



系統重構流程

改善方案內容與實務操作方法

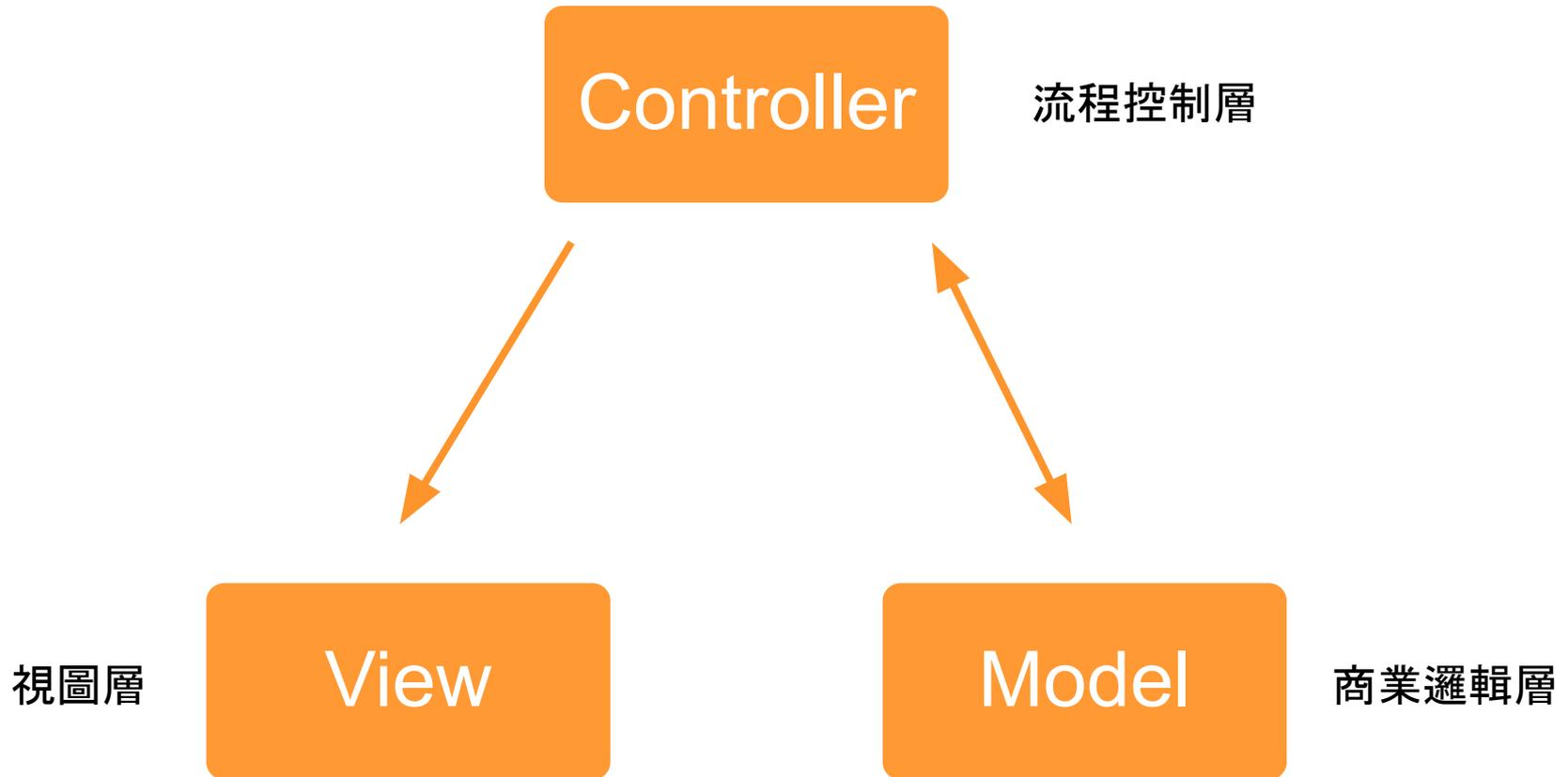
目錄

- 現有架構的限制
 - 原 CodeIgniter 架構:MVC
 - 現行架構:MVC + Service
 - 現行架構的限制
- 改善方案
 - 加入領域模型層(Domain Layer)
 - 領域模型
 - 改善項目
 - 引入領域模型的隱憂
 - 維護模型文件
 - 現況
 - 其他需求
- 實務操作
 - 重構範例:回歸測試
 - 重構範例:擷取函示
 - 重構範例:新類別(領域模型)
 - 重構範例:單元測試

