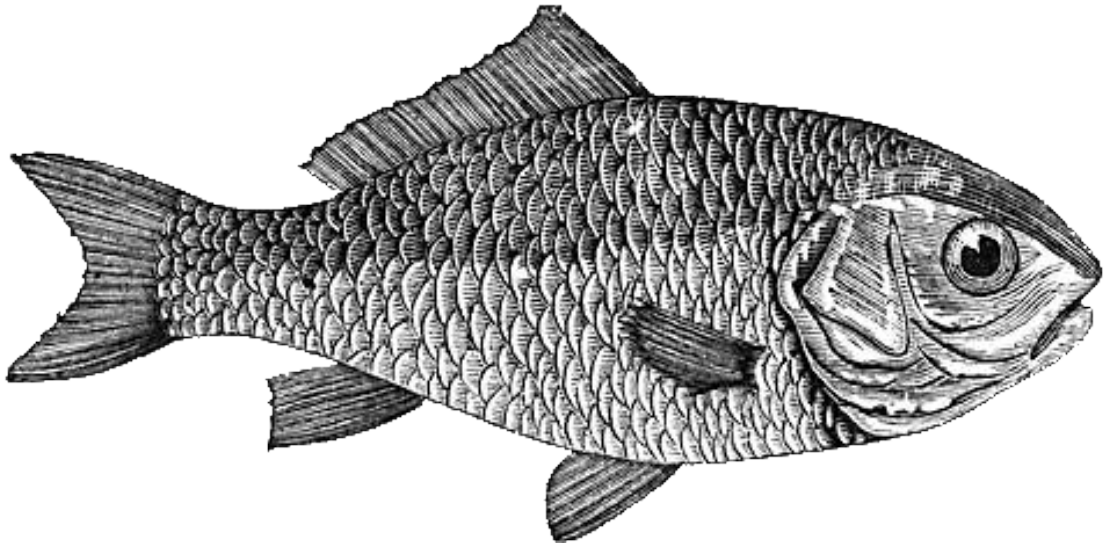


實戰 PHP 重構與模式

一次就瞭解三種模式的重構技法

O'REALLY

大澤木小鐵

一些經驗分享

關於重構的實戰經驗

重構時機：

- 完成某部份功能後，新增或修改某功能前。

必要條件：

- 一定要寫好對應的測試。
- 一定要瞭解程式碼。
- 一定要把重構時間加入時程裡。

注意事項：

- 要讓共事的伙伴也能看懂重構後的程式碼。
- 落落長的系統難以重構時，就重 GO 吧...

關於測試的實戰經驗

該測什麼？

- 不必鉅細靡遺地測試每行程式碼，只測試你覺得會出錯的部份。

常用技巧：

- 程式儘量以帶入參數的方式來注入不易測試的對象，例如 DAO 、 Mailer 等。
- 利用 Mock Data Object 來做測試。
- 善用 IDE 的測試機制及快捷鍵，例如 NetBeans 的 Ctrl + F6 (測試類別) ； Shift + F6 (執行當前的測試) ； Alt + F6 (執行所有測試)

關於模式的實戰經驗

應用時機：

- 設計階段就知道問題可用哪個模式解決。
- 重構時發現程式結構偏向哪個模式。

注意事項：

- 設計時，遵守物件導向設計原則。
- 撰寫程式碼時別去想模式，先專心處理問題。

實例

批次訂單狀態更新程式

背景：

- 某電子商務系統 Web 平台後端自動化系統。

功能說明：

- 金流及物流服務的排程程式，會將相關資訊寫入佇列資料表中。
- 本程式會依照佇列資料表中的資訊，批次更新訂單狀態。
- 訂單狀態：已付款、已開發票、已出貨、已結案。
- 訂單狀態更新後，要觸發對應的動作，例如寄信。
- 完成後的佇列要刪除，避免再次觸發。

初版程式碼
DEMO

怎麼重構？

1. 從現有版本分支。
2. 決定要實做的方向。
3. 小步前進。
4. 測試。
5. 完成一次小重構後，提交至版本控制系統。
6. 繼續步驟 2 ~ 5 。
7. 整個功能完成後，合併回主幹。

