Interacting with the User 18

Introduction

Unless programs are able to interact with users their use is very limited. PHP is able to interact with a user via a Web page by using forms. Forms allow data to be entered and submitted to the server for processing by PHP. Forms manifest themselves in all sorts of shapes and sizes. Some forms are easy to identify as being forms whereas others are less obvious. In this chapter we shall introduce the basic form elements and illustrate how they can be combined to enable quite complex interactions.

PHP and Forms

Forms are part of HTML and not part of PHP. While PHP can output forms in the same way that it can output all other HTML elements it is not adding anything new to the functionality offered by standard HTML. PHP can, however, be used to control forms and process the data which the user provides by interacting with the form, which is something that HTML cannot do. As PHP is a server-side scripting language form data must be transmitted back to the server for processing and then the output generated be transmitted back to the user. This is different from the way that JavaScript can be used, where the processing can be done on the client's machine within the browser.

A Simple Form

We shall begin by creating a standard HTML form and illustrating some of the data entry elements. Consider the following script:

```
<html>

<!- Interacting with the user - Example 18-1 ->
<!- ----------------------------------- ->
```

```
<body>
      <h2>Please enter your personal details:</h2>
      <form action='example18-2.php' method='post'>
      Firstname: <input type='text' name='firstname'>
      <br>
      Surname: <input type='text' name='surname'>
      <br>
      Username: <input type='text' name='username'>
      <br>
      Password: <input type='password' name='password'>
      <br>
      <input type='submit'>
      </form>
</body>
</html>
```

Note that this script is a HTML document and not a PHP script and thus it should be saved with a .html extension. The script illustrates the basic elements of a HTML form. Firstly, the form begins with a <form> element, which has two key attributes: action and method. Attribute action specifies where the form data is to be sent for processing. This can be a CGI application written in any programming language or it could be a PHP script. The method attribute specifies how the data from the form will be sent to the application. There are two main methods, POST and GET. PHP supports POST so we have to use this. In our example the form element specifies that the script to process the form data is called example18-2.php, but we haven't written this yet:

```
<form action='example18-2.php' method='post'>
..
</form>
```

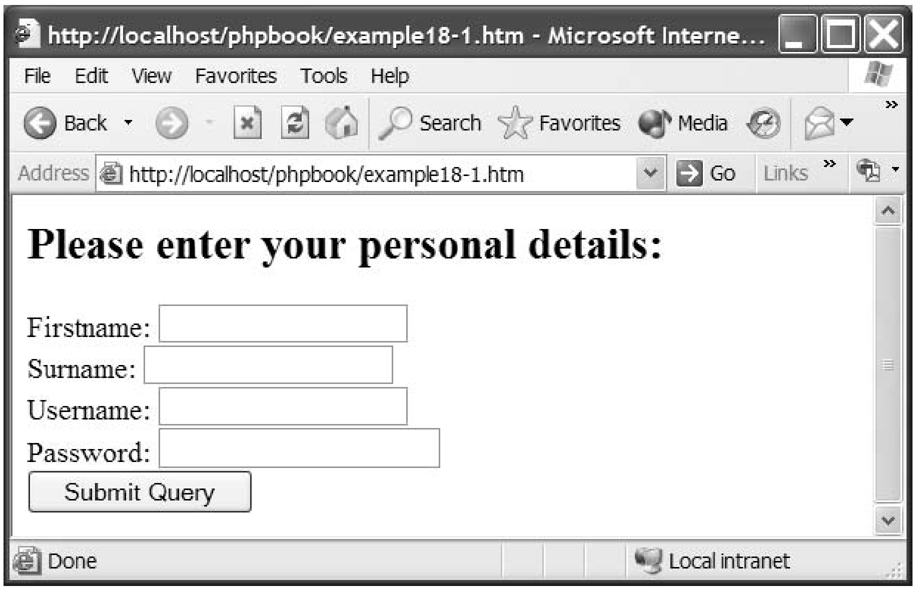
The rest of the form consists of four data entry fields, three of which are type text and one is of type password:

```
        Firstname: <input type='text' name='firstname'>
        <br>
        Surname: <input type='text' name='surname'>
        <br>
        Username: <input type='text' name='username'>
        <br>
        Password: <input type='password' name='password'>
        <br>
```

These are essentially the same type of data entry fields, accept that password fields hide the data that is typed into them and display "*" characters instead. Note that each data input field has a separate field name, which has been chosen to represent the data that each element represents. These form field names are important, as we shall need them to access the form data later in our PHP script. The form concludes with a input field of type submit:

```
<input type='submit'>
```

This field is required in order for the user to submit the form data once they have completed the form. The output from the above HTML script is shown in Figure 18.1. Essentially, this is as far as HTML goes with forms. While, we shall see later in this chapter that there are a few other HTML form field types, if we want to do anything with the form data then we will need to begin to create a PHP script to handle the data entered on the form.