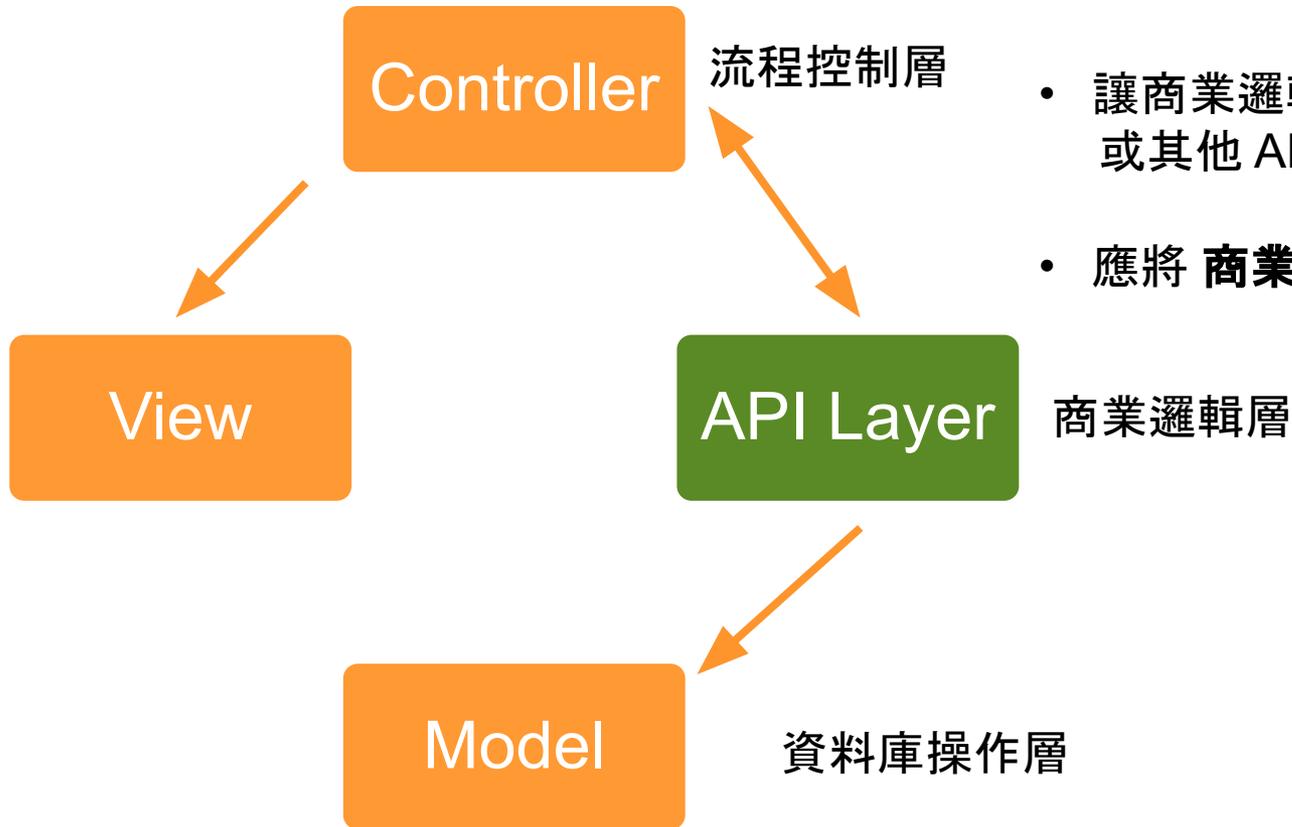


現行架構的限制

原 CodeIgniter 架構:MVC



現行架構: MVC + Service



- API 層為系統的商業邏輯(Service)層
- 讓商業邏輯可以共用於控制器或其他 API 之間
- 應將 **商業邏輯移至 API 層**

現行架構的限制 (1/4)

Controller

API Layer

Model

- API 只是控制器的延伸邏輯，**沒辦法直接代表某個系統功能**
- 系統功能的概念散落在 **控制器** 與 **API 層** 之間，導致 **沒辦法將系統功能概念化**
- API **缺少結構**，導致常須使用暫時變數或邏輯，**彌補缺少結構的不足**

現行架構的限制 (2/4)

經常需要臨時變數跟邏輯，
彌補缺少結構的不足

```
$form_group = $this->assessment_api->get_form_belonged_group($form_id);
$form_group_ids = count($form_group) ? array_column($form_group, 'group_id') : [];
$parent_group_list = $this->group_api->get_group_path_array($group_id);
$sub_group_list = $this->group_api->get_sub_group_id_array($group_id);
$search_group_data = array_merge((array)$group_id, $parent_group_list, $sub_group_list, $form_group_ids);
// 取得審核列表
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search_group_data);
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow['process'] = $this->workflow_api->get_form_default_workflow_selected($default_workflow['form_id'], $search_group_data);
    if (count($default_workflow['process'])) {
        foreach ($default_workflow['process'] as &$process) {
            $process['user'] = $this->workflow_api->get_form_default_workflow_selected($process['form_id'], $search_group_data);
            $user_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    $user_name[] = $user['name'];
                } else {
                    $user_name[] = $user['role_name'];
                }
            }
            $process['user_name'] = implode(' ', $user_name);
        }
    }
}
```

臨時
邏輯

臨時變數

現行架構的限制 (3/4)

系統功能的概念散落於控制器與 API 之間
導致沒辦法將系統功能 概念化

```
public function get_default_workflow_list()
```

```
$form_group = $this->assessment_api->get_form_belonged_group($form_id);  
$form_group_ids = count($form_group) ? array_column($form_group, 'group_id') : [];  
$parent_group_list = $this->group_api->get_group_path_array($group_id);  
$sub_group_list = $this->group_api->get_sub_group_id_array($group_id);  
$search_group_data = array_merge((array)$group_id, $parent_group_list, $sub_group_list, $form_group_ids);  
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search_group_data);  
foreach ($default_workflow_list as &$default_workflow) {  
    $default_workflow['process'] = $this->workflow_api->get_default_workflow_process($default_workflow);  
    if (count($default_workflow['process'])) {  
        foreach ($default_workflow['process'] as &$process) {  
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process['default_user_id']);  
            $user_name = array();  
            foreach ($process['user'] as $user) {  
                if ($user['default_user_id'] !== null) {  
                    $user_name[] = $user['name'];  
                } else {  
                    $user_name[] = $user['role_name'];  
                }  
            }  
            $process['user_name'] = implode( ' ', $user_name);  
        }  
    }  
}
```

現行架構的限制 (4/4)

```
public function get_default_workflow_list()
{
    // $user_id = $this->session->userdata('user_id');
    $form_id = (int)$this->input->post('form_id');
    $group_id = $this->input->post('group_id');
```