基礎重構

起手式 – Extract Method

1. 將相關的數行程式碼複製到一個新方法裡。
2. 不是屬於方法的變數，就當做參數傳入。
3. 將原來的程式碼註解起來，改為呼叫新的方法，**測試**。
4. 將註解起來的程式碼刪除，**測試**。

將 DAO (SQL) 封裝至 Model 中

1. 建立一個抽象 Model 類別，並將 DAO 改為外部可帶入的類別屬性。
2. 建立訂單子 Model 類別。
3. 在主程式中，將原來使用 DAO 的部份，改用 Model 類別，**測試**。

注意：

- 因為通常我們會使用 ORM 來取代 DAO ，因此這個重構的重點在於要讓 Model 類別是可以被測試的；請參考各位所使用的的 ORM 相關資訊。

將全域常數改為類別常數

1. 在訂單 Model 類別建立表示狀態的類別常數。
2. 在原本有使用到全域常數利用編輯器取代功能，改成類別常數，**測試**。
3. 移除原本用 define 定義的常數，**測試**。

將模式引入程式中

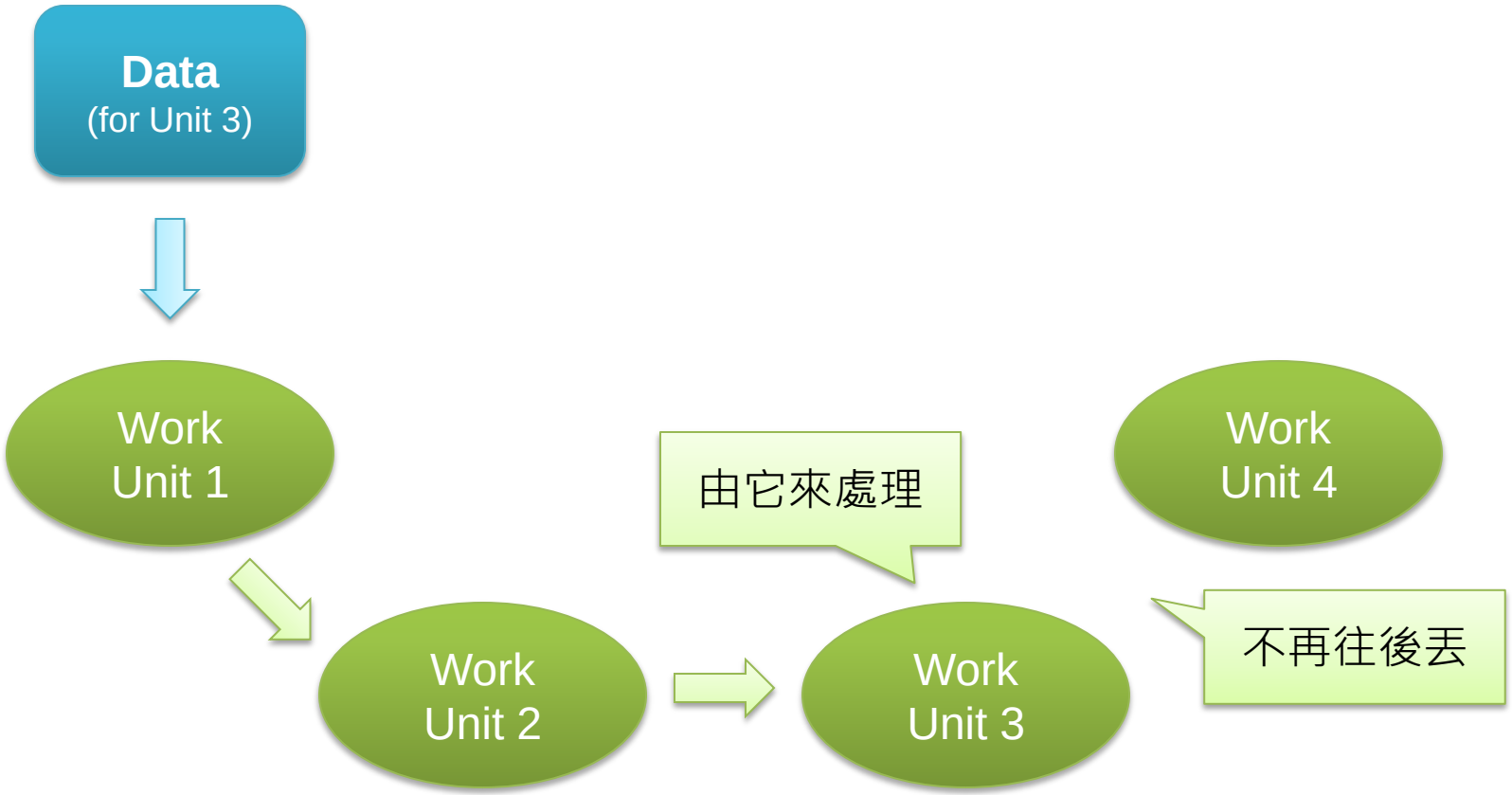
if … elseif … else

- 看到 if … elseif … else 就是一個徵兆。
- 資料在條件判斷下，有依序處理的狀況。
- 使用 Chain of Responsibility 模式。

Chain of Responsibility 原理

1. 將要做的工作單元串成一個鎖鏈。
2. 將資料拋給第一個工作單元。
3. 符合條件的資料就會被該工作單元處理。
4. 不符合條件的資料就會被轉向下一個工作單元。
5. 最後有可能出現沒被處理的資料。

Chain of Responsibility 圖解



如何做？

1. 在 Task 中，為 Dao 及 Mailer 建立 getter 。
2. 建立抽象工作單元類別，並引用目前的 Task 。
3. 建立子工作單元類別。
4. 將每個 if/elseif 區塊複製到工作單元類別裡。
5. 修改子工作單元類別裡對 dao 及 mailer 的呼叫方式。
6. 修改主程式，將所有工作單元依序串起，**測試**。
7. 將原有的 if … elseif … else 改用工作單元處理，**測試**。
8. 刪除不要的區段與方法，**測試**。

