PHP5&MySQL程式設計

第6章 物件導向

6-1　認識物件導向

- 「物件」(object) 或「案例」(instance)
- 「屬性」(property)、「欄位」(field) 或「成員變數」(member variable)
- 「方法」(method) 或「成員函式」(member function)
- 「事件」(event)
- 「類別」(class)

物件導向程式設計 (OOP) 具有下列特點：

- 封裝 (encapsulation)
- 繼承 (inheritance)
- 介面 (interface)
- 多型 (polymorphism)

6-2　定義類別

PHP的類別可以包含下列成員：

一、 屬性 (又稱爲欄位、成員變數)

二、 方法 (又稱爲成員函式)

三、 常數

四、 建構函式 (constructor)

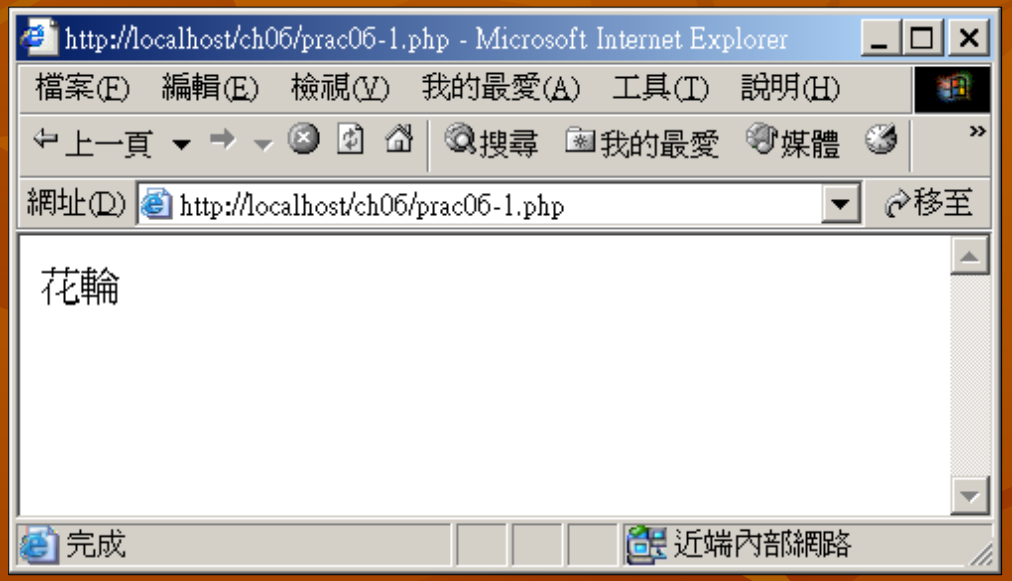
五、 解構函式 (destructor)

```
class class_name [extends parentclass_name]
{
  [public|private|protected|var $property_name [= value];]
  [[public|private|protected] function method_name(…){…}]
  […]
}

var $MyName = '小丸子';

class MyClass
{
  var $MyName = '小丸子';
  function Display()
  {
    echo $this->MyName;
  }
}
```

6-2-1 建立類別的物件

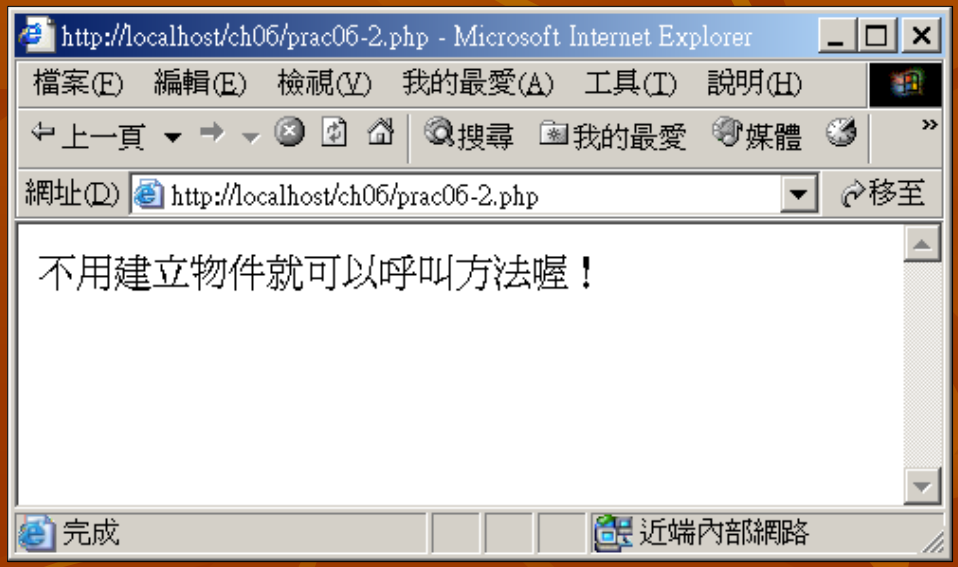
\ch06\prac06-1.php

```
<HTML>
 <BODY>
  <?php
    class MyClass
    {
      var $MyName = '小丸子';
      function Display()
      {
        echo $this->MyName;
      }
    }
    $MyObject = new MyClass();
    $MyObject->MyName = '花輪';
    $MyObject->Display();
  ?>
 </BODY>
</HTML>
```