這裏其實都是在操作 **預設簽核** 的邏輯

```
$form_group = $this->assessment_api->get_form_belonged_group($form_id);
$form_group_ids = count($form_group) ? array_column($form_group, 'group_id') : [];
$parent_group_list = $this->group_api->get_group_path_array($group_id);
$sub_group_list = $this->group_api->get_sub_group_id_array($group_id);
$search_group_data = array_merge((array)$group_id, $parent_group_list, $sub_group_list, $form_group_ids);

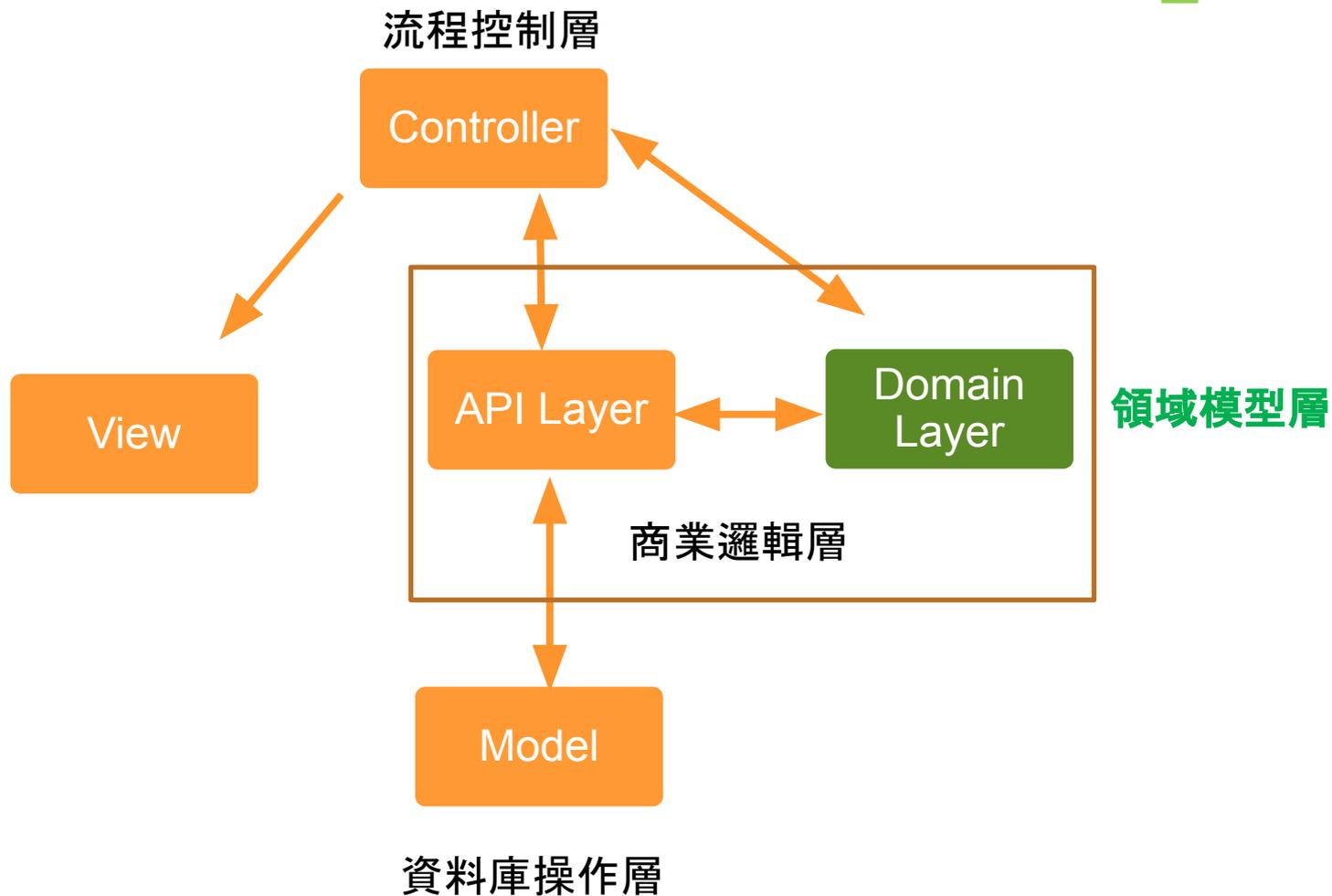
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search_group_data);
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow['process'] = $this->workflow_api->get_default_workflow_process($default_workflow);
    if (count($default_workflow['process'])) {
        foreach ($default_workflow['process'] as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process['default_workflow_id']);
            $user_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    $user_name[] = $user['name'];
                } else {
                    $user_name[] = $user['role_name'];
                }
            }
            $process['user_name'] = implode( ' ', $user_name);
        }
    }
}
```



改善方案

加入領域模型層

加入領域模型層



領域模型 (1/7)

領域模型 是由 **現實業務實體** 所歸類出來的

```
Default_workflow
├── f default_workflow_id
├── f create_user_id
├── f create_time
├── f group_id
├── f title
├── f is_delete
├── m __construct(default_workflow, repository)
├── m process()
├── m process_user(sequence)
```

- **領域模型**, 是一種歸類
- 為 **業務** 和 **技術實現** 之間的橋樑
- 可減輕認知的負擔
- **避免重複的工作** 和思考
- 提升工程師學習的速度
- 提供團隊一個共同的語言

預設簽核模型

* **領域模型** 應該捕捉 **業務規則** 或 **領域邏輯** (business rules / domain logic)

領域模型 (2/7)

```
// 取得簽核列表
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search)
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow['process'] = $this->workflow_api->get_default_workflow_process($default_workflow)
    if (count($default_workflow['process'])) {
        foreach ($default_workflow['process'] as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process['default_user_id'])
            $user_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    $user_name[] = $user['name'];
                } else {
                    $user_name[] = $user['role_name'];
                }
            }
            $process['user_name'] = implode( glue: ',', $user_name);
        }
    }
}
```

業務實體



由業務實體歸納出領域模型

```
Default_workflow
├── default_workflow_id
├── create_user_id
├── create_time
├── group_id
├── title
├── is_delete
├── __construct(default_workflow, repository)
├── process()
├── process_user(sequence)
```

領域模型

領域模型 (3/7)

```
// 取得簽核列表
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search)
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow['process'] = $this->workflow_api->get_default_workflow_process($default_workflow)
    if (count($default_workflow['process'])) {
        foreach ($default_workflow['process'] as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process['default_workflow_id'])
            $user_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    $user_name[] = $user['name'];
                } else {
                    $user_name[] = $user['role_name'];
                }
            }
            $process['user_name'] = implode( glue: ',', $user_name);
        }
    }
}
```

用 領域模型 取代業務實體



減輕認知的負擔

```
// 取得簽核列表
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search)
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow = Clouder\Workflow\Factory::Default_workflow($default_workflow);
}
}
```

領域模型 (4/7)

```
// 取得簽核列表
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search)
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow['process'] = $this->workflow_api->get_default_workflow_process($default_workflow)
    if (count($default_workflow['process'])) {
        foreach ($default_workflow['process'] as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process['default_workflow_id'])
            $user_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    $user_name[] = $user['name'];
                } else {
                    $user_name[] = $user['role_name'];
                }
            }
            $process['user_name'] = implode( glue: ',', $user_name);
        }
    }
}
```

×6

程式碼中有六段差不多的邏輯
都可以用 **領域模型** 取代 **實體**

工程師不須每次再學習重複的邏輯

避免重複的工作和思考

提升工程師學習系統的速度

```
// 取得簽核列表
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search)
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow = Clouder\Workflow\Factory::Default_workflow($default_workflow);
}
}
```

領域模型 (5/7)

