

12

CHAPTER

存取資料庫

12-1 PHP 與資料庫

12-2 建立與關閉資料連接

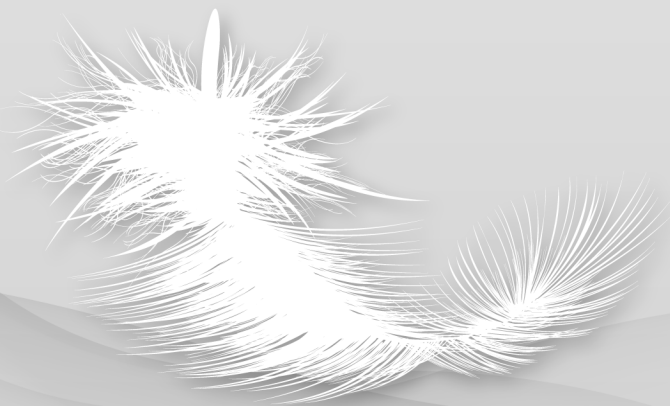
12-3 存取資料庫伺服器

12-4 執行 SQL 查詢

12-5 取得欄位資訊

12-6 取得記錄內容

12-7 分頁瀏覽



12-1 PHP 與資料庫

從 PHP 5.5 開始，過去用來存取 MariaDB/MySQL 資料庫的 `mysql_connect()`、`mysql_close()` 等數十個函式都被標示為過時 (`deprecated`)，而在 PHP 7 推出時，這些函式則被直接移除，建議勿再使用，避免程式出錯。

資料庫存取有物件導向和函式兩種語法，前者可以充分展示 PHP 的物件導向功能，而後者則是容易學習，因此，本書範例程式將使用函式語法。不過，在說明函式語法之前，我們先簡單介紹物件導向語法，好讓您有個基本概念，有興趣要進一步學習的讀者可以參考 PHP 文件 (<https://www.php.net/manual/en/book.mysql.php>)。

下面是一個例子，它會開啟 `product` 資料庫，從 `price` 資料表讀取 `category` 為「主機板」的資料並顯示在網頁上。

\ch12\mysql_oo.php (下頁續 1/2)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $mysqli = new mysqli("localhost", "root", "dbpwd1022", "product");
      if ($mysqli->connect_errno)
        die("無法建立資料連接: " . $mysqli->connect_error);

      $mysqli->query("SET NAMES utf8");
      $result = $mysqli->query("SELECT * FROM price WHERE category = '主機板'");

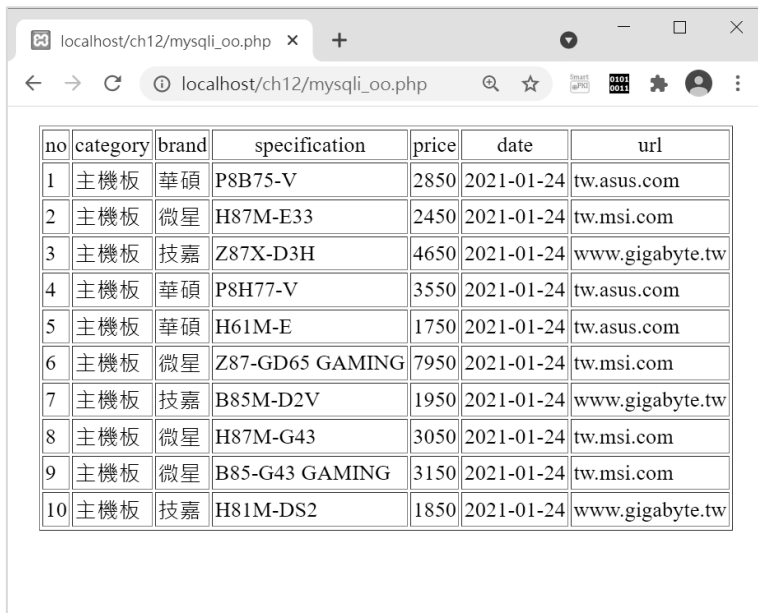
      echo "<table border='1' align='center'><tr align='center'>";
      // 顯示欄位名稱
      while ($field = $result->fetch_field())
        echo "<td>" . $field->name . "</td>";
      echo "</tr>";
```

\ch12\mysqli_oo.php (接上頁 2/2)