\ch06\prac06-2.php

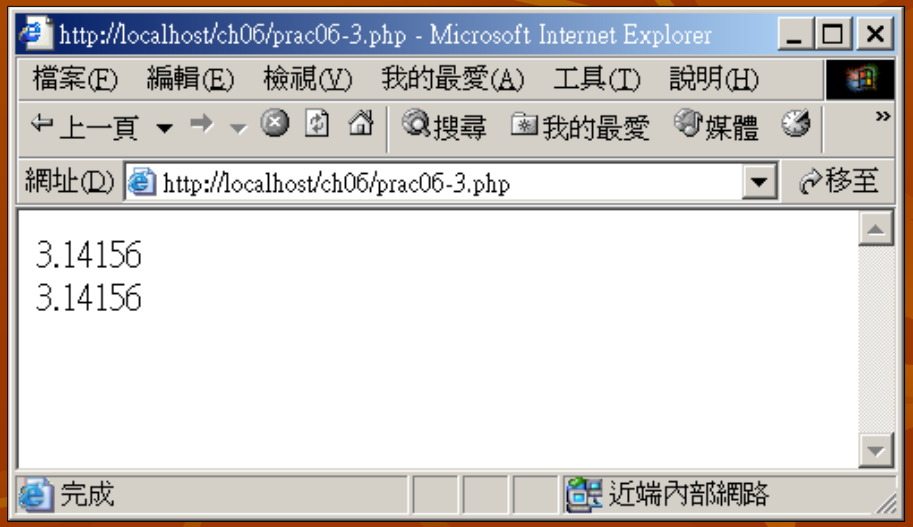
```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class MyClass
05:      {
06:        function Display()
07:        {
08:          echo '不用建立物件就可以呼叫方法喔！';
09:        }
10:      }
11:     MyClass::Display();
12:    ?>
13:  </BODY>
14:</HTML>
```



6-2-2　常數

\ch06\prac06-3.php

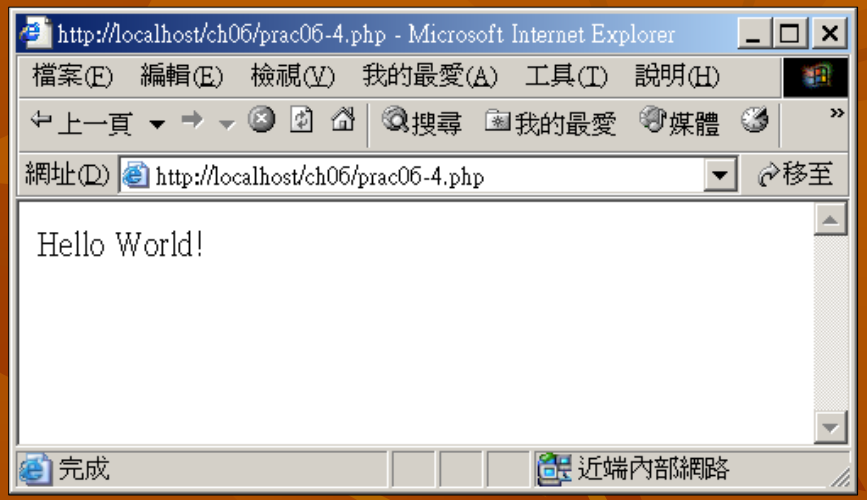
```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class MyClass
05:      {
06:        const PI = 3.14156;
07:        function Display()
08:        {
09:          echo MyClass::PI;
10:        }
11:      }
12:      echo MyClass::PI.'<BR>';
13:      MyClass::Display();
14:    ?>
15:  </BODY>
16:</HTML>
```



6-2-3 建構函式

\ch06\prac06-4.php

```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class MyClass
05:      {
06:        var $Message;
07:        function __construct($Msg)
08:        {
09:          $this->Message = $Msg;

10:        }
11:        function Display()
12:        {
13:          echo $this->Message;
14:        }
15:      }
16:      $MyObject = new MyClass('Hello World!');
17:      $MyObject->Display();
18:    ?>
19:  </BODY>
20:</HTML>
```



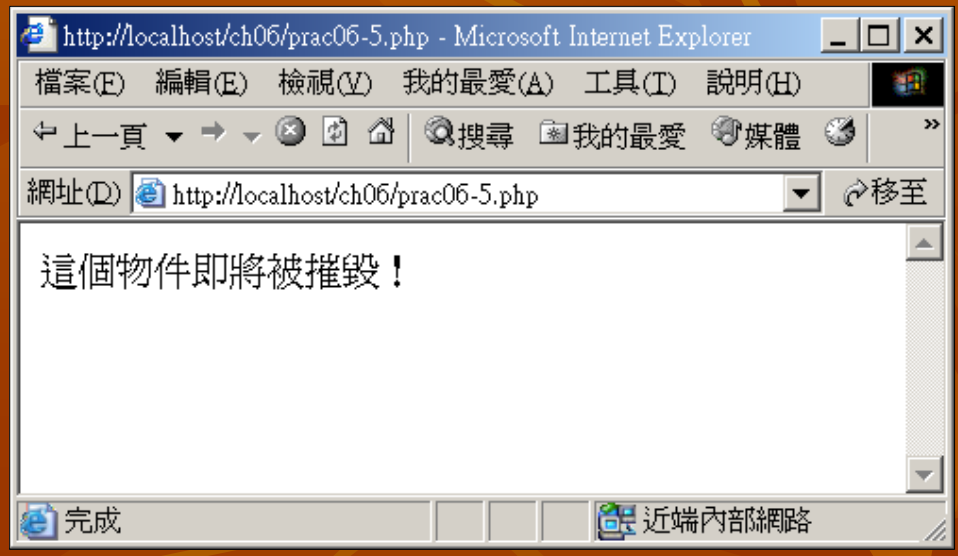
6-2-4　解構函式

\ch06\prac06-5.php (下頁續)

