```
        $this->_formBuilt = true;
        $this->addElement('header', null, 'Wizard page 3 of 3');
        $this->addElement('textarea','itxaTest' ,
        'Parting words:', array('rows' => 5, 'cols' => 40));
        $prevnext[]=&$this->createElement('submit' ,
                $this->getButtonName('back'), '<< Back');
        $prevnext[]=&$this->createElement('submit' ,$this->
                getButtonName('next'), 'Finish');
        $this->addGroup($prevnext, null, '', ' ', false);

$this->addRule('itxaTest' ,'Saysomething!' ,'required');

        $this->setDefaultAction('next');
    }
}

//改寫 HTML_QuickForm_Action 的 perform 方法
class ActionProcess extends HTML_QuickForm_Action
{
    function perform(&$page, $actionName)
    {
       //顯示出" Submit successful!" 字樣，並將輸入值用
      //exportValues 方法輸出，並加以用陣列的方式顯示
      //出來 (var_dump 方法)。
        echo "Submit successful!<br>\n<pre>\n";
        var_dump($page->controller->exportValues());
        echo "\n</pre>\n";
    }
}

//建立一個 controller 物件，並命名為 Wizard
```

```
$wizard =& new HTML_QuickForm_Controller('Wizard');

//將表單第一頁物件化，並命名為 page1，且放入 Wizard 中
$wizard->addPage(new PageFirst('page1'));
$wizard->addPage(new PageSecond('page2'));
$wizard->addPage(new PageThird('page3'));

//把所有 submit 按鈕的可能產生的 handler 加入到 Wizard 中
$wizard->addAction('display'，new
HTML_QuickForm_Action_Display());
$wizard->addAction('next'， new HTML_QuickForm_Action_Next());
$wizard->addAction('back'， new HTML_QuickForm_Action_Back());
$wizard->addAction('jump'， new HTML_QuickForm_Action_Jump());

//將 action 的處理機制加入到 Wizard 中
$wizard->addAction('process'， new ActionProcess());

$wizard->run();
?>
```

可以輕輕鬆鬆，簡簡單單的產生多頁的表單，又具有驗證的功能，套用 CSS 的功能，節省了許多撰寫程式碼的時間，功能又強大，這就是使用 PEAR 的魅力。

5-3 DB 套件

PHP 函式與資料庫的結合，是動態網頁程式相當重要的一項功能。但資料庫的產品非常多，例如:mysql， PostgreSQL，Oracle…等，對 PHP 而言，不同的資料庫的連結，需撰寫不同的程式碼，這對程式設計者是一件十分困擾的事，假如程式設計者要更換資料庫的類型，必需將所有與資料庫相關的 PHP 程式碼全部修改。

幸好 PEAR 的 DB 套件導入了資料庫抽象層(database abstraction layer)的概念，所謂抽象層，就是感覺上 PHP 程式並不直接與資料庫連結，

而是透過一個中介的介面。此介面可讓不同的資料庫用同一種方法存取，日後若程式設計者想更換資料庫的種類時，只需更改少許的程式碼即可。此舉將可減少程式設計者的許多負擔，也簡化撰寫程式碼的工作。

底下利用 Melonfire (2002)所撰寫的範例程式來加以說明如何使用資料庫抽象層。請注意，若以下範例以「//」作為一行的開頭，代表該行為「註解」，而研究者就利用註解來說明程式碼的意義。底下以範例說明:

5-3.1 介紹資料庫抽象層:首先，研究者用兩個資料表做為被存取的資料。第一個名為 cds，內容為 CD 唱片的名稱，以及該 CD 的編號與作者。

```
mysql> SELECT * FROM cds;
+----+-------------------+---------------+
| id | title             | artist        |
+----+-------------------+---------------+
| 16 | On Every Street   | Dire Straits  |
| 15 | Fever             | Kylie Minogue |
| 17 | Hell Freezes Over | The Eagles    |
+----+-------------------+---------------+
3 rows in set (0.06 sec)
```

第二個資料表名為 tracks，內容為 CD 唱片的歌曲名稱，與該歌曲屬於哪一片 CD，可將 cd 欄位的號碼比對上方資料表的 id 欄位而得知。

```
mysql> SELECT * FROM tracks;
+----+---------------------- +------+
| cd | track                  | indx |
+----+---------------------- +------+
| 17 | I Can't Tell You Why   |    9 |
| 15 | More More More         |    1 |
| 17 | Hotel California       |    6 |
| 15 | Love At First Sight    |    2 |
| 16 | Sultans Of Swing       |    1 |
| 16 | Lady Writer            |    2 |
| 16 | Romeo And Juliet       |    3 |
+----+-----------------------+------+
```

7 rows in set (0.00 sec)

將資料分開為兩個資料表，是為了避免不需要的資料被重覆抓取，這種作法有一個名詞稱為「正常化(normalization)」。

接著用 SQL 語法可輕易的將兩個資料表關聯起來，請注意「WHERE cds.id =tracks.cd」這一句可將兩個資料表透過 cds 資料表的 id 欄位與 tracks 資料表的 cd 欄位的互相對映，將兩個資料表關連起來。

mysql> SELECT title， artist， indx， track FROM cds， tracks WHERE cds.id =tracks.cd ORDER BY cd， indx;

```
+-------------+---------------+-----+----------------------+
| title       | artist        |indx  | track                |
+-------------+---------------+---+----------------------+
| Fever        | Kylie Minogue | 1 | More More More          |
| Fever        | Kylie Minogue  | 2 | Love At First Sight    |
| On Every Street | Dire Straits  | 1 | Sultans Of Swing   |
| On Every Street | Dire Straits  | 2 | Lady Writer        |
| On Every Street | Dire Straits  | 3 | Romeo And Juliet   |
| Hell Freezes Over | The Eagles    | 6 | Hotel California   |
| Hell Freezes Over | The Eagles | 9 | I Can't Tell You Why |
+---------------+-------------+------+--------------------+
```

7 rows in set (0.00 sec)

