){}
    public function getPaidUsers(){}
}
```

step 2.

```
$repository = new UserRepository();

$users = $repository->getPopularWomen();
```

Form

把參數驗證（validation）的邏輯（例如字串長度、日期、金額大小）抽離出來！

問題： 參數驗證的code，放哪好呢

```
$validation = Validator::make(
    array(
        'title' => $title,
        'content' => $content,
    ),
    array(
        'title' => array( 'required', 'alpha_dash' ),
        'content' => array( 'required' ),
    )
);

return $validation->passes();
```

step 1. 封進Form

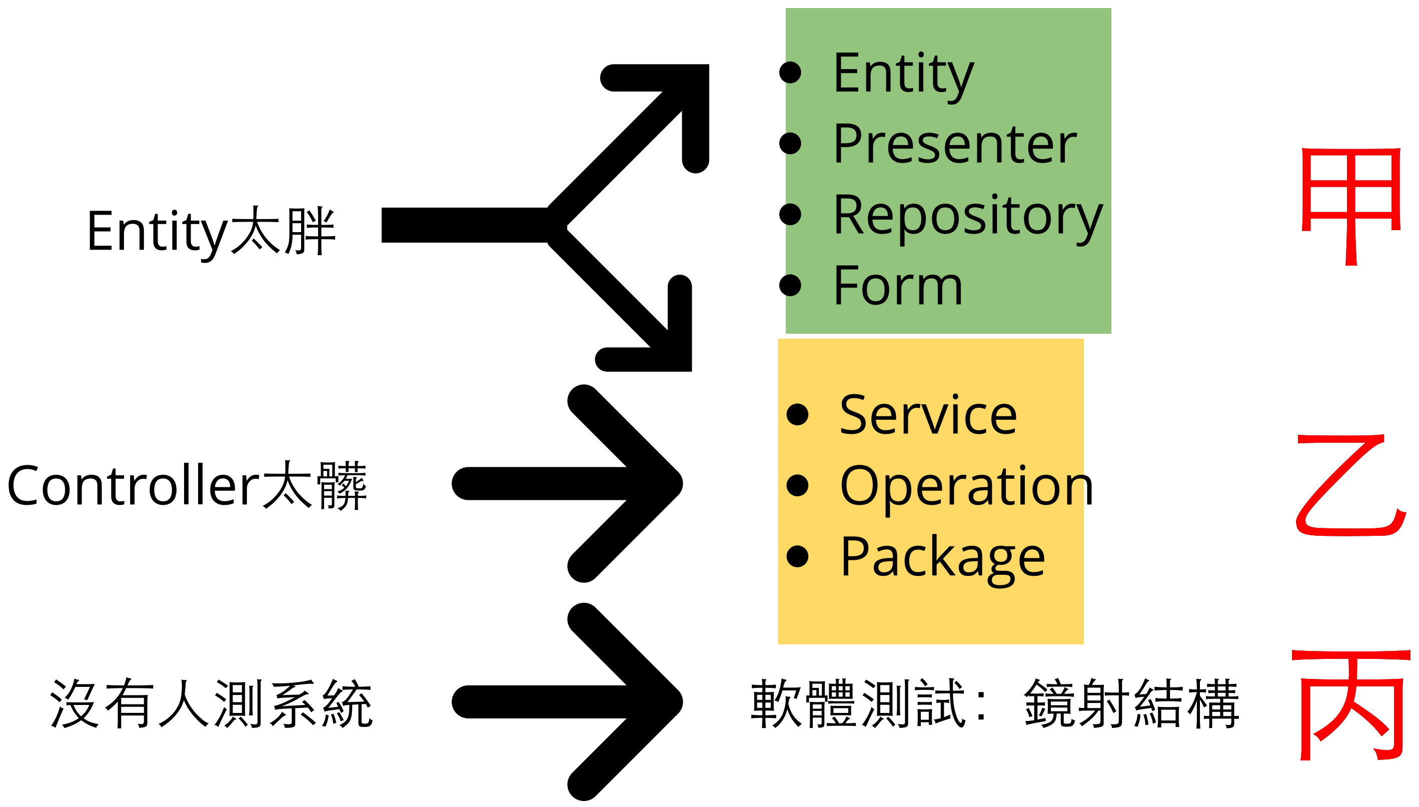
```
class ArticleForm
{
    protected $validationRules = [
        'title' =>  array( 'required', 'alpha_dash' )
        'content' => array( 'required' )
    ];
    protected $validator;
    public function isValid($input)
    {
        $this->validator = Validator::make($input, $t
        return $this->validator->passes();
    }
    public function getErrors()
    {
        return $this->validator->errors();
    }
```

step 2.