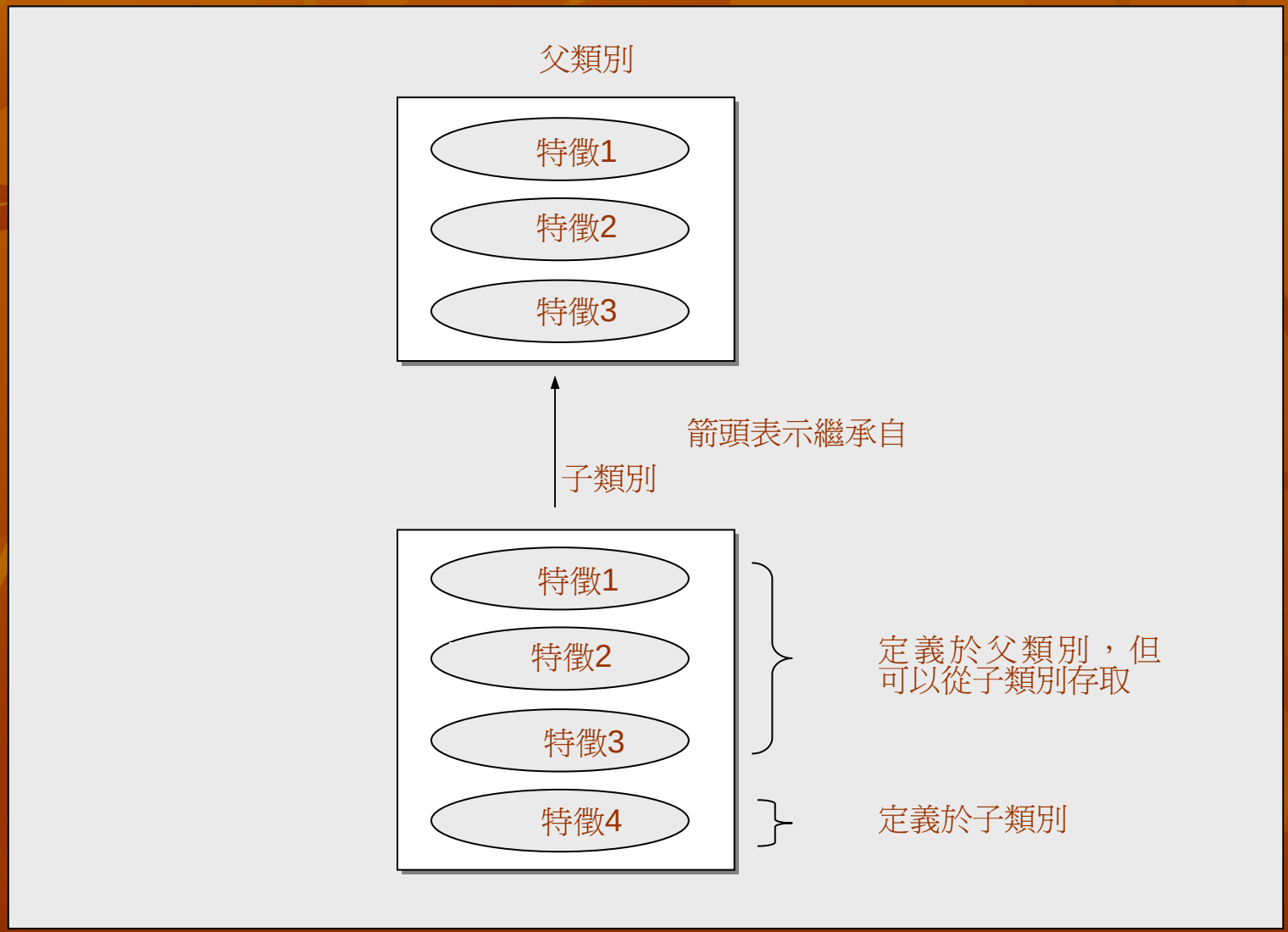
```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class MyClass
05:      {
06:        var $Message;
07:        function __construct($Msg)
08:        {
09:          $this->Message = $Msg;
10:        }
11:        function Display()
12:        {
13:          echo $this->Message;
14:        }
```

\ch06\prac06-5.php (接上頁)

```
14:       function __destruct()
15:       {
16:         $this->Message = NULL;
17:         echo '這個物件即將被摧毀！';
18:       }
19:     }
20:     $MyObject = new MyClass('Hello World!');
21:     $MyObject = NULL;
22:   ?>
23: </BODY>
24:</HTML>
```



6-3 繼承


父類別
特徵1
特徵2
特徵3
箭頭表示繼承自
子類別
特徵1
特徵2
特徵3
特徵4
定義於父類別，但
可以從子類別存取
定義於子類別

6-3-1　定義子類別

class *childclass_name* **extends** *parentclass_name*

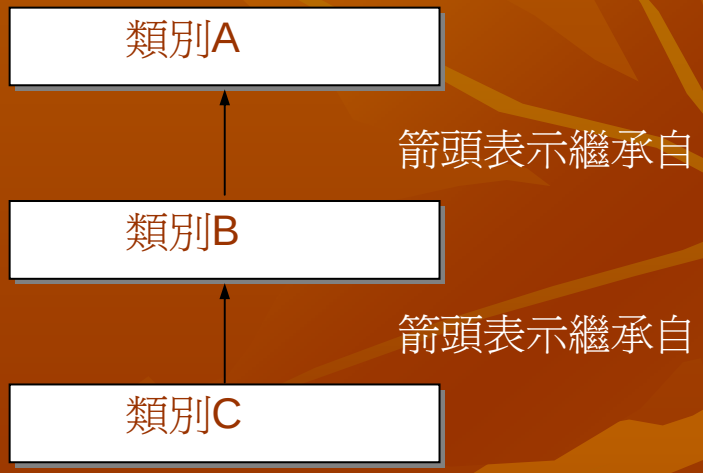
{

　[…]

