PHP物件導向程式設計

PHP雖說是在HTML中的一種Script語言，但他也是一個物件導向語言。物件導向程式設計(OOP:Object-Oriented Programming)是當前程式語言的發展主流，它更符合人類『跳動式』的思考模式!不過，對於已熟悉結構化（一個命令、一個動作）程式語言的人而言，因為思考模式的改變，確實需要花點時間來適應。

物件，從某個角度來看，可以看做是一種函式庫的延伸。讓我們試著這樣看:

字母符號：原物料。

指令 ：一個指令是一種零件。

函數 ：半成品。

物件導向：成品或半成品。

在函數階段，我們仍停留在因應需要而設計的階段。

而物件乃是以整個完整工作內容(稱為：**類別**)為考量，它提供一個完整工作內容的所有「可能」需求功能(**物件**)，每一個功能給予一個名稱，當需用某項功能時，只須呼叫「功能名稱」即可，（對於用不到的項目，也就不管它）。

上一段的說明，可能會讓同學感覺太過艱澀，我們換個方式來說：

　　在函數中，主程式和函數是「壁壘分明」，函數是透過主程式的呼叫才會執行的，而：

1. 函數中使用的變數與主程式中所使用的變數是各自獨立的
2. 函數執行中，所使用的變數其資料變換過程，與主程式並無直接關聯，主程式如果需要用到，則可能要先將中間變數值，放置到預定回傳的資料陣列變數中，等函數執行完成後再將資料回傳。
3. 函數功能執行完成並(如有需要)回傳指定的資料後，工作即告結束。其作業中的各項資料變數也宣告失效，後續如有需要，則須重新進行呼叫並重新傳送對應的資料，函數才會啟動對的新工作。
4. 簡單來說，就是主程式和函數不是「同步作業」

物件導向的觀念是：

1. 基本上他是一個「函數庫」(稱為:類別/物件集合)，在類別中放置了 各種各樣可能會用到的函數，且在類別中也有自行設定的一些變數。
2. 我們在主程式可以設定一個「物件聯結變數」與類別連結，來開啟同步作業
3. 主程式中，我們可以透過「物件聯結變數」隨時呼叫類別的函數或「設定/取得」類別中的變數資料。
4. 也可設定延伸類別(又稱:子類別)，在資料的使用上，可以互相連通應用。

下面我們以一些例子來做逐項說明。

物件導向程式設計中一個重要的觀念是 類別(class)和物件(object)，類別是物件導向中的一個定義型態(可以看成是 物件名稱)，而物件則是在類別中的實際執行單位(函數)。

首先要宣告類別的區域(定義)：

類別宣告指令的格式如下：

class 類別名稱

{

var $變數 A;

var $變數 B;

function 名稱1($參數1[,$參數2,…] )

{

名稱1程式執行區域;($this->A)

}

function 名稱2($參數1[,$參數2,…])

{

名稱2程式執行區域;

}

}

說明：在類別中的成員可以區分為「成員資料（member data）」與「成員函數（member function）」兩類

1.類別中的變數或常數，為靜態屬性，稱為【成員資料】

2.類別中的函數，為動態運算方法，稱為【成員函數】

3.類別中宣告的變數，前面必須加上（var）關鍵字。

4.成員函數中的『$this』表示類別本身；『->』後面接類別中宣告的變數，變數前面不加 $。

我們以範例說明如下：

class cart //cart為類別名稱

{

var $owner;

var $price=300; //每套產品的定價，變數前面加上 var

var $goods;

function add\_item($title, $amt){

//新增項目(數量)到購物車中

$this->goods[$title] +=$amt; //$this-> 表示「這個類別的」

//傳回數量

return $amt;

}

function remove\_item($title, $amt)

{

//由購物車內移除項目(資料)

$this->goods[$title] -=$amt;

//傳回數量

return $amt;

}

function get\_owner()

{

//傳回購物車擁有者的名稱

return $this->owner**;**

}

function get\_goods(){

//傳回購物車中的物件清單

return $this->goods;