更適合的模式

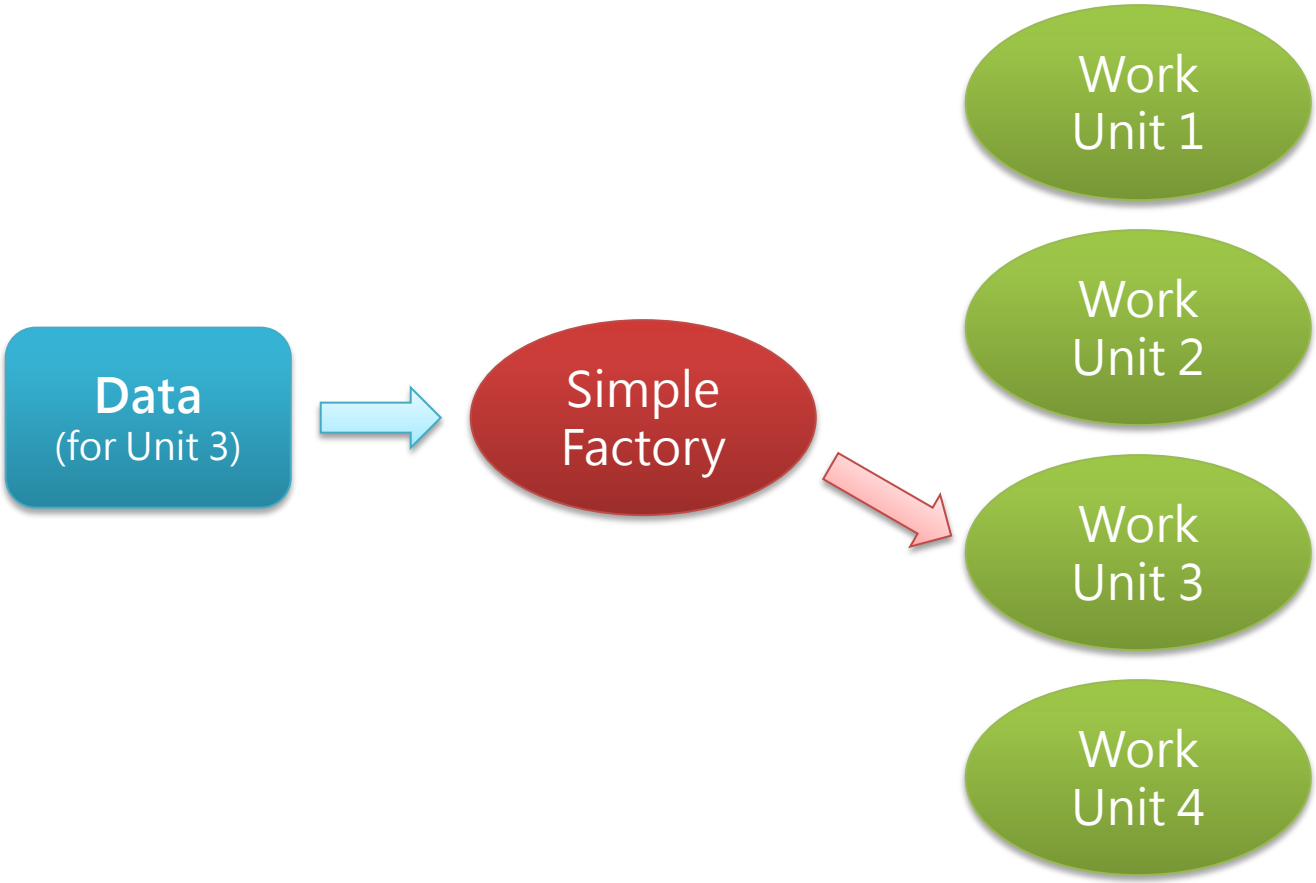
用 switch 代替

- 通常 if … elseif … else 可以用 switch 代替。
- 看到 switch ，表示有可能是切換處理方式。
- 使用 Strategy 模式。

Strategy 原理

- 使用者在面對某些狀況時，可以有不同的因應方式，這些因應方式通常稱為策略。
- 通常是用設定檔來決定採用何種策略，執行時期則是以資料內容或使用者動作來決定。
- 對使用者來說，雖然策略是不同的，但呼叫方法都一樣；通常是搭配 Simple Factory 模式來建立策略物件讓主程式使用。

Strategy 圖解


Data
(for Unit 3)
Simple
Factory
Work
Unit 1
Work
Unit 2
Work
Unit 3
Work
Unit 4

如何做？

1. 在 Task 中，為 Dao 及 Mailer 建立 getter 。
2. 將 if … elseif … else 轉換為 switch ，**測試**。
3. 建立抽象工作單元類別，並定義適當名稱的處理方法，例如 handle() 。
4. 將每個 case 區塊所呼叫方法內容，複製到子工作單元類別的 handle() 方法中。
5. 修改子工作單元類別 handle() 方法裡對 dao 及 mailer 的呼叫方式。
6. 改用新的子類別來取代原來 case 區塊裡的方法，**測試**。
7. 將 case 區塊改以 Simple Factory 模式處理，**測試**。
8. 移除不要的程式碼。

