B

抽象類別、介面與多型

本章學習目標

B-1 抽象類別與常數類別

「抽象類別」(Abstract Class)是一種不能完全代表物件的類別，抽象類別不能建立物件，主要目的是作為類別繼承的父類別，用來定義一些子類別的共同部分。常數類別表示類別不能被繼承，如果方法宣告成常數方法，表示方法不允許覆寫。

B-1-1 抽象類別與抽象方法

PHP 類別宣告成 abstract 表示是抽象類別，抽象類別不能建立物件，只能繼承抽象類別來宣告子類別。

在抽象類別可以使用 abstract 宣告抽象方法，表示方法只是原型宣告，實作程式碼是位在子類別，其繼承子類別一定要實作這些抽象方法。例如：在 PHP 程式宣告抽象類別 Shape，如下所示：

```
abstract class Shape {
   public $x;
   public $y;
   abstract function area();
}
```

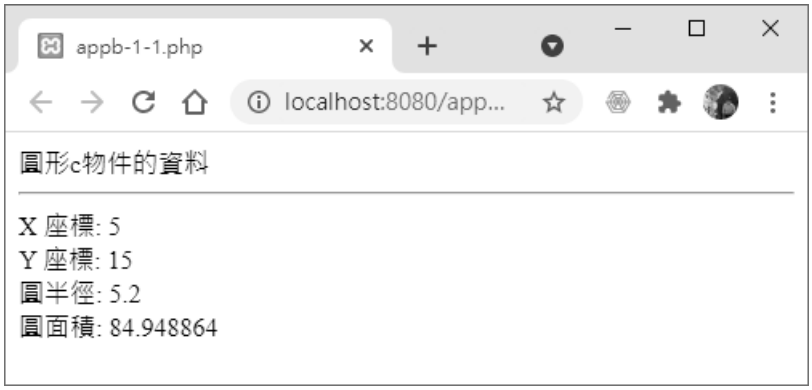
上述 Shape 類別定義點(Point)座標 $x 和 $y，和提供抽象方法 area() 計算形狀的面積，但是沒有方法的程式區塊。接著宣告 Circle 子類別繼承 Shape 類別，如下所示：

```
class Circle extends Shape {
   public $r;
   function __construct($x, $y, $r) { … }
   function area() {
      return 3.1416*$this->r*$this->r;
   }
}
```

上述子類別 Circle 定義圓形，除了圓心座標外，新增成員變數半徑 $r，和實作 area() 抽象方法來計算圓面積。

程式範例：appb-1-1.php

在 PHP 程式宣告 Shape 抽象類別，內含計算面積的 area() 抽象方法，然後建立 Circle 圓形類別繼承 Shape 類別，最後建立 Circle 物件顯示圓形的相關資訊，如下圖所示：



程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8" />
05: <title>appb-1-1.php</title>
06: <?php
07: abstract class Shape {    // Shape抽象類別宣告
08:     public $x;    // X座標
09:     public $y;    // y座標
10:     abstract function area(); // 抽象方法
11: }
12: class Circle extends Shape { // Circle類別宣告
13:     public $r;    // 半徑
14:     // 建構子方法
```

→ 接下頁

```
15:     function __construct($x, $y, $r) {
16:        $this->x = $x;
17:        $this->y = $y;
18:        $this->r = $r;
19:     }
20:     // 成員方法: 實作抽象方法area()
21:     function area() {
22:        return 3.1416*$this->r*$this->r;
23:     }
24: }
25: ?>
26: </head>
27: <body>
28: <?php
29: $c = new Circle(5.0, 15.0, 5.2);  // 建立物件
30: // 顯示圓形c的資料
31: echo "圓形c物件的資料<hr/>";
32: echo "X 座標: " . $c->x . "<br/>";
33: echo "Y 座標: " . $c->y . "<br/>";
34: echo "圓半徑: " . $c->r . "<br/>";
35: // 呼叫物件的成員方法
36: echo "圓面積: " . $c->area() . "<br/>";
37: ?>
38: </body>
39: </html>
```

程式說明

- 第 7~11 列：Shape 抽象類別的宣告，內含 area() 抽象方法。
- 第 12~24 列：繼承 Shape 類別的 Circle 子類別，第 21~23 列實作 area() 抽象方法。
- 第 29 列：使用 new 運算子建立 Circle 物件變數 $c。
- 第 32~36 列：顯示 Circle 物件的成員變數和呼叫成員方法 area()。

B-1-2 常數類別與常數方法

PHP 類別如果宣告成 final，表示類別不能被繼承；方法宣告成 final，表示方法不可以覆寫。例如：繼承父類別 Member 的 CarMember 類別，如下所示：