```
$form = new ArticleForm();

if ( ! $form->isValid(Input::all()) ){
    return Redirect::back()->with( [ 'errors' => $for
}

// Passed the validation.
// Create the article.
```


Entity太胖
Controller太髒
沒有人測系統
Entity
Presenter
Repository
Form
Service
Operation
Package
軟體測試：鏡射結構
甲
乙
丙

乙

Entity太胖、Controller太髒

- Service
- Operation
- Package

Service

1. 牽扯到外部行為
2. controller內像又不像的商業邏輯
3. 牽扯到多種entity

#1 牽扯到外部行為

- 跑測試時，不想觸發外部行為
  （送email、透過網路呼叫第三方API...etc）
- 應用constructor injection
- 測試時，做mocks傳進去

範例：過程中，呼叫Google Drive備份

step 1. 封成service

```php
class TranslatorAssign
{
    protected $googleDrive;

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

    public function execute($user, $document)
    {
        $user->doSomething();

        $this->googleDrive->backup($document->file_id
    }

}
```

step 2. 跑測試

```
//寫一個有backup方法的假類別GoogleDriveMock
$service = new TranslatorAssign(new GoogleDriveMock()

$service->execute($user, $document);

$this->assertEquals(Document::ASSIGNED_STATUS, $docum

```

```
/*  用test framework支援的mocks也可以
    $stub = $this->getMockBuilder('GoogleDrive')
                 ->getMock();

    $stub->method('backup')
         ->willReturn('foo');

    $service = new TranslatorAssign($stub);
*/
```

step 3.

```
$service = new TranslatorAssign(new GoogleDrive());

$service->execute($user, $document);

//Return successful page
```

#2 controller內 像又不像的商業邏輯

- 感覺封不封進service，都可以嗎?
- 想測試就封

範例：controller內，單純的條件判斷，該封嗎?

```
$document = Document::find(Input::get('id'));

if(Input::get('role') == 'translator'){
    $document->doSomethingForTranslator();
    $document->doAnotherThingForTranslator();
}else if(Input::get('role') == 'editor'){
    $document->doSomethingForEditor();
    $document->doAnotherThingForEditor();
}

//Return successful page
```

step 1. 封成service

```php
class GetAccessPermission
{
    //有想避開的dependency，就用constructor injection
    public function __construct(/*...*/)
    {
        //...
    }
    public function execute($document, $role)
    {
        if($role == 'translator'){
            $document->doSomethingForTranslator();
            $document->doAnotherThingForTranslator();
        }else if($role == 'editor'){
            $document->doSomethingForEditor();
            $document->doAnotherThingForEditor();
        }
    }
}
```

step 2. 跑測試

```
$service = new GetAccessPermission();

$service->execute($document, 'translator');

$this->assertTrue($document->has_translator);

$this->assertTrue($document->blah_blah);
```

step 3.

```
$document = Document::find(Input::get('id'));

$service = new GetAccessPermission();

$service->execute($document, Input::get('role'));

//Return successful page
```

#3 牽扯到多種entity

- 如果某段code放進哪種entity都有點怪的話...
- 索性獨立成service

範例：結帳牽扯到 User, Order, Product

```
class CheckoutBill
{

    public function execute($user, $order, $product)
    {
        //...
    }

}
```

Operation

1. 發現某些service必須連續執行
2. feature幾乎可以獨立開專案

#1 發現某些service 必須連續執行

- 總是連續執行某幾個service，發現各自幾乎不獨立
- 改寫成operation，用Facade Pattern封裝
- 整個operation都透過facade對外溝通

範例：計算專案金額、促銷價、交件日期

```
$calcDueDate = new CalcDueDate();

$calcPrice = new CalcPrice();

$calcDiscount = new CalcDiscount();
```

問題：好幾個地方duplicate

```
$calcDueDate->execute($order);

$calcPrice->execute($order);

$calcDiscount->execute($order);
```

step 1. Facade Pattern封裝

```
class QuotationManager
{
    protected $calcDueDate;
    protected $calcPrice;
    protected $calcDiscount;
    public __construct()
    {
        $this->calcDueDate = new CalcDueDate();
        $this->calcPrice = new CalcPrice();
        $this->calcDiscount = new CalcDiscount();
    }
    public function execute($order)
    {
        $this->calcDueDate($order);
        $this->calcPrice($order);
        $this->calcDiscount($order);
    }
}
```

step 2.