}

}//end class

定義一項類別名稱後，要使用該項類別(物件功能)的相關指令功能，必須先設定該項類別的連結變數，格式如下：

$物件名稱變數=new 類別名稱; //物件與類別配對

建立好 「連結物件」 的變數名稱後，就可以用下面指令來「設定/取用 物件屬性(資料)」 或是 「使用物件所提供的方法」

$物件名稱變數 ->屬性 = 屬性值; //傳送資料給類別

$物件名稱變數 ->名稱n($參數1[,$參數2,…]); //没有傳回值(在類別直接處理)

$變數=$物件名稱 ->名稱n($參數1[,$參數2,…]) //處理完成後之值要傳回

同樣的，我們用範例來做說明：

<?php

//建立一個新物件

$mycart=new cart;

//指定購物車名稱

$mycart->owner="Linda";

//新增物品到購物車

$item += $mycart->add\_item("Access",3);

$item += $mycart->add\_item("Word",2);

?>

我們將上面兩個部份結合再做一個較完整功能的例子來說明

例：設計一個cart購物車類別，內含有add\_item、remove\_item、get\_item和get\_goods等(函數)名稱，以及owner、price和goods等屬性，其中price設定是300元，而goods為陣列，用來儲存購物車中的貨品。以物件導向程式的指令，在購物車中新增和刪除項目，最後列出所有的項目清單和總價。

ch15-01.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>物件與類別</title>

</head>

<body>

<?php

//宣告類別

class cart

{

var $owner;

var $price=300; //每套產品的定價

var $goods;

function add\_item($title, $amt){

//新增項目到購物車中

$this->goods[$title] +=$amt;

//傳回數量

return $amt;

}

function remove\_item($title, $amt){

//由購物車內移除項目

if($this->goods[$title]>$amt)

{

$this->goods[$title] -=$amt;

//傳回數量

return $amt;

}else{

echo "錯誤!! $title 移除的量大於該項目既有的量!<br>";

return false;

}

}

function get\_owner(){

//傳回購物車擁有者的名稱

return $this->owner;

}

function get\_goods(){

//傳回購物車中的物件清單

return $this->goods;

}

}//end class

$item=0;

//建立一個新物件

$mycart=new cart;

//指定購物車名稱

$mycart->owner="Linda";

//新增物品到購物車

$item += $mycart->add\_item("Access",3);

$item += $mycart->add\_item("Word",2);

$item += $mycart->add\_item("PowerPoint",3);

$item += $mycart->add\_item("VB",4);

$item += $mycart->add\_item("Access",1);

//移除購物出車中的物品

$item -= $mycart->remove\_item("PowerPoint",1);

//取得購物車的擁有者

$na=$mycart->get\_owner();

echo $na."<p>";

//取得購物車內的物件清單

$list=$mycart->get\_goods();

print\_r($list)."<p>";

echo "<p>採購總金額 : ".$item \* $mycart->price;

?>

</body>

</html>

範例ch15-01.php是物件類別的基本用法，另外一些延伸功能，我們在下面逐一討論。

建構子(Constructor)

在類別的成員函數中，可以有一個(而且只能有一個)比較特別的函數，稱為「建構子」。建構子的名稱和類別同名，當以new方式建立一個該類別的物件實體時，建構子會被自動執行。

建構子的指令格式如下：

$物件名稱=new 類別名稱([$參數1,參數2,…]);

PHP5也允許以 \_\_construct( )方式來宣告建構子，指令格式如下：

function \_\_construct([$參數1,參數2,…]) // \_ \_為二個連續底線

{

建構子程式執行區塊;

}

例：設計一個購物車類別，含有一個建構子，它會顯示一個歡迎的訊息，購物車類別中其他的成員函數和成員資料和7-1-1.php中相同。

ch15-02.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>建構子</title>

</head>

<body>

<?php

//宣告類別

class cart

{

var $owner;

var $price=300; //每套產品的定價

var $goods;

function cart($name){

//建構子

$this->owner=$name;

echo "<h2><font color=purple>歡迎進入志凌資訊學習網!</h2>";

echo $name."建立了cart類別的物件</font><p>";