```
while ($row = $result->fetch_row())
{
    echo "<tr>";
    for ($i = 0; $i < $result->field_count; $i++)
        echo "<td>" . $row[$i] . "</td>";
    echo "</tr>";
}
echo "</table>";

$result->free();
$mysqli->close();
?>
</body>
</html>
```

提醒您，在執行這個程式之前，請記得匯入 **product** 資料庫 (\database\product.sql)，執行結果如下。



no	category	brand	specification	price	date	url
1	主機板	華碩	P8B75-V	2850	2021-01-24	tw.asus.com
2	主機板	微星	H87M-E33	2450	2021-01-24	tw.msi.com
3	主機板	技嘉	Z87X-D3H	4650	2021-01-24	www.gigabyte.tw
4	主機板	華碩	P8H77-V	3550	2021-01-24	tw.asus.com
5	主機板	華碩	H61M-E	1750	2021-01-24	tw.asus.com
6	主機板	微星	Z87-GD65 GAMING	7950	2021-01-24	tw.msi.com
7	主機板	技嘉	B85M-D2V	1950	2021-01-24	www.gigabyte.tw
8	主機板	微星	H87M-G43	3050	2021-01-24	tw.msi.com
9	主機板	微星	B85-G43 GAMING	3150	2021-01-24	tw.msi.com
10	主機板	技嘉	H81M-DS2	1850	2021-01-24	www.gigabyte.tw

PHP 提供數十個函式讓使用者存取 MariaDB/MySQL 資料庫，我們會在相關章節中介紹下列函式。

函式	說明	頁數
<code>mysqli_affected_rows()</code>	取得最近一次執行 INSERT、UPDATE 或 DELETE 指令時，被影響的記錄筆數。	12-18
<code>mysqli_close()</code>	關閉資料連接。	12-7
<code>mysqli_connect()</code>	建立資料連接。	12-5
<code>mysqli_data_seek()</code>	移動記錄指標。	12-31
<code>mysqli_connect_errno()</code>	傳回最近一次呼叫 <code>mysqli_connect()</code> 函式所產生的錯誤代碼。	12-12
<code>mysqli_connect_error()</code>	傳回最近一次呼叫 <code>mysqli_connect()</code> 函式所產生的錯誤訊息。	12-12
<code>mysqli_errno()</code>	傳回最近一次存取資料庫所產生的錯誤代碼。	12-12
<code>mysqli_error()</code>	傳回最近一次存取資料庫所產生的錯誤訊息。	12-12
<code>mysqli_fetch_array()</code>	將查詢結果存入結合陣列或數值陣列。	12-27
<code>mysqli_fetch_assoc()</code>	將查詢結果存入結合陣列 (associative array)。	12-30
<code>mysqli_fetch_field()</code>	從查詢結果中取得欄位資訊，傳回值為 object 型別。	12-24
<code>mysqli_fetch_object()</code>	從查詢結果中取得記錄資訊，傳回值為 object 型別。	12-30
<code>mysqli_fetch_row()</code>	從查詢結果中取得記錄資訊，傳回值為陣列型別。	12-25
<code>mysqli_field_seek()</code>	移動欄位指標。	12-24
<code>mysqli_free_result()</code>	釋放查詢結果所佔用的記憶體。	12-26
<code>mysqli_get_client_info()</code>	取得 MariaDB/MySQL 用戶端函式庫的版本資訊。	12-8
<code>mysqli_get_host_info()</code>	取得 MariaDB/MySQL 主機的相關資訊。	12-9
<code>mysqli_get_proto_info()</code>	取得 MariaDB/MySQL 資料庫協定的版本資訊。	12-10
<code>mysqli_get_server_info()</code>	取得 MariaDB/MySQL 資料庫的版本資訊。	12-11
<code>mysqli_num_fields()</code>	取得執行 SELECT 指令時，結果所包含的欄位數目。	12-18
<code>mysqli_num_rows()</code>	取得執行 SELECT 指令時，結果所包含的記錄筆數。	12-18
<code>mysqli_query()</code>	執行 SQL 查詢。	12-15
<code>mysqli_select_db()</code>	開啟資料庫。	12-13

12-2 建立與關閉資料連接

12-2-1 建立資料連接

在使用 PHP 存取 MariaDB/MySQL 資料庫之前，必須先建立「資料連接」，也就是使用指定的帳號與密碼登入 MariaDB/MySQL 資料庫伺服器。

我們可以使用 `mysqli_connect()` 函式建立資料連接，其語法如下，若建立資料連接成功，就會傳回連接識別字 (link identifier)，否則傳回 `FALSE`：