假設我要將我個人的 CD 列出在網頁上，但不用 PEAR 資料庫抽象層的方式，而用一般的 PHP 來連結 mySQL 資料庫的寫法，程式碼如下:

```
<?php
//假設用 mysql 資料庫，並與資料庫建立連結
$connection = mysql_connect("localhost"， "john"， "doe") or die ("Unable toconnect!");

//選取資料庫，假設資料庫名為 db278。
mysql_select_db("db278") or die ("Unable to select database!");

//設定 Query 並執行。
```

```
$query = "SELECT * FROM cds";
$result = mysql_query($query) or die ("Error in query: $query ".
mysql_error());

//遞迴地將資料取出並放入$row 中，並將 TITLE - ARTIS 兩欄位的內容
//列出，這兩個欄位位於第二欄與第三欄，所以用$row[1] - $row[2]
//第一欄為$row[0]
while ($row = mysql_fetch_row($result))
{
    echo "$row[1] - $row[2]\n";
}

//將有被執行的資料，共有幾筆，列出
echo "\n[" . mysql_num_rows($result) . " rows returned]\n";

//關閉與資料庫的連結
mysql_close($connection);
?>
```

執行的結果如下:

```
On Every Street - Dire Straits
Fever - Kylie Minogue
Hell Freezes Over - The Eagles
[3 rows returned]
```

補充說明一下，執行 Query 後抓取資料的方法有下列三種:

用 mysql_fetch_row()可得到用數字當 index 的陣列，上例用這一種。

用 mysql_fetch_assoc()可得到用 field 檔名當 index 的陣列

用 mysql_fetch_object()可得到用 field 檔名當 index 的物件

若程式設計者要改用 PostgreSQL 資料庫，以上程式碼需做非常大的更動，這將照成程式開發人員的困擾，所以研究者接下來改用 PEAR 的 DB 套

件來撰寫程式碼。

用 PEAR 的 Db 套件中的函式庫來連結 mySQL 資料庫的程式碼如下：

```
<?php
//引入 PEAR 的 DB.php 檔
include("DB.php");

//連結到資料庫
$dbh = DB::connect("mysql://john:doe@localhost/db287");

//執行 Query
$query = "SELECT * FROM cds";
$result = $dbh->query($query);

/遞迴地將資料取出並放入$row 中，並將 TITLE - ARTIS 列出
while($row = $result->fetchRow())
{
    echo "$row[1] - $row[2]\n";
}

//將有被執行的資料共有幾筆列出
echo "\n[" . $result->numRows() . " rows returned]\n";

//關閉與資料庫的連結
$dbh->disconnect();
?>
```

用上述寫法若要改為用 PostgreSQL 資料庫，只要將

```
$dbh = DB::connect("mysql://john:doe@localhost/db287");
```

改為

```
$dbh = DB::connect("pgsql://john:doe@localhost/db287");
```

即可，是不是方便許多。

補充說明:

$result->fetchRow()會將 Query 的結果，以一筆資料錄為單位，放入$row 陣列中，並以數字為 index 如上例。

```
while($row = $result->fetchRow())
{
   echo "$row[1] - $row[2]\n";
}
```

若改用$result->fetchRow(DB_FETCHMODE_ASSOC)會將 Query 的結果，以一筆資料錄為單位，放入$row 陣列中，並以 field 欄位名稱為 index 如下例：

```
while($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
{
   echo $row['title'] . " - " . $row['artist'] . "\n";
}
```

若改用$result->fetchRow(DB_FETCHMODE_OBJECT)會將 Query 的結果，以一筆資料錄為單位，放入$row 物件中，並以 field 欄位名稱為 index。如下例：

```
while($row = $result->fetchRow(DB_FETCHMODE_OBJECT))
{
   echo $row->title . " - " . $row->artist . "\n";
}
```

5-3.2:資料庫抽象層 DB 類別提供的一些方法介紹:

numCols()與 columns()方法，範例程式:

```
<?php
include("DB.php");
$dbh = DB::connect("mysql://john:doe@localhost/db287");
$query = "SELECT * FROM cds";
$result = $dbh->query($query);

//numRows()方法可得到 Query 結果共有多少筆資料錄
```

```
//numCols()方法可得到一筆資料錄有多少欄位
echo "Query returned " . $result->numRows() . " rows of " .
$result->numCols() . " columns each";

//關閉資料庫連線
$dbh->disconnect();
?>
```

LimitQuery()方法，範例程式如下:

```
<?php
include("DB.php");
$start = 2;  //由第三筆資料錄開始(由第 0 筆開始)
$num = 4;  //共抓四筆
$dbh = DB::connect("mysql://john:doe@localhost/db287");
$query = "SELECT track FROM tracks";

//請注意這個取代 query 用法，只抓第三筆起算的前四筆資料錄
$result = $dbh->LimitQuery($query， $start， $num);

echo "[Retrieving rows $start through " . ($start+$num) . " of
resultset]\n";

while($row = $result->fetchRow())
{
   echo "$row[0]\n";
}
$dbh->disconnect();
?>
```

tableInfo()方法，範例程式如下:

```
<?php
```

```
include("DB.php");
$dbh = DB::connect("mysql://john:doe@localhost/db287");
$query = "SELECT title，artist，track FROM cds，tracks WHERE cds.id
=tracks.cd";
$result = $dbh->query($query);

//請注意這個方法的用法，此方法可以取出 Query 結果的資料格式資訊
//並放入一個陣列中，再用 print_r 方法列出。
print_r($result->tableInfo());

//中斷資料庫連線
$dbh->disconnect();
?>
```

執行結果如下:

```
Array
(
    [0] => Array
        (
            [table] => cds    //資料表名
            [name] => title   //欄位名
            [type] => string    //資料型態
            [len] => 255    //可裝的資料長度
            [flags] => not_null    //不可空白
        )

    [1] => Array
        (
            [table] => cds    //資料表 table，名稱為 cds
            [name] => artist    //欄位名稱為 name，值為 artist
            [type] => string
            [len] => 255
```

```
            [flags] => not_null
        )

    [2] => Array
        (
            [table] => tracks //資料表 table，名稱為 tracks
            [name] => track
            [type] => string
            [len] => 255
            [flags] => not_null
        )

)
```