}

function add\_item($title, $amt){

//新增項目到購物車中

$this->goods[$title] +=$amt;

//傳回數量

return $amt;

}

function remove\_item($title, $amt){

//由購物車內移除項目

if($this->goods[$title]>$amt){

$this->goods[$title] -=$amt;

//傳回數量

return $amt;

}else{

echo "錯誤!! $title 移除的量大於該項目既有的量!<br>";

return false;

}

}

function get\_owner(){

//傳回購物車擁有者的名稱

return $this->owner;

}

function get\_goods(){

//傳回購物車中的物件清單

return $this->goods;

}

}//end class

$item=0;

//建立一個新物件並指定名稱

$mycart=new cart("Peter");

//新增物品到購物車

$item += $mycart->add\_item("ASP.NET",3);

$item += $mycart->add\_item("Java",2);

$item += $mycart->add\_item("JavaScript",3);

$item += $mycart->add\_item("VB",4);

$item += $mycart->add\_item("ASP",1);

//取得購物車的擁有者

$na=$mycart->get\_owner();

echo $na."<p>";

//取得購物車內的物件清單

$list=$mycart->get\_goods();

print\_r($list)."<p>";

echo "<p>採購總金額 : ".$item \* $mycart->price;

?>

</body>

</html>

解構子(Destructor)

物件導向程式語言，在程式結束時會自動回收各物件所佔用的系統資源。不過，PHP5提供了「解構子」函數，可以在程式結束之前，在移除物件並釋放所用的系統資源時，執行解構子的程式。

解構子指令格式如下：

Class 類別名稱

{

類別程式執行區塊;

function\_\_destruct( )

{

解構子程式執行區塊;

}

}

說明：解構子函數通常會放在類別宣告程式區塊的最後面

例：設計一個cart類別，其中含有一個解構子，在其中以echo顯示「即將結束程式!」的訊息，其餘內容和8-1-2.php相同。

ch15-03.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>解構子</title>

</head>

<body>

<?php

//宣告類別

class cart

{

var $owner;

var $price=300; //每套產品的定價

var $goods;

function cart($name){

//建構子

$this->owner=$name;

echo "<h2><font color=purple>歡迎進入志凌資訊學習網!</h2>";

echo $name."建立了cart類別的物件</font><p>";

}

function add\_item($title, $amt){

//新增項目到購物車中

$this->goods[$title] +=$amt;

//傳回數量

return $amt;

}

function remove\_item($title, $amt){

//由購物車內移除項目

if($this->goods[$title]>$amt)

{

$this->goods[$title] -=$amt;

//傳回數量

return $amt;

}else{

echo "錯誤!! $title 移除的量大於該項目既有的量!<br>";

return false;

}

}

function get\_owner(){

//傳回購物車擁有者的名稱

return $this->owner;

}

function get\_goods(){

//傳回購物車中的物件清單

return $this->goods;

}

function \_\_destruct(){

//解構子

echo "<p>即將結束程式 !....";

}

}//end class

//建立一個新物件並指定名稱

$mycart=new cart("Alice");

//新增物品到購物車

$item += $mycart->add\_item("PHP & MySQL",3);

//取得購物車的擁有者

$na=$mycart->get\_owner();

echo $na."<p>";

//取得購物車內的物件清單

$list=$mycart->get\_goods();

print\_r($list)."<p>";

?>

</body>

</html>

設定類別存取權限

PHP允許在建立類別時，加入「存取關鍵字」，以便設定屬性和方法的存取權限，關鍵字共有「public、protected、private和static」4個，存取關鍵字的指令格式用法如下：

　class 類別名稱

　　｛

　　　　存取關鍵字　$變數名稱；

　　　　存取關鍵字　function 函數名稱（參數）；

　　　 {

函數執行程式區塊；

　　　　　}　　//end function

……

} //end class

「存取關鍵字」功能分別敍述如下：

public：

類別中的成員，可以在程式碼中的任何位置被呼叫或存取

　　　　　以phblic關鍵字宣告

<? php

同一類別中存取

子類別中存取

外部區域直接存取

class 類別 A

{

public 變數 x；

　　　　　　　　　 function AA( )