}

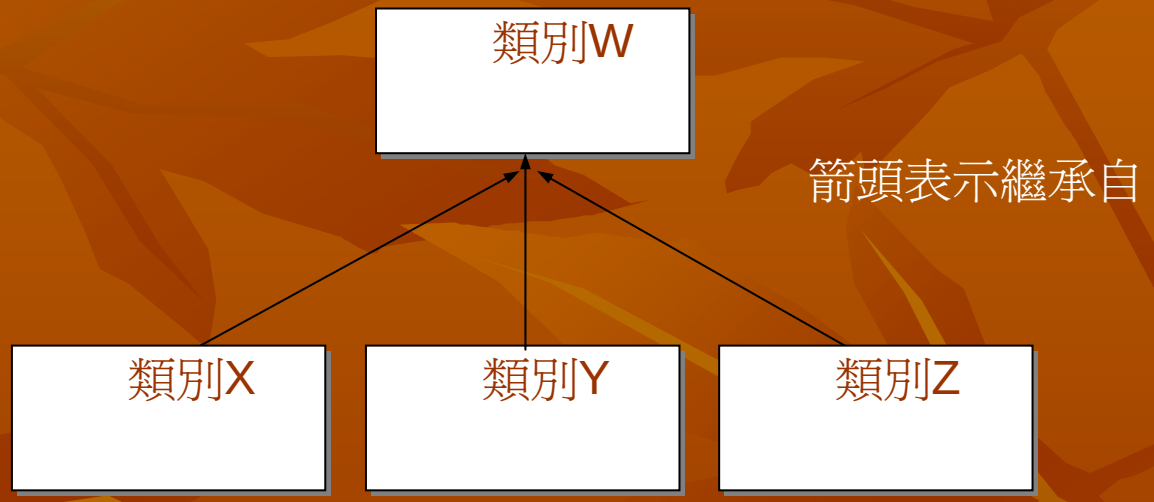
\ch06\prac06-7.php

```
<HTML>
 <BODY>
  <?php
   class A
   {
    //...
   }
   class B extends A
   {
    //...
   }
   class C extends B
   {
    //...
   }
  ?>
 </BODY>
</HTML>
```



ch06\prac06-8.php

```
<HTML>
 <BODY>
  <?php
    class W
    {
      //...
    }
    class X extends W
    {
      //...
    }
    class Y extends W
    {
      //...
    }
    class Z extends W
    {
      //...
    }
  ?>
 </BODY>
</HTML>
```



箭頭表示繼承自

6-3-2 設定成員的存取範圍

- public
- private
- protected

\ch06\prac06-9.php

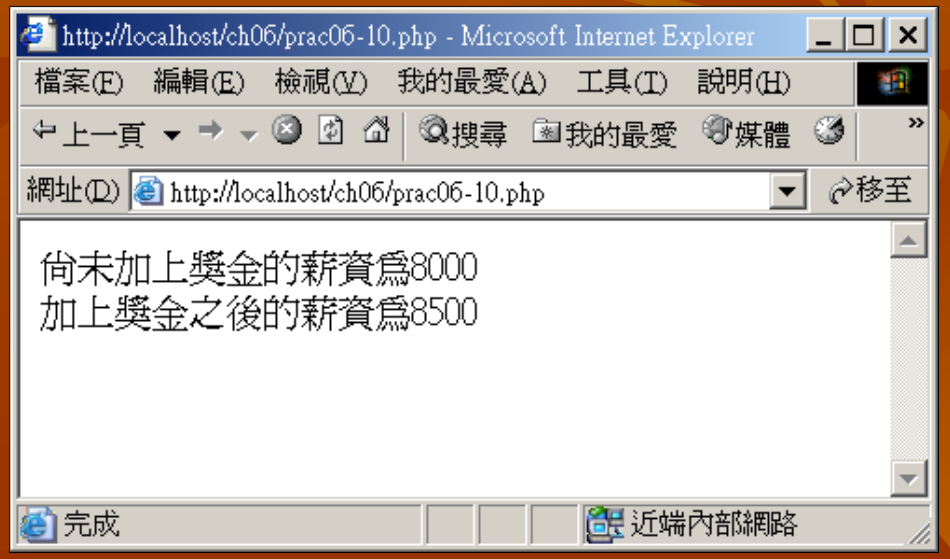
```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class ParentClass                          //定義父類別
05:      {
06:        public $Field1;                          //定義public屬性 (能夠被繼承)
07:        private $Field2;                         //定義private屬性 (不能被繼承)
08:        protected $Field3;                       //定義protected屬性 (能夠被繼承)
09:        public function Method1(){}              //定義public方法 (能夠被繼承)
10:        private function Method2(){}             //定義private方法 (不能被繼承)
11:        protected function Method3(){}           //定義protected方法 (能夠被繼承)
12:      }
13:
14:      class ChildClass extends ParentClass       //定義子類別
15:      {
16:        public $Field4;                          //定義public屬性 (能夠被繼承)
17:        private $Field5;                         //定義private屬性 (不能被繼承)
18:        protected $Field6;                       //定義protected屬性 (能夠被繼承)
19:        public function Method4(){}              //定義public方法 (能夠被繼承)
20:        private function Method5(){}             //定義private方法 (不能被繼承)
21:        protected function Method6(){}           //定義protected方法 (能夠被繼承)
22:      }
23:    ?>
24:  </BODY>
25:</HTML>
```