幫助系統功能概念化

新增課程

課程類別	一般課程 講座課程	
課程分類	請選擇課程分類	開課單位 請選擇開課單位
課程名稱		
教師身份	院內教師 單位: 帳號輸入	上課老師 輸入教師帳號
協同教師	單位: 帳號輸入	+
院內教育學分	無	繼續教育積分 無 無選項
參加人數限制	0 人 (0 表示為無人數限制)	
課程時數 (單位: 分鐘)		假別 無
課程開始時間	2019-01-30 0 時 0 分	課程結束時間 2019-01-30 0 時 0 分
報名開始時間	2019-01-27 0 時 0 分	報名結束時間 2019-01-29 23 時 59 分
學生自行報名	<input checked="" type="radio"/> 是 <input type="radio"/> 否	
課程日期(複選)		
課程地點	其他	請輸入地點
課程完成條件 (可複選)	評核	
	<input type="checkbox"/> 筆試測驗	<input type="checkbox"/> 實地操作
	<input type="checkbox"/> 書面報告	<input type="checkbox"/> 其他
	<input type="checkbox"/> mini-CEX 請選擇表	<input type="checkbox"/> Cbd 請選擇表
		<input type="checkbox"/> 口頭測驗
		<input type="checkbox"/> DOPS 請選擇表
		<input type="checkbox"/> 線上測驗 通過份數
	紀錄	
	<input type="checkbox"/> 心得報告	<input type="checkbox"/> 實例討論
	<input type="checkbox"/> 指定紀錄單 請選擇	<input type="checkbox"/> 附件上傳
	其他	
	<input type="checkbox"/> 出席參與	<input type="checkbox"/> 輔助教材
		<input type="checkbox"/> 其他表單 請選擇表
填寫問卷	<input type="radio"/> 是 <input checked="" type="radio"/> 否	
課程備註		

```
Course_creator
├── assistants
├── start_time
├── assessments
├── course_name
├── group_id
├── teacher_id
├── repo
├── end_time
├── teacher_type
├── place
├── __construct(repository)
├── set_teacher_type(teacher_type)
├── set_assistants(assistants)
├── set_place(place)
├── set_start_time(start_time)
├── set_end_time(end_time)
├── set_assessments(assessments)
└── create()
```

開課功能



課程建造器模型

領域模型 (6/7)

表單模型

```
Form
  __construct(failure_form_data, repository)
  set_workflow(workflow)
  workflow_process()
  workflow_process_user(sequence)
  update_process_user(sequence, sign_user_id)
  ready_to_send()
  make_form_format_workflow_process(workflow_id)
```

組合領域模型
減少重複的邏輯

委派 表單簽核模型

```
Form_workflow
  __construct(publish_failure_form_id, repository)
  process()
  process_user(sequence)
```

取得表單簽核流程資料

```
Workflow_api
  __construct()
  get_default_workflow(default_workflow_id, user_id, search_group)
  get_default_workflow_process(default_workflow_id)
  get_default_workflow_process_user(default_process_id)
  get_publish_failure_form_workflow_process(default_workflow_id)
  get_publish_failure_form_workflow_process_user(default_process_id)
```

領域模型 (7/7)

提供團隊一個共同的語言

Course_creator
課程建造器模型
根據開課畫面的設定
建立課程資料



create()
開始建立課程
[Observer 觀察者模式]

視情況加入：

Teacher_setter
負責設定老師資料

Assessment_creator
負責建立評核資料

Signup_processor
負責報名學生邏輯
(若有的話)

Course_copier
負責複製課程邏輯
(若有的話)

18

開發團隊開始可以用 領域模型 來溝通業務的處理流程或實現方法

改善項目

現行架構的限制	影響	引入領域模型
<ul style="list-style-type: none">系統功能的概念散落在控制器與API層之間	<ul style="list-style-type: none">寫一個功能需要遍歷多個檔案須使用暫時變數或邏輯彌補缺少結構的不足	有結構的 領域模型 改善： <ul style="list-style-type: none">減輕認知的負擔去除重複的工作和思考提升工程師學習系統的速度
<ul style="list-style-type: none">API 沒辦法直接代表某個系統功能	<ul style="list-style-type: none">沒辦法將系統功能概念化	提供團隊一個共同的語言 開發團隊可以用 領域模型 溝通業務的處理流程或實現方法

引入領域模型的隱憂

隱憂：	改善方法：
類別數量會變多	也是優點，類別符合單一職責，只有單一理由會修改類別 (Debug 或需求異動)
須維護領域模型文件	<p>領域模型必須被持續學習跟關注</p> <p>團隊新增/修改每個 領域模型 後，必須讓每位成員得知異動項目。</p> <p>文件內容應包含：</p> <ul style="list-style-type: none">• 領域模型的用途 (解決哪個業務需求)、• 領域模型的屬性與方法、• 領域模型如何組合其他模型 (參考:P.18 領域模型 (7/7)) <p>目前構想： 使用 DokuWiki 來做維護文件系統， 可每日發送「文件異動通知」信件， 取代 敏捷開發 的每日站立會議</p>

維護領域模型文件

Chrome 檔案 編輯 檢視 歷史記錄 書籤 人員 視窗 說明

方案 意思 - Google 搜尋 | 臨床教育e-Portfolio 2.0版 | chimei:workflow:default_workfl

不安全 | wade.playground.io/dokuwiki/doku.php?id=chimei:workflow:default_workflow

Default_workflow

namespace Clouder\Workflow;
實作 Workflow_interface 介面

屬性：

`public $default_workflow_id: integer`
預設簽核流水號

`public $group_id: integer`
訓練科室 ID

`public $title: string`
預設簽核標題

`public $is_delete: string`
是否已被刪除

方法：

`public function __construct`
定義：

目錄表

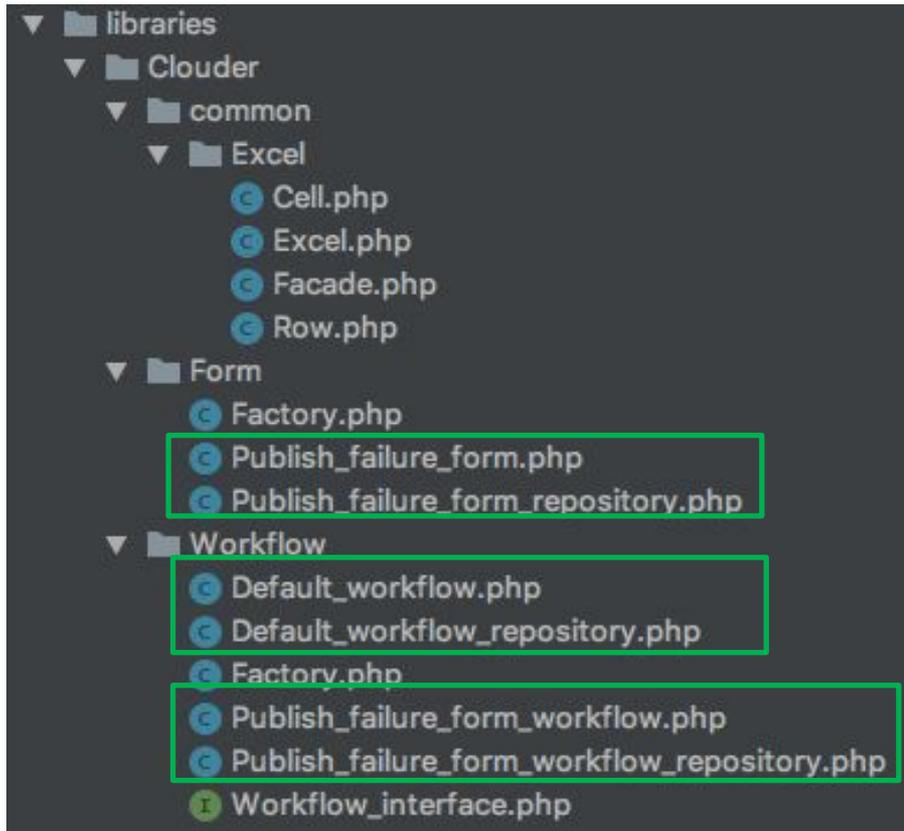
- Default_workflow
- 屬性：
- 方法：
 - public function __construct
 - 定義：
 - public function process()
 - 定義：
 - 範例：
 - public function process_user(\$sequence)
 - 定義：
 - 範例：