{ 程式區塊；}

}

子類別

　　　　　　　　　{

　　　　　　　　　　成員資料或成員函數；

　　　　　　　　　}

類別外部程式執行區塊； P.S.子類別的應用在下一節

　　　　　？＞ 再解說

　　　　　　　　PHP程式

protected：

類別中的成員，只可以在同一類別或衍生子類別中被呼叫或存取

　　　　　以protected關鍵字宣告

<? php

同一類別中存取

子類別中存取

外部區域不能存取

class 類別 A

{

protected 變數 x；

　　　　　　　　　 function AA( )

{ 程式區塊；}

}

子類別

　　　　　　　　　{

　　　　　　　　　　成員資料或成員函數；

　　　　　　　　　}

類別外部程式執行區塊；

　　　　　？＞

　　　　　　　　PHP程式

private：

類別中的成員，只可以在同一類別中被呼叫或存取

　　　　　以protected關鍵字宣告

<? php

同一類別中存取

子類別中存取

外部區域不能存取

class 類別 A

{

private 變數 x；

　　　　　　　　　 function AA( )

{ 程式區塊；}

}

子類別

　　　　　　　　{

成員資料或成員函數；

　　　　　　　　　}

類別外部程式執行區塊；

　　　　　？＞

　　　　　　　　PHP程式

static：

如果宣告為static，主程式中可以直接呼叫或存取類別成員，而不用事先建立對應的物件實體。

呼叫static類別的指令格式如下：

類別名稱**::**成員資料=屬性值；

類別名稱**::**成員函數（參數）；

例：設計一個vehicle類別，包含三個變數(屬性)和六個成員函數(方法)，其中三個變數分別設定不同的存取權限，只有public所設定$type變數可以在類別外的程式區直接呼叫，其餘只能透過類別所提供的方法去設定。

ch15-04.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>設定類別存取權限</title>

</head>

<body>

<h2><font color="purple">類別存取權限</font></h2>

<?php

//宣告類別

class vehicle

{

//設定不同的存取權限

public $type;

protected $model;

private $color;

static $test;

function set\_type($tt){

//設定交通工具屬性

$this->type=$tt;

}

function set\_model($mm){

//設定車子的廠牌

$this->model=$mm;

}

function set\_color($cc){

//設定車子顏色

$this->color=$cc;

}

function getcartype() {

//傳回$type屬性

return $this->type;

}

function getmodel(){

//傳回$model屬性

return $this->model;

}

function getcolor(){

//傳回$color屬性

return $this->color;

}

}//end class

$Lindacar=new vehicle;

$Lindacar->type="Van";

$Lindacar->set\_model("TOYOTA");

$Lindacar->set\_color("silver");

//$Lindacar->model="TOYOTA";

//$Lindacar->color="red";

vehicle::$test="測試資料";

//$ttt=vehicle::$test;

echo "Linda的車型是 : ".$Lindacar->getcartype()."<p>";

echo "Linda車的顏色是 : ".$Lindacar->getcolor()."<p>";

echo "Linda車的廠牌是 : ".$Lindacar->getmodel()."<p>";

//echo $ttt;

echo vehicle::$test;

?>

</body>

</html>

類別與物件的應用

繼承

繼承是物件導向最重要的特性，藉由類別的繼承，可以使用其他類別中已經宣告的變數(成員資料)或函數(成員函數)。

繼承類別稱為 衍生類別(Derived Class) 或 子類別(Child Class)

被繼承類別稱為 基礎類別(Bass Class) 或 父類別(Parent Class)

<? php

父類別

class 類別 A

{

父類別成員資料

var 變數 x；

呼叫父類別的函數

父類別成員函數

　　　　　　　　　 function AA( )