```
mysqli_connect([string host [, string username [, string password [,string dbname]]]])
```

- ❏ *host*：MariaDB/MySQL 資料庫伺服器的電腦名稱、DNS 名稱或 IP 位址，例如 `localhost`，參數可以包含 `port` 資訊，例如 `localhost:1000`，若省略此參數，則預設值為 `localhost:3306`。
- ❏ *username*：登入 MariaDB/MySQL 資料庫伺服器的帳號。
- ❏ *password*：登入 MariaDB/MySQL 資料庫伺服器的密碼。
- ❏ *dbname*：預設的資料庫名稱。

下面是一個例子，它會試著建立資料連接，其中第 09 行是呼叫 `mysqli_connect()` 函式建立資料連接，此處的主機名稱、登入帳號與密碼請根據您的實際情況做設定，若建立資料連接失敗，就會執行 `or` 後面的 `die("無法建立資料連接")`，終止程式並顯示「無法建立資料連接」，否則會執行第 10 行，顯示「成功建立資料連接」。

\ch12\mysqli_connect.php (下頁續 1/2)

```
01:<!DOCTYPE html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title>建立資料連接</title>
06:  </head>
```

\ch12\mysqli_connect.php (接上頁 2/2)

```

07: <body>
08:   <?php
09:       $link = mysqli_connect("localhost", "root", "dbpwd1022") or die("無法建立資料連接");
10:       echo "成功建立資料連接";
11:   ?>
12: </body>
13:</html>

```

**NOTE**

為了方便起見，本書的 PHP 程式在建立資料連接時，均是以 root 帳號登入資料庫伺服器，不過，從安全性角度來看，這樣的做法並不妥當，因為 root 帳號的權限太大，通常只有在管理資料庫伺服器時，才會使用 root 帳號登入，避免密碼外洩。若要新增帳號，可以登入 phpMyAdmin，然後在右窗格中點取 [使用者帳號]、[新增使用者帳號]，再依照畫面的提示操作即可。



12-2-2 關閉資料連接

雖然使用 `mysqli_connect()` 函式所建立的資料連接在所有程式碼執行完畢後會自動關閉，但我們建議您在不需要存取資料庫時，就使用 `mysqli_close()` 函式關閉資料連接，無須等到所有程式碼執行完畢，其語法如下：

```
mysqli_close(resource link_identifier)
```

下面是一個例子，它和 `\ch12\mysqli_connect.php` 的差別在於第 11 行是以手動的方式關閉資料連接。

`\ch12\mysqli_close.php`

```
01:<!DOCTYPE html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title>關閉資料連接</title>
06:  </head>
07:  <body>
08:    <?php
09:      $link = mysqli_connect("localhost", "root", "dbpwd1022") or die("無法建立資料連接");
10:      echo "成功建立資料連接";
11:      mysqli_close($link);
12:    ?>
13:  </body>
14:</html>
```



12-4 執行 SQL 查詢

在執行 SQL 查詢之前，除了要與 MariaDB/MySQL 資料庫伺服器建立資料連接，還要開啟指定的資料庫，才能對該資料庫執行 SQL 查詢。

12-4-1 使用 `mysqli_select_db()` 函式開啟資料庫

我們可以使用 `mysqli_select_db()` 函式開啟資料庫，其語法如下，若開啟資料庫成功，就傳回 `TRUE`，否則傳回 `FALSE`：

```
mysqli_select_db(resource link_identifier, string database_name)
```

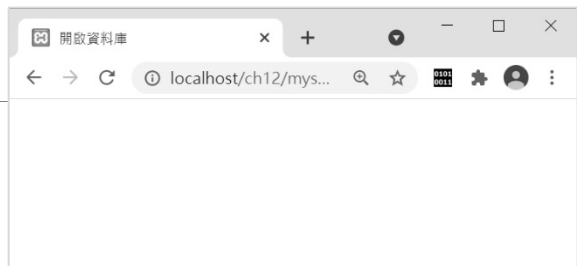
❏ *link_identifier*：連接識別字。

❏ *database_name*：欲開啟的資料庫名稱。

下面是一個例子，它會呼叫 `mysqli_select_db()` 函式開啟 `students` 資料庫。

`\ch12\mysqli_select_db.php`

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>開啟資料庫</title>
  </head>
  <body>
    <?php
      $link = mysqli_connect("localhost", "root", "dbpwd1022")
        or die("無法建立資料連接: " . mysqli_connect_error());
      mysqli_select_db($link, "students")
        or die("無法開啟 students 資料庫: " . mysqli_error($link));
      mysqli_close($link);
    ?>
  </body>
</html>
```