5-3.3:資料庫抽象層的新增與刪除

若要將許多資料一次放入 tracks 資料表中，範例程式如下:

```
<?php
include("DB.php");

//若你要插入許多資料到 tracks 資料表中的 track 欄位
//你可以一次動作就可以了
//先將要插入的值放在一個叫$tracks 的陣列中
$tracks = array("Track A", "Track B", "Track C", "Track D");
$dbh = DB::connect("mysql://john:doe@localhost/db287");

//請注意，用 prepare 來取代 query，用?來取代”值”
$query = $dbh->prepare("INSERT INTO tracks (cd, track) VALUES
(12, ?)");

//執行取多次的 query 如下，將$tracks 的值當作$t 代入$query 中的?
foreach($tracks as $t)
```

```
{
    $result = $dbh->execute($query， $t);
}

//中斷資料庫連線
$dbh->disconnect();
?>
```

也可以用二維陣列依次新增兩個值到兩個欄位中，範例程式如下:

```
<?php
include("DB.php");
$cds = array();

//用二維陣列來一次插入兩個值到兩個欄位中
$cds[] = array("CD A"， "Artist A");
$cds[] = array("CD B"， "Artist B");
$cds[] = array("CD C"， "Artist C");
$dbh = DB::connect("mysql://john:doe@localhost/db287");

//請注意有兩個問號
$query = $dbh->prepare("INSERT INTO cds (title， artist) VALUES
(?， ?)");

//用遞迴的方式執行多次的 query
//將$c(一個子陣列，含兩個值)代入$query 的兩個問號中
foreach($cds as $c)
{
    $result = $dbh->execute($query， $c);
}

//中斷資料庫連線
```

```
$dbh->disconnect();
?>
```

刪除的方法與新增相同，只是將$query 的語法改變即可。

例如:$query=$dbh->prepare(“DELETE FROM cds WHERE title=?”);

5-3.4:資料庫抽象層的偵錯

偵錯是程式設計者非常重要的機制，有良好的偵錯功能，才能增加工作效率與正確性。範例程式如下:

```
<?php
include("DB.php");
$dbh = DB::connect("mysql://john:doe@localhost/db287");

//故意將 SELECT 打錯字為 RE-ELECT
$query = "RE-ELECT * FROM cds";
$result = $dbh->query($query);

//資料 query 的偵錯
if ($dbh->isError($result))
{

//若有錯，則顯示出底下字樣
   echo "Take cover。 Something bad just happened。\n";
   echo "You said: $query\n";
   echo "The database said: " . DB::errorMessage($result) . "\n";
   exit();
}
else
{
   while($row = $result->fetchRow())
   {
```

```
        echo "$row[1] - $row[2]\n";
    }
}

//中斷資料庫連線
$dbh->disconnect();
?>
```

當錯誤發生時，網頁執行時將出現底下錯誤訊息:

```
    Take cover。 Something bad just happened。
    You said: RE-ELECT * FROM cds          //這一行錯了
    The database said: syntax error        //資料庫產生的錯誤訊息
```

由以上所有例子可發現用資料庫抽象層，可讓程式開發者將程式用物件導向的方式撰寫，以及不同的資料庫可用相同的存取方式，讓程式開發者可方便抽換資料庫。

5-4 Benchmark 套件

我們可以利用此套件，計算網頁程式執行的時間，藉以了解伺服器的運作效率，或檢查程式執行速度的瓶頸何在。底下利用官方網站提供的手冊中 Martin Jansen，Alexander Merz(2004)兩位所撰寫的範例程式來加以說明 Benchmark 套件的用法。請注意，若以下範例以「//」作為一行的開頭，代表該行為「註解」，而研究者就利用註解來說明程式碼的意義。

```
<?php
//引入 Benchmark 的 Timer.php 檔
Require_once ("Benchmark/Timer.php");

//建立一個測試用的函式名為 wait
function wait($amount) {
    for ($i=0; $i < $amount; $i++) {
```

```
        for ($i=0; $i < 100; $i++);
        //執行兩個迴圈。
        }
}

//實體化一個 Benchmark_Timer 的物件
$timer = new Benchmark_Timer();

//啟動此物件的 start 方法，開始計算時間
$timer->start();

//執行 wait 函式，讓程式跑 10 次 0~99 的迴圈
wait(10);

//執行此物件的 setMarker 方法，設立一個標記，名為 Marquel
$timer->setMarker('Marquel');

//在網頁上顯示出遊 start 到 Marquel 的時間間隔
echo "從 start 到 Marquel 所花時間: " 。
$timer->timeElapsed('Start', 'Marquel');
wait(50);
$timer->stop();
$timer->display();
?>
```

網頁程式執行的結果如下:

從 start 到 Marquel 所花時間: 0.000422954559326

	time index	ex time	%
Start	1081950720.51193100	-	0.00%
Marquel	1081950720.51235400	0.000422954559326	38.34%
Stop	1081950720.51303400	0.000680208206177	61.66%

total	-	0.0011031627655	100.00%

以上表格會由$time->display()來產生,「而從 start 到 Marquel 所花時間: 」後的秒數是從 start 開始到 Marquel 的時間間隔。

5-5 HTML Templeate Flexy 套件

這是一個網頁的樣板套件，所謂「網頁樣板」是用來讓網頁設計人員與程式設計者共同分工合作的工具，一般網頁設計者較專注於美工，排版方面的技術，對於網頁程式較不專精，相反的，程式設計者對網頁的設計較不拿手，當兩者要一起開發網站時要如何協調呢?「網頁樣板」可以讓網頁版面與程式分開，讓網頁設計者專心設計精美的網頁，而程式設計者只負責開發程式即可。