http://localhost/phpbook/example18-1.htm - Microsoft Interne...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-1.htm Go Links
Done
Local intranet

Please enter your personal details:

Firstname:
Surname:
Username:
Password:
Submit Query

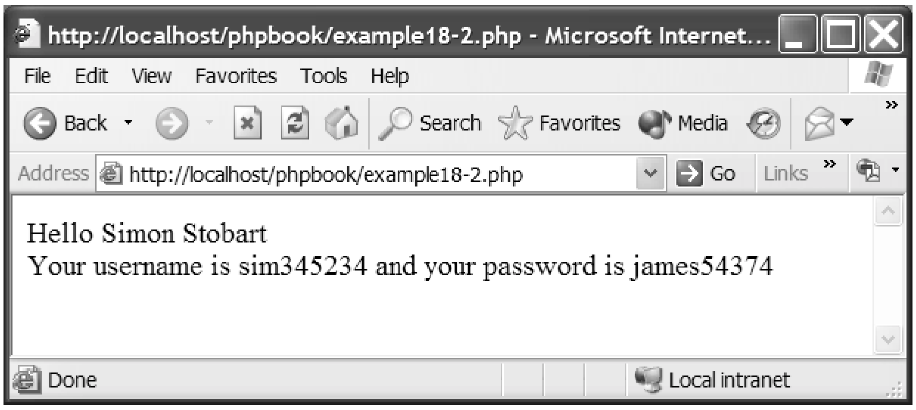
Figure 18.1 A simple form.

We shall now create a PHP script to handle the form data passed to it. The script is very simple as all it does is echo out the data it receives, as shown in Figure 18.2:

```
<?php

// Interacting with the user - Example 18-2
//---------------------------------


$firstname = $_POST["firstname"];
$surname = $_POST["surname"];
$username = $_POST["username"];
$password = $_POST["password"];

echo("Hello $firstname $surname<br>");
echo("Your username is $username and your password is $password");
?>
```

Accessing form data is essentially very easy. When a form is created an associated array of variables is created. This array is called $_POST. Each HTML form element that has a unique name is stored in the array and can be accessed by the PHP script. To access, for example, the form element "firstname" we use the syntax:

```
$firstname = $_POST["firstname"];
```


http://localhost/phpbook/example18-2.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-2.php Go Links
Done Local intranet

```
Hello Simon Stobart
Your username is sim345234 and your password is james54374
```

Figure 18.2 Form data output.

Combining Forms and PHP

In our previous example the HTML form and the PHP script were written as two separate files. The first one contained the HTML form and the second the PHP script to process the data. This is not the best way of implementing this as it is difficult to refer back to the form if we need the user to re-enter some new data. The best method is to combine the form and the PHP processing together in the same script, as shown below:

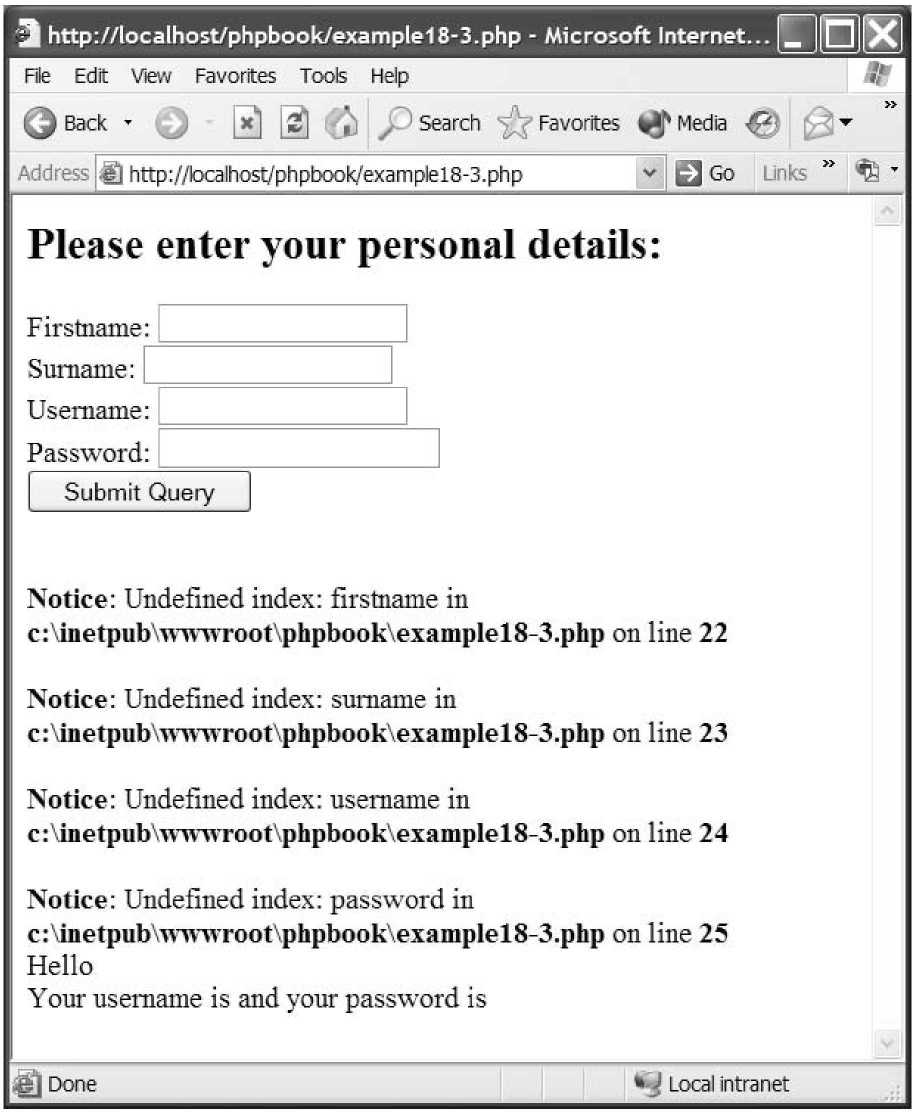
```
<html>
<body>
    <h2>Please enter your personal details:</h2>
    <form action='example18-3.php' method='post'>
    Firstname: <input type='text' name='firstname'>
    <br>
    Surname: <input type='text' name='surname'>
    <br>
    Username: <input type='text' name='username'>
    <br>
    Password: <input type='password' name='password'>
    <br>
    <input type='submit'>
    </form>
```

```
</body>
</html>
<?php

// Interacting with the user - Example 18-3
//-----------------------------------

$firstname = $_POST["firstname"];
$surname = $_POST["surname"];
$username = $_POST["username"];
$password = $_POST["password"];

echo("Hello $firstname $surname<br>");
echo("Your username is $username and your password is $password");
?>
```

The above script combines the HTML form and the PHP script together into one single place. However, there is a problem as illustrated in Figure 18.3.

The problem is that when the script is processed the form is output first then the PHP echo statements are processed. This is not a problem after the first time the form is submitted but the first time the page is loaded the form variables contain no data and this results in the display of an incomplete message. What we need is a means of determining if the form data was submitted or not. Consider the following script:

```
<html>
<body>
   <h2>Please enter your personal details:</h2>
   <form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>
   Firstname: <input type='text' name='firstname'>
   <br>
   Surname: <input type='text' name='surname'>
   <br>
   Username: <input type='text' name='username'>
   <br>
   Password: <input type='password' name='password'>
   <br>
   <input type='submit' name='submit'>
   </form>
</body>
</html>
<?php

// Interacting with the user - Example 18-4
//-----------------------------------
```


http://localhost/phpbook/example18-3.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-3.php Go Links
Submit Query
Done
Local intranet

Please enter your personal details:

Firstname:
Surname:
Username:
Password:

```
Notice: Undefined index: firstname in
c:\inetpub\wwwroot\phpbook\example18-3.php on line 22

Notice: Undefined index: surname in
c:\inetpub\wwwroot\phpbook\example18-3.php on line 23

Notice: Undefined index: username in
c:\inetpub\wwwroot\phpbook\example18-3.php on line 24

Notice: Undefined index: password in
c:\inetpub\wwwroot\phpbook\example18-3.php on line 25
Hello
Your username is and your password is
```

Figure 18.3 The form problem.

```
if(isset($_POST["submit"])){
   $firstname = $_POST["firstname"];
   $surname = $_POST["surname"];
   $username = $_POST["username"];
   $password = $_POST["password"];

   echo("Hello $firstname $surname<br>");
   echo("Your username is $username and your password is $password");
}
?>
```

This is a rewrite of the previous script and solves the issue of displaying blank variable values when the form has not been submitted. The first change in the program has nothing to do with this problem, but makes life far easier for the developer. The change replaces the name of the program which is invoked with the variable PHP_SELF:

```
<?php echo $_SERVER["PHP_SELF"]; ?>
```

This is a server variable (introduced in Chapter 9) and contains the name of the current script and by including this ensures that the script will always call itself, no matter what name it has been saved as. The next change involves providing the submit button with a name:

```
<input type='submit' name='submit'>
```

Here, the name has been set to "submit". Finally, an if statement has been included to check the result of invoking the function isset(), which was first mentioned in Chapter 9. Function isset() has the format:

Function prototype:

```
Int isset(mixed value);
```

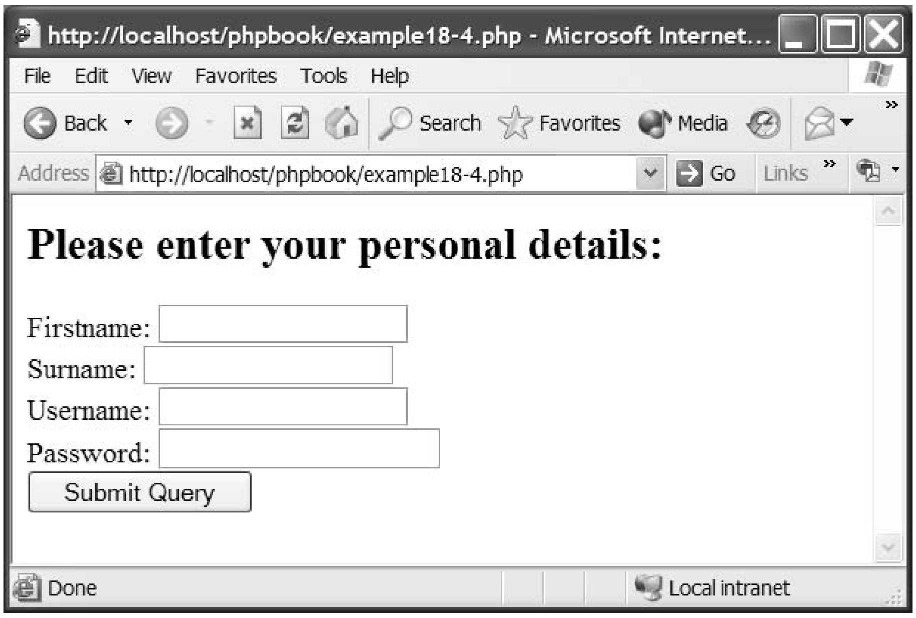
Function arguments and return details:

Name	Type	Description
value	Mixed	Variable to check if exists or not
isset() returns	Int	TRUE if the variable exists and FALSE if not

Function example:

```
$set = isset($_POST["submit"]);
```

The function requires a variable to be passed to it, which it then determines if this has been set or not. If the variable is set it returns a TRUE value, otherwise it returns a FALSE value. The program checks that the value of variable $submit is set. This is only true if the form has been submitted, thus solving our problem, as illustrated in Figure 18.4.


http://localhost/phpbook/example18-4.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-4.php Go Links
Done Local intranet

Please enter your personal details:

Firstname:
Surname:
Username:
Password:
Submit Query

Figure 18.4 The form problem solved.

Minimum Coin Calculator

So far our form examples have been very basic. Let's see if we can create something that is a little more interesting. The following script illustrates a very simple form combined with a "useful" function. The script allows a user to enter a value via a form field and submit this. The value represents a price. For example 134 represents 1 pound and 34 pence and 56 represents 56 pence. The script calculates the minimum number of coins that would be required to present the amount entered via the form. The program uses the images (saved in the graphics directory) illustrated in Table 18.1.

Table 18.1 Coin images and filenames.

1.jpg	2.jpg	5.jpg	10.jpg	18.jpg	50.jpg	100.jpg	180.jpg

Take a look at the script which accomplishes this:

```
<h2>Minimum Coin Calculator</h2>
<form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>
Enter Amount: <input type='text' name='amount'>
```

```
<input type='submit'>
</form>

<?php

// Interacting with the user - Example 18-5
//---------------------------------


if(isset($_POST["amount"]))
        coins($_POST["amount"]);


function coins($amount) {
        while($amount >= 200){
              $amount=$amount-200;
              echo("<img src='graphics/200p.jpg'>");
        } // while
        while($amount >= 100){
              $amount=$amount-100;
              echo("<img src='graphics/100p.jpg'>");
        } // while
        while($amount >= 50){
              $amount=$amount-50;
              echo("<img src='graphics/50p.jpg'>");
        } // while
        while($amount >= 20){
              $amount=$amount-20;
              echo("<img src='graphics/20p.jpg'>");
        } // while
        while($amount >= 10){
              $amount=$amount-10;
              echo("<img src='graphics/10p.jpg'>");
        } // while
        while($amount >= 5){
              $amount=$amount-5;
              echo("<img src='graphics/5p.jpg'>");
        } // while
        while($amount >= 2){
              $amount=$amount-2;
              echo("<img src='graphics/2p.jpg'>");
        } // while
        if($amount > 0){
              echo("<img src='graphics/1p.jpg'>");
        }
}
?>
```

The script begins by outputting a simple form which consists of a text input field and a submit button:

```
<h2>Minimum Coin Calculator</h2>
<form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>
Enter Amount: <input type='text' name='amount'>
<input type='submit'>
</form>
```

The PHP script begins by determining if an amount has been entered via the form, by checking the $amount variable. If this is set then function coins() is invoked passing it the amount entered on the form:

```
<?php

if(isset($_POST["amount"]))
        coins($_POST["amount"]);
```