還記得 `mysqli_connect()` 函式的語法嗎？其語法如下，其中 *dbname* 參數可以用來指定預設的資料庫名稱：

```
mysqli_connect([string host [, string username [, string password [,string dbname]]]])
```

因此，`\ch12\mysqli_select_db.php` 也可以寫成如下，在使用 `mysqli_connect()` 函式建立資料連接時，直接使用第 4 個參數來指定要使用哪個資料庫。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>開啟資料庫</title>
  </head>
  <body>
    <?php
      $link = mysqli_connect("localhost", "root", "dbpwd1022", "students")
        or die("無法建立資料連接: " . mysqli_connect_error());
      mysqli_close($link);
    ?>
  </body>
</html>
```

12-4-2 使用 `mysqli_query()` 函式執行 SQL 查詢

開啟資料庫後，我們可以使用 `mysqli_query()` 函式執行 SQL 查詢，其語法如下：

```
mysqli_query(resource link_identifier, string query)
```

❏ *link_identifier*：連接識別字。

❏ *query*：欲執行的 SQL 查詢。

`mysqli_query()` 函式的執行結果有下列兩種：

❏ 失敗：一律傳回 `FALSE`。

❏ 成功：傳回 `TRUE`，當 `mysqli_query()` 函式執行 `SELECT`、`SHOW`、`EXPLAIN` 或 `DESCRIBE` 指令時，會傳回資源識別字（`mysqli_result` 物件），指向查詢結果，您可以將它想像成位於記憶體內的資料庫。

下面是一個例子，它會使用 `mysqli_query()` 函式執行 `SELECT` 指令。

\ch12\mysqli_query.php (下頁續 1/2)

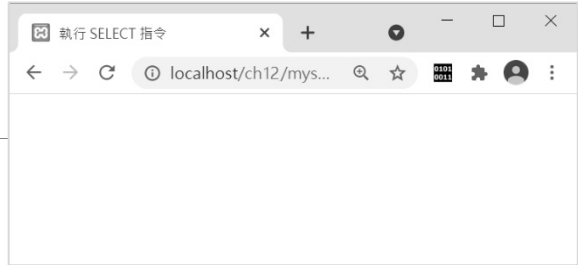
```
01:<!DOCTYPE html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title>執行 SELECT 指令</title>
06:  </head>
07:  <body>
08:    <?php
09:      $link = mysqli_connect("localhost", "root", "dbpwd1022")
          or die("無法建立資料連接: " . mysqli_connect_error());
10:
11:      mysqli_select_db($link, "product")
          or die("無法開啟 product 資料庫: " . mysqli_error($link));
12:
```

\ch12\mysqli_query.php (接上頁 2/2)

```

13:     $sql = "SELECT * FROM price WHERE category = '主機板'";
14:     $result = mysqli_query($link, $sql);
15:
16:     mysqli_close($link);
17:     ?>
18: </body>
19:</html>

```



- ❏ 09：呼叫 `mysqli_connect()` 函式建立資料連接，此處的主機名稱、登入帳號與密碼請根據您的實際情況做設定，當建立資料連接失敗時，會顯示「無法建立資料連接」，並呼叫 `mysqli_connect_error()` 函式顯示錯誤訊息，然後終止程式。
- ❏ 11：開啟 `product` 資料庫，當開啟 `product` 資料庫失敗時，會執行顯示「無法開啟 `product` 資料庫」，並呼叫 `mysqli_error()` 函式顯示錯誤訊息，然後終止程式。
- ❏ 13：設定欲執行的 SQL 查詢。
- ❏ 14：呼叫 `mysqli_query()` 函式執行 SQL 查詢，傳回值為資源識別字，指向查詢結果。

請注意，建議您在第 11 行的前面呼叫 `mysqli_query()` 函式執行「`SET NAMES utf8`」指令，設定查詢所要使用的字元集名稱，因為我們通常會將資料庫的編碼方式設定為 `UTF-8`，才能支援多國字元，故須將查詢所要使用的字元集名稱設定為 `utf8`，否則一旦查詢結果包含非 `ASCII` 字元，將會出現亂碼。