底下利用官方網站提供的手冊中 Martin Jansen，Alexander Merz(2004)兩位所撰寫的範例程式來加以說明 HTML Templeate Flexy 套件 Benchmark 套件的用法。請注意，若以下範例以「//」作為一行的開頭，代表該行為「註解」，而研究者就利用註解來說明程式碼的意義。

```
<?php
//引入 HTML_Template_Flexy 的檔案
require_once 'HTML/Template/Flexy.php';

//為 HTML_Template_Flexy 類別設定一個靜態屬性名為$option
$options = &PEAR::getStaticProperty('HTML_Template_Flexy',
'options');

//將 example.ini 檔加以解析，並變成$config 陣列，example.ini 放在
//本範例後
$config = parse_ini_file('example.ini',TRUE);

//將$config['HTML_Template_Flexy']的值放入$option 屬性中
$options = $config['HTML_Template_Flexy'];
```

```
//建立一個網頁的控制類別
class controller_test
{
    var $template = "home.html";     //由網頁設計者設計的樣版檔
    var $title;      //設定「標題」資料
    var $numbers = array();
    var $anObject;
    var $element = array();  //表單元件的陣列

    //本類別的建構子
    function controller_test()
    {
     //啟動 start 方法
        $this->start();
        //啟動 output 方法
        $this->output();
    }

     //建立一個函式
    function start()
    {
        //設定網頁標題
        $this->title = "Hello World";
        //儲存一個物件
        $this->anObject = new StdClass;

        //將'Object Member'字串放入上個物件的 member 屬性中
        $this->anObject->member = 'Object Member';

        //建立一個表單物件，此物件是文字輸入元件(input)
     $this->elements['input'] = new HTML_Template_Flexy_Element;
```

```
        //在 input 元件中設並預設值 Hello
        $this->elements['input']->setValue('Hello');

        //將 numbers 資料陣列放入 4 個值，例如：numbers[1]=Number1
        for ($i = 1;$i< 5;$i++) {
            $this->numbers[$i] = "Number $i";
        }
    }

    //output 函式-用來輸出程式的運算結果到樣板檔
    function output() {
        //建立一個 HTML_Template_Flexy 物件
        $output = new HTML_Template_Flexy();

       //編譯樣板
        $output->compile($this->template);

       //將 input 元件放到類別中，透過$output 配合樣板輸出
        $output->outputObject($this，$this->elements);
    }

     //建立一個函式
    function someMethod() {
        echo "<B>Hello From A Method</B>";
    }

}

//物件化 controller_test 類別，執行建構子
new controller_test;
```

```
?>
```

範例檔:example.ini 檔的內容

```
[HTML_Template_Flexy]

//template 檔的放置處，本例即為 home.html 檔所放置的位置
templateDir = /home/me/Projects/myapplication/templates

//編譯過程中，編譯檔放置的位置，該目錄需 web 的使用者有寫入權限
compileDir =
/home/me/Projects/myapplication/compiled_templates

//預設為 0
forceCompile = 0

//若設為 1 會顯示除錯訊息
debug = 0

//用來讀寫 templates 的語言，預設為英語
locale = en

//編譯引擎的方式
compiler = Standard
```

此檔案經過$config=parse_ini_file(example.ini)函式解析後放入$config 陣列，結果會變成如下:

```
$config['HTML_Template_Flexy']= array(
   'templateDir'=>'/home/me/Projects/myapplication/templates',
   'compileDir'=>
'/home/me/Projects/myapplication/compiled_templates',
   'forceCompile'  => 0,
```

```
   'debug'          => 0，
   'locale'         => 'en'，
   'compiler'       => 'Standard'，
  );
```

接著進入樣板檔 home.html，原始檔內容如下:

```
<html>
  <head>

//controller 物件的 title 屬性，會被代入{title}中
    <title>{title}</title></head>
  <body>
  <H1>{title}</H1>
      <form name="form">
    <table>
      <tr><td><?php

/*底下會將$element['input']物件放到<in put name="input">中 t*/
?>
          Input Box: <input name="input">
        </td></tr>
      <?php

/*底下利用 flexy:foreach 來做迴圈，將 numbers 陣列中的 index 放入
number 變數中，且?id=" number" 陣列的值，當作 string 變數作為外顯
文字*/
?>
      <tr flexy:foreach="numbers，number，string"><td>
          <a href="mypage.html?id=%7Bnumber%7D">{string}</a>
        </td></tr>
    </table>
```

```
    <?php /*不用迴圈只顯示出 numbers[2]陣列中的某一個值*/ ?>
    this is number 2 : {numbers[2]}
    <?php /*顯示出某個物件的 member 屬性*/ ?>
  This is a {anObject.member}
    <?php /*也可以用底下的方式呼叫 someMethod 方法*/ ?>
    {someMethod()}
    <?php /*若加上:h 可以將輸出的結果變成有 HTML 的效果*/ ?>
    {someMethod():h}
  </body>
</html>
```

<?php /*上面用「/*…*/」的註解方式取代「//」是為了將來可以用 phpDocumentor 套件將註解轉成文字網頁做成說明手冊*/ ?>

本網頁執行結果如下:

```
    Hello World

    Input Box : [Hello     ]
    Number 1
    Number 2
    Number 3
    Number 4

    this is number 2 : Number 2

    This is a member Variable

    <B>Hello From A Method</B>

    Hello From A Method
```

上述只是 flexy 樣板的簡易使用範例，還有許多用法未能詳敘，請參閱 PEAR 手冊，上例中，網頁設計者只需專心製作樣版檔 home.html 就可以了，其他的就交給程式開發者，這樣就可以達到專業分工的目的了。

5-6 phpDocumenentor 套件

5-6.1 phpDocumentor 套件介紹

phpDocumentor 是一個最先進的自動由程式的註解轉變為產生說明文件的套件，若將 php 程式檔中的註解，用一定的格式撰寫，此套件將可以把註解文字輸出成說明文件，它具有以下特色：(Jesus M. Castagnetto，2003)

