第六章

製作網站表單

處理表單，是任何網路應用程式必要的組成內容之一。表單（form）是使用者與伺服器互動的方式，我們可以：登記新的使用者帳號、搜尋論壇上有關特定主題的文章、重新取得忘記的密碼、找出附近的餐廳或修鞋舖、購買書籍……等等。

在 PHP 程式裡使用表單有兩個步驟。步驟一：呈現表單。包括以 HTML 標籤建構完善的使用者介面元素，例如文字方塊（text box）、核取方塊（checkbox）、按鈕（button）。如果你對建立表單必備的 HTML 不甚熟悉，就從參考《HTML & XHTML 大全》的＜表單＞一章開始吧。（Chuck Musciano 與 Bill Kennedy 著，歐萊禮出版）

使用者看到內含表單的頁面後，開始輸入資訊，接著按下「送出」（submit）按鈕或 Enter 鍵送出表單資訊。伺服器處理傳來的表單資訊即為步驟二。

範例 6-1 是對使用者打招呼的網頁。如果使用者送出姓名，該頁面會呈現問候語。如果沒送出姓名，該頁面會呈現使用者可以輸入姓名的表單。

範例 6-1 打聲招呼

```
if (array_key_exists('my_name',$_POST)) {
    print "Hello, ". $_POST['my_name'];
} else {
    print<<<_HTML_
<form method="post" action="$_SERVER[PHP_SELF]">
 Your name: <input type="text" name="my_name">
<br/>
<input type="submit" value="Say Hello">
</form>
_HTML_;
}
```

還記得第一章的用戶端與伺服端溝通圖嗎？圖 6-1 顯示必要的用戶端與伺服端對話，用以呈現及處理範例 6-1 的表單。第一組申請（request）和回應（response）使瀏覽器顯示表單。第二組申請和回應中，伺服器處理傳入的表單資料，瀏覽器則呈現結果。

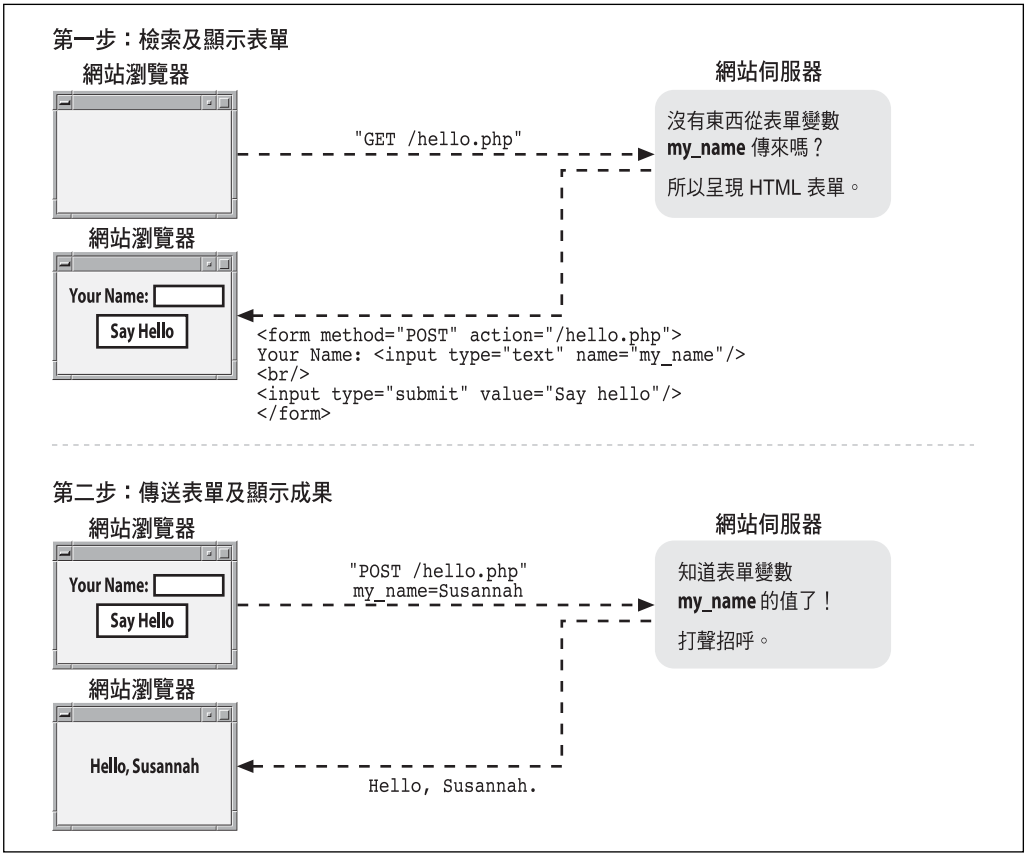


圖 6-1 簡易的表單顯示與處理

對第一組申請的回應是組成表單的 HTML 碼。圖 6-2 顯示瀏覽器收到回應後顯示的畫面。

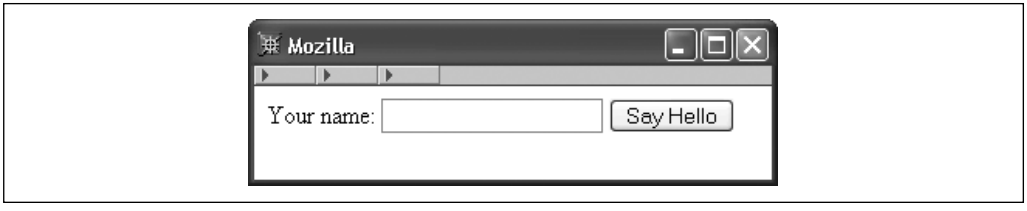


圖 6-2 一份簡易表單

對第二組申請的回應是處理傳回的表單資料。圖 6-3 呈現鍵入 Susannah 得到的輸出結果。

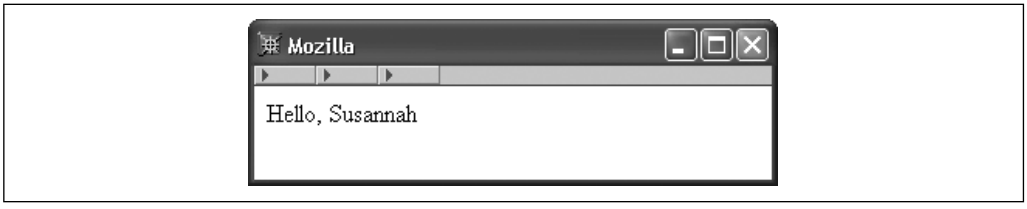


圖 6-3 送出表單

範例 6-1 的模式，「如果傳入表單資料，則予以處理；其他狀況，則呈現表單」，是種常見的 PHP 程式。建立基本表單時，維持表單的顯示，與處理該表單的程式碼於同一頁，較容易保持表單和相關邏輯的同步。

表單輸入的資料將送至與申請表單時相同的 URL，因為 `<form>` 標籤 `action` 屬性的值是特殊變數：`$_SERVER[PHP_SELF]`。`$_SERVER` 是自動全域變數陣列（auto-global array），持有伺服器和 PHP 直譯器處理申請所用的各種資訊。`$_SERVER` 陣列的元素 `PHP_SELF`，持有目前申請的 URL 路徑名稱。例如說，如果存取一段位於 http://www.example.com/store/catalog.php 的程式碼，`$_SERVER['PHP_SELF']` 即為 `/store/catalog.php`。

`$_POST` 陣列是持有表單輸入資料的自動全域變數。`$_POST` 的「鍵」就是表單組件名稱，相對應的 `$_POST` 元素值就是表單的組件值。把自己的名字輸入範例 6-1 的文字方塊，並按下「送出」按鈕，送出的資料成為 `$_POST['my_name']` 的值，因為文字方塊的 `name` 屬性名稱是 `my_name`。

接下來，測試 `$_POST` 陣列裡是否有叫做 `my_name` 的鍵，用以確認表單參數 `my_name` 是否已送出。即使文字方塊 `my_name` 維持空白，`array_key_exists()` 仍舊傳回 `true` 並顯示問候語。