6-3-3覆蓋繼承自父類別的方法

\ch06\prac06-10.php (下頁續)

```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class Payroll                                          //定義父類別
05:      {
06:        public $EmpName;                                     //定義屬性 (能夠被繼承)
07:        public function Payment($Hours, $PayRate)  //定義方法 (能夠被繼承)
08:        {
09:          return $Hours * $PayRate;
10:        }
11:      }
```

```
12:     class BonusPayroll extends Payroll                  //定義子類別
13:     {
14:      public function Payment($Hours, $PayRate, $Bonus)//覆蓋父類別的方法
15:       {
16:         return $Hours * $PayRate + $Bonus;
17:       }
18:     }
19:     echo '尚未加上獎金的薪資爲'.Payroll::Payment(100, 80).'<BR>';
20:     echo '加上獎金之後的薪資爲'.BonusPayroll::Payment(100, 80, 500).'<BR>';
21:   ?>
22: </BODY>
23:</HTML>
```



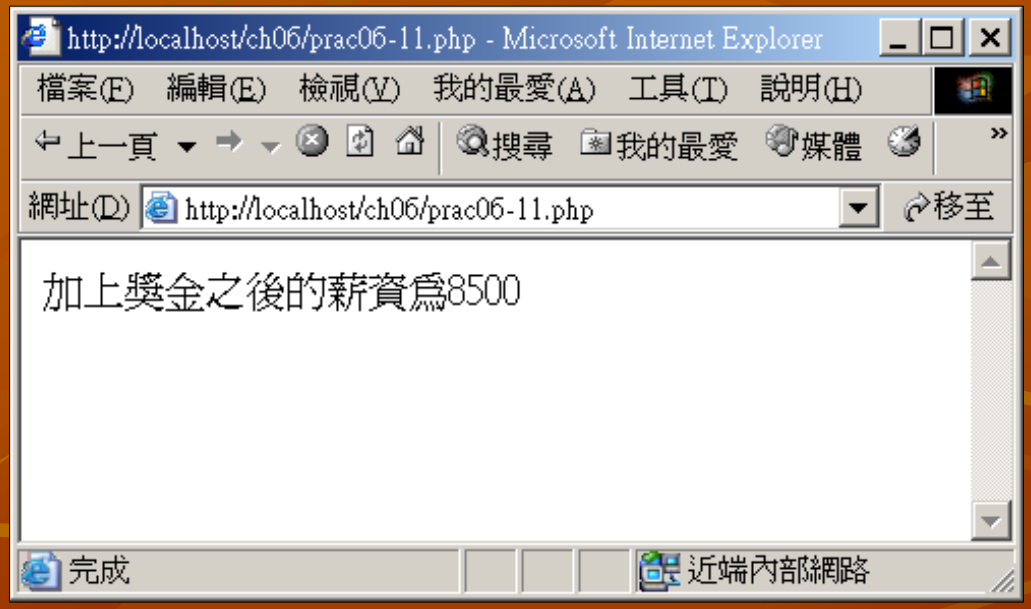
6-3-4　呼叫父類別內被覆蓋的方法

```
public function Payment($Hours, $PayRate, $Bonus)
{
  return $Hours * $PayRate + $Bonus;
}

public function Payment($Hours, $PayRate, $Bonus)
{
  return parent::Payment($Hours, $PayRate) + $Bonus;
}

public function Payment($Hours, $PayRate, $Bonus)
{
  return Payroll::Payment($Hours, $PayRate) + $Bonus;
}
```

6-3-5　抽象方法

\ch06\prac06-11.php

```
01:<HTML>
02:  <BODY>
03:    <?php
04:      abstract class Payroll
05:      {
06:        public $EmpName;
07:        abstract public function Payment($Hours, $PayRate, $Bonus);
08:      }
09:      class BonusPayroll extends Payroll
10:      {
11:        public function Payment($Hours, $PayRate, $Bonus)
12:        {
13:          return $Hours * $PayRate + $Bonus;
14:        }
15:      }
16::      echo '加上獎金之後的薪資爲'.BonusPayroll::Payment(100, 80, 500).'<BR>';
17:    ?>
18:  </BODY>
19:</HTML>
```

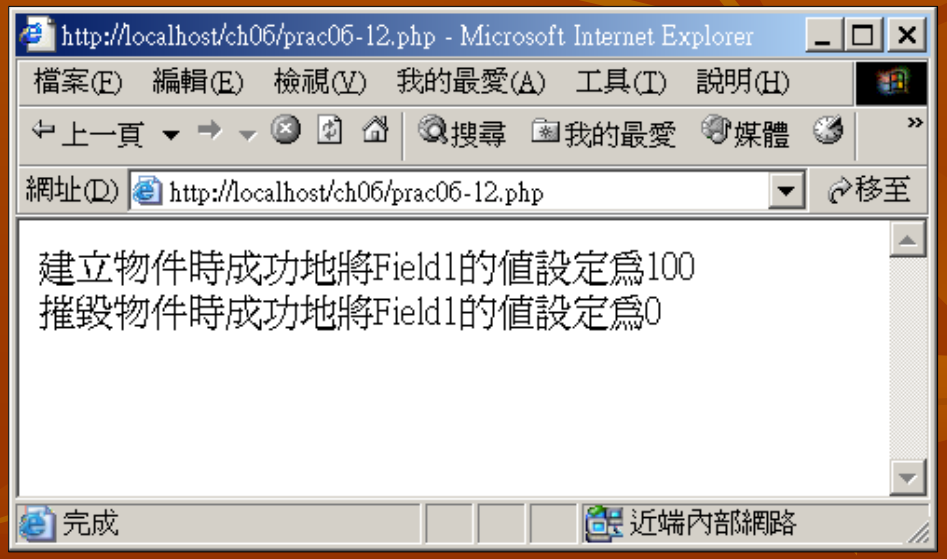

http://localhost/ch06/prac06-11.php - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛 媒體
網址(D) http://localhost/ch06/prac06-11.php
移至
加上獎金之後的薪資爲8500
完成
近端內部網路