目前推薦使用 DokuWiki 來做維護文件

DokuWiki 優點：

- 支持 Markdown
- 文件系統架構相當容易編寫跟維護
- **每日寄送文件異動 Mail** (有訂閱的話)

現況：領域模型(1/3)



- 目前 **領域模型** 都放在 application/libraries/Clouder 目錄下
- 為了讓領域模型可執行 **單元測試**，故設計每個 **領域模型** **必須** 搭配一個 **Repository** 資料庫操作層
- 盡量使用 **Factory** 來建立領域模型物件，減少複雜的初始化程序。

現況：領域模型(2/3)

領域模型

```
Default_workflow
├── default_workflow_id
├── create_user_id
├── process
├── create_time
├── group_id
├── repo
├── title
├── is_delete
├── __construct(default_workflow, repository)
├── process()
├── process_user(sequence)
```

組合

```
Default_workflow_repository
├── workflow_api
├── __construct()
├── get_process(default_workflow_id)
```

領域模型資料庫操作層

範例：

領域模型**必須**搭配一個 Repository 資料庫操作層

現有架構之 API

```
Workflow_api
├── common/modules/workflow_api/controllers/Workflow_api.php
├── workflow
├── __construct()
├── get_default_workflow(default_workflow_id, user_id, search_group)
├── get_default_workflow_process(default_workflow_id)
├── get_default_workflow_process_user(default_process_id)
├── get_publish_failure_form_workflow_process(default_workflow_id)
├── get_publish_failure_form_workflow_process_user(default_process_id)
├── insert_default_workflow(data)
├── insert_form_workflow_process_set(default_workflow_process, workflow_id)
├── insert_publish_failure_form_workflow_process_set(publish_failure_form_id, default_workflow_id)
├── insert_default_workflow_process(data)
├── insert_default_workflow_process_user(data)
├── delete_default_workflow(default_workflow_id)
├── get_workflow(workflow_id, ai_id, state)
├── get_workflow_process(workflow_id, process_state, sequence, process_id)
├── get_workflow_process_user(process_id, sign_user_id, sign_state, process_user_id)
├── insert_workflow(data)
├── insert_workflow_process(data)
```

調用

現況：領域模型(3/3)

範例：
在 Factory 內初始化 Form 物件

```
<?php
namespace Clouder\Form;
use \Clouder\Workflow\Factory as Workflow;

class Factory
{
    /**
     * 建立表單物件
     *
     * @param array $form_data
     *
     * @return Form
     */
    public static function Form($failure_form_data)
    {
        // 初始化 表單物件 並配置 資料庫操作層物件
        $form = new Form($failure_form_data, new Form_repository());
        // 組合 表單審核物件
        $form->set_workflow(Workflow::Form_workflow($form->form_id));

        return $form;
    }
}
```

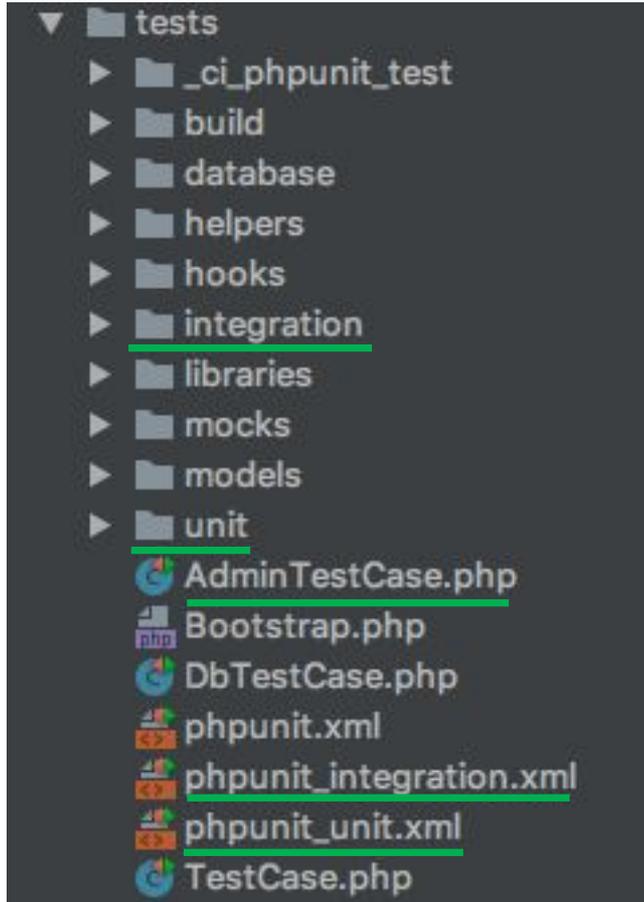
盡量使用 **Factory** 來建立領域模型物件

範例：
透過 Factory 取得 Form 物件

```
public function get_base_info($assessment_data, $is_assessor = null, $data = null)
{
    $form_data = $this->assessment_form_api->get_form_data($assessment_data['form_id']);
    $form = \Clouder\Form\Factory::Form($form_data);
}
```

避免複雜的初始化程序。

現況：自動化測試環境



目錄：

- 測試程式一律放在 application/tests 目錄
- **integration** 為整合測試目錄，**迴歸測試** 應放在此目錄下
- **unit** 為單元測試目錄，每個 **領域模型** 應該都要有 **單元測試**

檔案：

- **AdminTestCase.php**：
測試案例都要**繼承**此類別，此類別提供：
 - 登入後台功能（供迴歸測試用）
 - 資料庫交易機制
（測試異動資料庫的功能後，須做滾回）
- **phpunit_integration.xml**：
PHPUnit 設定檔，用來執行所有 **整合測試**
- **phpunit_unit.xml**：
PHPUnit 設定檔，用來執行所有 **單元測試**