{ 程式區塊；}

}

子類別

class 類別 B extends 類別 A

　　　　　　　　　{

子類別成員函數

子類別成員資料

var 變數 y；

　　　　　　　　　　function BB

{

parent**::**AA()

程式區塊；

}

　　　　　　　　　}

類別外部程式執行區塊；

　　　　　？＞

　　　　　　　　PHP程式

由上圖中，應該可以瞭解到：「子類別除了擁有自己的『成員資料』及『成員函數』外，也同時可以使用父類別的『成員資料』及『成員函數』。」

子類別的語法如下：

class 類別 B extends 類別 A

　　　　{

var 變數；

　　　　　function 函數名稱

{程式區塊；}

$this->變數名稱； //呼叫父類別成員資料

parent**::**函數名稱(參數)； //呼叫父類別成員函數

　　　　}

例：設計一個含有vehicle類別，包含車子的型態(type)、廠牌(model)和顏色(color)屬性。另外設計一個car類別，它繼承vehicle類別，同時有車輪(wheel)型態，和相關的成員函數。

ch15-05.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>類別繼承</title>

</head>

<body>

<h2><font color="#CC00CC">類別繼承</font></h2>

<?php

//宣告類別

class vehicle

{

//相關屬性

var $type;

var $model;

var $color;

function set\_type($tt){

//設定交通工具屬性

$this->type=$tt;

}

function set\_model($mm){

//設定車子的廠牌

$this->model=$mm;

}

function set\_color($cc){

//設定車子顏色

$this->color=$cc;

}

function getcartype() {

//傳回$type屬性

return $this->type;

}

function getmodel(){

//傳回$model屬性

return $this->model;

}

function getcolor(){

//傳回$color屬性

return $this->color;

}

}//end vehicle class

class car extends vehicle

{//繼承自vehicle 的子類別

var $wheel;

function showcar($x,$y,$z)

{

//是四輪的車子

$this->wheel=4;

//呼叫父類別的函數

parent::set\_type($x);

parent::set\_model($y);

parent::set\_color($z);

//顯示車子的相關資料

echo "你車子的類型是 : <font color='blue'> ".$this->type."</font><p>";

echo "你車子的輪子數是 : <font color='blue'> ".$this->wheel."</font><p>";

echo "你車子的廠牌是 : <font color='blue'> ".$this->model."</font><p>";

echo "你車子的顏色是 : <font color='blue'> ".$this->color."</font><p>";

} //end function

} //end car class

//建立新的 car 物件

$Alicecar=new car;

//使用物件所提供的方法

$Alicecar->showcar("家用房車","Nissan","red");

?>

</body>

</html>

多形性的應用(polymorphism)

多形性的觀念是指：如果子類別中的成員函數和父類別中的成員函數名稱相同時，php程式會以子類別的成員函數為主。

說明:可能在父類別中有設定10個函數，而在子類別中，會用到8個函數，另

外有1個函數操成步驟不同，如此，在子類別中，只需重新定義這個不同  
函數的操作步驟，其餘的仍可直接承續應用父類別的函數。

例：設計一個cart類別，在其中定義get\_owner()成員函數。另外設計兩個子類別，分別繼承自cart類別，兩個子類別中都具有自己的get\_owner()成員函數，在類別外呼叫子類別時，會使用自己的get\_owner()成員函數。

ch15-06.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>多形性的應用</title>

</head>

<body>

<h2><font color="#CC00FF">多形性的應用</font></h2>

<?php

//宣告類別

class cart

{//建立購物車類別

var $owner;

var $store;

function get\_owner($x)

{

$this->owner=$x;

return $his->owner;

} //end function

} //end cart class

class store\_cart extends cart

{//繼承自cart類別的商店購物車

function get\_owner($x)

{

//設定商店名稱

$this->store="大家發商店";

echo "<font color='green'>歡迎光臨 <i>".$this->store."</i><p>";

echo $x." 客戶您好</font><p>";

} //end function

} //end store\_cart class

class book\_cart extends cart

{//繼承自cart類別的書店購物車

function get\_owner($x)

