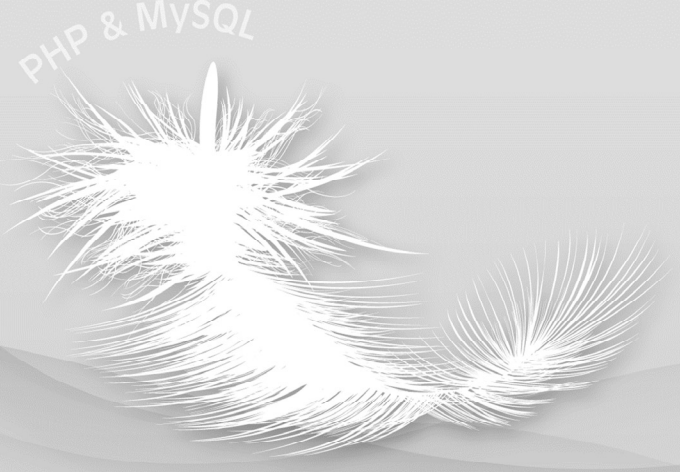
06 CHAPTER

例外與錯誤處理



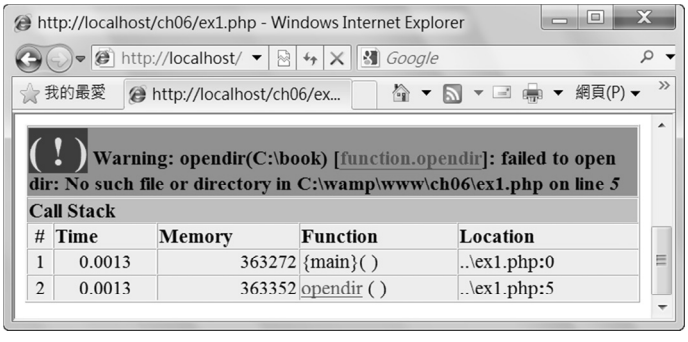
6-1 結構化例外處理

「結構化例外處理」(structured exception handling) 對 Java、C# 等程式語言來說，早已行之數年，其優點是使程式的開發、偵錯與維護變得更有彈性和效率，但對 PHP 來說，還是到了 PHP 5 才開始支援結構化例外處理。

所謂「例外」(exception) 指的並不單純是錯誤 (error)，舉凡程式沒有處理到的情況或錯誤，都可以稱為例外。為了讓您瞭解結構化例外處理的用途，我們來看個例子，其中第 05 行試圖開啟不存在的資料夾，因而得到如下圖的一串錯誤訊息。

\ch06\ex1.php

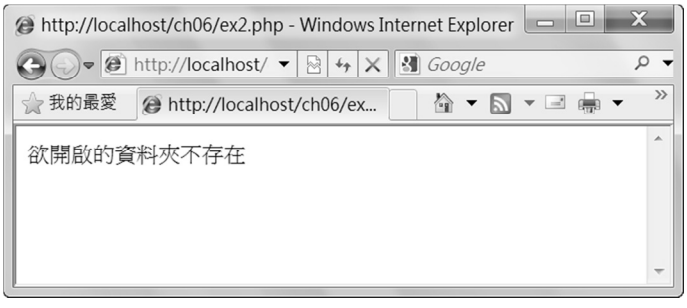
```
01:<html>
02:   <head><meta http-equiv="content-type" content="text/html; charset=utf-8"></head>
03:   <body>
04:      <?php
05:         opendir("C:\\book");          //試圖開啟不存在的資料夾
06:      ?>
07:   </body>
08:</html>
```



然這樣的一串錯誤訊息實在不太友善，為此，我們將程式碼改寫成 <\ch06\ex2.php>，也就是在開啟資料夾之前，先檢查資料夾是否存在 (第 09 行)，若存在，就加以開啟 (第 10 行)，否則顯示「欲開啟的資料夾不存在」(第 12 行)，因而得到了如下圖的結果。

