

PHP REFACTORING

程式重構



本投影片簡報內容按 **Apache-2.0** 授權。所提及或者引用的公司名稱、產品名稱以及所引用的文字、商標、影片、產品相片或者網站頁面，均為其所屬公司所擁有。

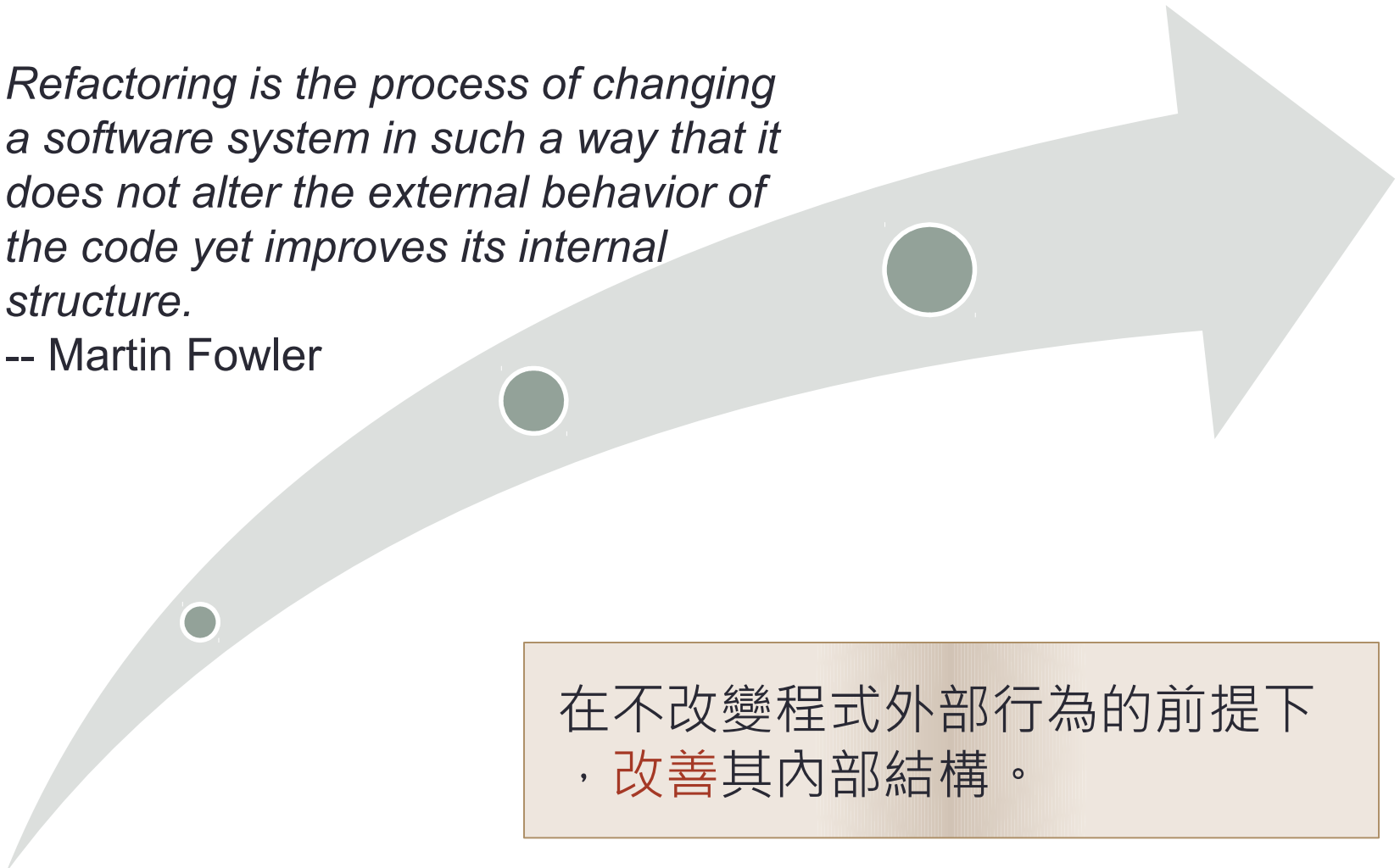
議程與大綱

- 重構 (Refactoring) 觀念
- 進行重構的時機
- 重構與程式寫作規範
- 重構實例與技巧

重構 (Refactoring)

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.