範例 6-1 的結構即為本章表單處理素材的骨幹。然而，這個範例有個缺陷：直接顯示未經修正的外部輸入資料不甚安全，具有表單參數 `my_name` 值的 `print "Hello, " . $_POST['my_name'];` 即為一例。來自程式外的資料，如傳入的表單參數，有可能內嵌 HTML 或 JavaScript。「HTML 與 JavaScript」一節，將解釋如何清除外部表單的輸入資料，以確保程式更為安全。

本章其餘部分則詳述表單處理的各種面向。「存取表單參數」深入探討表單輸入資料的特色，如可以傳入多個值的表單參數。「以函式處理表單」則描繪有彈性、以函式為基礎的架構，用以簡化某些表單維護的任務。以函式為基礎的架構還可檢查傳入的表單資料，以確認資料不含有意外事物。「資料核查」一節，解釋用來檢查表單傳入資料的各

種不同方式。「顯示預定值」則示範如何提供表單元素的預定值，及如何在重現表單時保存使用者輸入的值。最後一節「整合運用」，則呈現包含本章所有議題的表單，包括：以函式為基礎的組織架構、核查資料及錯誤訊息的顯示、通用預定值及保存使用者輸入的資料，最後是處理傳入的資料。

常用伺服器變數

除了 PHP_SELF，自動全域變數陣列 $_SERVER 還包含不少有用的元素，可提供網站伺服器和申請的相關資訊。表 6-1 列出常用的元素。

表 6-1 $_SERVER 的 entry

元素	範例	說明
QUERY_STRING	category=kitchen&price=5	在問號後的這部分URL 即為 URL 參數。本例的查詢字串應用的 URL 為 *http://www.example.com/catalog/store.php?category=kitchen&price=5*。
PATH_INFO	/browse	附加的路徑資訊依附於 URL 末端，路徑前要加斜線。這是把資訊傳給一段程式，但又不需用到查詢字串的方式。本例的 PATH_INFO 應用的 URL 為 *http://www.example.com/catalog/store.php/browse*。
SERVER_NAME	www.example.com	這是 PHP 直譯器運行的網站名稱。如果網站伺服器管理數個虛擬網域，本項則為被直譯器存取的虛擬網域名稱。
DOCUMENT_ROOT	/usr/local/htdocs	存有可讀取網站文件的伺服器目錄。如果網站 *http://www.example.com* 的文件根目錄是 */usr/local/htdocs*，則讀取頁面 *http://www.example.com/catalog/store.php* 時，與之相應的檔案位於 */usr/local/htdocs/catalog/store.php*。
REMOTE_ADDR	175.56.28.3	向網站伺服器送出申請的使用者對應 IP 位址。
REMOTE_HOST	pool0560.cvx.dialup.verizon.net	如果網站伺服器調整為把使用者的 IP 位址轉譯成主機名稱，這就是向網站伺服器發出申請的主機名稱。因為位址與名稱的轉譯所費不貲（我是指運算時間），多數網站不會這麼做。

表 6-1 $_SERVER 的 entry（續）

元素	範例	說明
HTTP_REFERER【註】	http://directory.google.com/Top/Shopping/Clothing/	如果有人點選可觸及現行 URL 的鏈結，HTTP_REFERER 即包含含有該鏈結的頁面的 URL。這個值可以偽造，所以別當成存取個人網頁的唯一校標（criteria/criterion）。不過，這項功能可用來找出鏈結到網站的人。
HTTP_USER_AGENT	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	用來檢索頁面的網站瀏覽器。範例的值是 Windows XP 搭配 Internet Explorer 6.0 時產生的簽章（signature）。它像 HTTP_REFERER 一樣可以假造，但常用於分析。

註　正確的拼法是 HTTP_REFERRER，可是很多早期的網路專門文件都拼錯了，所以在寫網路程式時會常常看到只有三個 R 的拼法。

存取表單參數

每串申請的開始，PHP 直譯器都會設定一些全域陣列變數，內含表單或 URL 傳入的任何變數。URL 傳入的變數，和用 GET 方法傳送的表單變數，都放在陣列 $_GET。用 POST 方法傳送的表單變數，則放在 $_POST。

http://www.example.com/catalog.php?*product_id=21&category=fryingpan* 把兩個變數放入 $_GET：$_GET['product_id'] 被設為 21；$_GET['category'] 被設為 fryingpan。送出範例 6-2 的表單，會使同樣的值存入 $_POST（假設文字方塊也輸入 21，下拉式選單中也選了 Frying Pan）。

範例 6-2 有兩個組件的表單

```
<form method="POST" action="catalog.php">
<input type="text" name="product_id">
<select name="category">
<option value="ovenmitt">Pot Holder</option>
<option value="fryingpan">Frying Pan</option>
<option value="torch">Kitchen Torch</option>
</select>
<input type="submit" name="submit">
</form>
```

範例 6-3 將範例 6-2 的表單與完整的 PHP 程式結合，該程式會在列出表單後呈現適當的 $_POST 值。因為 <form> 標籤的 action 屬性是 catalog.php，所以 PHP 程式需儲存在網站伺服器的 catalog.php 裡。如果程式存放的檔名不同，則視情況調整 action 屬性。

範例 6-3 呈現送入的表單參數

```
<form method="POST" action="catalog.php">
<input type="text" name="product_id">
<select name="category">
<option value="ovenmitt">Pot Holder</option>
<option value="fryingpan">Frying Pan</option>
<option value="torch">Kitchen Torch</option>
</select>
<input type="submit" name="submit">
</form>
Here are the submitted values:

product_id: <?php print $_POST['product_id']; ?>
<br/>
category: <?php print $_POST['category']; ?>
```

一個表單組件可能攜有很多個值，在命名時需要以 `[]` 結尾；用來告訴 PHP 直譯器把這些值視為陣列元素。範例 6-4 即把 `<select>` 選單的送出值放入 `$_POST['lunch']` 陣列中。

範例 6-4 有很多個值的表單組件

```
<form method="POST" action="eat.php">
<select name="lunch[]" multiple>
<option value="pork">BBQ Pork Bun</option>
<option value="chicken">Chicken Bun</option>
<option value="lotus">Lotus Seed Bun</option>
<option value="bean">Bean Paste Bun</option>
<option value="nest">Bird-Nest Bun</option>
</select>
<input type="submit" name="submit">
</form>
```

如果以範例 6-4 的表單選擇 `Chicken Bun`、`Bird-Nest Bun` 送出，`$_POST['lunch']` 則會是帶有兩個元素的陣列，元素值為 `chicken` 和 `nest`。用一般的多維陣列語法即可讀取。範例 6-5 將範例 6-4 的表單與完整的 PHP 程式結合，該程式呈現以下拉式選單選到的值。（同樣的規則也可應用在檔案名稱和 `action` 屬性上。請把範例 6-5 的程式碼存在名為 `eat.php` 的檔案中，或調整 `<form>` 標籤的 `action` 屬性為正確的檔名。）

範例 6-5 存取多個送入的值

```
<form method="POST" action="eat.php">
<select name="lunch[]" multiple>
<option value="pork">BBQ Pork Bun</option>
```

```
<option value="chicken">Chicken Bun</option>
<option value="lotus">Lotus Seed Bun</option>
<option value="bean">Bean Paste Bun</option>
<option value="nest">Bird-Nest Bun</option>
</select>
<input type="submit" name="submit">
</form>
Selected buns:
<br/>
<?php
foreach ($_POST['lunch'] as $choice) {
    print "You want a $choice bun. <br/>";
}
?>
```

假設選了範例 6-5 下拉式選單裡的 `Chicken Bun` 和 `Bird-Nest Bun`，則（在列出表單後）呈現的結果為：

```
Selected buns:
You want a chicken bun.
You want a nest bun.
```