6-3-6　子類別的建構函式與解構函式

\ch06\prac06-12.php (下頁續)

```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class ParentClass                          //定義父類別
05:      {
06:        protected $Field1;                       //定義父類別的屬性
07:        function __construct($Value)            //定義父類別的建構函式
08:        {
09:          $this->Field1 = $Value;
10:          echo '建立物件時成功地將Field1的值設定爲'.$this->Field1.'<BR>';
11:        }
```

```
12:       function __destruct()                    //定義父類別的解構函式
13:       {
14:         $this->Field1 = 0;
15:         echo '摧毀物件時成功地將Field1的值設定爲'.$this->Field1.'<BR>';
16:       }
17:     }
18:     class ChildClass extends ParentClass      //定義子類別
19:     {
20:       protected $Field2;                       //定義子類別的屬性
21:     }
22:     $MyObject = new ChildClass(100);          //會自動執行父類別的建構函式
23:     $MyObject = NULL;                         //會自動執行父類別的解構函式
24:    ?>
25:  </BODY>
26:</HTML>
```

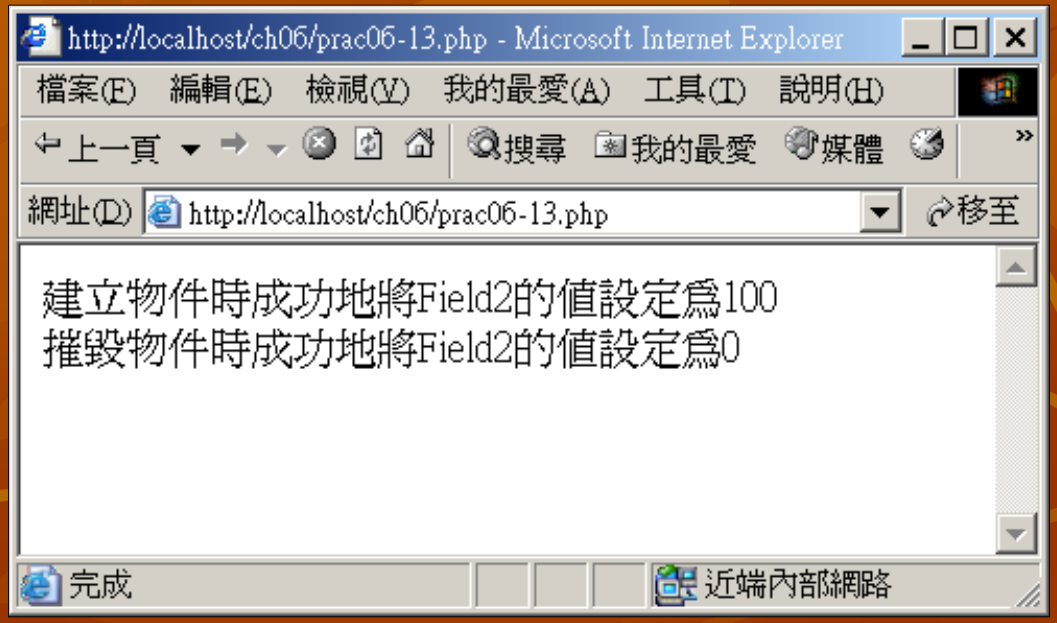


\ch06\prac06-13.php (下頁續)