- 文件可將常數，函式，靜態函式，類別，方法，靜態變數，類別變數，gloval 變數和外部的指引全部包括進去。
- 可輸出為 html，chm，pdf，xml，docbook 等格式的文件。
- 若有 CSV:dia2code Convertor 應用程式可以用來產生 UML 圖案，用 Harald Fielkers dia2code 應用程式可由圖案產生程式碼。
- 具有錯誤與警告的追蹤系統
- 按照元件的字母順序作文件的排序的索引
- 可將輸出文件編排成樹狀結構。

底下將 php 的註解可以給 phpDocumentor 套件解析用的特殊標籤條列出來：

第一類.Regular tags

標籤名(Tag)	功能介紹(Description)
@abstract	指出一個抽象方法
@access	變數為公用或私有
@author	作者姓名
@copyright	版本名稱與版權宣告日期
@deprecated	描述
@deprec	“反對”的化名
@example	範例檔路徑
@exception	與 java 可共用

@global 在函式中用來寫 global 變數的名稱與描述

@ignore 文件中可被忽略的元件

@param 參數型態(type)的描述

@return 型態的描述

@name 變數的化名

@magic php 文件的協調

@package 包裹名

@see 其他可被文件化的元件，產生一個關聯到文件中

@since 版本或日期

@static 指出靜態方法或變數

@staticvar 寫出函式的靜態變數的描述

@subpackage 子包裹的名稱，在專案中的群組

@throws 文件的協和

@todo 文件的協和

@var 寫出類別變數的 data 的形式

@version 版本

第二類.Inline tags

標籤名(Tag) 功能介紹(Description)

@link URL 超連結

@source 在描述中顯示出函式的來源

@tutorial 說明文件的路徑

@inheritdoc 從父文件區塊拷貝到文件中

@internal 只用在進階開發者的私有訊息

底下利用 Gregory Beaver(2004)所撰寫的範例程式來加以說明 phpDocumentor 套件的用法。請注意，以下範例以「/*…*/」作為「註解」，而註解在底下範例中需遵循一定的格式，才能讓 phpDocumentor 套件來做解析，格式請對照上方的 tag 表。

```
<?php
/**
* A set of assert methods。
*
* @package PHPUnit //套件名稱
* @author  Sebastian Bergmann sb@sebastian-bergmann.de  //作者
* Based upon JUnit， see http://www.junit.org/ for details。
*/
class PHPUnit_Assert {
    /**
    * @var    boolean    //變數型態
    * @access private    //變數範圍
    */
    var $_looselyTyped = false;

    /**
    * Asserts that two variables are equal。
    *
    * @param  mixed    //第一個參數的型態
    * @param  mixed   //第二個參數的型態
    * @param  string    //第三個參數的型態
    * @param  mixed    //第四個參數的型態
    * @access public    //此函示可供其他類別存取
    */
    function assertEquals($expected， $actual， $message = ''，
        $delta = 0) {
        if ((is_array($actual)  && is_array($expected)) ||
            (is_object($actual) && is_object($expected))) {
            if (is_array($actual) && is_array($expected)) {
                ksort($actual);
                ksort($expected);
```

```
            }
?>
```

若要使用 phpDocumentor 套件還需注意 php 程式書寫的格式，需做到以下五點：

1. 類別名稱需含功能的描述，例如:HTML_Table。即可知是一個做表格的類別。
2. Public 方法和變數名稱需遵循一定的規則，如： addRow()，HTML_Table()格式除了解構子，名稱開頭不用底線。
3. Private 方法與變數必須加上底線，例如： _formatBorder()，$this->_numRows。
4. 常數應全部大寫且將 package 的名字放在前頭，例如:MATH_STATS_CUMMULATIVE
5. 若你需要定義 gloval 變數，則前頭需加上 package 名稱且開頭為底線，例如： $GLOBALS['_PEAR_default_error_mode']

5-6.2 如何將 php 程式中的註解轉成說明文件

若有一個範例程式如下，註解皆已按規定寫好，姑且不論內容如何，因為範例的重點在於註解的格式：

```
<?php
* @package SDPHP::MondoRegex    //套件名稱
* @version 0.1.0    //套件版本
* @copyright Jesus M. Castagnetto， 2003    //版權所有者
* @author Jesus M. Castagnetto    //作者
*
* License: PHP License 2.0 or later
* Last Updated: Thursday 2003-06-05 06:15:25 PDT。
*/
class MondoRegex {    /*{{{*/


    /**
```

```
     * List of regular expressions    //說明底下資料變數的意義
     *
     * @var array    //變數型態
     * @access private    //變數存取範圍
     */
var $_reList = array;
    /**
     * List of matches
     *
     * @var array
     * @access private
     */
var $_matchList = array();
    /**
     * Class constructor    //建構子
     *
     * @param optional array $reList associative array of regular
expressions    //說明參數的意義
     * @return MondoRegex object    //傳回值說明
     * @access public    //存取範圍
     */
function MondoRegex($reList = "")/*{{{*/
{
  if (!empty($reList) && is_array($reList)) {
   $this->setRegexList($reList);
        }
}/*}}}*/
    /**
     * Sets the regular expression list
     *
     * @param optional array $reList associative array of regular
```

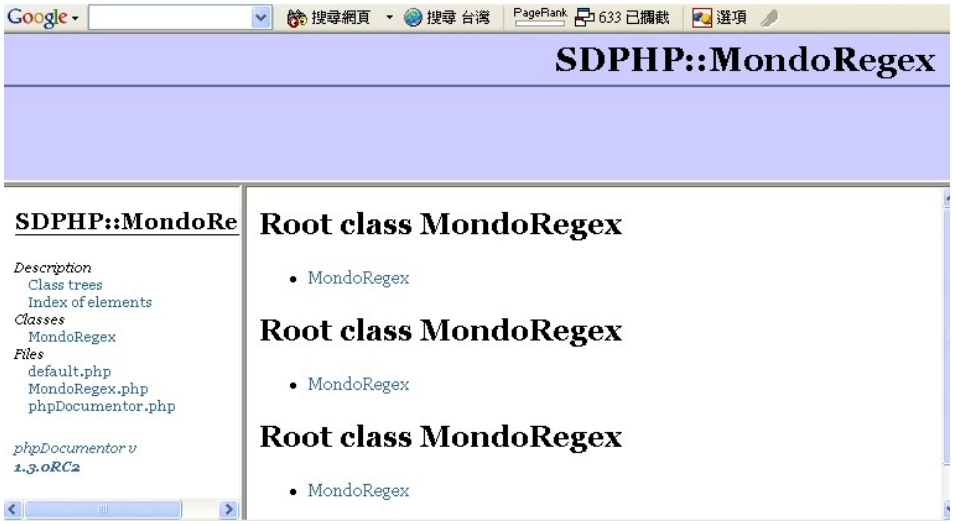
```
expressions
     * @return void
     * @access public
     */
    function setRegexList($reList)
    {
        if (is_array($reList)) {
            $this->_reList = $reList;
        }
    }
…

?>
```