其他需求

- 需要一個專門給**自動化測試**使用的 DataBase, 否則 DataBase 不斷異動, 會造成測試案例失敗。
- 需要一個可訂閱的文件系統, 共**維護領域模型文件**用。
- 要找一個方便維護文件的 SOP、套件。(最好一次不要超過 5 分鐘)



實務操作

持續重構系統，逐漸引入領域模型

運用重構技術，產生領域模型

```
// 取得簽核列表
$default_workflow_list = $this->workflow_api->get_form_default_workflow_selected(null, $form_id, $search)
foreach ($default_workflow_list as &$default_workflow) {
    $default_workflow['process'] = $this->workflow_api->get_default_workflow_process($default_workflow_id)
    if (count($default_workflow['process'])) {
        foreach ($default_workflow['process'] as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process['default_workflow_id'])
            $user_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    $user_name[] = $user['name'];
                } else {
                    $user_name[] = $user['role_name'];
                }
            }
            $process['user_name'] = implode( glue: ',', $user_name);
        }
    }
}
```

業務實體

由業務實體歸納出領域模型

```
Default_workflow
├── default_workflow_id
├── create_user_id
├── create_time
├── group_id
├── title
├── is_delete
```

領域模型

其實就是運用重構技術，產生領域模型

重構技術

不改變軟體外部行為的前提下，**改變其內部結構**，使其**更容易理解且易於修改**。

取得預設簽核流程邏輯

取得預設簽核流程邏輯

取得預設簽核流程邏輯

重構前

- 程式碼散落、沒有結構
- 沒有程式碼風格
- 不容易閱讀的程式碼
- 程式邏輯與責任很模糊
- 不容易擴充

多次重構

- 程式碼逐漸變得易讀
- 程式與責任越來越清楚

完成重構

- 容易閱讀
- 有結構的程式碼
- 程式責任分離清楚
- 高內聚/低耦合
- 彈性高易擴充
- 程式碼可重複利用

重構流程

* 必要

建立回歸測試

- 確認修改部分沒有影響到既有功能。

* 必要

擷取函式

- 將相關的程式碼複製到一個新方法裡
不屬於方法的變數，就當作參數傳入。

* 必要

替換程式碼

- 將原本的程式碼註解(或移除)，改呼叫新的方法。

(可選)

需建立新類別時

建立新類別

- 若被重構的程式不符合**單一職責原則**，
可將**擷取函式**產生的新方法**搬移**進適合的類別。

(可選)

需建立新類別時

替換程式碼

- 步驟類似**擷取函式**，在程式碼內引入上一步驟的類別。
移除原程式碼的方法，改呼叫引入類別的方法。

(視情況)

有辦法撰寫就要

建立單元測試

- 若有辦法的話，
務必替新程式撰寫單元測試，以提升系統的測試覆蓋率

註：建立新類別，就是產生**領域模型**的步驟

示範重構流程

以 預設簽核設定畫面 為例

預計重構 取得預設簽核流程 邏輯

預設簽核設定

預設簽核列表 新增/修改預設簽核 表單預設簽核

簽核主旨: 西醫職類-實習醫學生(學生-課程教師-科主任)

簽核順序	順序	方式	成員/角色	管理	調整順序
	1	填寫	學生	編輯 刪除	↑ ↓
	2	填寫	課程教師	編輯 刪除	↑ ↓
	3	填寫	臨床教師	編輯 刪除	↑ ↓
	4	填寫	長期導師	編輯 刪除	↑ ↓
	5	填寫	教學負責人	編輯 刪除	↑ ↓
	6	填寫	計畫主持人	編輯 刪除	↑ ↓

新增成員: 填寫 審核 通知 受評者

新增角色: 填寫 審核 通知 受評者

確認修改 取消

```
public function get_default_workflow_process()
{
    // 取得簽核列表
    $default_workflow_id = (int)$this->input->post('default_workflow_id');
    $read_mode = (int)$this->input->post('read_mode');
    $default_workflow_process = $this->workflow_api->get_default_workflow_process($def

    if (count($default_workflow_process)) {
        foreach ($default_workflow_process as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user(
                $user_id = $user_name = $role_id = $role_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] != null) {
                    array_push(&array: $user_id, $user['default_user_id']);
                    array_push(&array: $user_name, $user['name']);
                } else {
                    array_push(&array: $role_id, $user['default_role_id']);
                    array_push(&array: $role_name, $user['role_name']);
                }
            }
            $process['user_id'] = implode(' ', $user_id);
            $process['user_name'] = implode(' ', $user_name);
            $process['role_id'] = implode(' ', $role_id);
            $process['role_name'] = implode(' ', $role_name);
        }
        $this->tplVar['default_workflow_id'] = $default_workflow_id;
    }
    $this->tplVar['default_workflow_process'] = $default_workflow_process;

    switch ($read_mode) {
        case 1:
            $page_name = 'set_workflow';
            break;
        case 2:
            $page_name = 'preview_default_workflow';
            break;
        default:
            $page_name = 'set_default_workflow';
    }
    echo json_encode(array(
        'default_workflow_process' => $this->load->view('default_workflow/' . $page_na
```

重構目標

- 建立領域模型
- 提升可讀性

* 以下範例將假設您已經安裝好 PHPUnit 。

示範：建立回歸測試(1/7)

回歸測試：

用來確認 **整個功能** 在重構過程中，有沒有影響到原本的功能。

確認方法：

在重構過程中，**每做一次修改** 就驗證**相同的輸入**是否能得到**相同的輸出結果**。

建立 **回歸測試** 流程：

1. 新增 **測試程式** 與 **測試案例**
2. 取得程式的 **輸入** 與 **輸出**，並在 **測試案例** 中給予 **輸入** 與 **預期的輸出結果**
3. 在 **測試案例** 中引入被重構的程式
4. 執行測試案例，驗證程式執行結果是否與 **預期的輸出結果** 相符

示範：建立回歸測試(2/7)

步驟 1:

新增 **測試程式** 與 **測試案例**

測試程式 都要放在 **application/tests** 目錄下，
且測試檔案應該模擬「受測程式」所在目錄結構。

受測程式(application/) :

```
modules
├── admin
│   ├── common
│   │   ├── modules
│   │   │   ├── form
│   │   │   └── controllers
│   │       └── Default_workflow.php
```

測試程式 (application/tests/):

```
modules
├── admin
│   ├── common
│   │   ├── modules
│   │   │   ├── form
│   │   │   └── controllers
│   │       └── Default_workflow_test.php
```