```
mysqli_query($link, "SET NAMES utf8");
```

由於每次存取資料庫都必須建立資料連接及開啟資料庫，因此，我們撰寫下列函式，在需要建立資料連接或執行 SQL 查詢時，就可以直接呼叫使用，減少重複撰寫程式碼。

`\ch12\dbtools.inc.php`

```
01:<?php
02:  function create_connection()
03:  {
04:      $link = mysqli_connect("localhost", "root", "dbpwd1022")
          or die("無法建立資料連接: " . mysqli_connect_error());
05:      mysqli_query($link, "SET NAMES utf8");
06:      return $link;
07:  }
08:
09:  function execute_sql($link, $database, $sql)
10:  {
11:      mysqli_select_db($link, $database)
          or die("開啟資料庫失敗: " . mysqli_error($link));
12:      $result = mysqli_query($link, $sql);
13:      return $result;
14:  }
15:?>
```

- ◀ 02 ~ 07：定義 `create_connection()` 函式，用來建立資料連接，第 05 行可以解決資料庫中文亂碼問題。
- ◀ 09 ~ 14：定義 `execute_sql()` 函式，用來執行指定的 SQL 查詢，此函式包含 3 個參數，*link* 用來指定欲使用的資料連接，*database* 用來指定資料庫名稱，*sql* 用來指定欲執行的 SQL 查詢。舉例來說，假設您要對 `prodcut` 資料庫的 `price` 資料表執行「`SELECT * FROM price WHERE category = '主機板'`」查詢，可以寫成如下：

```
execute_sql($link, "prodcut", "SELECT * FROM price WHERE category = '主機板'");
```

12-4-3 取得執行 SQL 查詢被影響的記錄筆數或欄位數目

PHP 提供下列三個函式，讓我們能夠得知在執行 SQL 查詢後，有多少筆記錄或多少個欄位受到影響：

- ❏ `mysqli_num_rows()`：適用於執行 `SELECT` 指令，可以傳回選取的記錄筆數，其語法如下，參數 *result* 為資源識別字 (resource identifier)：

```
mysqli_num_rows(resource result)
```

- ❏ `mysqli_num_fields()`：適用於執行 `SELECT` 指令，可以傳回選取的欄位數目，其語法如下，參數 *result* 為資源識別字：

```
mysqli_num_fields(resource result)
```

- ❏ `mysqli_affected_rows()`：適用於執行 `INSERT`、`UPDATE`、`REPLACE`、`DELETE` 等指令，可以傳回有多少筆記錄受到該指令的影響，其語法如下，參數 *link_identifier* 為連接識別字 (link identifier)：

```
mysqli_affected_rows(resource link_identifier)
```

若在執行 `DELETE` 指令時沒有指定 `WHERE` 子句，將導致資料表的所有記錄被刪除，此時，`mysqli_affected_rows()` 函式會傳回 0，而不是實際刪除的記錄筆數。

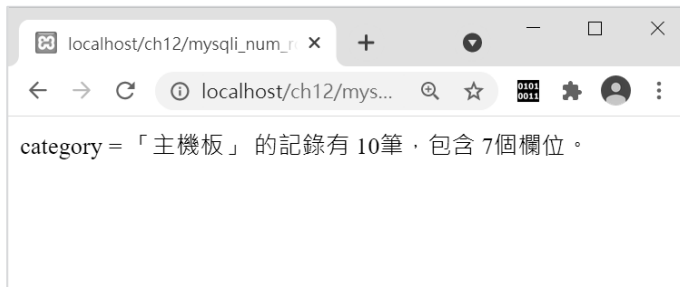
若最近一次執行 SQL 查詢的結果為失敗，那麼 `mysqli_affected_rows()` 函式會傳回 -1。

此外，在執行 `UPDATE` 指令時，`mysqli_affected_rows()` 函式傳回的是實際更新的記錄筆數，而不是符合 `WHERE` 子句的記錄筆數，因為當指定的新值與舊值相同時，並不會有更新的動作，傳回值將為 0。

下面是一個例子，它會執行 `SELECT` 指令，然後在網頁上顯示選取的記錄筆數及欄位數目。

\ch12\mysqli_num_rows.php