只要 php 文件的格式與 phpDocumenter 的 Tags 正確，則可以在指令列中輸入底下指令：

```
$ phpdoc -d /path/to/my/devel/dir \
-t /path/to/my/devel/dir/apidoc \
--pear -ti 'MondoRegex' \
-o 'HTML:frames:default，XML:DocBook/peardoc2:default'
```

-d 後參數代表 php 檔放置目錄，-t 後參數代表產生的文件檔要放置的目錄，--pear -ti 後參數代表 php 檔中類別的名稱，最後一行設定生成的說明文件要有 HTML 與 XML 兩種格式。

執行後即可在你所指定的 apidoc 目錄下產生兩個說明文件的目錄，一個裝的是 HTML 格式，一個裝的是 XML 格式。上例所產生的 HTML 格式的說明文件如下圖。

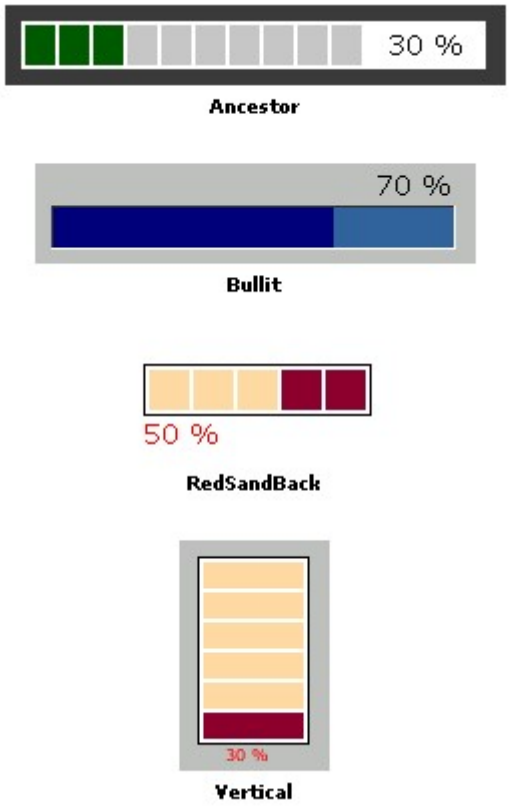


圖三十二 以 phpDocumentor 所自動產生的說明網頁

用這種方法可以節省許多程式開發者重複撰寫註解與說明文件的時間，讓這兩個工作可以合併起來只要做一次即可。對寫說明文件很傷腦筋的開發者而言，這個套件真是一個省時省力的好工具。

5-7 HTML_Progress 套件

此套件可讓使用者在網頁上顯示出 loading bad(如圖)，方便使用者了解一個較長時間的執行狀態，需等待大約多少時間。



圖三十三 各種 HTML_Progress 形式的 progress bar

這些 loading bar 具有以下幾個特徵：

- 可以是垂直的也可以是水平的
- 可以使用外部的 css 或 javascript 碼
- 可以更改每個屬性值，例如顏色，文字。
- 百分比與文字接附著於 loading bar 旁
- 可輕易的整合各種模板(template)
- 可在一個網頁中使用多個 loading bar
- 可以讓 loading bar 增強功能，具有可控制性(如圖三十四)

圖三十四 HTML_Progress 的含控制功能的 progress bar

底下利用 Laurent Laville (2004)所撰寫的範例程式來加以說明 HTML_Progress 套件的用法。請注意，若以下範例以「//」作為一行的開頭，代表該行為「註解」，而研究者就利用註解來說明程式碼的意義。

```
<?php
//引入 progress.php 檔
require_once( “HTML/Progress.php” )

//建立一個 HTML_Progress 物件，並採用預設的設定
$bar= new HTML_Progress();

//也可修改設定，範例如下:
//將 loading bar 用垂直方式顯現:如下
//$bar= new HTML_Progress(HTML_PROGRESS_BAR_VERTICAL);
//將 loading bar 的百分比由 0%到 100%：如下
//$bar= new HTML_Progress(0，100);
//用垂直方式並由 0%~100%：如下
//$bar= new HTML_Progress(HTML_PROGRESS_BAR_VERTICAL，0，100);

//或用物件的方法修改屬性值，如:
//設定最小值為 0
$bar->setMininum(0);

//設定最大值為 60
$bar->setMaxinum(60);

//設定由多少數字開始起跳
$bar->setValue(10);

//設定每此跳動的百分筆數目
```

```
$bar->setIncrement(10);

//設定 bar 的外觀為$ui
$ui=& $bar->getUI();

//設定 Progress 的背景顏色屬性
$ui->setProgressAttributes(array(
     'background-color' => '#e0e0e0' ));

//設定文字的屬性
$ui->setStringAttributes(array(
      'valign' => 'left' //文字靠左對齊
      'color' => 'red'
      'background-color' => 'lightblue' ));

//設定框線的屬性
$ui->setBorderAttributes(array(
      'width' => 1
      'style' => 'solid'
      'color' => 'navy' ));

//設定 cell 的屬性
$ui->setCellAttributes(array(
      'active-color' => '#3874B4'
      'inactive-color' => '#EEEECC' ));

//設定 Cell 的數目
$ui->setCellCount(20);

//設定 bar 的方向改為垂直
$ui->setOrientation(HTML_PROGRESS_BAR_VERTICAL);
```

```
//填充 cell 的方式
$ui->setFillWay( 'reverse' );
?>

//設定 CSS 樣式
<style type=" text/css" >
<?php echo $bar->getStyle(); ?>
</style>

//設定 javascript
<script type=" text/javascript" >
<?php echo $ui->getScript(); ?>
</script>

//顯示出 bar
<?php
ehco $bar->toHTML();
do {

//重新顯示 bar-跳一格
    $bar->display();

//若執行完成停止程式
    if ($bar->getPersentComplete()==1){
        break;
    }

//調整為新的 progress 值
    $bar->incValue();
}while(1);
?>
```