The function coins() calculates which coin images to display. It consists of a number of while loops, which subtracts the value of the current coin being processed from the total amount. The function is designed to subtract the higher coin values first and thus calculate the minimum number of coins required to equal the value entered:

```
function coins($amount) {
        while($amount >= 200){
              $amount=$amount-200;
              echo("<img src='graphics/200p.jpg'>");
        } // while
        while($amount >= 100){
              $amount=$amount-100;
              echo("<img src='graphics/100p.jpg'>");
        } // while
        while($amount >= 50){
              $amount=$amount-50;
              echo("<img src='graphics/50p.jpg'>");
        } // while
        while($amount >= 20){
              $amount=$amount-20;
              echo("<img src='graphics/20p.jpg'>");
        } // while
        while($amount >= 10){
              $amount=$amount-10;
              echo("<img src='graphics/10p.jpg'>");
        } // while
```

```
          while($amount >= 5){
                $amount=$amount-5;
                echo("<img src='graphics/5p.jpg'>");
          } // while
          while($amount >= 2){
                $amount=$amount-2;
                echo("<img src='graphics/2p.jpg'>");
          } // while
          if($amount > 0){
                echo("<img src='graphics/1p.jpg'>");
          }
  }
  ?>
```

When first run, the output from the above program is as illustrated in Figure 18.5. If the user enters a value in the form field and clicks the "submit" button the program calculates the minimum number of coins, which would be required to make up that amount. For example, typing 99 (representing 99p) results in the output shown in Figure 18.6.

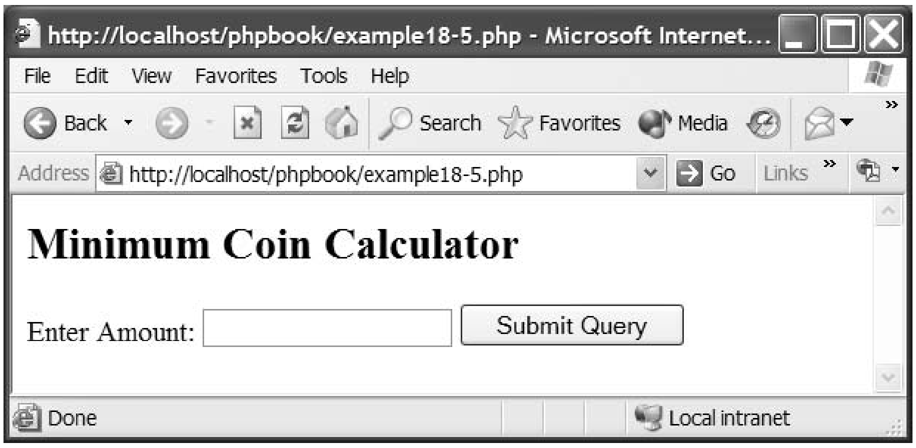
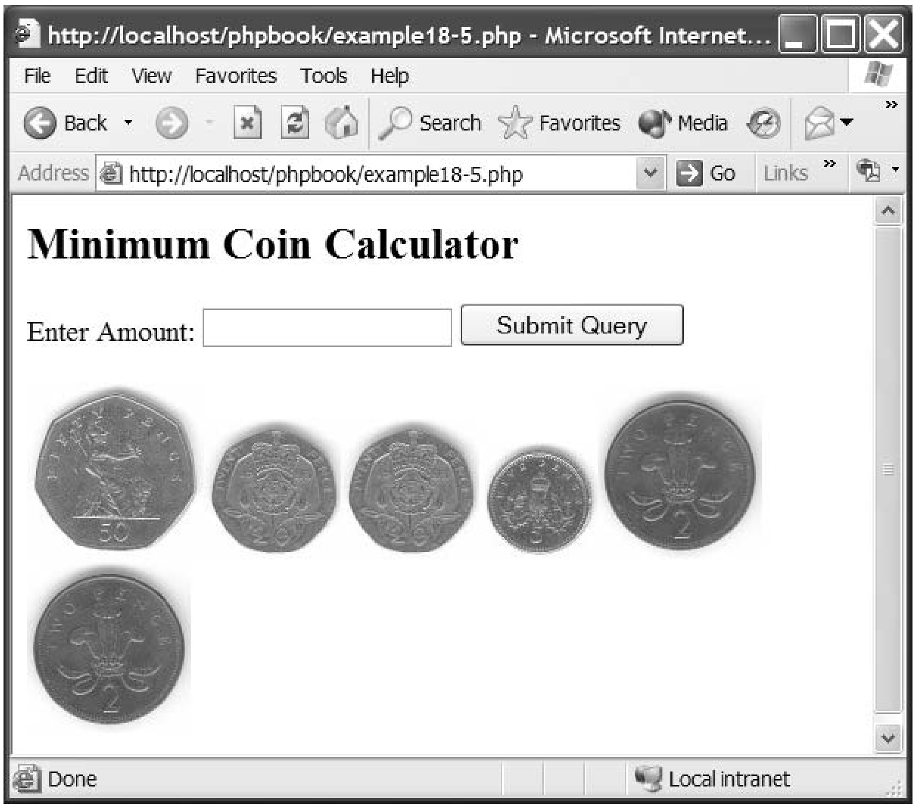