```
01:<!DOCTYPE html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:  </head>
06:  <body>
07:    <?php
08:      require_once("dbtools.inc.php");
09:      $link = create_connection();
10:      $sql = "SELECT * FROM price WHERE category = '主機板'";
11:      $result = execute_sql($link, "product", $sql);
12:      echo "category = 「主機板」的記錄有". mysqli_num_rows($result) . "筆";
13:      echo "，包含 ". mysqli_num_fields($result) . "個欄位。";
14:      mysqli_close($link);
15:    ?>
16:  </body>
17:</html>
```



- ❖ 08：呼叫 `require_once()` 函式將 `dbtools.inc.php` 引用進來，這樣就能呼叫自行定義的 `create_connection()` 與 `execute_sql()` 兩個函式。
- ❖ 09：呼叫自行定義的 `create_connection()` 函式建立資料連接。
- ❖ 10 ~ 11：呼叫自行定義的 `execute_sql()` 函式對 `product` 資料庫執行「`SELECT * FROM price WHERE category = '主機板'`」查詢。

12-5 取得欄位資訊

12-5-1 使用 `mysqli_fetch_field_direct()` 函式取得欄位資訊

在使用 `mysqli_query()` 函式執行 `SELECT` 指令後，該函式會傳回資源識別字，此時可以使用 `mysqli_fetch_field_direct()` 函式取得欄位資訊，其語法如下：

```
mysqli_fetch_field_direct(resource result, int field_offset)
```

- ❖ *result*：資源識別字。
- ❖ *field_offset*：欄位的序號，0 表示第一個欄位，1 表示第二個欄位，2 表示第三個欄位，依此類推。

`mysqli_fetch_field_direct()` 函式的傳回值為 `object` 型別，常用的屬性如下。

屬性	說明
<code>name</code>	欄位名稱。
<code>orgname</code>	欄位的原始名稱。
<code>table</code>	欄位所屬的資料表名稱。
<code>orgtable</code>	欄位所屬的原始資料表名稱。
<code>db</code>	欄位所屬的資料庫名稱。
<code>max_length</code>	欄位內容實際存放的最大長度，不是資料庫內設定的資料長度。
<code>length</code>	欄位在資料庫內設定的資料長度。
<code>type</code>	欄位型態，3 代表 <code>Integer</code> 、10 代表 <code>DATE</code> 、253 代表 <code>VARCHAR</code> 。

舉例來說，假設要取得第 2 個欄位資訊，可以寫成如下：

```
$meta = mysqli_fetch_field_direct($result, 1);
```

假設要取得第 2 個欄位資訊並顯示其欄位名稱及資料型態，可以寫成如下：

```
$meta = mysqli_fetch_field_direct($result, 1);  
echo "欄位名稱：$meta->name";  
echo "資料型態：$meta->type";
```

下面是一個例子，它會顯示資料表內所有欄位的欄位名稱、資料型態及最大長度。

\ch12\mysqli_fetch_field_direct.php

```
01:<!DOCTYPE html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title>顯示欄位資訊</title>
06:  </head>
07:  <body>
08:    <?php
09:      require_once("dbtools.inc.php");
10:
11:      $link = create_connection();
12:      $sql = "SELECT * FROM price WHERE category = '主機板'";
13:      $result = execute_sql($link, "product", $sql);
14:
15:      echo "<table width='400' border='1'><tr align='center'>";
16:      echo "<td>欄位名稱</td><td>資料型態</td><td>最大長度</td></tr>";
17:      $i = 0;
18:      while ($i < mysqli_num_fields($result))
19:      {
20:        $meta = mysqli_fetch_field_direct($result, $i);
21:        echo "<tr>";
22:        echo "<td>$meta->name</td>";
23:        echo "<td>$meta->type</td>";
24:        echo "<td>$meta->max_length</td>";
25:        echo "</tr>";
26:        $i++;
27:      }
28:      echo "</table>";
29:
30:      mysqli_close($link);
31:    ?>
32:  </body>
33:</html>
```



欄位名稱	資料型態	最大長度
no	3	2
category	253	9
brand	253	6
specification	253	15
price	3	4
date	10	10
url	253	15

- ❖ 09：呼叫 `require_once()` 函式將 `dbtools.inc.php` 引用進來，這樣就能呼叫自行定義的 `create_connection()` 與 `execute_sql()` 兩個函式。
- ❖ 11：呼叫自行定義的 `create_connection()` 函式建立資料連接。
- ❖ 12 ~ 13：叫定自行定義的 `execute_sql()` 函式對 `product` 資料庫執行「`SELECT * FROM price WHERE category = '主機板'`」查詢。
- ❖ 17 ~ 27：顯示資料表內所有欄位的欄位名稱、資料型態及最大長度，其中資料型態的傳回值為數值，不同數字代表不同資料型態，例如 3 代表 Integer、10 代表 DATE、253 代表 VARCHAR，詳細資訊可以參考 PHP 文件 (<https://www.php.net/manual/en/mysqli-result.fetch-field-direct.php>)。
- ❖ 30：關閉資料連接。

12-5-2 使用 `mysqli_fetch_field()` 函式取得欄位資訊

我們也可以使用 `mysqli_fetch_field()` 函式取得欄位資訊，其語法如下，此函式會傳回與 `mysqli_fetch_field_direct()` 函式相同的物件：