```
final class CarMember extends Member { … }
```

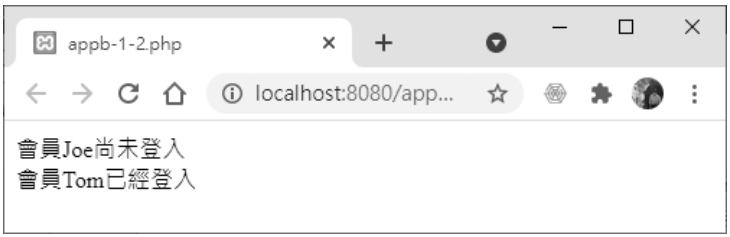
上述 final 宣告表示 CarMember 類別不能再有子類別，我們不能再繼承 CarMember 建立 NissanMember 子類別。在 Member 類別的方法是宣告成 final，如下所示：

```
class Member {
   ……
   final function isLogon() { return $this->status; }
   final function setStatus($s) { $this->status=$s; }
}
```

上述 2 個成員方法都宣告成 final，表示子類別 CarMember 不允許覆寫這些方法。

程式範例：appb-1-2.php

在 PHP 程式建立繼承 Member 類別的 CarMember 類別宣告，CarMember 類別是宣告成 final，Member 類別的方法也是宣告成 final，如下圖所示：



上述圖例是會員狀態，顯示會員 Joe 和 Tom 是否已經登入網站。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8" />
05: <title>appb-1-2.php</title>
06: <?php
07: class Member {      // Member類別宣告
08:    var $username;  // 成員資料
09:    var $password;
10:    private $status;
11:    // 成員方法
12:    final function isLogon() { return $this->status; }
13:    final function setStatus($s) { $this->status=$s; }
14: }
15: // CarMember類別宣告, 繼承自Member類別
16: final class CarMember extends Member {
17:    private $age;
18:    // 建構子方法
19:    function __construct($name, $pass, $age) {
20:       $this->username = $name;
21:       $this->password = $pass;
22:       $this->age = $age;
23:       $this->setStatus(false);
24:    }
25: }
26: ?>
27: </head>
28: <body>
29: <?php
30: $joe = new CarMember("Joe","5678",36);  // 建立物件
31: $tom = new CarMember("Tom","1234",28);
32: if ( $joe->isLogon() ) echo "會員Joe已經登入<br/>";
33: else echo "會員Joe尚未登入<br/>";
34: $tom->setStatus(true);   // 呼叫成員方法
35: if ( $tom->isLogon() ) echo "會員Tom已經登入<br/>";
36: else echo "會員Tom尚未登入<br/>";
37: ?>
38: </body>
39: </html>
```

程式說明

- 第 12~13 列：2 個宣告成 final 的成員方法。
- 第 16~25 列：使用 final 宣告 CarMember 類別。

B-2 介面

介面（Interface）是在類別繼承架構定義類別的行為，在介面宣告的方法是一種抽象方法，實作介面的類別需要實作「所有」抽象方法，如下所示：

```
interface AreaInterface {
   abstract function area();
}
```

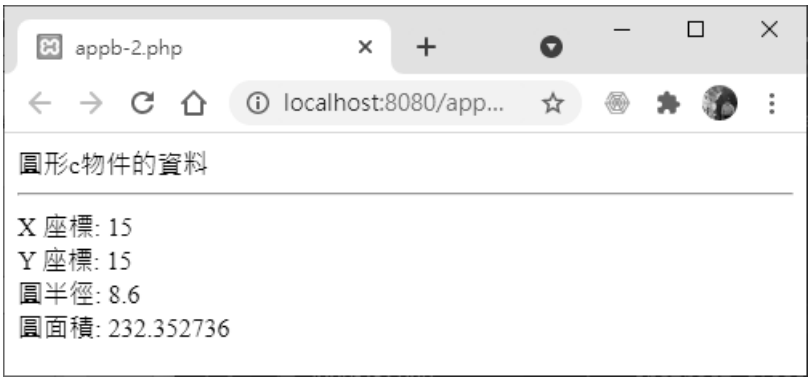
上述 PHP 介面宣告是使用 interface 關鍵字，類似類別架構，只是宣告內容是抽象方法（表示尚未實作）。類別可以實作一個或多個介面，如果實作多個介面，請使用「,」號分隔。例如：宣告 Circle 類別繼承 Shape 類別和實作 AreaInterface 介面，如下所示：

```
class Circle extends Shape implements AreaInterface {
   public $r;
   function __construct($x, $y, $r) {   }
   function area() {
      return 3.1416*$this->r*$this->r;
   }
}
```