從下列被轉譯成 PHP 的程式碼，可以猜到在表單送出時有個命名為 `lunch[]` 的表單組件（假設表單組件的送出值是 `chicken` 和 `nest`）。

```
$_POST['lunch'][] = 'chicken';
$_POST['lunch'][] = 'nest';
```

本例正如範例 4-5，利用在陣列末端加入新元素的語法。

以函式處理表單

範例 6-1 的基本表單還可以更有彈性，呈現表單外觀和處理資料用的程式碼可分別做成不同的函式。範例 6-6 即為 6-1 的函式版。

範例 6-6 用函式打聲招呼吧

```
// 根據傳入的表單參數，執行正確的邏輯
if (array_key_exists('my_name',$_POST)) {
    process_form();
} else {
    show_form();
}

// 表單送出後要做的事
function process_form() {
```

範例 6-6 用函式打聲招呼吧（續）

```
    print "Hello, ". $_POST['my_name'];
}

// 呈現表單外觀
function show_form() {
print<<<_HTML_
<form method="POST" action="$_SERVER[PHP_SELF]">
Your name: <input type="text" name="my_name">
<br/>
<input type="submit" value="Say Hello">
</form>
_HTML_;
}
```

要改變表單，或改變表單送出時的動作，則改變 `process_form()` 或 `show_form()` 的程式本體。函式使得程式更清楚，但範例起始處的的邏輯判斷仍需依賴表單特有的資訊：`my_name` 參數。在測試表單的送出時，把參數隱藏在表單裡可以解決這個問題。如果隱藏的參數在 `$_POST`，則處理表單資訊；反之，則呈現表單。範例 6-7 即以命名為 `_submit_check` 的參數，應用前述邏輯。

範例 6-7 以隱藏參數指示表單的送出資料

```
// 根據隱藏參數 _submit_check，執行正確的邏輯
if ($_POST['_submit_check']) {
    process_form();
} else {
    show_form();
}

// 表單送出後要做的事
function process_form() {
    print "Hello, ". $_POST['my_name'];
}

// 呈現表單外觀
function show_form() {
    print<<<_HTML_
<form method="POST" action="$_SERVER[PHP_SELF]">
Your name: <input type="text" name="my_name">
<br/>
<input type="submit" value="Say Hello">
<input type="hidden" name="_submit_check" value="1">
</form>
_HTML_;
}
```

在這段範例程式開頭的 `if()` 敘述裡只用簡寫法，而不寫 `array_key_exists` 是沒問題的。表單參數 `_submit_check` 只能有一個值：1。不用擔心這個參數被展現在 `$_POST` 中時，會有個被計算為 `false` 的值。

除了把頁面使用的主要邏輯，從任何會變化的表單組件獨立出來之外，使用隱藏參數測試表單送出資訊，還能保證表單在使用者按下 Enter（非點按瀏覽器上的「送出」按鈕時）之後也會處理。如果表單是以 Enter 送出，有些瀏覽器不會隨著其它表單資訊一起傳送「送出」按鈕的名稱和值。但是隱藏的參數則永遠都會包含在資訊內。

表單的處理與顯示分離成函式，也能使資料核查的階段更容易。後面的「資料核查」一節，是所有網站應用程式存取表單傳送資料的關鍵點。資料應在表單送出後、資料處理前先行核對。範例 6-8 把一個核查資料用的函式加入範例 6-7 中。

範例 6-8 表單資料核查

```
// 根據隱藏參數 _submit_check，執行正確的邏輯
if (array_key_exists('_submit_check', $_POST)) {
    if (validate_form()) {
        process_form();
    } else {
        show_form();
    }
} else {
    show_form();
}

// 表單送出後要做的事
function process_form() {
    print "Hello, ". $_POST['my_name'];
}

// 呈現表單外觀
function show_form() {
    print<<<_HTML_
<form method="POST" action="$_SERVER[PHP_SELF]">
Your name: <input type="text" name="my_name">
<br/>
<input type="submit" value="Say Hello">
<input type="hidden" name="_submit_check" value="1">
</form>
_HTML_;
}
```

範例 6-8 表單資料核查（續）

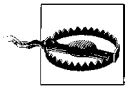
```
// 檢查表單資料
function validate_form() {
    // my_name 是否至少有 3 個字元？
    if (strlen($_POST['my_name']) < 3) {
        return false;
    } else {
        return true;
    }
}
```

範例 6-8 的 `validate_form()`，會在 `$_POST['my_name']` 少於三個字元時傳回 `false`；多於三個字元則傳回 `true`。程式開頭即規定表單送出時就呼叫 `validate_form()`。如果函式傳回 `true`，則呼叫 `process_form()`；反之，則呼叫 `showform()`。這是指，如果送出的表單名稱至少有三個字元，如 `Bob`、`Bartholomew`，則會與上個範例的呈現結果一樣，出現問候語：`Hello, Bob`（或 `Hello, Bartholomew`）。如果送出的名稱像 `BJ` 或乾脆任由文字方塊空白，`validate_form()` 則傳回 `false`，也不會呼叫 `process_form()`。相反地，會被呼叫的函式是 `show_form()`，再呈現一次表單。

範例 6-8 不會說明輸入的名稱無法通過 `validate_form()` 的原因。理想中，如果有人送出過不了 `validate_form()` 這關的資料，程式設計者應該解釋原因，重新呈現表單，如果可以的話，還應該在適當的表單組件中重新呈現輸入的值。「資料核查」一節將說明如何呈現錯誤訊息，「顯示預設值」則會解釋如何安全地再現使用者輸入的值。

資料核查

本節中討論的部分核查策略，將用到**正規運算式**（regular expression，後文簡稱正規式），這是種威力強大的文字比對模式，以獨特的語言寫成。如果對正規式不太熟悉，附錄 B 裡有入門導引。



資料核查是網站應用程式裡最重要的一部分。怪異、錯誤、損壞的資料會在最意想不到的地方出現。使用者可能粗心大意、也可能包藏禍心，更可能比程式原初設計時更有想像力與創意（多半是不小心的）。我很想學「發條橘子」的強力洗腦方式，強制灌輸各位未核查資料的危險性，所以，只好一再強調嚴厲檢查外部傳入資料的重要性。有些外部來源清楚易辨，大部分

> 傳入應用程式的資料，都來自網站表單。但還有很多能把資料傳給應用程式的方法，如：與其他人或其他應用程式共享的資料庫、網站伺服器、遠端主機，甚至還包括 URL 及其參數。

之前提過，如果過不了 `validate_form()` 的檢查，範例 6-8 不會指明表單出錯的地方。範例 6-9 更動了 `validate_form()` 和 `show_form()`，用以控制及呈現備用錯誤訊息的陣列。

範例 6-9 呈現錯誤訊息

```
// 根據隱藏參數 _submit_check，執行正確的邏輯
if ($_POST['_submit_check']) {
    // 如果 validate_form() 回傳錯誤訊息，則傳給 show_form()
    if ($form_errors = validate_form()) {
        show_form($form_errors);
    } else {
        process_form();
    }
} else {
    show_form();
}

// 表單送出後要做的事
function process_form() {
    print "Hello, ". $_POST['my_name'];
}

// 呈現表單外觀
function show_form($errors = '') {
    // 如果有錯誤訊息送入，呈現出來
    if ($errors) {
        print 'Please correct these errors: <ul><li>';
        print implode('</li><li>', $errors);
        print '</li></ul>';
    }

    print<<<_HTML_
<form method="POST" action="$_SERVER[PHP_SELF]">
Your name: <input type="text" name="my_name">
<br/>
<input type="submit" value="Say Hello">
```

範例 6-9 呈現錯誤訊息（續）

```
<input type="hidden" name="_submit_check" value="1">
</form>
_HTML_;
}

// 檢查表單資料
function validate_form() {
    // 以一個錯誤訊息的空陣列起始
    $errors = array();

    // 如果名稱太短則加入一個錯誤訊息
    if (strlen($_POST['my_name']) < 3) {
        $errors[] = 'Your name must be at least 3 letters long.';
    }
    // 回傳錯誤訊息陣列（可能是空的）
    return $errors;
}
```