```
mysqli_fetch_field(resource $result)
```

下面是一個例子，它會顯示資料表內所有欄位的欄位名稱、資料型態及最大長度，執行結果與第 12-5-1 節的例子一樣。

\ch12\mysqli_fetch_field.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>顯示欄位資訊</title>
  </head>
  <body>
    <?php
      require_once("dbtools.inc.php");

      $link = create_connection();
      $sql = "SELECT * FROM price WHERE category = '主機板'";
      $result = execute_sql($link, "product", $sql);

      echo "<table width='400' border='1'><tr align='center'>";
      echo "<td>欄位名稱</td><td>資料型態</td><td>最大長度</td></tr>";
      while ($meta = mysqli_fetch_field($result))
      {
        echo "<tr>";
        echo "<td>$meta->name</td>";
        echo "<td>$meta->type</td>";
        echo "<td>$meta->max_length</td>";
        echo "</tr>";
      }
      echo "</table>";

      mysqli_close($link);
    ?>
  </body>
</html>
```



欄位名稱	資料型態	最大長度
no	3	2
category	253	9
brand	253	6
specification	253	15
price	3	4
date	10	10
url	253	15

12-5-3 使用 `mysqli_field_seek()` 函式移動欄位指標

請您回想一下 `mysqli_fetch_field()` 函式的語法：

```
mysqli_fetch_field(resource result)
```

此函式可以用來取得目前欄位的資訊，那麼 `mysqli_fetch_field()` 函式如何知道目前位於哪個欄位？！喔，是這樣的，事實上是有一個欄位指標記錄著目前位於哪個欄位。

`mysqli_field_seek()` 函式可以讓我們移動欄位指標，其語法如下，若移動欄位指標成功，就傳回 `TRUE`，否則傳回 `FALSE`：

```
mysqli_field_seek(resource result, int field_offset)
```

◀ *result*：資源識別字。

◀ *field_offset*：欄位的序號，0 表示第一個欄位，1 表示第二個欄位，2 表示第三個欄位，依此類推。

舉例來說，在下面的程式碼執行完畢後，變數 `$meta` 是存放了第幾個欄位的資訊呢？答案是第 5 個欄位，因為第一個敘述已經先將欄位指標移到第 5 個欄位了：

```
$seek_result = mysqli_field_seek($result, 4);  
$meta = mysqli_fetch_field($result);
```