http://localhost/phpbook/example18-5.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-5.php Go Links
Minimum Coin Calculator
Enter Amount:
Submit Query
Done
Local intranet

Figure 18.5 The coin calculator user input.

Radio Buttons

In addition to the text, password and submit form fields we have introduced thus far there are a number of other types. The first of these is the radio type. The radio entry type does not allow the user to enter any text but provides a series of "radio buttons" from which the user can make a selection. Only one of the buttons grouped together can be selected. In order to ensure that the form knows which buttons form a group all grouped buttons must share the same name. An example of the radio type is shown in the script below:


http://localhost/phpbook/example18-5.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address http://localhost/phpbook/example18-5.php
Go
Links
Done
Local intranet

Minimum Coin Calculator

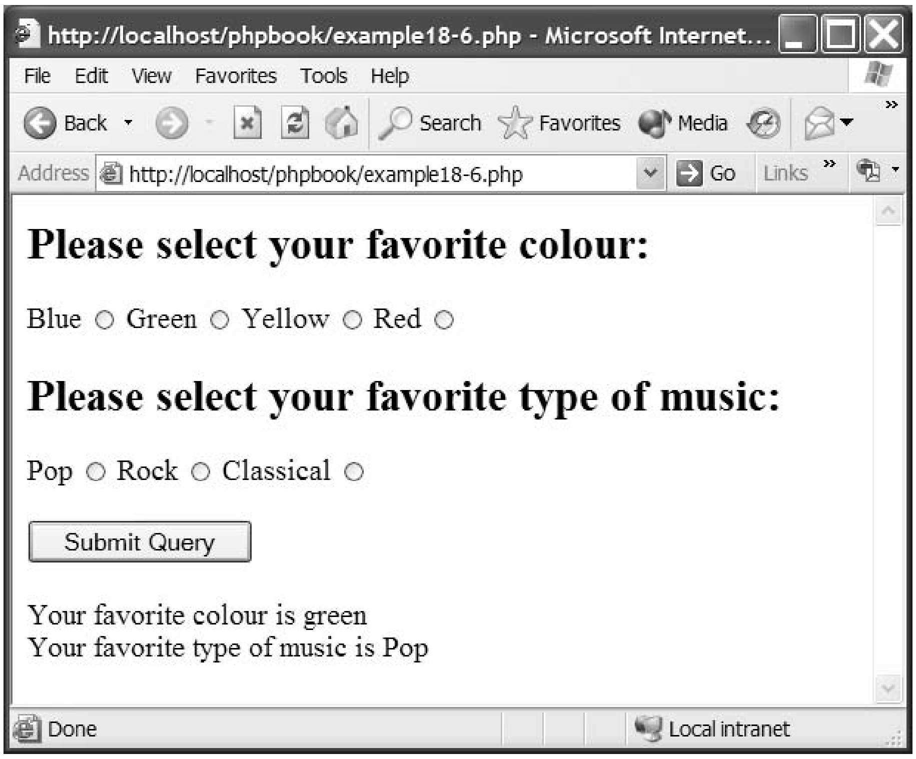
Enter Amount: Submit Query

Figure 18.6 The coin calculator output.

```
    <form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>
    <h2>Please select your favorite color:</h2>
    Blue <input type='radio' name='col' value='blue'>
    Green <input type='radio' name='col' value='green'>
    Yellow <input type='radio' name='col' value='yellow'>
    Red <input type='radio' name='col' value='red'>
    <br>
    <h2>Please select your favorite type of music:</h2>
    Pop <input type='radio' name='mus' value='Pop'>
    Rock <input type='radio' name='mus' value='rock'>
    Classical <input type='radio' name='mus' value='classical'>
    <br><br>
    <input type='submit' name='submit'>
    </form>

<?php
```

```
// Interacting with the user - Example 18-6
//----------------------------------

if(isset($_POST["submit"])){
    $col = $_POST["col"];
    $mus = $_POST["mus"];
    echo("Your favorite color is $col<br>");
    echo("Your favorite type of music is $mus");
}
?>
```

The script displays two sets of radio buttons, one given the name "col" and the other "mus". A check is made to determine if the submit button has been pressed and if so the values stored in variables $col and $mus are displayed. The output from running the above script is shown in Figure 18.7.