範例 6-9 的程式碼利用了空陣列會被視為 `false` 的好處。`if ($form_errors = validate_form())` 這串敘述會判斷要呼叫 `show_form()` 或呼叫 `process_form()`。`validate_form()` 回傳的陣列被指定給 `$form_errors`。`if()` 檢測式的邏輯值則如第 3 章的「瞭解邏輯值」一節所示，是被指定的結果。如果 `$form_errors` 內有元素，則 `if()` 檢測式為 `true`；如果是空的，則檢測式為 `false`。如果 `validate_form()` 沒檢查出任何錯誤，回傳的陣列會是空的。

一口氣檢查所有表單組件，比起每次發現錯誤時就重新呈現一次表單要好得多。使用者可以在送出表單後，一口氣找出自己的所有錯誤，不用為了一個新出現的錯誤，反覆送出表單。範例 6-9 的函式 `validate_form()` 在表單組件出現問題時，於 `$errors` 後加上新的元素。接著，`show_form()` 呈現錯誤訊息列表。

本處呈現的核查方式都在 `validate_form()` 內進行。如果表單組件通不過測試，就在 `$_errors` 末端增加一個訊息。

必填元素

以 `strlen()` 檢查長度，可以確認必填元素是否有被輸入，如範例 6-10 所示。

範例 6-10 檢驗必填元素

```
if (strlen($_POST['email']) == 0) {
   $errors[] = "You must enter an email address.";
}
```

在檢查必填元素時，記得用 strlen()，而不是以 if() 測試。如 if (! $_POST['quantity']) 的檢測方式會將 false 值視為錯誤。改用 strlen()，能讓使用者在要求的元素裡輸入 0 這類值。

數字或字串元素

確保傳入的值是整數或小數的方式，需用轉換函式 intval()、floatval()。這兩個函式能提取字串內的數字（整數或小數），不管其他的文字或非數字的格式。

傳入的表單值，應與函式 intval()、floatval() 回傳，再透過 strval() 處理後得到的結果相比較。strval() 函式會把清理過的數字轉換成字串，才能順利與 $_POST 儲存的表單參數比較。如果傳入的字串和清理過的字串無法吻合，表示傳入值帶著陰陽怪氣的東西，程式應該拒絕受理。範例 6-11 展示如何檢查傳入的表單組件是否為整數。

範例 6-11 檢查整數

```
if ($_POST['age'] != strval(intval($_POST['age']))) {
    $errors[] = 'Please enter a valid age.';
}
```

如果 $_POST['age'] 是整數，如 59、0、-32，而 intval($_POST['age']) 回傳值也是那個整數；兩個值相吻合，便不會在 $error 裡加入新的元素。但 $_POST['age'] 如果是 52-pickup，intval($_POST['age']) 卻是 52；兩個值不相等，if() 檢測式便會成立，並在 $error 裡加入新元素。如果 $_POST['age'] 未包含任何數值，intval($_POST['age']) 則會傳回 0。舉例來說，傳入 $_POST['age'] 的值如果是 old，intval($_POST['age']) 就會傳回 0。

範例 6-12 呈現如何使用 floatval() 和 strval() 檢查傳入的值是小數或十進位數。

範例 6-12 檢查小數

```
if ($_POST['price'] != strval(floatval($_POST['price']))) {
    $errors[] = 'Please enter a valid price.';
}
```

floatval() 的運作方式就像 intval()，但它還可以解析小數。範例 6-12 裡，如果 $_POST['price'] 包含小數或整數（如 59.2、12、-23.2），floatval($_POST['price']) 則會等於 $_POST['price']，不會對 $_errors 加入任何元素。但 $_POST['price'] 如果有字元或其他怪東西則會觸發錯誤訊息。

核對元素時（特別是字串元素），用 trim() 函式移除頭尾的留白字元會很有幫助。這個函式可與 strlen() 合併使用，不讓只有空白字元的必填元素過關。trim() 與 strlen() 的結合請見範例 6-13。

範例 6-13 結合 trim() 與 strlen()

```
if (strlen(trim($_POST['name'])) == 0) {
    $errors[] = "Your name is required.";
}
```

如果想在後續程式使用清理過留白的值，則到 $_POST 改變值的內容，並測試改變後的值，如範例 6-14 所示。

範例 6-14 在 $_POST 裡改變值

```
$_POST['name'] = trim($_POST['name']);

if (strlen($_POST['name']) == 0) {
    $errors[] = "Your name is required.";
}
```

因為 $_POST 是個自動全域變數，牽一髮動全身，改變 validate_form() 裡使用的這個變數，將持續到其他在改變後使用 $_POST 的函式，如 process_form()

數值範圍

要檢查數值是否落在某段範圍中時，先確認輸入資料是否為數值。再使用 if() 敘述測試輸入的值，如範例 6-15。

範例 6-15 檢查數值範圍

```
if ($_POST['age'] != strval(intval($_POST['age']))) {
    $errors[] = "Your age must be a number.";
} elseif (($_POST['age'] < 18) || ($_POST['age'] > 65)) {
    $errors[] = "Your age must be at least 18 and no more than 65.";
}
```

測試資料範例前，先將輸入的資料值轉換成電腦紀元（epoch timestamp），再檢查時間是否合適。（關於範例 6-16 用的電腦紀元和 strtotime() 函式，在第 9 章有更詳細的說明。）因為電腦紀元是整數，在時間範圍橫跨月份或年份時，不需什麼特殊動作。範例 6-16 檢查提供的資料是否少於六個月內。

範例 6-16 檢查時間範圍

```
// 取得六個月前的電腦紀元
$range_start = strtotime('6 months ago');
// 取得現在的電腦紀元
$range_end = time();
```

```
// 4 位數紀年，如 2005，存放於 $_POST['yr']
// 2 位數紀月，如 11，存放於 $_POST['mo']
// 2 位數紀日，如 07，存於於 $_POST['dy']
$submitted_date = strtotime($_POST['yr'] . '-' .
                            $_POST['mo'] . '-' .
                            $_POST['dy']);

if (($range_start > $submitted_date) || ($range_end < $submitted_date)) {
     $errors[] = 'Please choose a date less than six months old.';
}
```

電子郵件

檢查電子郵件，是最有商確餘地的常見表單核查任務。因為，沒有確認電子郵件是否有效的捷徑。「有效」（valid）可以表示很多種情況，端視設計者的目標而定。如果真的想確認郵件位址是否有在使用，而且提供郵件位址的人也是該位址的使用者，必需做到兩件事。第一，電子郵件位址傳入的同時，送出一份包含隨機字串的訊息至該位址；或是在訊息內加入一段可由使用者點選的 URL，或是內嵌一段程式碼。如果程式碼能被送出（或 URL 被點選），表示有人收到訊息和並送入同樣的電子郵件位址到網站上（或者說，有人知道並允許送出動作的進行）。

如果不想麻煩地用另一份訊息確認電子郵件，也有些在表單核查中能用的語法，可以排除輸入錯誤的位址。正規式 `^[^@\s]+@([-a-z0-9]+\.)+[a-z]{2,}$`，符合大多數常見的電子郵件位址，而不會與常見的手誤相吻合。範例 6-17 將此正規式與 `preg_match()` 合用。

範例 6-17 檢查電子郵件的語法

```
if (! preg_match('/^[^@\s]+@([-a-z0-9]+\.)+[a-z]{2,}$/i',
                 $_POST['email'])) {
    $errors[] = 'Please enter a valid e-mail address';
}
```

但這個正規式有一點危險，它不允許使用者名稱（@ 前的部分）有空白字元。像 `"Marles Pickens"@sludge.example.com` 這類位址，根據網際網路郵件位址的標準，算是合格的；但它卻無法通過正規式的測試，只因為內含空白字元。幸好，含有留白的郵件位址很少見，大概不會因此惹上麻煩。