將模式複合在一起

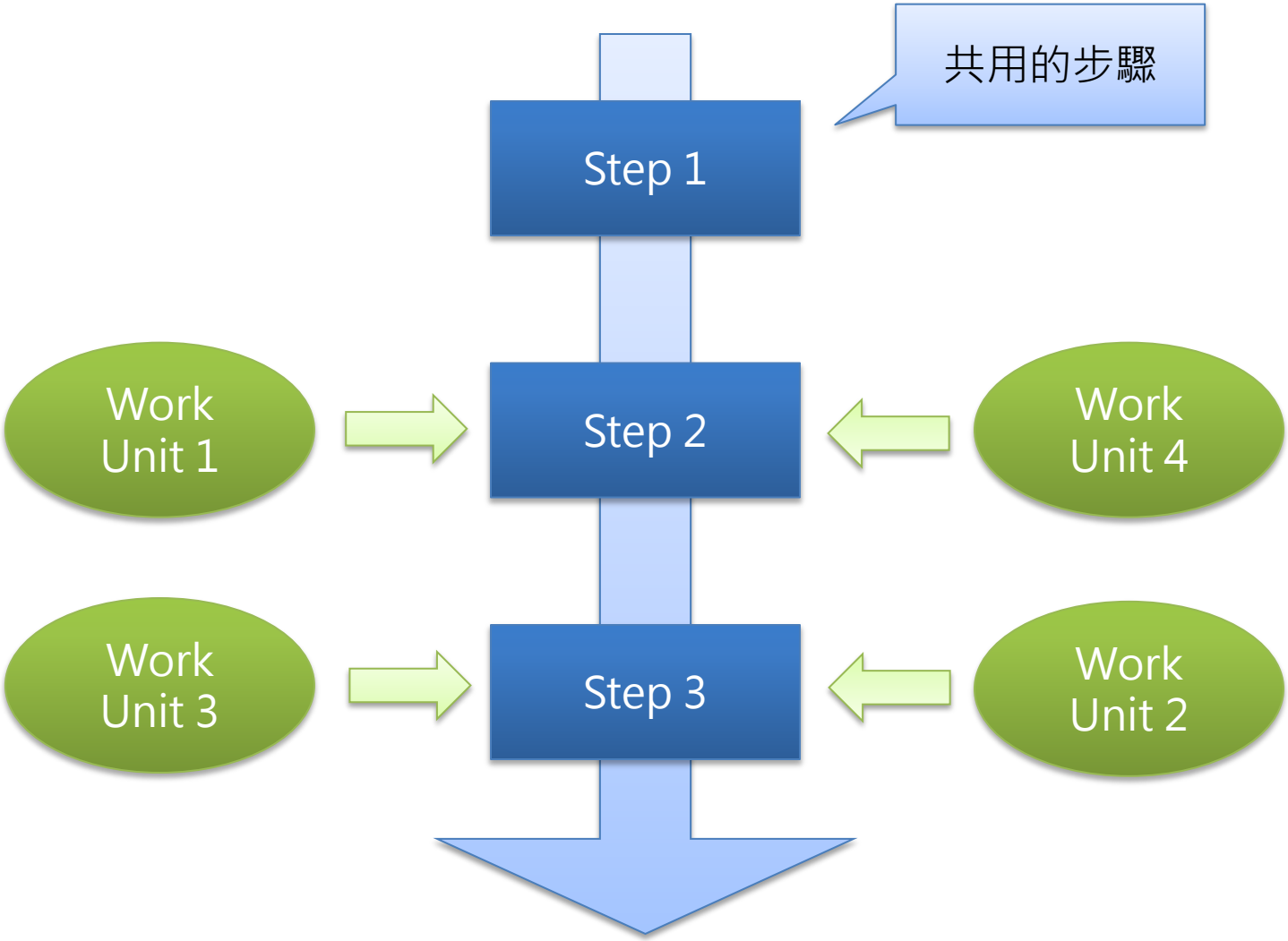
相似的流程

- 每個訂單狀態處理的流程很類似。
- 在少數的部份有差異。
- 使用 Template Method 模式。

Template Method 原理

- 由抽象類別的某方法中定義主要流程的骨架；骨架上會提供一些需要被覆寫的方法，稱為 Hook 。
- 子類別只覆寫自己需要處理的 Hook 方法。
- 使用者呼叫定義主流程的方法，子類別會負責處理其他細部差異的部份。
- Don’t Call Me, I call you.

Template Method 圖解



如何做？

1. 對 Strategy 重構後的子工作單元類別實施 Extract Method 重構。
2. 分析流程步驟，找出重複的部份，及有差異的部份。
3. 在抽象工作單元類別中定義流程方法及完成流程骨架。
4. 修改子工作單元類別，將重複的部份移到抽象工作類別上，保留有差異的部份。
5. 修改 Task 的 run() ，將共用的部份移到抽象工作類別中。

搞定收工？

還有其他該注意的事

- 見好就收，別花 80% 的力氣來得到 20% 的效益。
- 不要硬套模式的形，能更清楚表達意圖的程式碼才是最有效的。
- 別忘了把分支合併回主幹。
- 讓伙伴知道你重構了些什麼；重構時可以採用 Pair Programming ，重構後可以用文件或註解。
- 重構後如果伙伴看不懂，要嘛就放棄這次重構，要嘛就放棄這個伙伴 (誤) 。
- 讓你的伙伴跟著你一起成長！

範例下載

https://github.com/jaceju/PHP-Refactoring-And-Patterns

謝謝大家