http://localhost/phpbook/example18-6.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-6.php Go Links
Done
Local intranet

Please select your favorite colour:

Blue ○ Green ○ Yellow ○ Red ○

Please select your favorite type of music:

Pop ○ Rock ○ Classical ○

Submit Query

```
Your favorite colour is green
Your favorite type of music is Pop
```

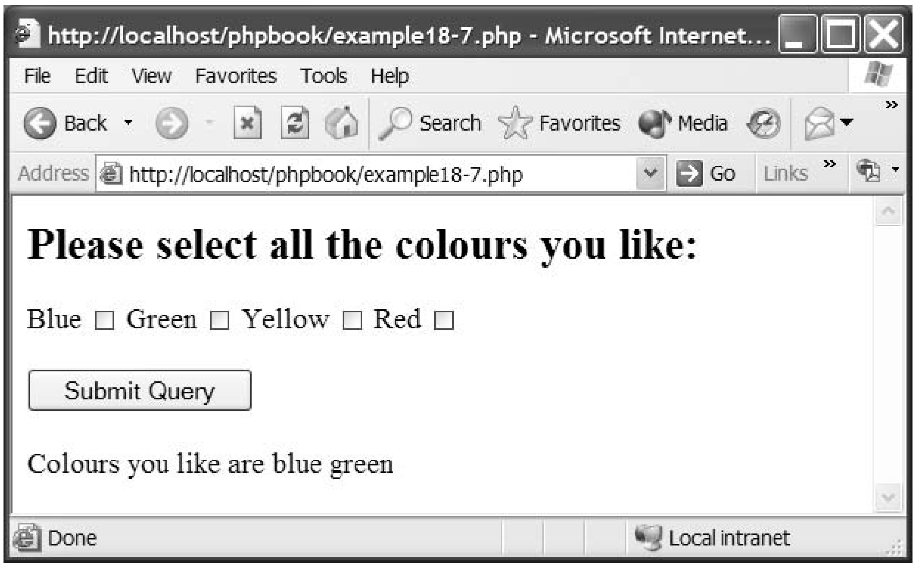
Figure 18.7 Radio buttons.

Checkbox Fields

Checkbox fields enable the user to select as many options as they like from the form. Each checkbox is given a unique name. An example of the use of the form field is shown in the script below:

```
    <form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>
    <h2>Please select all the colors you like:</h2>
    Blue <input type='checkbox' name='cb1' value='blue'>
    Green <input type='checkbox' name='cb2' value='green'>
    Yellow <input type='checkbox' name='cb3' value='yellow'>
    Red <input type='checkbox' name='cb4' value='red'>
    <br><br>
    <input type='submit' name='submit'>
    </form>

<?php

// Interacting with the user - Example 18-7
//-----------------------------------

if(isset($_POST["submit"])){
    if (isset($_POST["cb1"]))
          $cb1 = $_POST["cb1"];
    else
          $cb1 = "";
    if (isset($_POST["cb2"]))
          $cb2 = $_POST["cb2"];
    else
          $cb2 = "";
    if (isset($_POST["cb3"]))
          $cb3 = $_POST["cb3"];
    else
          $cb3 = "";
    if (isset($_POST["cb4"]))
          $cb4 = $_POST["cb4"];
    else
          $cb4 = "";
    echo("Colors you like are $cb1 $cb2 $cb3 $cb4");
}
?>
```

The script displays a form consisting of four checkbox fields which represent different colours. The user is asked to select any of these colours. If the submit button has been clicked then the values of the checkbox variables ($cb1, $cb2, $cb3 and $cb4) are displayed. If a checkbox was not selected the corresponding variable will contain no value and thus nothing will be displayed. The output from running the above script is shown in Figure 18.8.


http://localhost/phpbook/example18-7.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-7.php Go Links
Done
Local intranet

Please select all the colours you like:

Blue ☐ Green ☐ Yellow ☐ Red ☐

Submit Query

Colours you like are blue green

Figure 18.8 Checkbox fields.