\ch06\ex2.php

```
01:<html>
02:   <head><meta http-equiv="content-type" content="text/html; charset=utf-8"></head>
03:   <body>
04:      <?php
05:         open_folder("C:\\book");                    //試圖開啟不存在的資料夾
06:
07:         function open_folder($folder)
08:         {
09:            if (file_exists($folder))                //檢查資料夾是否存在
10:               opendir($folder);                     //存在的話就加以開啟
11:            else
12:               echo '欲開啟的資料夾不存在';          //不存在的話就顯示錯誤訊息
13:         }
14:      ?>
15:   </body>
16:</html>
```



嗯，這樣的錯誤訊息看起來清楚多了，原本我們可以就此打住，但若我們希望除了錯誤訊息之外，還可以顯示檔案路徑、錯誤代碼、錯誤行數等資訊，那麼 <\ch06\ex2.php> 是沒有辦法達成這個任務的，此時，結構化例外處理就可以派上用場了。

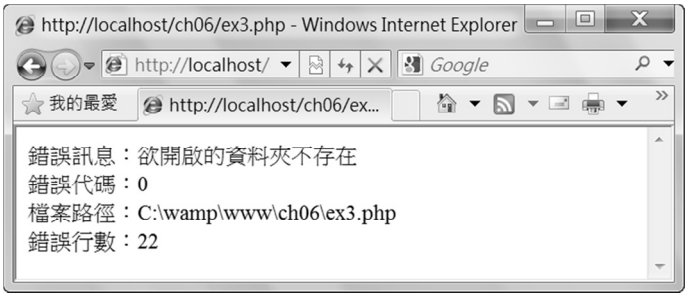
PHP 的結構化例外處理主要包含下列兩個部分：

- Exception 物件：這是例外本身，包含錯誤訊息、錯誤代碼、檔案路徑、錯誤行數等資訊。
- try…catch 語法：用來捕捉代表例外本身的 Exception 物件。

現在，我們就以結構化例外處理的方式將 <\ch06\ex2.php> 改寫成如下。

\ch06\ex3.php

```
01:<html>
02:   <head><meta http-equiv="content-type" content="text/html; charset=utf-8"></head>
03:   <body>
04:      <?php
05:         try
06:         {
07:            open_folder("C:\\book");                          //試圖開啟不存在的資料夾
08:         }
09:         catch(Exception $ex)                                 //捕捉可能產生的例外
10:         {
11:            echo '錯誤訊息：'.$ex->getMessage().'<br>';       //顯示例外的錯誤訊息
12:            echo '錯誤代碼：'.$ex->getCode().'<br>';          //顯示例外的錯誤代碼
13:            echo '檔案路徑：'.$ex->getFile().'<br>';          //顯示例外的檔案路徑
14:            echo '錯誤行數：'.$ex->getLine().'<br>';          //顯示例外的錯誤行數
15:         }
16:
17:         function open_folder($folder)
18:         {
19:            if (file_exists($folder))                         //檢查資料夾是否存在
20:               opendir($folder);                              //存在的話就加以開啟
21:            else
22:               throw new Exception('欲開啟的資料夾不存在');  //不存在的話就丟出例外
23:         }
24:      ?>
25:   </body>
26:</html>
```



- 05 ~ 15：第 05 ~ 08 行的 try 區塊必須放在可能產生例外的程式碼周圍，在此，第 07 行的 open_folder("C:\\book"); 敘述就是可能產生例外的程式碼，而第 09 行的 catch(Exception $ex) 敘述則是用來捕捉可能產生的例外，若沒有捕捉到例外，就跳出第 10 ~ 15 行的 catch 區塊。

  相反的，若有捕捉到例外，就將它存放在物件變數 ex 中，然後將程式的控制權轉移到第 10 ~ 15 行的 catch 區塊，執行第 11 ~ 14 行，透過物件變數 ex 的 getMessage()、getCode()、getFile()、getLine() 等方法，取得例外的錯誤訊息、錯誤代碼、檔案路徑、錯誤行數等資訊並顯示出來。