{

//設定商店名稱

$this->store="志凌書店";

echo "<font color='blue'>歡迎光臨 <i>".$this->store."</i><p>";

echo $x." 客戶您好</font><p>";

} //end function

} //end book\_cart class

//宣告新的物件

$Linda=new book\_cart;

//使用類別內自訂的成員函數

$Linda->get\_owner("林錦雀");

//宣告新的物件

$Caroline=new store\_cart;

//使用類別內自訂的成員函數

$Caroline->get\_owner("羅微娟");

?>

</body>

</html>

抽象類別(Abstract class)

抽象類別是PHP 5新增的功能，設計者可以先定義一個 「只包含成員資料，和『不包含任何內容』的成員函數」的父類別，稱為 【類別原型】，又稱【抽象類別】。

設計者必須在子類別中再重新撰寫成員函數所需要的內容。抽象類別的宣告，必須在class之前加上「abstract」關鍵字，指令格式如下：

abstract class 抽象類別名稱

{

var $變數 A;

var $變數 B;

abstract function 成員函數名稱()

}

例：定義一個cart抽象類別，其中內含get\_owner($x)成員函數。另外設計兩個子類別。在兩個子類別中，重新去定義get\_owner()抽象函數的內容。

ch15-07.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>抽象類別</title>

</head>

<body>

<h2><font color="#CC00FF">抽象類別</font></h2>

<?php

//宣告抽象類別(類別的原型)

abstract class cart

{//建立購物車類別

var $owner;

var $store;

//定義抽象成員函數

abstract function get\_owner($x);

} //end abstract class

class store\_cart extends cart

{ //繼承自cart類別的商店購物車

//重新定義抽象成員函數

function get\_owner($x)

{

//設定商店名稱

$this->store="大順利網路商店";

echo "<font color='green'><h3>".$this->store."</h3>";

echo $x." 客戶您好</font><p>";

} //end function

} //end store\_cart class

class book\_cart extends cart

{//繼承自cart類別的書店購物車

//重新定義抽象成員函數

function get\_owner($x)

{

//設定商店名稱

$this->store="私房教師書店";

echo "<font color='blue'><h3>".$this->store."</h3>";

echo $x." 客戶您好</font><p>";

} //end function

} //end book\_cart class

//宣告新的物件

$Linda=new book\_cart;

//使用類別內自訂的成員函數

$Linda->get\_owner("廖佐育");

//宣告新的物件

$Caroline=new store\_cart;

//使用類別內自訂的成員函數

$Caroline->get\_owner("江高舉");

?>

</body>

</html>

介面(interface)類別

介面類別和抽象類別功用非常類似，差別在於

1.介面類別不含 成員資料(變數)。

2.宣告關鍵字：interface(介面類別)，abstract class(抽象類別)。

3.子類別引用關鍵字：implements(介面類別)、extends(抽象類別)

4.子類別可以同時引用多個介面類別，但只能繼承一個抽象類別。

介面類別指令格式如下：

interface 介面名稱

{

function 函數名稱()；

}

子類別引用指令格式如下：

class 類別名稱 implements 介面1**,**介面2[**,…]**

{

var $變數；

　function 函數名稱

{程式區塊；}

　　function 函數名稱

{程式區塊；}

}

例：宣告itemtotal和owner兩個介面，另外cart類別引用前面的兩個介面類別。類別中有自己的成員資料和成員函，另外還須重新定義介面中的成員函數。再應用cart類別建立對應的物件實體，並顯示相關的訊息。

ch15-08.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>介面</title>

</head>

<body>

<h2><font color="#CC00FF">介面</font></h2>

<?php

//宣告介面(類別的原型,不含任何屬性定義)

interface itemtotal

{

//宣告介面成員函數

function item\_total($x);

}

//宣告另一個介面

interface owner

{

//宣告介面成員函數

function get\_owner($x);

}

//cart類別引用itemtotal和owner介面

class cart implements itemtotal,owner

{

//宣告自已的成員資料

var $owner;

var $store;

var $item=0;