下拉式選單

在表單內使用 <select> 下拉式選單，需要確保送出的值，是允許使用的選項之一。雖然市佔率廣、行為良好的瀏覽器（Mozilla、IE 等），不會允許使用者送出選單上沒有的值，但有心攻擊的人就算不用瀏覽器，還是能建構包含任意值的申請訊息。

簡化呈現和查核 <select> 選單的方式，是把選項放入陣列中，以呈現 show_form() 函式裡的 <select> 下拉式選單。在 validate_form() 使用相同的陣列，以確認傳入的值。範例 6-48 展示如何用這項技巧呈現 <select> 下拉式選單。

範例 6-18 呈現 <select> 下拉式選單

```
$sweets = array('Sesame Seed Puff','Coconut Milk Gelatin Square',
                'Brown Sugar Cake','Sweet Rice and Meat');

// 呈現表單外觀
function show_form() {
    print<<<_HTML_
<form method="post" action="$_SERVER[PHP_SELF]">
Your Order: <select name="order">

_HTML_;
foreach ($GLOBALS['sweets'] as $choice) {
    print "<option>$choice</option>\n";
}
print<<<_HTML_
</select>
<br/>
<input type="submit" value="Order">
<input type="hidden" name="_submit_check" value="1">
</form>
_HTML_;
}
```

範例 6-18 的 show_form() 函式將呈現如下 HTML：

```
<form method="post" action="order.php">
Your Order: <select name="order">
<option>Sesame Seed Puff</option>
<option>Coconut Milk Gelatin Square</option>
<option>Brown Sugar Cake</option>
<option>Sweet Rice and Meat</option>
</select>
<br/>
<input type="submit" value="Order">
<input type="hidden" name="_submit_check" value="1">
</form>
```

在 validate_form() 裡運用 <select> 選項陣列的方式如下：

```
if (! in_array($_POST['order'], $GLOBALS['sweets'])) {
    $errors[] = 'Please choose a valid order.';
}
```

如果希望 <select> 下拉式選單有不同的選項值和實際呈現選項，則要用到更複雜的陣列。讓每個陣列元素鍵都是給某個選項的屬性值，相對應的陣列元素值則是實際呈現的選項。範例 6-19 的選項屬性值為 full、square、cake、ricemeat，實際呈現的選擇為 Sesame Seed Puff、Coconut Milk Gelatin Square、Brown Sugar Cake、Sweet Rice and Meat。

範例 6-19 選項和值不同的 <select> 下拉式選單

```
$sweets = array('puff' => 'Sesame Seed Puff',
                'square' => 'Coconut Milk Gelatin Square',
                'cake' => 'Brown Sugar Cake',
                'ricemeat' => 'Sweet Rice and Meat');

// 呈現表單外觀
function show_form() {
    print<<<_HTML_
<form method="post" action="$_SERVER[PHP_SELF]">
Your Order: <select name="order">
_HTML_;
// $val 是選項值， $choice 是實際顯示的文字
foreach ($GLOBALS['sweets'] as $val => $choice) {
    print "<option value=\"$val\">$choice</option>\n";
}
print<<<_HTML_
</select>
<br/>
<input type="submit" value="Order">
<input type="hidden" name="_submit_check" value="1">
</form>
_HTML_;
}
```

範例 6-19 的表單呈現為下列 HTML：

```
<form method="post" action="order.php">
Your Order: <select name="order">
<option value="puff">Sesame Seed Puff</option>
<option value="square">Coconut Milk Gelatin Square</option>
<option value="cake">Brown Sugar Cake</option>
<option value="ricemeat">Sweet Rice and Meat</option>
```

```
</select>
<br/>
<input type="submit" value="Order">
<input type="hidden" name="_submit_check" value="1">
</form>
```

範例 6-19 的下拉式選單傳入的值應為 `puff`、`square`、`cake` 或 `ricemeat`。範例 6-20 以 `validate_form()` 查核相同的範例。

範例 6-20 確認 <select> 下拉式選單傳入的值

```
if (! array_key_exists($_POST['order'], $GLOBALS['sweets'])) {
    $errors[] = 'Please choose a valid order.';
}
```

HTML 與 JavaScript

送出包含 HTML 或 JavaScript 的表單資料，可能會捅出大紕漏。以「訪客留言板」（一種能讓使用者上傳意見並成串顯示在網頁上的程式）為例。如果使用者行為良好，只輸入純文字，留言板可以正常運作。有個使用者送出 `Cool page! I like how you list the different ways to cook fish`，這就是你在留言板上見到的留言。

但在送出的留言不是純文字時，情況就複雜得多了。如果某位熱情的使用者輸入 `This page <b>rules!!!!</b>` 這串留言，而且由留言板應用程式逐字執行，你會在留言板上看到粗體的 `rules!!!!`。瀏覽器區分不出留言板應用程式本身的 HTML 標籤（可能用表格或清單形式排出版面），或剛好嵌在留言板呈現訊息裡的 HTML 標籤。

雖然看到粗體字是個小問題，但未經過濾就呈現使用者輸入的資料，等於大開留言板的門戶，帶來更多頭痛問題。如果不只粗體的 `<b></b>` 標籤，使用者還輸入了會使表單功能失常的標籤，或少輸入結束標籤（如沒有 `"` 或 `>` 的超鏈結 `<a href="`），瀏覽器將無法正確呈現頁面。更糟的事還在後面，輸入的資料可能包含 JavaScript 程式碼（在查看留言板時執行），一段能把個人電腦裡的 cookie 寄給陌生人，或是偷偷摸摸把使用者引導到其他網站的程式。

留言板成為幫兇，讓心懷惡意的使用者上傳 HTML 或 JavaScript，稍後再由毫不知情的使用者以瀏覽器執行。這種問題稱為「跨網站指令碼攻擊（cross-site scripting attack）」。先天不良的留言板程式，讓來自 A 來源（惡意使用者）的程式得以偽裝成來自 B 來源（留言板網站）。

想避免跨網站指令碼攻擊的發生，千萬別呈現未經調整的外部輸入資料。PHP 有兩種方式可簡化防備任務。`strip_tags()` 函式會移除字串裡的 HTML 標籤；`htmlentities()` 能編寫特殊的 HTML 碼。

範例 6-21 示範 strip_tags() 的使用。

範例 6-21 從字串中剝除 HTML 標籤

```
// 從留言裡移入 HTML
$comments = strip_tags($_POST['comments']);
// 現在可以呈現 $comments 了
print $comments;
```

如果 $_POST['comments'] 包含訊息 I love sweet rice & tea.，範例 6-21 的呈現結果是：

```
I love sweet rice & tea.
```

所有 HTML 標籤和屬性都被移除，但標籤間的純文字則被完好無缺地保留下來。

範例 6-22 示範 htmlentities() 的使用。

範例 6-22 編寫在字串裡的 HTML

```
$comments = htmlentities($_POST['comments']);
// 現在可以呈現 $comments 了
print $comments;
```

如果 $_POST['comments'] 包含 I love sweet rice & tea.，範例 6-22 的呈現結果是：

```
I &lt;b&gt;love&lt;/b&gt; sweet &lt;div class=&quot;fancy&quot;
      &gt;rice&lt;/fancy&gt; &amp; tea.
```

在 HTML 裡有特殊意義的字元（<、>、&、"）都被轉換成相等的字元實體：

- < → <
- > → >
- & → &
- " → "

當瀏覽器看到 <，會呈現 < 字元，而不是解讀成：「哦！有個 HTML 標籤要處理。」轉換雙括號字串裡的 " 或 $ 字元，也是相同的概念（只不過語法不同），如同第 2 章「文字」一節所述。圖 6-4 顯示，範例 6-22 在網站瀏覽器上看來會是什麼樣子。

在大多數應用程式裡，應該用 htmlentities() 消毒外來資料。這個函式不會丟棄任何內容，也能防止跨網站指令碼攻擊。舉例來說，一個討論 HTML 的討論板（有「 能做什麼？」這類內容），或討論數學的地方（「如果 x<y，則 2x>z？」），不太適合用 strip_tags() 清除張貼文章。問題的呈現會變成「能做什麼？」或「如果，則？」