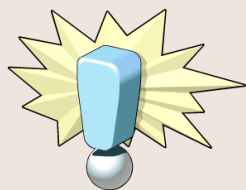
-- Martin Fowler



在不改變程式外部行為的前提下，改善其內部結構。

什麼不是重構

- 增加新功能
- 改善安全性
- 提升效能
- 程式除錯



重構是軟體設計期間持續進行的工作，進行上述活動時，自然也會做程式重構。
重構後的程式，有助於進行上述活動。

何時進行重構

- 增加新功能

- 增加新功能時，不可能完全與舊程式無關。
- 必須了解舊程式，也才知道該不該以及如何重構。
 - 如果有需要，重構舊程式；
如果新功能很容易就加得進去，自然就不更動舊程式。

- 修正錯誤

- 不可能只從介面除錯，必須了解內部程式運作。
- 理解後，也才知道怎麼重構成比較好讀的程式。
- 結果是：好讀的程式，好除錯的程式。

來看一則例子吧



```
$Data = readDataFile("pick3.txt");  
showHotColdList($Data);
```

```
// 讀取資料檔  
function readDataFile($sFilename) {  
    // ...  
    return $result;  
}
```

```
// 顯示冷號熱號數列  
function showHotColdList($Data)  
{  
    // ...  
}
```

```
// 讀取資料檔
```

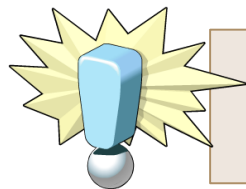
```
$sData = "";  
$f = fopen ( "pick3.txt", "r" );  
while ( ! feof ( $f ) ) {  
    $line = fgets ( $f );  
    $sData .= Trim ( $line );  
}  
fclose ( $f );
```

```
// 產生冷號熱號數列
```

```
$result = "0123456789";  
$iLen = strlen ( $sData );  
for($iPos = 0; $iPos < $iLen; $iPos ++ ) {  
    $sch = substr ( $sData, $iPos, 1 );  
    $result = $sch .  
        str_replace ( $sch, "", $result );  
}  
echo substr ( $result, 0, 5 ) . "-"  
    . substr ( $result, 5, 5 );
```

重構的目的

- Simplicity
- Clarity
- Maintainability
- Efficiency
- Flexibility
 - Reusability
 - Extensibility



簡單地說，將程式改成比較好讀、好維護。

什麼是好的程式（一）

- Readability
- Extensibility
- Efficiency
- and More... （課堂討論）

什麼是好的程式 (二)

- 您偏好哪一種寫法？ (A or B)

```
// A  
if (empty($stringVar))  
{  
    // do something  
}  
  
// B  
if ($stringVar == "")  
{  
    // do something  
}
```

```
<?php  
  
$name = "Chien";  
  
// A  
echo "Hello! " . $name;  
  
// B  
echo "Hello! $name";  
  
?>
```

什麼是好的程式（三）

- 團隊共識
 - 例如：在可讀性的前提下，程式短一點好一點。
- 程式寫作規範
 - <http://www.php-fig.org/psr/psr-1/>
 - <http://www.php-fig.org/psr/psr-2/>
(類別、函式、變數命名慣例、控制結構的樣式、註解等等)
- 可重用 (re-usable)
- 設計模式 (design patterns)

如何重構

- Extract Method
- Inline Method
- Inline Temp
- Replace Temp with Query
- Introduce Explaining Variable
- Split Temporary Variable

Extract Method

- 適用對象：
 - 過長的函式
 - 需要一段註解才能理解的程式
- 作法：
 1. 針對打算「移出」的程式，依其主旨新建一個函式。
 2. 將程式移入新的函式。
 3. 移入新函式的程式若會用到既有的變數，宣告為新建函式的參數。
- 重點：
 - 新函式的名稱要表達「**做什麼**」而不是「怎麼做」。
 - 最高境界，**Client** 端這邊的程式**讀起來像是一系列的註解**。