測試程式名稱: XXXX_test.php, 為 **受測檔案** 的名稱加上 _test 為後綴。

示範：建立回歸測試(3/7)

步驟 1-2:

在測試檔案中新增 **測試案例**

測試案例 必須為 public 方法，
且方法名稱須以 test_* 作為前綴。

建議：

以 BDD 的格式撰寫測試案例：

@scenario: 使用情境
@given : 某個條件下
@when: 當使用者執行某些動作
@then: 預期的執行結果

提高測試案例的可讀性。

Default_workflow_test.php

```
<?php
class Default_workflow_test extends AdminTestCase
{
    public function setUp()
    {
        parent::setUp();
    }

    public function tearDown()
    {
        parent::tearDown();
    }

    /**
     * @scenario 取得 預設簽核流程畫面
     *
     * @given 預設簽核 id
     * @when 索取 預設簽核流程 畫面
     * @then 取得 預設簽核流程
     */
    public function test_get_default_workflow_process()
    {
        // @given 預設簽核 id

        // @when 索取 預設簽核流程 畫面

        // @then 取得 預設簽核流程
    }
}
```

示範：建立回歸測試(4/7)

步驟 2-1: 取得程式的 輸入 與 輸出

取得程式的 輸入

取得程式的 輸出

籤核主旨	西醫職類-實習醫學生(學生-課程教師-科主任)			
籤核順序	順序	方式	成員/角色	管理
	1	填寫	學生	編輯 刪除
	2	填寫	課程教師	編輯 刪除
	3	填寫	臨床教師	編輯 刪除
	4	填寫	長期導師	編輯 刪除
	5	填寫	教學負責人	編輯 刪除
	6	填寫	計畫主持人	編輯 刪除

建議找可明顯判斷的差異部分，當作程式的 輸出

示範：建立回歸測試(5/7)

步驟 2-2:

在 測試案例 中給予 輸入 與 預期的輸出結果

簽核主旨	西醫職類-實習醫學生(學生-課程教師-科主任)		
簽核順序	順序	方式	成員/角色
	1	填寫	學生
	2	填寫	課程教師
	3	填寫	臨床教師
	4	填寫	長期導師
	5	填寫	教學負責人
	6	填寫	計畫主持人

```
/**
 * @scenario 取得 預設簽核流程畫面
 *
 * @given 預設簽核 id 212
 * @when 索取 預設簽核流程 畫面
 * @then 取得預設簽核流程為「1.填寫:學生 2.填寫:課程教師 3.填寫:長期導師 4.簽核:
 */
public function test_get_default_workflow_process()
{
    // @given 預設簽核 id 212
    $post_data = [
        'default_workflow_id' => 212,
    ];

    // @when 索取 預設簽核流程 畫面

    // @then 取得預設簽核流程為「1.填寫:學生 2.填寫:課程教師 3.填寫:長期導師 4.簽核:
    $this->assertContains( needle: '學生', $default_workflow_process);
    $this->assertContains( needle: '課程教師', $default_workflow_process);
    $this->assertContains( needle: '長期導師', $default_workflow_process);
    $this->assertContains( needle: '臨床教師', $default_workflow_process);
    $this->assertContains( needle: '教學負責人', $default_workflow_process);
    $this->assertContains( needle: '計畫主持人', $default_workflow_process);
}
```

示範：建立回歸測試(6/7)

步驟 3:

在 **測試案例** 中引入被重構的程式

引入被重構的程式，
並調用程式

```
/**
 * @scenario 取得 預設簽核流程畫面
 *
 * @given 預設簽核 id 212
 * @when 索取 預設簽核流程 畫面
 * @then 取得預設簽核流程為「1.填寫:學生 2.填寫:課程教師 3.填寫:長期導師 4.簽核:臨床教師 5.簽核:
 */
public function test_get_default_workflow_process()
{
    // @given 預設簽核 id 212
    $post_data = [
        'default_workflow_id' => 212,
    ];
    $this->login( account: 'admin', password: COMMON_PASSWORD, keep_db: true);

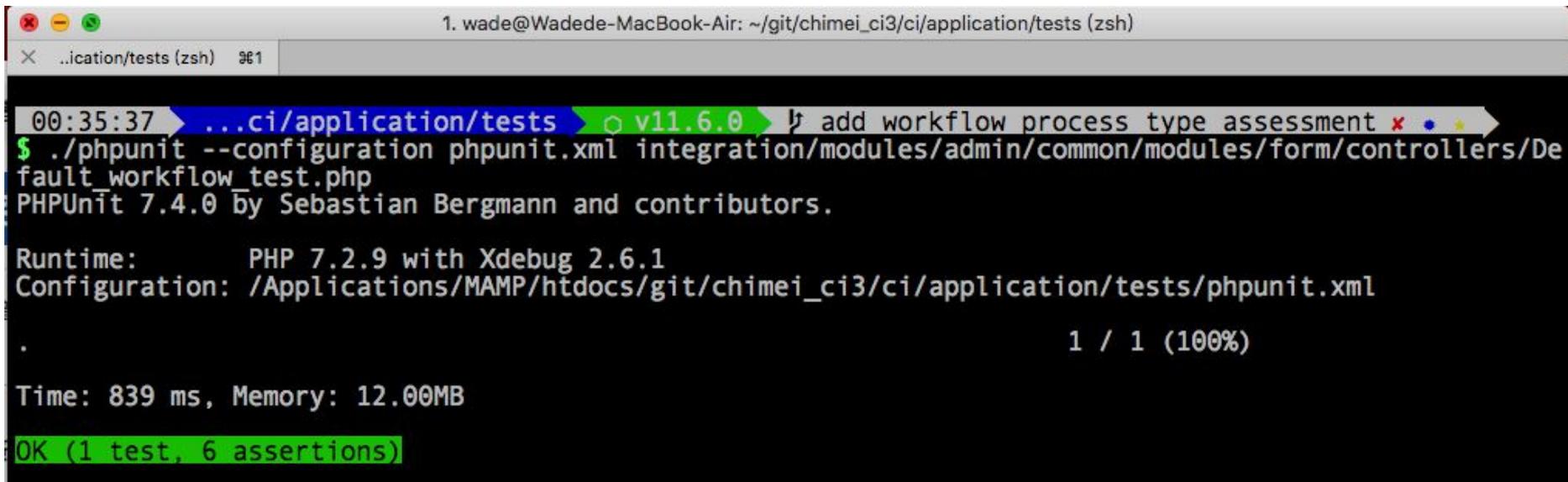
    // @when 索取 預設簽核流程 畫面
    $output = $this->request(
        http_method: 'POST',
        argv: 'admin/common/form/default_workflow/get_default_workflow_process',
        $post_data
    );
    $output = json_decode($output, assoc: true);
    $default_workflow_process = $output['default_workflow_process'];

    // @then 取得預設簽核流程為「1.填寫:學生 2.填寫:課程教師 3.填寫:長期導師 4.簽核:臨床教師 5.簽核:
    $this->assertContains( needle: '學生', $default_workflow_process);
    $this->assertContains( needle: '課程教師', $default_workflow_process);
    $this->assertContains( needle: '長期導師', $default_workflow_process);
    $this->assertContains( needle: '臨床教師', $default_workflow_process);
    $this->assertContains( needle: '教學負責人', $default_workflow_process);
    $this->assertContains( needle: '計畫主持人', $default_workflow_process);
}
```