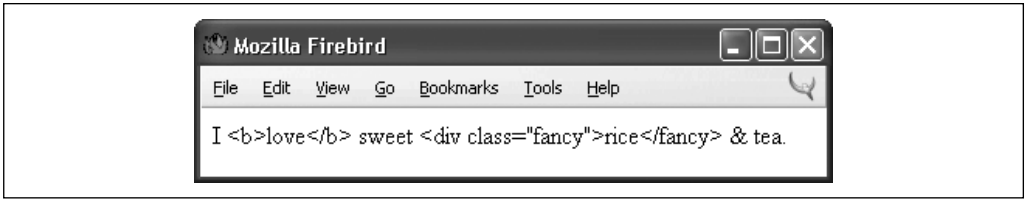


圖 6-4 呈現編寫成實體字元的文字

在語法之外

本章中討論的多數核查策略，目前都是檢查傳入值的語法；確認傳入的資料符合特定的格式。然而，有時候想確認的不只是傳入值的正確語法，還要檢查其意義。`<select>` 下拉式選單的核對，做的就是這件事；不只確認傳入值是字串，還與特定的值相比對。檢查電子郵件時採用的確認信策略，則是另一個不只檢查語法的例證。如果只確認傳入的郵件位址格式正確，愛搗蛋的使用者就能提供 `president@whitehouse.gov` 這種八成不屬於他的郵件位址。確認信能保證郵件位址的含義正確，換句話說，保證「這個郵件位址屬於提供它的使用者」。

呈現預定值

有時候，我們會想呈現帶有預設值的表單組件，組件可能是文字方塊、核取方塊、單選圓鈕、下拉式選單。保留使用者上次輸入的資料，對重載內有錯誤訊息的表單會是極大的幫助。範例 6-23 為達成這項功能的程式碼，屬於 `show_form()` 的一部分，並使陣列 `$defaults` 成為與表單組件一起使用的值。

範例 6-23 建立儲存預設值的陣列

```
if ($_POST['_submit_check']) {
    $defaults = $_POST;
} else {
    $defaults = array('delivery'  => 'yes',
                      'size'      => 'medium',
                      'main_dish' => array('taro','tripe'),
                      'sweet'     => 'cake');
}
```

如果 `$_POST['_submit_check']` 已設定，即為表單已送出之意。本例中，預設值應該是使用者送入的任何值。如果 `$_POST['_submit_check']` 沒有設定，才可採用程式設計者預設的值。大部分表單參數的預設值是字串或數字。對可以有多個值的表單元素而言，預設值是陣列，如複選的 `main_dish` 下拉式選單。

設過預設值後，在 $defaults 裡提供適當的值，以供呈現表單組件的 HTML 標籤利用。記得在需要時以 htmlentities() 重新編排預設值，預防跨網站指令碼攻擊。因為 HTML 標籤結構的關係，將需要以不同的方式處理文字方塊、下拉式選單、多行文字區域 (textarea)，及核取方塊／單選圓鈕。

文字方塊的 <input> 標籤 value 屬性，應設為 $defaults 裡的適當元素。請參考範例 6-24。

範例 6-24 設定文字方塊的預設值

```
print '<input type="text" name="my_name" value="' .
      htmlentities($defaults['my_name']). '">';
```

多行文字區域的話，需在 <textarea> 和 </textarea> 兩個標籤中，放入全都經過編譯過的值，如範例 6-25 所示。

範例 6-25 設定多行文字區域的預設值

```
print '<textarea name="comments">';
print htmlentities($defaults['comments']);
print '</textarea>';
```

下拉式選單則需在呈現 <option> 標籤的迴圈裡，加入選取的項目；<option> 迴圈負責在合適時印出 selected="selected" 屬性。範例 6-26 是在單選的下拉式選單裡執行上述動作的程式碼。

範例 6-26 在下拉式選單裡設定一個預設值

```
$sweets = array('puff' => 'Sesame Seed Puff',
                'square' => 'Coconut Milk Gelatin Square',
                'cake' => 'Brown Sugar Cake',
                'ricemeat' => 'Sweet Rice and Meat');

print '<select name="sweet">';
// $val 是選項值 , $choice 是呈現出的項目。
foreach ($sweets as $option => $label) {
    print '<option value="' .$option .'"';
    if ($option == $defaults['sweet']) {
        print ' selected="selected"';
    }
    print "> $label</option>\n";
}
print '</select>';
```

如果下拉式選單可以複選，需將儲存預設值的陣列，轉換成相對的陣列，陣列轉換後的每個鍵都是選取項目。接著，為每個陣列中存有的選項加上 selected="selected" 屬性。請見範例 6-27。

範例 6-27 為下拉式選單設定多個預設值

```
$main_dishes = array('cuke' => 'Braised Sea Cucumber',
                     'stomach' => "Sauteed Pig's Stomach",
                     'tripe' => 'Sauteed Tripe with Wine Sauce',
                     'taro' => 'Stewed Pork with Taro',
                     'giblets' => 'Baked Giblets with Salt',
                     'abalone' => 'Abalone with Marrow and Duck Feet');

print '<select name="main_dish[]" multiple="multiple">';

$selected_options = array();
foreach ($defaults['main_dish'] as $option) {
    $selected_options[$option] = true;
}

// 列出 <option> 標籤
foreach ($main_dishes as $option => $label) {
    print '<option value="' . htmlentities($option) . '"';
    if (array_key_exists($option, $selected_options))
        print ' selected="selected"';
    }
    print '>' . htmlentities($label) . '</option>';

    print "\n";
}
print '</select>';
```

核取方塊和單選圓鈕的預設值，則是在 `<input>` 標籤裡加上 `checked="checked"` 屬性。用在核取方塊和單選圓鈕的語法幾乎相同，只有 `type` 屬性的差異而已。範例 6-28 呈現預設值名稱為 `delivery` 的核取方塊，以及三個可供預設選取的單選圓鈕，每個圓鈕的名稱均為 `size`，但值彼此相異。

範例 6-28 為核取方塊和單選圓鈕設定預設值

```
print '<input type="checkbox" name="delivery" value="yes";
if ($defaults['delivery'] == 'yes') { print ' checked="checked"'; }
print '> Delivery?';

print '<input type="radio" name="size" value="small";
if ($defaults['size'] == 'small') { print ' checked="checked"'; }
print '> Small ';
print '<input type="radio" name="size" value="medium";
if ($defaults['size'] == 'medium') { print ' checked="checked"'; }
print '> Medium';
print '<input type="radio" name="size" value="large";
if ($defaults['size'] == 'large') { print ' checked="checked"'; }
print '> Large';
```

整合運用

把粗糙的網路表單轉換成多功應用程式，能做資料檢核、呈現預設值、處理送出結果等等事務，看來是件恐怖的差事。為了減輕大家的負荷，本節的完整範例將能執行下列功能：

- 呈現表單，包括預設值
- 核查送入資料
- 再現附有錯誤訊息的表單，並在送入資料未通過核查時，加工處理使用者輸入的資料
- 處理通過核查的送入資料

上述功能的完整範例需依賴輔助函式，以簡化表單組件呈現的部分。函式已列在範例 6-29 內。

範例 6-29 輔助呈現表單組件的函式

```
// 文字方塊
function input_text($element_name, $values) {
    print '<input type="text" name="' . $element_name .'" value="';
    print htmlentities($values[$element_name]) . '"/>';
}

// 送出按鈕
function input_submit($element_name, $label) {
    print '<input type="submit" name="' . $element_name .'" value="';
    print htmlentities($label) .'"/>';
}

// 多行文字區域
function input_textarea($element_name, $values) {
    print '<textarea name="' . $element_name .'">';
    print htmlentities($values[$element_name]) . '</textarea>';
}

// 單選圓鈕或核取方塊
function input_radiocheck($type, $element_name, $values, $element_value) {
    print '<input type="' . $type . '" name="' . $element_name .'"
            value="' . $element_value . '" ';
    if ($element_value == $values[$element_name]) {
        print ' checked="checked"';
    }
    print '/>';
}
```