Extract Method



```
$Data = readDataFile("pick3.txt");  
showHotColdList($Data);
```

```
// 讀取資料檔  
function readDataFile($sFilename) {  
    // ...  
    return $result;  
}
```

```
// 顯示冷號熱號數列  
function showHotColdList($Data)  
{  
    // ...  
}
```

```
// 讀取資料檔
```

```
$sData = "";  
$f = fopen ( "pick3.txt", "r" );  
while ( ! feof ( $f ) ) {  
    $line = fgets ( $f );  
    $sData .= Trim ( $line );  
}  
fclose ( $f );
```

```
// 產生冷號熱號數列
```

```
$result = "0123456789";  
$iLen = strlen ( $sData );  
for($iPos = 0; $iPos < $iLen; $iPos ++ ) {  
    $sch = substr ( $sData, $iPos, 1 );  
    $result = $sch .  
        str_replace ( $sch, "", $result );  
}  
echo substr ( $result, 0, 5 ) . "-"  
    . substr ( $result, 5, 5 );
```

Inline Method

- 函式的內容十分簡明，其簡明的程度就與函式名稱無異。

Before

```
$amount = 7;  
$rating = MoreThan5($amount) ? 'A' : 'B';  
echo $rating;  
  
function MoreThan5($amount) {  
    return $amount > 5;  
}
```

After

```
$amount = 7;  
$rating = ($amount > 5) ? 'A' : 'B';  
echo $rating;
```

Inline Temp

- 變數的內容只被設定一次，而且也只被參用一次。

Before

```
function calculateTotal($unitPrice, $count)
{
    $total = $unitPrice * $count;
    return $total;
}
```

After

```
function calculateTotal($unitPrice, $count)
{
    return $unitPrice * $count;
}
```

Replace Temp with Query (一)

- 變數的內容存的是一個運算式，將運算式轉成函式。
- 找出內容只被設定過一次的臨時變數
 - 將等號右邊的運算式，擷取成一個 `private` 獨立函數。

Before

```
function getPrice() {  
    $basePrice = $this->_quantity * $this->_itemPrice;  
    if ($basePrice > 10000)  
        return $basePrice * 0.90;  
    else  
        return $basePrice * 0.95;  
}
```


Replace Temp with Query (二)

- 變數的內容存的是一個運算式，將運算式轉成函式。
- 找出內容只被設定過一次的臨時變數
 - 將等號右邊的運算式，擷取成一個 **private** 獨立函數。

After

```
function getPrice() {  
    if ($this->basePrice() > 10000)  
        return $this->basePrice() * 0.90;  
    else  
        return $this->basePrice() * 0.95;  
}  
  
private function basePrice() {  
    return $this->_quantity * $this->_itemPrice;  
}
```

Introduce Explaining Variable (—)

- 將複雜的運算式的結果，存入一個意義簡明的變數。

Before

```
$url = "http://" . str_replace (
    basename($_SERVER ["REQUEST_URI"]),
    "dest.php",
    $_SERVER ["SERVER_NAME"] . ":" .
    $_SERVER ["SERVER_PORT"] .
    $_SERVER ["REQUEST_URI"] );
```

```
// ...
```

Introduce Explaining Variable (二)

- 將複雜的運算式的結果，存入一個意義簡明的變數。

After

```
$sourceUrl = $_SERVER ["SERVER_NAME"] . ":" .  
    $_SERVER ["SERVER_PORT"] .  
    $_SERVER ["REQUEST_URI"];  
  
$url = "http://" . str_replace (  
    basename($_SERVER ["REQUEST_URI"]),  
    "dest.php",  
    $sourceUrl );  
  
// ...
```

Split Temporary Variable

- 同一個變數先後被設定過兩次內容，兩次的意義各不相同。
- 應該取兩個不同的變數名稱。

Before

```
$quantity = 2;  
$price = 25;  
  
$x = $quantity * $price;  
$x = $x * 1.05;  
  
echo $x;
```

After

```
$quantity = 2;  
$price = 25;  
  
$preTax = $quantity * $price;  
$afterTax = $preTax * 1.05;  
  
echo $afterTax;
```