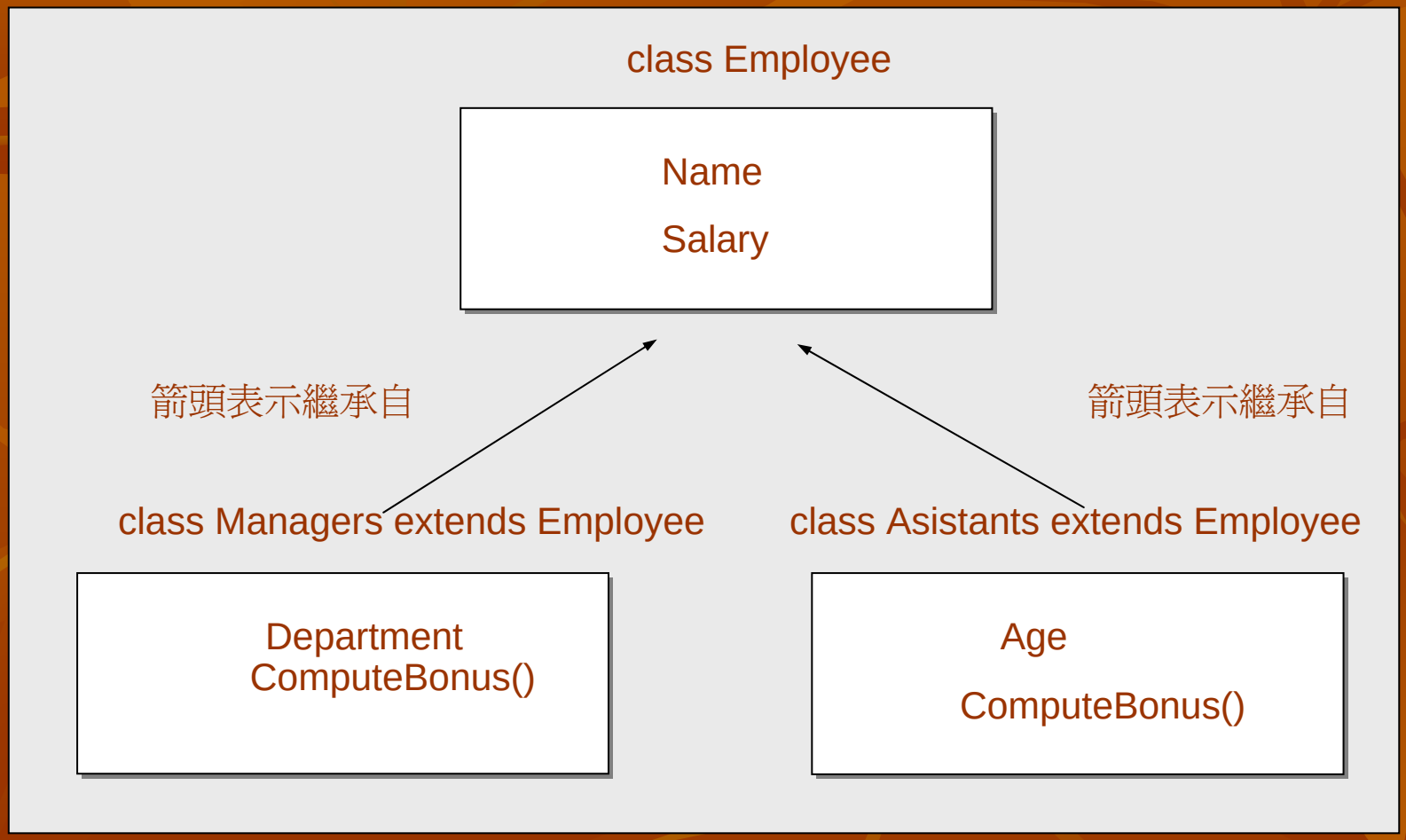
```
01:<HTML>
02:  <BODY>
03:    <?php
04:      class ParentClass                          //定義父類別
05:      {
06:        protected $Field1;                       //定義父類別的屬性
07:        function __construct($Value)            //定義父類別的建構函式
08:        {
09:          $this->Field1 = $Value;
10:          echo '建立物件時成功地將Field1的值設定爲'.$this->Field1.'<BR>';
11:        }
12:        function __destruct()                   //定義父類別的解構函式
13:        {
14:          $this->Field1 = 0;
15:          echo '摧毀物件時成功地將Field1的值設定爲'.$this->Field1.'<BR>';
16:        }
17:      }
```

\ch06\prac06-13.php (接上頁)

```
18:     class ChildClass extends ParentClass        //定義子類別
19:     {
20:       protected $Field2;                        //定義子類別的屬性
21:       function __construct($Value)             //定義子類別的建構函式
22:       {
23:         $this->Field2 = $Value;
24:         echo '建立物件時成功地將Field2的值設定爲'.$this->Field2.'<BR>';
25:       }
26:       function __destruct()                    //定義子類別的解構函式
27:       {
28:         $this->Field2 = 0;
29:         echo '摧毀物件時成功地將Field2的值設定爲'.$this->Field2.'<BR>';
30:       }
31:     }
32:     $MyObject = new ChildClass(100);          //會自動執行子類別的建構函式
33:     $MyObject = NULL;                         //會自動執行子類別的建構函式
34:   ?>
35: </BODY>
36:</HTML>
```


http://localhost/ch06/prac06-13.php - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛 媒體
網址(D) http://localhost/ch06/prac06-13.php
移至
建立物件時成功地將Field2的值設定爲100
摧毀物件時成功地將Field2的值設定爲0
完成
近端內部網路

6-3-7　類別階層


class Employee
Name
Salary
箭頭表示繼承自
箭頭表示繼承自
class Managers extends Employee
class Asistants extends Employee
Department
ComputeBonus()
Age
ComputeBonus()

6-3-8　使用繼承的時機

- 子類別應該屬於父類別的一種，而不只是和父類別有關聯
- 繼承可以提高程式的重複使用性
- 繼承適用於類別階層少的情況
- 繼承可以用來實作多型

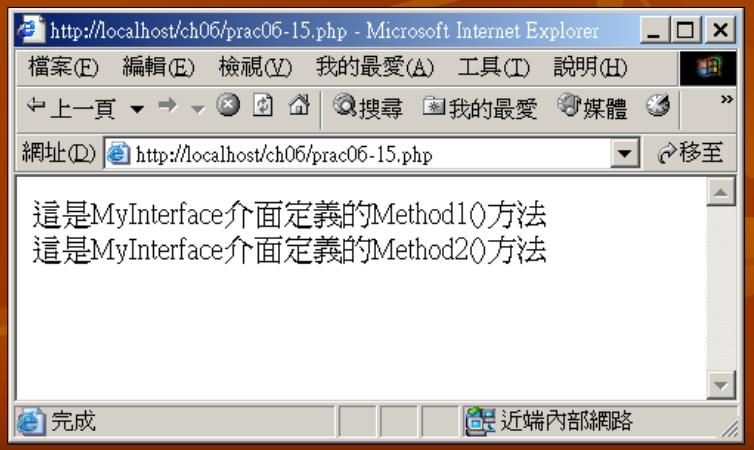
6-4　介面

interface *interface_name*
{
　[定義成員但不提供實作方式]
}

class *class_name* **implements** *interface_name*
{
　[提供成員的實作方式]
}

\ch06\prac06-15.php

```
01:<HTML>
02:  <BODY>
03:    <?php
04:     interface MyInterface                    //定義名稱爲MyInterface的介面
05:     {
06:       function Method1();                    //定義方法但不提供實作方式
07:       function Method2();                    //定義方法但不提供實作方式
08:     }
09:
10:     class MyClass implements MyInterface  //此類別用來實作MyInterface介面的成員
11:     {
12:       function Method1()                     //提供Method1()方法的實作方式
13:       {
14:         echo '這是MyInterface介面定義的Method1()方法<BR>';
15:       }
16:       function Method2()                     //提供Method2()方法的實作方式
17:       {
18:         echo '這是MyInterface介面定義的Method2()方法<BR>';
19:       }
20:     }
21:
22:     MyClass::Method1();
23:     MyClass::Method2();
24:    ?>
25:  </BODY>
26:</HTML>
```