範例 6-29 輔助呈現表單組件的函式（續）

```
// 下拉式選單
function input_select($element_name, $selected, $options, $multiple = false) {
    // <select> 標籤
    print '<select name="' . $element_name;
    // 如果可複選，加入複選屬性，並在標籤名稱後加上 []
    if ($multiple) { print '[]" multiple="multiple'; }
    print '">';

    // 設定可供選擇的清單
    $selected_options = array();
    if ($multiple) {
        foreach ($selected[$element_name] as $val) {
            $selected_options[$val] = true;
        }
    } else {
    $selected_options[ $selected[$element_name] ] = true;
    }

    // <option> 標籤
    foreach ($options as $option => $label) {
        print '<option value="' . htmlentities($option) . '"';
        if ($selected_options[$option]) {
            print ' selected="selected"';
        }
        print '>' . htmlentities($label) . '</option>';
    }
    print '</select>';
     }
```

範例 6-29 列出的每個輔助應用程式，各自適用「呈現預設值」一節裡提到的相應邏輯，以呈現不同種類的表單元素。因為範例 6-30 的表單程式碼有很多種不同的組件，把重複出現的表單組件放入函式裡，會比徒手複製需要的程式碼簡便得多。

函式 `input_text()` 需要兩個引數：文字組件的名稱，和一個裝載表單組件值的陣列。函式會呈現出 `<input type="text">` 標籤，即單行文字方塊。如果在表單元素值陣列裡有符合文字組件名稱的記錄（entry），該記錄則為 `<input type="text">` 的屬性值；值裡的特殊字元都要以 `htmlentities()` 先行編碼。

函式 `input_submit()` 會呈現 `<input type="submit">` 標籤，即送出按鈕。本函式需要兩個引數：按鈕名稱，和按鈕上出現的標籤名稱。

函式 input_textarea() 需要兩個引數，跟 input_text() 一樣，包括：組件名稱，和表單組件值陣列。不過這個函式不會呈現單行文字方塊，而是多行文字區域所用的 <textarea></textarea> 標籤。如果在表單組件值陣列裡有符合組件名稱的記錄，該記錄則為多行文字區域的預設值。在值裡的特殊字元都要以 htmlentities() 先行編碼。

單選圓鈕和核取方塊都由 input_radiocheck() 處理。這個函式的第一個引數可以是 radio（單選圓鈕）或 checkbox（核取方塊），用來決定函式會呈現哪個標籤，<input type="radio"> 或 <input type="checkbox">。接下來的引數是組件名稱、表單組件值陣列，和整個組件的值。需要傳入完整的表單組件值陣列和指定組件的值，函式才知道給組件使用的陣列記錄是否符合傳入的值。例如範例 6-30 呈現的三個單選圓鈕，名稱均為 size，值各自相異（small、medium、large）。只有一個圓鈕可設有 checked="checked" 屬性：符合傳入值的表單組件值記錄。

函式 input_select() 呈現下拉式選單。本函式需要三個引數：組件名稱、表單組件值陣列、選項清單陣列。還可以把 true 當做第四個引數，開啟選取多個選項的功能。函式會用範例 6-26 和 6-27 的邏輯，為每個可標上 selected="selected" 屬性的選項，建立 $selected_options 陣列與一項記錄。然後，函式會重複讀取 $option 陣列，為每個選項加上 <option></option> 標籤。

範例 6-30 的程式碼依賴表單輔助函式，並呈現一份簡短的點餐表單。只要送出正確的表單資料，程式便會在瀏覽器上顯示結果，同時寄出電子郵件給 process_form() 函式裡定義的位址（假定是廚師的郵件信箱，讓廚師能動手準備餐點）。因為程式碼在 PHP 模式裡出出入入，所以在範例開頭處有起始標籤 <? php，在結尾處有 ?> 結束標籤，讓程式更易閱讀。

範例 6-30 完整的表單：呈現預設值、核查資料、處理資料

```
<?php
// 別忘記納入範例 6-29 定義過的表單輔助函式
//
// 下拉式選單的選項陣列設在全域範圍內，因為陣列將會用在：
// display_form()、validate_form()、process_form()
$sweets = array('puff' => 'Sesame Seed Puff',
                'square' => 'Coconut Milk Gelatin Square',
                'cake' => 'Brown Sugar Cake',
                'ricemeat' => 'Sweet Rice and Meat');

$main_dishes = array('cuke' => 'Braised Sea Cucumber',
                     'stomach' => "Sauteed Pig's Stomach",
                     'tripe' => 'Sauteed Tripe with Wine Sauce',
                     'taro' => 'Stewed Pork with Taro',
                     'giblets' => 'Baked Giblets with Salt',
                     'abalone' => 'Abalone with Marrow and Duck Feet');
```

範例 6-30 完整的表單：呈現預設值、核查資料、處理資料（續）

```
// 主頁面邏輯：
// 如果表單送出，核查資料後，處理或重載頁面；
// 如果表單未送出，則呈現如下頁面
if ($_POST['_submit_check']) {
    // 如果 validate_form() 回傳錯誤，則傳給 show_form()
    if ($form_errors = validate_form()) {
        show_form($form_errors);
    } else {
        // 如果傳入的資料通過核查，則開始處理
        process_form();
    }
} else {
    // 表單尚未送出，則呈現表單
    show_form();
}

function show_form($errors = '') {
    // 如果表單已送出，從送出的參數裡取得預設值
    if ($_POST['_submit_check']) {
        $defaults = $_POST;
    } else {

        // 否則，改設為自訂的預設值：中等份、要快遞
        $defaults = array('delivery' => 'yes',
                          'size'     => 'medium');
    }

    // 如果傳入錯誤訊息，則放入 $error_text（包括 HTML 標記）
    if ($errors) {
        $error_text = '<tr><td>You need to correct the following errors:';
        $error_text .= '</td><td><ul><li>';
        $error_text .= implode('</li><li>',$errors);
        $error_text .= '</li></ul></td></tr>';
    } else {
        // 沒有錯誤嗎？$error_text 會是空的
        $error_text = '';
    }

    // 離開 PHP 模式，讓標上 HTML 標籤容易點
?>
<form method="POST" action="<?php print $_SERVER['PHP_SELF']; ?>">
<table>
<?php print $error_text ?>
```

```
<tr><td>Your Name:</td>
<td><?php input_text('name', $defaults) ?></td></tr>

<tr><td>Size:</td>
<td><?php input_radiocheck('radio','size', $defaults, 'small');  ?> Small <br/>
<?php input_radiocheck('radio','size', $defaults, 'medium'); ?> Medium <br/>
<?php input_radiocheck('radio','size', $defaults, 'large');  ?> Large
</td></tr>

<tr><td>Pick one sweet item:</td>
<td><?php input_select('sweet', $defaults, $GLOBALS['sweets']); ?>
</td></tr>

<tr><td>Pick two main dishes:</td>
<td>
<?php input_select('main_dish', $defaults, $GLOBALS['main_dishes'], true) ?>
</td></tr>

<tr><td>Do you want your order delivered?</td>
<td><?php input_radiocheck('checkbox','delivery', $defaults, 'yes'); ?> Yes
</td></tr>

<tr><td>Enter any special instructions.<br/>
If you want your order delivered, put your address here:</td>
<td><?php input_textarea('comments', $defaults); ?></td></tr>

<tr><td colspan="2" align="center"><?php input_submit('save','Order'); ?>
</td></tr>

</table>

<input type="hidden" name="_submit_check" value="1"/>
</form>
<?php
      } // show_form() 結束

function validate_form() {
    $errors = array();

    // 需要名稱
    if (! strlen(trim($_POST['name']))) {
        $errors[] = 'Please enter your name.';
    }
    // 需要餐點份量
    if (($_POST['size'] != 'small') && ($_POST['size'] != 'medium') &&
```