有以上範例可看出，很簡單的就可以在網頁執行期間產生，執行狀態的動態 bar。

Progress 還提供許多進階用法，請參閱 http://pear.laurent-laville.org/HTML_Progress/apidoc/

5-9 Pager 套件

有了與資料庫的連結，有了網頁的樣板，但若資料量相當大的情況下，一頁的網頁裝不下，需要做分頁效果時，該怎麼辦呢?

Pager 套件就是將陣列中的資料作成分頁的類別，將輸入的資料，依據參數做成各種不同的分頁，並可以在特別的範圍內建立連結，也可以客製化(customization)輸出的表現方式。

底下利用官方網站提供的手冊中 Martin Jansen，Alexander Merz(2004)兩位所撰寫的範例程式來加以說明 Pager 套件的用法。請注意，若以下範例以「//」作為一行的開頭，代表該行為「註解」，而研究者就利用註解來說明程式碼的意義。

```
<?
require_once 'Pager/Pager.php';
$params = array(
            'mode'       => 'Jumping',  //翻頁的模式
            'perPage'    => 3,   //每頁 3 筆
            'delta'      => 2,    //前後兩頁做超連結
            'itemData'   => array('a','b','c','d','e','f',
'g','h','i','j','k','l','m','n','o','p','q','r','s',
't','u','v','w','x','y','z')   //資料內容的陣列
            );

//建立一個新的 Pager 物件，以上述陣列為參數
$pager = & new Pager($params);

//用 getPageData 方法抓資料到$data 中
$data  = $pager->getPageData();
```

```
//用 getLinks 方法抓超連結
$links = $pager->getLinks();

//所謂$links 是一個順序加「前一頁」,「後一頁」,「第一頁」,「最後頁」,
//「所有頁」的組合的陣列
//將到其他頁的 links 顯示出來
echo $links['all'];
echo "<br>";

//將目前頁的資料顯示出
echo 'PAGED DATA: ' ; print_r($data);

//各種顯示資料的方法如下:
// 1.顯示出目前的頁碼
echo 'getCurrentPageID(): ';
var_dump($pager->getCurrentPageID());

// 2.顯示出下一頁的頁碼
echo 'getNextPageID()......: ';
var_dump($pager->getNextPageID());

// 3.顯示出前一頁的頁碼
echo 'getPreviousPageID()..: ';
var_dump($pager->getPreviousPageID());

// 4.陣列中資料總筆數
echo 'numItems(): '; var_dump($pager->numItems());

// 5.資料總頁數
echo 'numPages(): '; var_dump($pager->numPages());
```

```
// 6.此頁是第一頁嗎?
echo 'isFirstPage(): '; var_dump($pager->isFirstPage());

// 7.此頁是最後一頁嗎?
echo 'isLastPage(): '; var_dump($pager->isLastPage());

// 8.最後一頁是否剛好三筆資料(設定的「每頁筆數」)
echo 'isLastPageComplete().: ';
var_dump($pager->isLastPageComplete());

// 9.目前頁是兩個頁碼中的哪一頁
echo '$pager->range: '; var_dump($pager->range);
?>
```

本網頁的執行結果如下：

圖三十五 Pager 範例程式產生的結果

用非常簡單的方式就可以達到分頁的效果。而且只要建立兩個 Pager 物件就可以在同一頁中產生兩個分頁效果。

5-10 Cache 套件

若 php 產生的動態網頁，不同的人取得相同的結果，但程式卻必須每次取得都要執行一次，若能將結果快取起來，將可結省網路流量與系統效能。Cache 套件就是提供快取的功能，這個套件包含一般的 cache 類別與特殊的子類別，這些類別可以儲存和管理快取的資料。

底下利用 Sebastian Bergmann (2001)所撰寫的範例程式來加以說明 Cache 套件的用法。請注意，若以下範例以「//」作為一行的開頭，代表該行為「註解」，而研究者就利用註解來說明程式碼的意義。

第一個範例，將網頁的執行結果快取。

```
<?php
//引入 Cache 套件的 Output.php 檔
require_once 'Cache/Output.php';

//建立一個 Cache_Output 物件，將快取的結果存成檔案，並放在與
//被快取的網頁相同的路徑
$cache = new Cache_Output('file', array('cache_dir' => '.') );

//製造出一個獨一無二的快取識別碼給被快取的檔案
//假設利用 URL，POST，cookies 變數來產生此識別碼
$cache_id = $cache->generateID(array('url' => $REQUEST_URI,
'post' => $HTTP_POST_VARS, 'cookies' => $HTTP_COOKIE_VARS) );

//若這個識別碼存在，則存取這個 cache，並顯示出來
if ($content = $cache->start($cache_id)) {
echo $content;

//顯示後停止執行網頁程式
die();
}

// 若找不到此識別碼
```

```
…[網頁的內容]…

// 執行完畢將內容快取起來。
echo $cache->end();
?>
```