6-5　多型

6-5-1　使用繼承實作多型

\ch06\prac06-16.php (下頁續)

```
<HTML>
 <BODY>
  <?php
   abstract class Transport
   {
    public abstract function Launch();
    public abstract function Stop();
   }
   class Bicycle extends Transport        //定義繼承自交通工具父類別的腳踏車子類別
   {
    public function Launch()              //覆蓋繼承自交通工具父類別的啓動方法
    {
     //寫上啓動腳踏車的程式碼
    }
```

\ch06\prac06-16.php (接上頁)

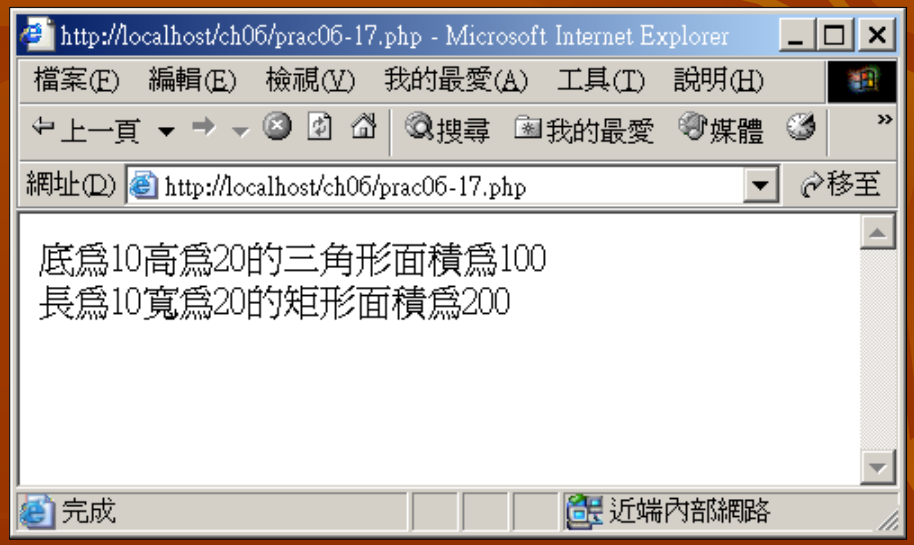
```
    public function Stop()                    //覆蓋繼承自交通工具父類別的停止方法
    {    //寫上停止腳踏車的程式碼 }
  }
  class Motorcycle extends Transport          //定義繼承自交通工具父類別的摩托車子類別
  {
    public function Launch()                  //覆蓋繼承自交通工具父類別的啓動方法
    {      //寫上啓動摩托車的程式碼   }
    public function Stop()                    //覆蓋繼承自交通工具父類別的停止方法
    {      //寫上停止摩托車的程式碼   }
  }

  class Car extends Transport                 //定義繼承自交通工具父類別的汽車類別
  {
    public function Launch()                  //覆蓋繼承自交通工具父類別的啓動方法
    {      //寫上啓動汽車的程式碼     }
    public function Stop()                    //覆蓋繼承自交通工具父類別的停止方法
    {     //寫上停止汽車的程式碼      }
  }
 ?>
</BODY>
</HTML>
```

6-5-2　使用介面實作多型

\ch06\prac06-17.php (下頁續)

```
01:<HTML>
02:  <BODY>
03:    <?php
04:    interface Shape
05:    {
06:      public function CalculateArea($X, $Y);
07:    }
08:
09:    class Triangle extends Shape                  //此三角形類別用來實作Shape介面
10:    {
11:      public function CalculateArea($X, $Y)       //提供CalculateArea()方法的實作方式
12:      {
13:         return ($X * $Y) / 2;                    //計算三角形面積 (底乘以高除以2)
14:      }
15:    }
16:
```

```
17:    class Rectangle extends Shape                //此矩形類別用來實作Shape介面
18:    {
19:      public function CalculateArea($X, $Y)      //提供CalculateArea()方法的實作方式
20:      {
21:        return ($X * $Y);                        //計算矩形的面積 (長乘以寬)
22:      }
23:    }
24:    $Obj1 = new Triangle()                       //建立三角形類別的物件
25:    $Obj2 = new Rectangle();                     //建立矩形類別的物件
26:    echo '底爲10高爲20的三角形面積爲'.$Obj1->CalculateArea(10, 20).'<BR>';
27:    echo '長爲10寬爲20的矩形面積爲'.$Obj2->CalculateArea(10, 20).'<BR>';
28:  ?>
29: </BODY>
30:</HTML>
```



6-6　類別相關函式

- is_a(*obj*, *class_name*)
- is_subclass_of(*obj*, *class_name*)
- class_exists(*class_name*)
- method_exists(*obj*, *method_name*)
- get_class(*obj*)
- get_class_methods(*class_name*)
- get_class_vars(*class_name*)
- get_declared_classes()
- get_declared_interfaces()
- get_object_vars(*obj*)
- get_parent_class(*arg*)
- call_user_method(*method_name*, *obj* [, *arg1* [, a*rg2* [,…]]])
- call_user_method_array(*method_name*, *obj* [, array *arg*])