上述 Circle 類別使用 implements 關鍵字實作 AreaInterface 介面，在類別宣告需要實作介面宣告的方法，即 area() 方法。

程式範例：appb-2.php

這個 PHP 程式是修改自 appb-1-1.php，改為使用介面宣告 area() 方法，如下圖所示：



程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8" />
05: <title>appb-2.php</title>
06: <?php
07: abstract class Shape {    // Shape抽象類別宣告
08:     public $x;    // X座標
09:     public $y;    // y座標
10: }
11: interface AreaInterface { // AreaInterface介面宣告
12:     function area(); // 介面方法
13: }
14: class Circle extends Shape    // Circle類別宣告
15:               implements AreaInterface {
16:     public $r;    // 半徑
17:     // 建構子方法
18:     function __construct($x, $y, $r) {
```

→ 接下頁

```
19:         $this->x = $x;
20:         $this->y = $y;
21:         $this->r = $r;
22:     }
23:     // 成員方法: 實作介面方法area()
24:     function area() {
25:         return 3.1416*$this->r*$this->r;
26:     }
27: }
28: ?>
29: </head>
30: <body>
31: <?php
32: $c = new Circle(15.0, 15.0, 8.6);  // 建立物件
33: // 顯示圓形c的資料
34: echo "圓形c物件的資料<hr/>";
35: echo "X 座標: " . $c->x . "<br/>";
36: echo "Y 座標: " . $c->y . "<br/>";
37: echo "圓半徑: " . $c->r . "<br/>";
38: // 呼叫物件的成員方法
39: echo "圓面積: " . $c->area() . "<br/>";
40: ?>
41: </body>
42: </html>
```

程式說明

- 第 11~13 列：在 AreaInterface 介面宣告 area() 方法。
- 第 14~27 列：Circle 類別繼承 Shape 抽象類別和實作 AreaInterface 介面，在第 24~26 列實作 area() 介面方法。
- 第 32 列：使用 new 運算子建立 Circle 物件變數 $c。
- 第 35~39 列：顯示 Circle 物件的成員變數和呼叫 area() 成員方法。

B-3 多型

「多型」(Polymorphism)是物件導向程式設計相當重要和複雜的觀念，可以讓應用程式更容易擴充，一個同名方法，就可以處理不同資料型態的物件，產生不同的操作。物件導向程式語言實作多型有兩種方法，如下所示：

- 類別繼承的成員方法覆寫：繼承基礎類別覆寫同名的成員方法來處理不同資料型態的物件，如果有新的資料型態，即物件，只需新增繼承的子類別和覆寫成員方法。
- 介面的成員方法實作：介面是指同一物件擁有多種型態，換個角度，不同物件可以擁有相同的介面型態，一樣可以透過介面來實作多型。

說明 PHP 的 Object 資料型態變數可以儲存各種物件，兩個類別只需擁有同名方法，就算沒有任何關係，在 PHP 也可以建立多型。

B-3-1 類別繼承實作多型

多型是物件導向技術中最複雜的觀念，在這一節筆者準備使用類別繼承的覆寫來實作多型，這是使用抽象類別繼承的多型實例。例如：Shape 抽象類別宣告，如下所示：

```
abstract class Shape {
   public $x;
   public $y;
   abstract function area();
}
```

上述抽象類別定義 area() 抽象方法。在 PHP 程式可以繼承此抽象類別建立 Circle（圓形）和 Rectangle（長方形）兩個子類別，其類別宣告如下所示：

```
class Circle extends Shape {
   ……
   function area() { return 3.1416*$this->r*$this->r; }
}
class Rectangle extends Shape {
   ……
   function area() { return $this->height*$this->width; }
}
```

上述 2 個子類別都實作 area() 抽象方法，只是內含程式碼不同，可以計算不同圖形的面積。現在我們可以建立 Circle 和 Rectangle 物件，如下所示：