```
$quotation = new QuotationManager();

$quotation->execute($order);
```

#2 feature幾乎可以獨立開專案

- 即將開發的功能幾乎獨立於原本的專案之外
- 甚至可以放entity進去
- 用Facade Pattern封裝。盡量以此對外溝通

範例：幾乎獨立的翻譯輔助工具

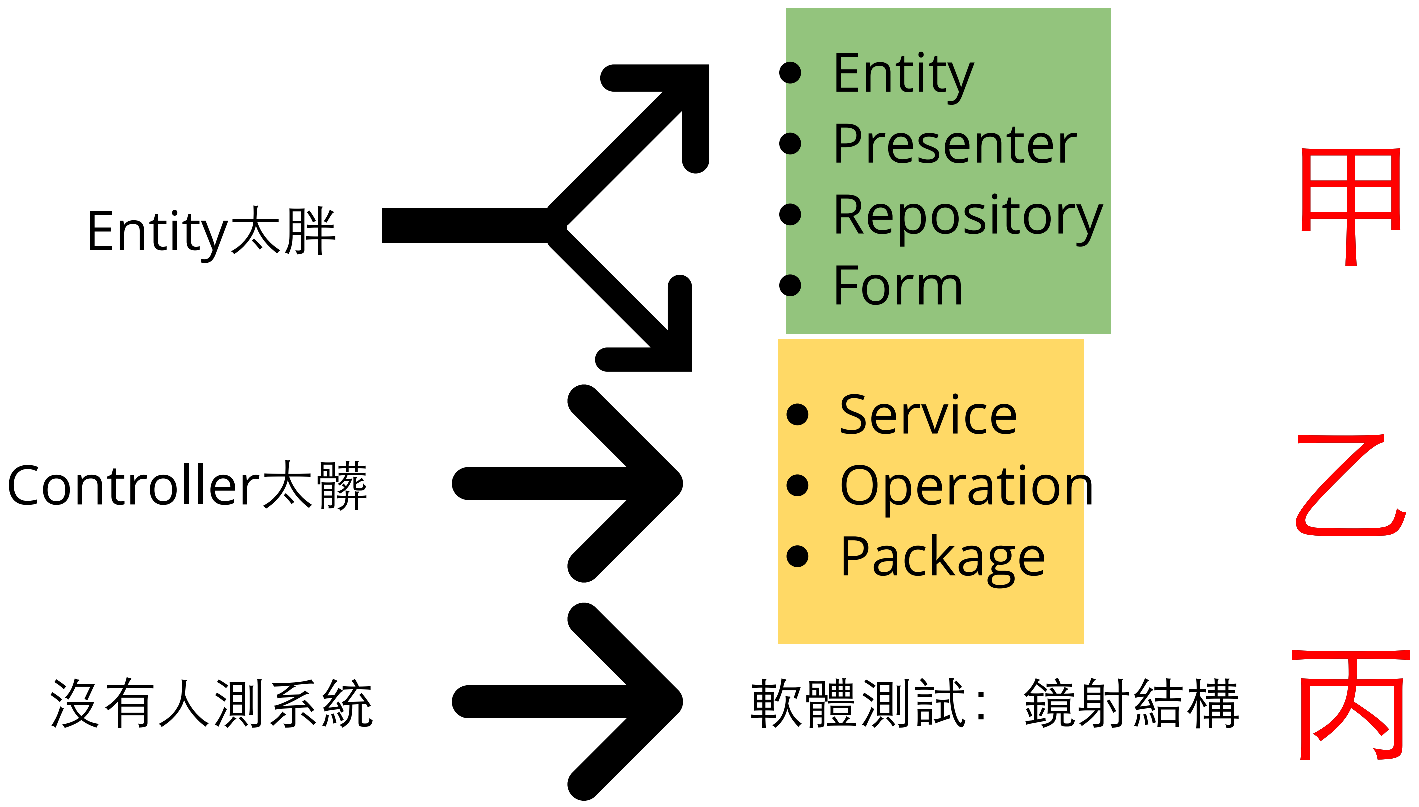
```
/MyApp
    /GlossarySystem
        /Manager.php
        /Analyzer.php
        /Generator.php
        /Parser.php
        /Splitter.php
        /Entity
            /Text.php
            /Segment.php
```

Package

把其他公司也能使用、

概念上獨立於當前專案的程式碼抽離出來！