範例 6-30 完整的表單：呈現預設值、核查資料、處理資料（續）

```
            ($_POST['size'] != 'large')) {
            $errors[] = 'Please select a size.';
        }
        // 需要甜度
        if (! array_key_exists($_POST['sweet'], $GLOBALS['sweets'])) {
            $errors[] = 'Please select a valid sweet item.';
        }
        // 需要明確指定兩道主菜
        if (count($_POST['main_dish']) != 2) {
            $errors[] = 'Please select exactly two main dishes.';
        } else {
            // 已經選擇了兩道主菜，再來進行核查
            if (! (array_key_exists($_POST['main_dish'][0], $GLOBALS['main_dishes']) &&
                   array_key_exists($_POST['main_dish'][1], $GLOBALS['main_dishes']))) {
                $errors[] = 'Please select exactly two valid main dishes.';
            }
        }
        // 如果勾選外送，需填寫住址
        if (($_POST['delivery'] == 'yes') && (! strlen(trim($_POST['comments'])))) {
            $errors[] = 'Please enter your address for delivery.';
        }

        return $errors;
    }

    function process_form() {
        // 到陣列 $GLOBALS['sweets'] 和 $GLOBALS['main_dishes']
        // 尋找甜點和主菜的全名
        $sweet = $GLOBALS['sweets'][ $_POST['sweet'] ];
        $main_dish_1 = $GLOBALS['main_dishes'][ $_POST['main_dish'][0] ];
        $main_dish_2 = $GLOBALS['main_dishes'][ $_POST['main_dish'][1] ];
        if ($_POST['delivery'] == 'yes') {
            $delivery = 'do';
        } else {
            $delivery = 'do not';
        }
        // 建立點餐訊息

        $message=<<<_ORDER_
    Thank you for your order, $_POST[name].
    You requested the $_POST[size] size of $sweet, $main_dish_1, and $main_dish_2.
    You $delivery want delivery.
    _ORDER_;
        if (strlen(trim($_POST['comments']))) {
```

```
            $message .= 'Your comments: '.$_POST['comments'];
        }

        // 把點餐訊息寄給廚師
        mail('chef@restaurant.example.com', 'New Order', $message);
        // 列出訊息，但事先編排所有 HTML 實體，把換行轉換成 <br/>
        print nl2br(htmlentities($message));
    }
    ?>
```

範例 6-30 的程式碼共有四個部分：頂端的全域作用程式碼、函式 `show_form()`、函式 `validate_form()`、函式 `process_form()`。

全域作用範圍的程式碼執行兩件事。第一件是設定描述兩個下拉式選單所需選項的兩個陣列，這兩個陣列會被 `show_form()`、`validate_form()`、`process_form()` 函式運用，所以必需定義在全域作用範圍。全域程式碼的另一項任務是處理 `if()` 敘述，決定要呈現表單、核查資料，或處理表單。

呈現表單的任務由 `show_form()` 執行。首先，函式會製造持有預設值的陣列 `$defaults`。如果表單已經送出，而且被重新下載的話，預設值則來自 `$_POST`。否則，各預設值會分別被設定：是否外送（`delivery`）的核取方塊設為 `yes`、決定份量（`size`）的單選圓鈕被設為 `medium`。接下來，`show_form()` 會列出傳入的任何錯誤訊息。HTML 列出的錯誤訊息是由 `$errors` 陣列架構而成，使用與範例 4-21 的 `implode()` 相似的技巧。再來，`show_form()` 跳出 PHP 模式，呈現表單。在 HTML 的表格標籤中，又重複回到 PHP 模式，呼叫能適當呈現出各個表單組件的輔助函式。表單結尾處隱藏的 `_submit_check` 組件，則不需以輔助函式呈現。

在送出的資料不符合規則時，`validate_form()` 函式負責建立儲存錯誤訊息的陣列。請注意檢查餐點份量、甜點，和主餐的部分，不只是檢查有無參數送入而已，還檢查送入的資料是否為合格的參數值。以餐點份量而言，參數必為 `small`、`medium`、`large` 之一。以甜點和主菜而為，參數必為全域陣列 `$sweets` 或 `$main_dishes` 的陣列鍵。即使表單帶有預設值，還是要核查輸入資料。有心潛入他人網站的人，可以迂迴避開一般網路瀏覽器，並構成一段申請（request），送入不在下拉式選單或單選圓鈕項目內的任意值。

最後一個函式 `process_form()`，在送入合格資料時發生動作。函式會建立一個字串 `$message`，內含傳入命令的敘述。然後函式會寄送 `$message` 給 `chef@restaurant.example.com`。內建的 `mail()` 函式負責送出電子郵件。（關於 `mail()`，請參考第 13 章的「收發電子郵件」一節。）在列出 `$message` 前，函式 `process_form()` 會將它傳送給另外兩個函式：一個是 `htmlentities()`，用來重新編碼任何 HTML 的特殊字元實體；一個是 `nl2br()`，用來把 `$message` 裡的任何換行字元，轉換成 `<br>` 標籤，讓訊息換行的模式也能在瀏覽器上正常顯示。

章節摘要

第 6 章包括：

- 瞭解瀏覽器和網站伺服器之間的對話，對話用來呈現表單、處理送出的表單參數，最後呈現處理結果。
- 建立表單 `<form>` 標籤的 `action` 屬性，和表單參數傳入的 URL 間的連結。
- 使用來自 `$_SERVER` 自動全域陣列的值。
- 存取 `$_GET` 和 `$_POST` 自動全域陣列裡的表單參數。
- 存取多重值的表單參數。
- 使用函式 `show_form()`、`validate_form()`、`process_form()` 模式化表單處理過程。
- 使用隱藏表單組件，以確認表單是否經過「送出」。
- 連同表單一起呈現錯誤訊息。
- 核查表單組件：必填組件、整數、小數、字串、日期範圍、郵件位址、下拉式選單。
- 在執行 HTML 和 JavaScript 前，先清除除潛在的危機。
- 呈現表單組件的預設值。
- 使用輔助函式來呈現表單組件。

習題

1. 在選擇 `Braised Noodles` 的第三項，`Sweet` 的最後一項，並在文字方塊鍵入 4 後，送出下列表單內容。`$_POST` 將有何面貌？

```
<form method="POST" action="order.php">
Braised Noodles with: <select name="noodle">
<option>crab meat</option>
<option>mushroom</option>
<option>barbecued pork</option>
<option>shredded ginger and green onion</option>
</select>
<br/>
Sweet: <select name="sweet[]" multiple>
<option value="puff"> Sesame Seed Puff
```

```
<option value="square"> Coconut Milk Gelatin Square
<option value="cake"> Brown Sugar Cake
<option value="ricemeat"> Sweet Rice and Meat
</select>
<br/>
Sweet Quantity: <input type="text" name="sweet_q">
<br/>
<input type="submit" name="submit" value="Order">
</form>
```

2. 撰寫能呈現所有送入表單參數及值的 `process_form()` 函式。可以假設表單參數只有純量值。

3. 撰寫執行基礎運算的程式。能輸入運算元的文字方塊，能選擇運算符號的下拉式選單：加、減、乘、除。核查輸入數據是否為數字，是否適合所選的運算符號。處理結果的函式應該呈現運算元、運算子，和運算結果。例如，運算元是 4 和 2，運算子是乘號時，函式應該呈現出 "4 * 2 = 8"。

4. 撰寫呈現、核查並處理船運貨物的表單程式。表單應包含填寫送貨人和收貨人地址、貨物尺寸、貨物重量的欄位。核查功能（至少）該檢查貨物的重量是否大於 150 磅，貨物的長寬高均不得大於 36 吋。可以假設輸入表單的地址都採美式規則，但是必需核對州名和郵遞區號的語法是否正確。處理結果的函式應呈現經過組織、格式化的貨物報告。

5. 調整第二題的 `process_form()` 函式，讓函式正確地處理帶有陣列值的表單參數。請記得，陣列內部還可以包含陣列。