```
$c = new Circle(15.0, 15.0, 8.0);
$r = new Rectangle(10.0, 10.0, 20.0, 15.0);
```

上述程式碼建立 Circle 和 Rectangle 物件。因為 PHP 的 Object 資料型態的變數可以指定成不同的物件參考，所以可以宣告 $obj 物件變數來參考 Circle 或 Rectangle 物件，如下所示：

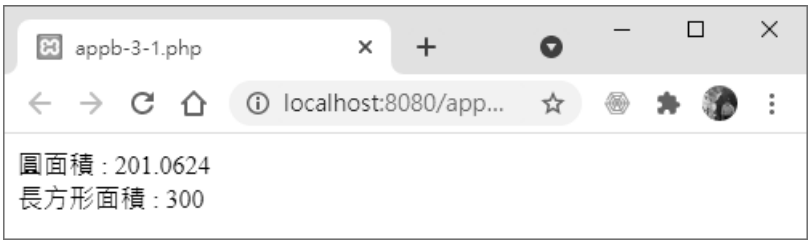
```
$obj = $c;
if ($obj instanceof Circle) echo "圓面積 : ";
echo $obj->area() . "<br/>";
```

上述 $obj 物件變數是參考 Circle 物件，可以使用 instanceof 運算子判斷物件變數是哪一種物件，然後呼叫 $obj->area() 方法取得圓形面積。

同樣的，當物件變數 $obj 是 Rectangle 物件時，我們仍然可以呼叫相同的 $obj->area() 方法取得長方形面積。兩種物件呼叫方法的程式碼相同，只是目標物件不同，這個 area() 方法稱為多型，即同名異式。

程式範例：appb-3-1.php

在 PHP 程式建立 Shape 抽象類別後，建立 Circle 和 Rectangle 子類別來建立 area() 多型方法，如下圖所示：



上述圖例顯示 2 個圖形面積，PHP 程式碼都是呼叫 $obj->area() 方法計算面積。多型方法是在執行階段才依照物件變數是哪一種物件，來執行該物件的方法，雖然方法名稱相同，但是執行結果卻不同。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8" />
05: <title>appb-3-1.php</title>
06: <?php
07: abstract class Shape {    // Shape抽象類別宣告
08:     public $x;    // X座標
09:     public $y;    // y座標
10:     abstract function area(); // 抽象方法
11: }
12: class Circle extends Shape {    // Circle類別宣告
13:     private $r;    // 半徑
14:     // 建構子方法
15:     function __construct($x, $y, $r) {
16:         $this->x = $x;     $this->y = $y;
17:         $this->r = $r;
18:     }
19:     // 成員方法: 實作抽象方法area()
```

→ 接下頁

```
20:     function area() { return 3.1416*$this->r*$this->r; }
21: }
22: class Rectangle extends Shape { // Rectangle類別宣告
23:     private $width;      // 寬
24:     private $height;     // 高
25:     // 建構子方法
26:     function __construct($x, $y, $w, $h) {
27:        $this->x = $x;      $this->y = $y;
28:        $this->width = $w; $this->height = $h;
29:     }
30:     // 成員方法: 實作抽象方法area()
31:     function area() { return $this->height*$this->width; }
32: }
33: ?>
34: </head>
35: <body>
36: <?php
37: $c = new Circle(15.0, 15.0, 8.0);  // 建立物件
38: $r = new Rectangle(10.0, 10.0, 20.0, 15.0);
39: // 呼叫物件的方法area()
40: $obj = $c;    // 圓形
41: if ($obj instanceof Circle) echo "圓面積 : ";
42: echo $obj->area() . "<br/>";
43: $obj = $r;    // 長方形
44: if ($obj instanceof Rectangle) echo "長方形面積 : ";
45: echo $obj->area() . "<br/>";
46: ?>
47: </body>
48: </html>
```

程式說明

- 第 7~11 列：Shape 抽象類別宣告擁有 area() 抽象方法。
- 第 12~21 列：繼承 Shape 類別的 Circle 子類別，第 20 列是 area() 抽象方法的實作。
- 第 22~32 列：繼承 Shape 類別的 Rectangle 子類別，第 31 列是 area() 抽象方法的實作。

- 第 37~38 列：使用 new 運算子建立 Circle 和 Rectangle 物件變數 $c 和 $r。
- 第 40~42 列：$obj 是 Circle 物件變數 $c，if 條件使用 instanceof 運算子判斷是否是 Circle 物件，$obj->area() 是呼叫 Circle 物件的 area() 方法。
- 第 43~45 列：$obj 是 Rectangle 物件變數 $r，if 條件使用 instanceof 運算子判斷是否是 Rectangle 物件，$obj->area() 是呼叫 Rectangle 物件的 area() 方法。