- 19 ~ 22：第 19 行用來檢查資料夾是否存在，若存在，就執行第 20 行加以開啟，否則執行第 22 行丟出一個例外 (隸屬於 Exception 類別)，並設定其錯誤訊息為「欲開啟的資料夾不存在」。現在，您總明白第 09 行所捕捉到的例外，其實就是第 22 行所丟出的。

注意

- 所有例外均隸屬於 Exception 類別，我們可以使用這個類別的 getMessage()、getCode()、getFile()、getLine() 等方法，取得例外的錯誤訊息、錯誤代碼、檔案路徑、錯誤行數。
- 我們可以使用下列語法丟出例外，其中參數 *error_message* 用來設定例外的錯誤訊息：

```
throw new Exception(error_message);
```

6-2 錯誤處理

PHP 程式所產生的錯誤可以分成下列三種類型：

一、 Notice（注意)：由於這種錯誤不會影響程式的執行，所以會被忽略，不會顯示錯誤訊息，也不會終止程式。舉例來說，假設我們在 PHP 程式中撰寫類似 echo TRUE / FALSE; 的敘述，就會被忽略。

二、 Warning (警告)：這種錯誤會顯示錯誤訊息，但不會終止程式。舉例來說，假設我們在 PHP 程式中進行除法運算，而且除數為 0，就會得到類似 "Warning: Division by zero in C:\wamp\www\ch06\a.php on line 7" 的警告，若這個敘述後面還有其它程式碼，就會繼續執行。

三、 Fatal error (嚴重錯誤)：這種錯誤不僅會顯示錯誤訊息，同時會終止程式，舉例來說，假設我們在 PHP 程式中呼叫沒有定義的函式，就會得到類似 "Fatal error: Call to undefined function open_folder() in C:\wamp\www\ch06\a.php on line 2" 的嚴重錯誤，若這個敘述後面還有其它程式碼，就會終止執行。

在示範如何在 PHP 程式中進行「錯誤處理」(error handling) 之前，我們先來介紹幾個相關函式：

- error_reporting([int *error_type*]])：用來設定 PHP 會報告哪些類型的錯誤，例如：

```
//關閉所有錯誤報告
error_reporting(0);
//報告所有錯誤
error_reporting(E_ALL);
//報告執行期間 (runtime) 的錯誤
error_reporting(E_ERROR | E_WARNING | E_PARSE);
//報告除了 Notice 之外的錯誤
error_reporting(E_ALL ^ E_NOTICE);
```

參數 *error_type* 代表的是錯誤類型 (常數)，如下表。

值	常數	說明
1	E_ERROR	執行期間嚴重錯誤 (runtime fatal errors)，會顯示錯誤訊息並終止程式。
2	E_WARNING	執行期間警告 (runtime warnings)，會顯示錯誤訊息，但不會終止程式。
4	E_PARSE	編譯期間剖析錯誤 (compile-time parse errors)。
8	E_NOTICE	執行期間注意 (runtime notices)，不會顯示錯誤訊息，也不會終止程式。
16	E_CORE_ERROR	PHP 初始啟動時所產生的嚴重錯誤，有點像 E_ERROR，但是由 PHP 的核心所產生。
32	E_CORE_WARNING	PHP 初始啟動時所產生的警告，有點像 E_WARNING，但是由 PHP 的核心所產生。
64	E_COMPILE_ERROR	編譯期間嚴重錯誤 (compile-time fatal errors)，有點像 E_ERROR，但是由 PHP 的 Zend Scripting Engine 所產生。
128	E_COMPILE_WARNING	編譯期間警告 (compile-time warnings)，有點像 E_WARNING，但是由 PHP 的 Zend Scripting Engine 所產生。
256	E_USER_ERROR	使用者所產生的嚴重錯誤 (fatal errors)，有點像 E_ERROR，但是由使用者呼叫 PHP 的內建函式 trigger_error() 所產生。
512	E_USER_WARNING	使用者所產生的警告 (warnings)，有點像 E_WARNING，但是由使用者呼叫 PHP 的內建函式 trigger_error() 所產生。
1024	E_USER_NOTICE	使用者所產生的注意 (notices)，有點像 E_NOTICE，但是由使用者呼叫 PHP 的內建函式 trigger_error() 所產生。
2047	E_ALL	所有錯誤。
2048	E_STRICT	執行期間注意 (runtime notices)。