示範：建立回歸測試(7/7)

步驟4:

執行 **測試案例**，驗證程式執行結果是否與 **預期的輸出結果** 相符



```
1. wade@Wadede-MacBook-Air: ~/git/chimei_ci3/ci/application/tests (zsh)
...ci/application/tests v11.6.0 add workflow process type assessment
$ ./phpunit --configuration phpunit.xml integration/modules/admin/common/modules/form/controllers/Default_workflow_test.php
PHPUnit 7.4.0 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.9 with Xdebug 2.6.1
Configuration: /Applications/MAMP/htdocs/git/chimei_ci3/ci/application/tests/phpunit.xml

.                                                       1 / 1 (100%)

Time: 839 ms, Memory: 12.00MB

OK (1 test, 6 assertions)
```

* 只有回歸測試顯示通過時，才可繼續重構的流程。

示範：擷取函式

```
public function get_default_workflow_process()
{
    // 取得簽核列表
    $default_workflow_id = (int)$this->input->post('default_workflow_id');
    $read_mode = (int)$this->input->post('read_mode');
    $default_workflow_process = $this->workflow_api->get_default_workflow_process($default_workflow_id);

    if (count($default_workflow_process)) {
        foreach ($default_workflow_process as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process);
            $user_id = $user_name = $role_id = $role_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    array_push( @array: $user_id, $user['default_user_id']);
                    array_push( @array: $user_name, $user['name']);
                } else {
                    array_push( @array: $role_id, $user['default_role_id']);
                    array_push( @array: $role_name, $user['role_name']);
                }
            }
            $process['user_id'] = implode( glue: ',', $user_id);
            $process['user_name'] = implode( glue: ',', $user_name);
            $process['role_id'] = implode( glue: ',', $role_id);
            $process['role_name'] = implode( glue: ',', $role_name);
        }
        $this->tplVar['default_workflow_id'] = $default_workflow_id;
        $this->tplVar['default_workflow_process'] = $default_workflow_process;
    }
}
```

擷取前

```
/**
 * @param $default_workflow_id
 * @param $process
 * @return mixed
 */
private function get_workflow_process($default_workflow_id, $process)
{
    $default_workflow_process = $this->workflow_api->get_default_workflow_process($default_workflow_id);

    if (count($default_workflow_process)) {
        foreach ($default_workflow_process as &$process) {
            $process['user'] = $this->workflow_api->get_default_workflow_process_user($process);
            $user_id = $user_name = $role_id = $role_name = array();
            foreach ($process['user'] as $user) {
                if ($user['default_user_id'] !== null) {
                    array_push( @array: $user_id, $user['default_user_id']);
                    array_push( @array: $user_name, $user['name']);
                } else {
                    array_push( @array: $role_id, $user['default_role_id']);
                    array_push( @array: $role_name, $user['role_name']);
                }
            }
            $process['user_id'] = implode( glue: ',', $user_id);
            $process['user_name'] = implode( glue: ',', $user_name);
            $process['role_id'] = implode( glue: ',', $role_id);
            $process['role_name'] = implode( glue: ',', $role_name);
        }
        $this->tplVar['default_workflow_id'] = $default_workflow_id;
        $this->tplVar['default_workflow_process'] = $default_workflow_process;
        return $process;
    }
}
```

把要重構的程式碼擷取到 **新方法** 裡面。

```
/**
 * 取得預設簽核流程
 *
 * @author aihcuyz 2018/03/21
 */
public function get_default_workflow_process()
{
    // 取得簽核列表
    $default_workflow_id = (int)$this->input->post('default_workflow_id');
    $read_mode = (int)$this->input->post('read_mode');

    $process = $this->get_workflow_process($default_workflow_id, $process);

    switch ($read_mode) {
        case 1:
            $page_name = 'set_workflow';
            break;
        case 2:
            $page_name = 'preview_default_workflow';
            break;
        default:
            $page_name = 'set_default_workflow';
    }
    echo json_encode(array(
        'default_workflow_process' => $this->load->view('default_workflow/' . $page_name)
    ));
}
```

擷取後

強烈建議使用 IDE 提供的重構功能：
各家 IDE 都有提供 **擷取函式** 的重構功能
讓 IDE 自動幫你找出哪些變數要變成參數

* 記得執行回歸測試，確認輸出是否符合預期

示範：建立新類別(領域模型)

領域模型

```
Default_workflow
├── default_workflow_id
├── create_user_id
├── process
├── create_time
├── group_id
├── repo
├── title
├── is_delete
├── __construct(default_workflow, repository)
├── process()
├── process_user(sequence)
```

組合

```
Default_workflow_repository
├── workflow_api
├── __construct()
├── get_process(default_workflow_id)
```

領域模型資料庫操作層

現有架構之 API

```
Workflow_api
├── workflow
├── __construct()
├── get_default_workflow(default_workflow_id, user_id, search_group)
├── get_default_workflow_process(default_workflow_id)
├── get_default_workflow_process_user(default_process_id)
├── get_publish_failure_form_workflow_process(default_workflow_id)
├── get_publish_failure_form_workflow_process_user(default_process_id)
├── insert_default_workflow(data)
├── insert_form_workflow_process_set(default_workflow_process, workflow_id)
├── insert_publish_failure_form_workflow_process_set(publish_failure_form_id, default_workflow_id)
├── insert_default_workflow_process(data)
├── insert_default_workflow_process_user(data)
├── delete_default_workflow(default_workflow_id)
├── get_workflow(workflow_id, ai_id, state)
├── get_workflow_process(workflow_id, process_state, sequence, process_id)
├── get_workflow_process_user(process_id, sign_user_id, sign_state, process_user_id)
├── insert_workflow(data)
├── insert_workflow_process(data)
```

調用



謝謝觀賞