```
/MyApp
    /...
    /...
/Howtomakeaturn (Github帳號)
    /MyPackage1
        /...
    /MyPackage2
        /...
    /MyPackage3
        /...
```


Entity太胖
Controller太髒
沒有人測系統
Entity
Presenter
Repository
Form
Service
Operation
Package
軟體測試：鏡射結構
甲
乙
丙

丙

軟體測試：鏡射結構

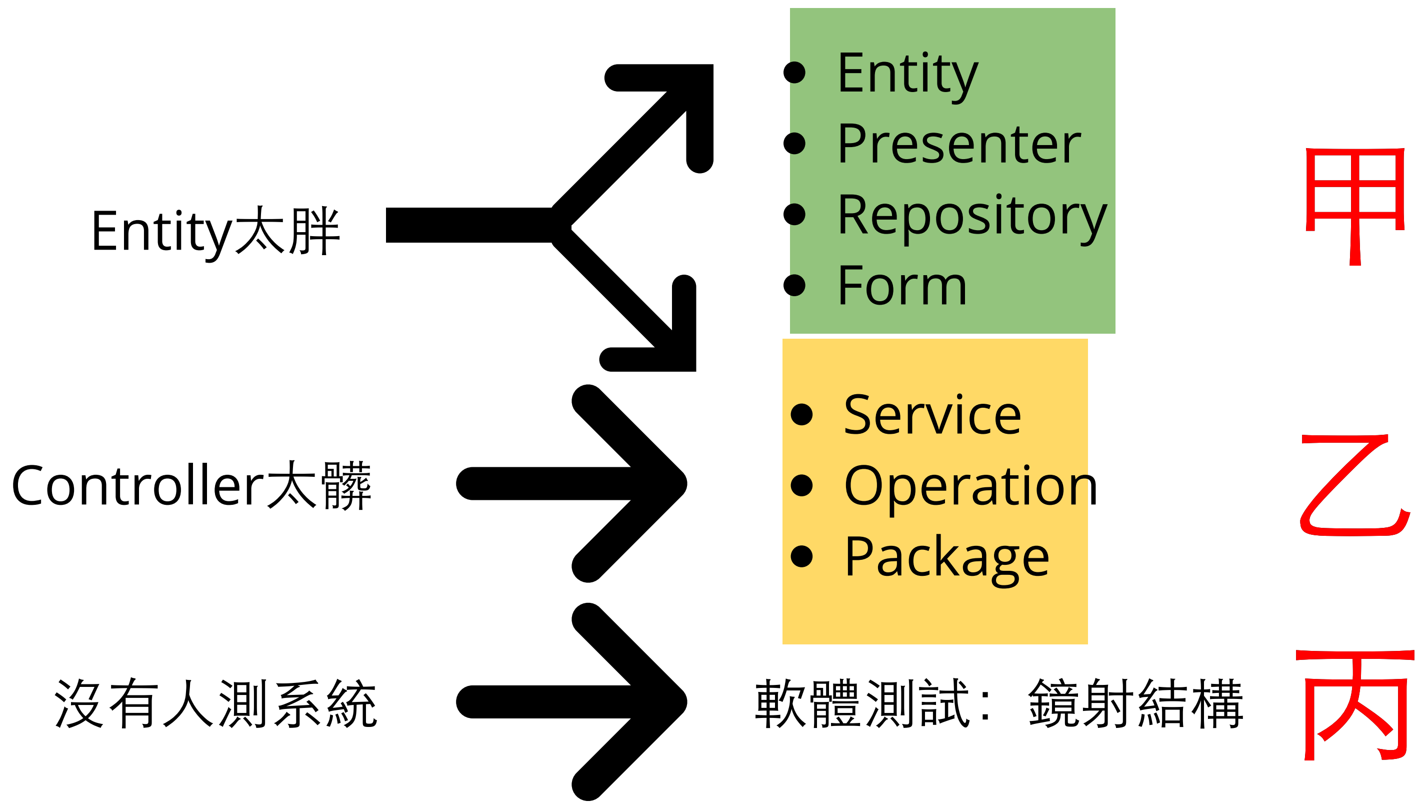
tests的檔案結構，與App核心一模一樣

```
/MyApp
    /Order
        /Order.php
        /OrderRepository
    /Product
        /Product.php
        /ProductPresente
    /Service
        /TranslatorAssig
        /GetAccessPermis
        /CheckoutBill.ph
    /...
```

```
tests/MyApp
    /Order
        /OrderTest.php
        /OrderRepositoryTest
    /Product
        /ProductTest.php
        /ProductPresenterTes
    /Service
        /TranslatorAssignTes
        /GetAccessPermission
        /CheckoutBillTest.ph
    /...
```

tests的檔案結構，與App核心一模一樣

- 設計方便、開發速度快
- 不細分測試種類（unit tests/integration tests...etc）


Entity太胖
Controller太髒
沒有人測系統
Entity
Presenter
Repository
Form
Service
Operation
Package
軟體測試：鏡射結構
甲
乙
丙

延伸閱讀

http://blog.turn.tw/?page_id=2742

檔案結構範例

https://github.com/howtomakeaturn/phpconf2015

謝謝大家<(_ _)>