- set_error_handler(callback *error_handler*)：將錯誤處理程式設定為參數 *error_handler* 所設定的函式。
- restore_error_handler()：將錯誤處理程式恢復為之前的設定。
- trigger_error(string *error_msg* [, int *error_type*])：觸發一個錯誤情況，進而呼叫預設的錯誤處理程式或使用者自訂的錯誤處理程式，參數 *error_msg* 用來設定錯誤訊息，參數 *error_type* 用來設定錯誤類型，意義和 error_reporting() 函式的參數一樣。請注意，PHP 還有一個函式叫做 user_error()，這個函式其實是 trigger_error() 函式的別名，使用方式均相同。
- error_log(string *message* [, int *message_type* [, string *destination* [, string *extra_headers*]]])：將參數 *message* 指定的錯誤訊息傳送至 Web 伺服器的錯誤記錄檔案、TCP 連接埠或指定的檔案，由參數 *message_type* 來決定，它的值如下表。

值	說明
0	將參數 *message* 指定的錯誤訊息傳送至 Web 伺服器的錯誤記錄檔案，這是預設值。
1	將參數 *message* 指定的錯誤訊息傳送至參數 *destination* 指定的電子郵件地址，若要傳送額外的郵件標頭資訊，可以透過參數 *extra_headesr* 來指定，這個參數的意義和 mail() 函式的第三個參數相同，詳細的說明可以參考第 18-2 節。
2	將參數 *message* 指定的錯誤訊息傳送至參數 *destination* 指定的遠端主機 IP 位址 (或許還會包含連接埠)。
3	將參數 *message* 指定的錯誤訊息傳送至參數 *destination* 指定的檔案。

現在，我們就換以錯誤處理的方式來改寫前一節的 <\ch06\ex3.php>，其中第 05 行用來關閉所有錯誤報告，第 06 行用來設定錯誤處理程式是一個名稱為 error_handler 的函式，第 15 行用來觸發一個 E_USER_ERROR 錯誤，而第 18 ~ 24 行用來定義錯誤處理程式 error_handler()。

\ch06\ex4.php

```
01:<html>
02:   <head><meta http-equiv="content-type" content="text/html; charset=utf-8"></head>
03:   <body>
04:      <?php
05:         error_reporting(0);                              //關閉所有錯誤報告
06:         set_error_handler('error_handler');              //設定錯誤處理程式
07:
08:         open_folder("C:\\book");                         //試圖開啟不存在的資料夾
09:
10:         function open_folder($folder)
11:         {
12:            if (file_exists($folder))
13:               opendir($folder);
14:            else
15:               trigger_error('欲開啟的資料夾不存在', E_USER_ERROR);  //觸發一個錯誤
16:         }
17:
18:         function error_handler($errno, $errmsg, $filename, $linenum)  //定義錯誤處理程式
19:         {
20:            echo '錯誤代碼：'.$errno.'<br>';              //參數 $errno 代表錯誤代碼
21:            echo '錯誤訊息：'.$errmsg.'<br>';             //參數 $errmsg 代表錯誤訊息
22:            echo '檔案路徑：'.$filename.'<br>';           //參數 $filename 代表檔案路徑
23:            echo '錯誤行數：'.$linenum.'<br>';            //參數 $linenum 代表錯誤行數
24:         }
25:      ?>
26:   </body>
27:</html>
```