B-3-2 使用介面實作多型

PHP 也可以實作介面方法來建立多型。例如：IArea 介面宣告，如下所示：

```
interface IArea {
   function area();
}
```

上述介面定義 area() 介面方法。在 PHP 程式的 Circle（圓形）和 Rectangle（長方形）類別都實作 IArea 介面，其類別宣告如下所示：

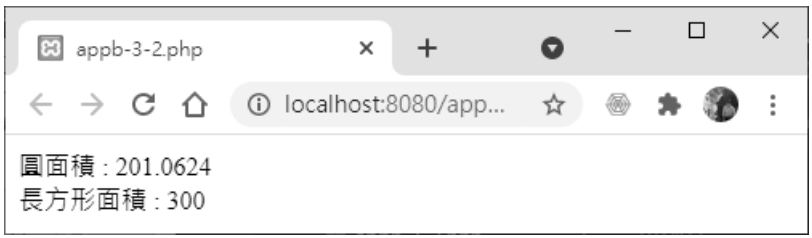
```
class Circle implements IArea {
   ......
   function area() { return 3.1416*$this->r*$this->r; }
}
class Rectangle implements IArea {
   ......
   function area() { return $this->height*$this->width; }
}
```

上述 2 個類別都實作 area() 方法，只是內含程式碼不同，可以分別計算不同圖形的面積。

如同 B-3-1 節，我們一樣可以將物件變數 $obj 指定成 Circle 或 Rectangle 物件，然後使用相同的 $obj->area() 程式碼呼叫來計算圖形面積，換句話說，area() 方法就是多型。

程式範例：appb-3-2.php

在 PHP 程式建立 IArea 介面後，建立 Circle 和 Rectangle 類別實作此介面，以便建立 area() 多型方法，如下圖所示：



上述圖例顯示 2 個圖形的面積，而且都是呼叫 $obj->area() 方法的執行結果，這就是多型。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8" />
05: <title>appb-3-2.php</title>
06: <?php
07: interface IArea {     // IArea介面宣告
08:     function area();  // 介面方法
09: }
10: class Circle implements IArea { // Circle類別宣告
11:     private $r;       // 半徑
12:     // 建構子方法
13:     function __construct($r) { $this->r = $r; }
14:     // 成員方法: 實作介面方法area()
15:     function area() { return 3.1416*$this->r*$this->r; }
```

→ 接下頁

```
16: }
17: class Rectangle implements IArea { // Rectangle類別宣告
18:     private $width;      // 寬
19:     private $height;     // 高
20:     // 建構子方法
21:     function __construct($w, $h) {
22:        $this->width = $w;  $this->height = $h;
23:     }
24:     // 成員方法: 實作介面方法area()
25:     function area() { return $this->height*$this->width; }
26: }
27: ?>
28: </head>
29: <body>
30: <?php
31: $c = new Circle(8.0);  // 建立物件
32: $r = new Rectangle(20.0, 15.0);
33: // 呼叫物件的方法area()
34: $obj = $c;    // 圓形
35: if ($obj instanceof Circle) echo "圓面積 : ";
36: echo $obj->area() . "<br/>";
37: $obj = $r;    // 長方形
38: if ($obj instanceof Rectangle) echo "長方形面積 : ";
39: echo $obj->area() . "<br/>";
40: ?>
41: </body>
42: </html>
```

程式說明

- 第 7~9 列：IArea 介面宣告擁有 area() 介面方法。
- 第 10~16 列：Circle 類別實作 IArea 介面，第 15 列是 area() 介面方法的實作。
- 第 17~26 列：Rectangle 類別實作 IArea 介面，第 25 列是 area() 介面方法的實作。

- 第 31~32 列：使用 new 運算子建立 Circle 和 Rectangle 物件變數 $c 和 $r。
- 第 34~36 列：$obj 是 Circle 物件變數 $c，if 條件使用 instanceof 運算子判斷是否是 Circle 物件，$obj->area() 是呼叫 Circle 物件的 area() 方法。
- 第 37~39 列：$obj 是 Rectangle 物件變數 $r，if 條件使用 instanceof 運算子判斷是否是 Rectangle 物件，$obj->area() 是呼叫 Rectangle 物件的 area() 方法。