第二個範例，將網頁的執行結果快取並加以客製化。

```
<?php
//引入 Cache.php 檔
require_once 'Cache.php';

//建立一個名為 MySQL_Query_Cache 類別，繼承自 Cache 類別
class MySQL_Query_Cache extends Cache {
  var $connection = null;

//Cache 的有效期限為 3600 秒
  var $expires    = 3600;
  var $cursor = 0;
  var $result = array();

//建構子，設定快取結果存成檔案，並設定此快取檔的參數
//包括快取檔與網頁檔同路徑，快取檔檔名前加上 cache_字樣，
//快取檔的有效期限為 3600 秒。
  function MySQL_Query_Cache($container  = 'file',
      $container_options = array('cache_dir'=> '.',
      'filename_prefix' => 'cache_'), $expires = 3600)
  {
    $this->Cache($container, $container_options);
    $this->expires = $expires;
  }
```

```
//解構子
  function _MySQL_Query_Cache() {
//若資料庫為連結狀態，中斷連結
      if (is_resource($this->connection)) {
        mysql_close($this->connection);
      }
      $this->_Cache();
  }
}

//建立一個連結函式
function connect($hostname， $username， $password， $database)
{
$this->connection = mysql_connect($hostname， $username，
$password) or trigger_error('Could not connect to database.'，
E_USER_ERROR);
mysql_select_db($database， $this->connection)   or
trigger_error('Could not select database.'， E_USER_ERROR);
}

//建立一個抓取資料的函式
function fetch_row() {
  if ($this->cursor < sizeof($this->result)) {
    return $this->result[$this->cursor++];
  } else {
    return false;
  }
}

//建立一個抓取資料比數的函式
```

```
function num_rows() {
  return sizeof($this->result);
}

//建立一個 query 函式
function query($query) {
  if (stristr($query, 'SELECT')) {
//若$query 中有 SELECT 這個字在其中，以$query 產生一個快取識別
//碼
    $cache_id = md5($query);

//用 get 方法去抓是否有此識別碼，有代表已快取，無代表尚未快取
//若有則將快取的內容放入$result 中
    $this->result = $this->get($cache_id, 'mysql_query_cache');
    if ($this->result == NULL) {
      // 若沒有抓到任何快取檔，則設定 cursor 為 0
      $this->cursor = 0;
      $this->result = array();
    if (is_resource($this->connection)) {
//連結資料庫，若可連上，則使用 mysql_unbuffered_query 方法
//若無，則用 mysql_query 方法
        if (function_exists('mysql_unbuffered_query')) {
           $result =  mysql_unbuffered_query($query,
$this->connection);
           } else {
           $result = mysql_query($query, $this->connection);
          }

        // 將$result 中的資料錄抓出放入 result[]陣列
        while ($row = mysql_fetch_assoc($result)) {
              $this->result[] = $row;
```

```
        }

        //將資料庫的存取結果由記憶體中釋放出來
        mysql_free_result($result);
        // 將 result 陣列快取起來
        $this->save($cache_id, $this->result, $this->expires,
'mysql_query_cache');
      }
    }
  } else {
    //若$query 中沒有 SELECT 字樣，則單純執行 mysql_query
    return mysql_query($query, $this->connection);
  }
}
?>
```

5-12 Translation 套件

這個類別可讓網頁呈現多種語言，這個多語言的解決方案是將字串儲存在資料庫中，使用資料庫的目的在於抓字串的速度快，可將許多種語言存在資料庫中。只要給予適當的參數給建構子，加上網頁有識別的訊息物件，就會抓取正確的字，底下利用 Wojciech Zieliński(2004)所撰寫的範例程式來加以說明 Translation 套件的用法。請注意，若以下範例以「//」作為一行的開頭，代表該行為「註解」，而研究者就利用註解來說明程式碼的意義。

先建立一個名為 translarion 的資料庫，內含三個資料表，第一個叫 tr_langsavail，資料內容為可用的語言，目前先輸入英語與中文兩種選擇，請注意 lang_id 用 en 代表英語，用 tw 代表繁體中文，如圖：

←T→		lang_id	name	metatags	errortext
編輯	刪除	en	English version		English version is not available yet.
編輯	刪除	tw	繁體中文		繁體中文

圖三十六 tr_langsavail 資料表內容

第二個表格名為 tr_strings_en，內容中 string 欄位為英語網頁中的文字，另外 page_id 為這些文字所在的網頁，範例中有兩頁為 MAIN 與 SECOND，string_id 為網頁文字的 id 代號。

←T→		page_id	string_id	string
編輯	刪除		TITLE	Use-case of Translation class
編輯	刪除		LINK	Go to the &&1&& homepage
編輯	刪除		FIRST	first
編輯	刪除		SECOND	second
編輯	刪除	MAIN	FIRST_STRING	Homepage for user &&1&&
編輯	刪除	SECOND	FIRST_STRING	Welcome on your second homepage !!!
編輯	刪除	SECOND	SECOND_STRING	This phrase will see everyone in his own language.

圖三十七 tr_strings_en 資料表內容

第三個表格名為 tr_strings_tw，內容與上一個表格相似，只是文字為中文。

←T→		page_id	string_id	string
編輯	刪除		TITLE	Translation 翻譯類別
編輯	刪除		LINK	回首頁
編輯	刪除		FIRST	第一頁
編輯	刪除		SECOND	第二頁
編輯	刪除	MAIN	FIRST_STRING	這是首頁- 使用者 &&1&&
編輯	刪除	SECOND	FIRST_STRING	歡迎這是第二頁!!!
編輯	刪除	SECOND	SECOND_STRING	這一段將可看到

圖三十八 tr_strings_tw 資料表內容

接著來看程式碼，第一頁名為 first.php:

```
<?php
//引入 Translation 套件的 translation.class.php 檔
require_once ("Translation/translation.class.php");

//建立一個 Translation 物件，抓取資料庫的 tr_strings_en 或
//tr_strings_tw 中 page_id 為 MAIN 的文字。若$_GET['Lang']沒有設定
//使用那一種語言，則預設為英語。
$tr = new Translation ("MAIN", (isset($_GET['Lang']) ?
$_GET['Lang'] : 'en'),
"mysql://user:password@localhost/translation");

//顯示網頁的表頭訊息
echo "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//EN\"><HTML><HEAD>";
echo "<TITLE>" . $tr->gstr("TITLE") . "</title>";
echo $tr->getMetaTags();
echo "</head><BODY>";

//引入 langs.php 檔，此檔可讓使用者切換語言。
```