//自訂自己的成員函數

function get\_store()

{

//指定商店的名稱

$this->store="私房教師數位學習網";

return $this->store;

}

//重新定義介面的item\_total()函數

function item\_total($x)

{

//將購買項目加總到原有的$item屬性

$this->item +=$x;

//傳回$item屬性值

return $this->item;

} //end function

//重新定義介面的get\_owner()函數

function get\_owner($x)

{

$this->owner=$x;

//傳回owner屬性值

return $this->owner;

}

} //end class

//建立新的物件

$Elearning=new cart;

echo "這家網路商店的名稱 : <font color='blue'>".$Elearning->get\_store()."</font><p>";

echo "目前購物車的車主是 : <font color='blue'>".$Elearning->get\_owner("Elvin")."</font><p>";

$total=$Elearning->item\_total(100);

echo "加買100項後<br/>";

echo "目前購買的數量是 : <font color='blue'>".$total."</font><p>";

$total=$Elearning->item\_total(5);

echo "加買5項後<br/>";

echo "目前購買的數量是 : <font color='blue'>".$total."</font><p>";

?>

</body>

</html>

類別、物件相關函數

PHP有提供一些與類別及物件相關的內建函數，如下表：

|  |  |
| --- | --- |
| 內建函數指令格式 | 功能說明 |
| class\_exists(類別名稱) | 判斷指定的類別是否存在 |
| interface\_exists(介面名稱) | 判斷指定的介面是否存在 |
| method\_exists(類別名稱**,**函數名稱) | 判斷指定類別中的指定函數是否存在 |
| property\_exists(類別名稱**,**屬性名稱) | 判斷指定類別中的指定屬性是否存在 |
| is\_subclass\_of(物件名稱**,**類別名稱) | 判斷指定物件是否為指定類別的「子類別物件」 |
| get\_class(物件名稱) | 傳回物件對應的類別名稱 |
| get\_class\_methods(類別名稱) | 以陣列模式傳回類別中的所有成員函數清單 |
| get\_class\_vars(類別名稱) | 以陣列模式傳回類別中的所有成員資料清單 |
| get\_object\_vars(物件名稱) | 以陣列模式傳回物件中的所有成員資料清單 |
| get\_parent\_class(類別名稱) | 傳回父類別的名稱 |

例：宣告一類別，其中含有成員資料和成員函數。利用get\_class函數傳回物件對應的類別名稱，並列出該類別中所含的所有成員資料和成員函數。

ch15-09.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>類別相關函數</title>

</head>

<body>

<h2><font color="#CC00FF">類別相關函數</font></h2>

<?php

//宣告類別

class vehicle

{

//相關屬性

var $type;

var $model;

var $color;

function set\_type($tt){

//設定交通工具屬性

$this->type=$tt;

}

function set\_model($mm){

//設定車子的廠牌

$this->model=$mm;

}

function set\_color($cc){

//設定車子顏色

$this->color=$cc;

}

function getcartype() {

//傳回$type屬性

return $this->type;

}

function getmodel(){

//傳回$model屬性

return $this->model;

}

function getcolor(){

//傳回$color屬性

return $this->color;

}

}//end vehicle class

//建立物件實體

$PeterCar=new vehicle;

//使用類別相關函數

$clsname=get\_class($PeterCar);

echo "\$PeterCar物件對應的類別是 : ".$clsname."<p>";

//顯示類別中的所有方法清單

$method\_array=get\_class\_methods($clsname);

echo "$clsname 類別的所有成員函數: <p>";

print\_r($method\_array);

//顯示類別中的所有屬性清單

$property\_array=get\_class\_vars($clsname);

echo "<p>$clsname 類別的所有成員資料: <p>";

print\_r($property\_array);

?>

</body>

</html>

單元作業1：將猜數字程式的guess2.php包裝成物件類別模式

(guessob0.htm、guessob1.php、guess\_ob.php)、

單元作業2：將樂透號碼程式以物件導向模式指令來設計