12-7 分頁瀏覽

「分頁瀏覽」是網頁資料庫經常使用的功能，當選取的記錄太多時，我們通常不會一次全部顯示，而是以分頁的方式來顯示，才不會造成瀏覽單一網頁的速度過慢。

下圖是我們在本節中所要製作的分頁網頁，每頁顯示 5 筆記錄，網頁最下方有一個導覽列，可以讓您快速瀏覽其它頁次的記錄，預設會顯示第一頁。



1. 點取頁次，例如第 2 頁



2. 顯示第 2 頁的記錄

\ch12\show_record.php (下頁續 1/3)

```
01:<!DOCTYPE html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title>分頁瀏覽</title>
06:  </head>
07:  <body>
08:    <?php
09:      require_once("dbtools.inc.php");
10:
11:      // 設定每頁顯示幾筆記錄
12:      $records_per_page = 5;
13:
14:      // 取得要顯示第幾頁的記錄
15:      if (isset($_GET["page"]))
16:        $page = $_GET["page"];
17:      else
18:        $page = 1;
19:
20:      // 建立資料連接
21:      $link = create_connection();
22:
23:      // 執行 SQL 查詢
24:      $sql = "SELECT category AS '零組件種類', brand AS '品牌', specification AS
           '規格', price AS '價格', date AS '報價日期' FROM Price";
25:      $result = execute_sql($link, "product", $sql);
26:
27:      // 取得欄位數目
28:      $total_fields = mysqli_num_fields($result);
29:
30:      // 取得記錄筆數
31:      $total_records = mysqli_num_rows($result);
32:
33:      // 計算總頁數
34:      $total_pages = ceil($total_records / $records_per_page);
35:
```

\ch12\show_record.php (下頁續 2/3)

```
36:      // 計算本頁第一筆記錄的序號
37:      $started_record = $records_per_page * ($page - 1);
38:
39:      // 將記錄指標移至本頁第一筆記錄的序號
40:      mysqli_data_seek($result, $started_record);
41:
42:      // 顯示欄位名稱
43:      echo "<table border='1' align='center' width='800'>";
44:      echo "<tr align='center'>";
45:      for ($i = 0; $i < $total_fields; $i++)
46:          echo "<td>" . mysqli_fetch_field_direct($result, $i)->name . "</td>";
47:      echo "</tr>";
48:
49:      // 顯示記錄
50:      $j = 1;
51:      while ($row = mysqli_fetch_row($result) and $j <= $records_per_page)
52:      {
53:          echo "<tr>";
54:          for($i = 0; $i < $total_fields; $i++)
55:              echo "<td>$row[$i]</td>";
56:          $j++;
57:          echo "</tr>";
58:      }
59:      echo "</table>";
60:
61:      // 產生導覽列
62:      echo "<p align='center'>";
63:      if ($page > 1)
64:          echo "<a href='show_record.php?page=" . ($page - 1) . "'>上一頁</a> ";
65:      for ($i = 1; $i <= $total_pages; $i++)
66:      {
67:          if ($i == $page)
68:              echo "$i ";
69:          else
70:              echo "<a href='show_record.php?page=$i'>$i</a> ";
71:      }
```

\ch12\show_record.php (接上頁 3/3)

```

72:     if ($page < $total_pages)
73:         echo "<a href='show_record.php?page=". ($page + 1) . "'>下一頁</a> ";
74:     echo "</p>";
75:     // 釋放記憶體
76:     mysqli_free_result($result);
77:     // 關閉資料連接
78:     mysqli_close($link);
79:     ?>
80: </body>
81:</html>

```

- ◀ 12：設定每頁顯示幾筆記錄，此處是 5，您可以視實際情況做設定。
- ◀ 15 ~ 18：設定要顯示第幾頁的記錄，一開始會先取得網址參數 `page`，我們使用 `isset()` 函式判斷變數 `$_GET["page"]` 是否有取得數值，若有取得數值，表示瀏覽者有指定要觀看第 `page` 頁的記錄，就將變數 `page` 設定為取得的數值，否則將變數 `page` 設定為 1，讓網頁顯示第一頁的記錄。
- ◀ 24：設定 SQL 查詢，用來取得 `category`、`brand`、`specification`、`price`、`date` 五個欄位的所有記錄並指定其中文別名。
- ◀ 34：計算總頁數，此處使用 `ceil()` 函式，一旦總頁數出現小數點，就無條件進位。
- ◀ 37：計算在目前要顯示的頁次中，第一筆記錄是位於查詢結果的第幾筆。
- ◀ 50 ~ 58：顯示某一區間範圍內的記錄，第 51 行是當有讀取到記錄且 `$j <= $records_per_page` 時，才會執行 `while` 區塊內的程式碼來顯示記錄，其中 `$j <= $records_per_page` 用來控制每頁顯示的記錄筆數，此處是 5。
- ◀ 63 ~ 73：產生導覽列，讓瀏覽者快速換頁。第 63 ~ 64 行是當目前頁次大於第一頁時，就插入「上一頁」超連結；第 72 ~ 73 行是當目前頁次小於最後一頁時，就插入「下一頁」超連結；第 65 ~ 71 行是產生所有頁碼，目前頁次的頁碼為純文字，而非目前頁次的頁碼則為超連結。