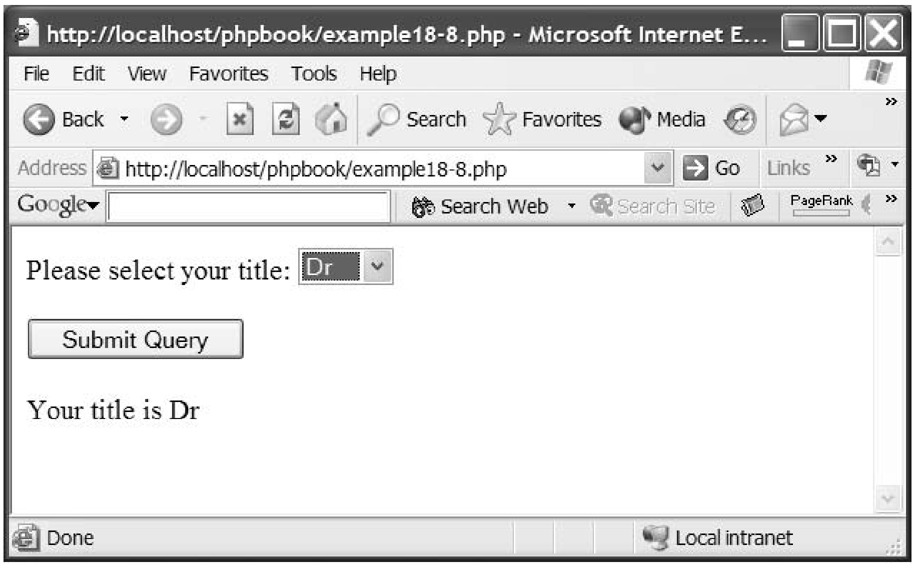
Selection Fields

The form selection field uses a different format from the other form fields, which we have come across before. The field uses two HTML elements: select and option. The select element surrounds all of the option elements making up the menu. For each item you wish to appear on the pull down selection menu you need to include an option element. Because each option element surrounds the data there is no need to include any value attributes. An example of the use of this element is shown in the following script:

```
    <form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>
    Please select your title: <select name='title'>
          <option>Mr</option>
          <option>Miss</option>
          <option>Ms</option>
          <option>Mrs</option>
          <option>Dr</option>
    </select>
    <br><br>
    <input type='submit' name='submit'>
    </form>


<?php

// Interacting with the user - Example 18-8
//----------------------------------
```

```
if(isset($_POST["submit"])){
    $title = $_POST["title"];
    echo("Your title is $title");
}
?>
```

The script displays a form consisting of a single selection menu called title. The menu is made up of a selection of titles. The script checks to see if the submit button has been clicked and if so displays the value of $title. The output from running the above script is shown in Figure 18.9.


http://localhost/phpbook/example18-8.php - Microsoft Internet E...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-8.php Go Links
Google Search Web Search Site PageRank

Please select your title: Dr

Submit Query

Your title is Dr

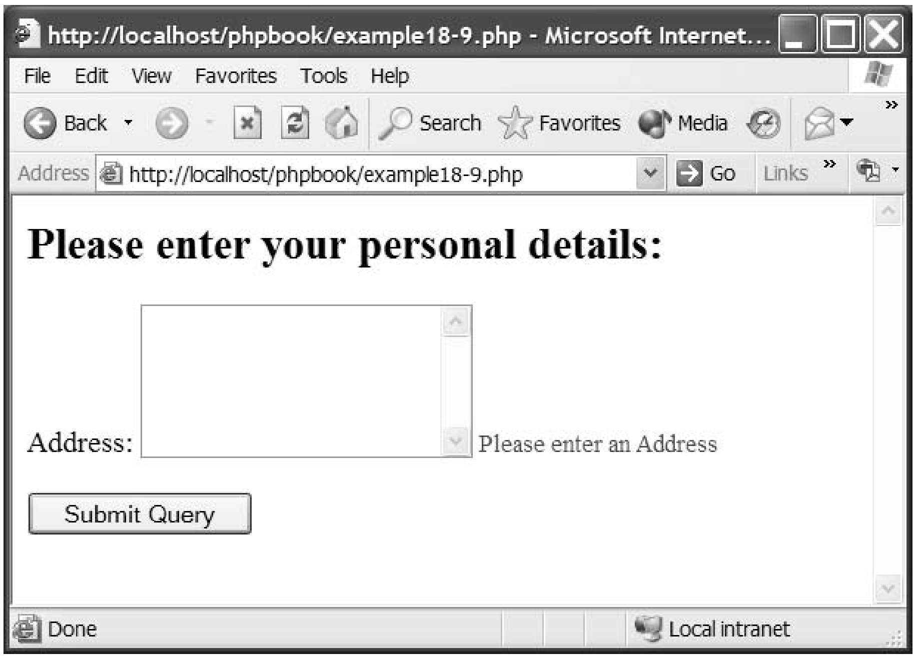
Figure 18.9 Selection fields.

Textarea Fields

The textarea form element allows a text box to be created with a certain number of rows and columns. The user can enter any text in this box. Consider the following script:

```
<?php

// Interacting with the user - Example 18-9
//---------------------------------

if(isset($_POST["submit"])){
    if($_POST["address"] == "")
          $addressErr = 1;
}
?>
    <form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>
    <h2>Please enter your personal details:</h2>
    Address: <textarea name='address' rows='5'></textarea><?php
if(isset($addressErr)) echo("<font color='red' size='2'>
Please enter an Address</font>");?>
    <br><br>
    <input type='submit' name='submit'>
    </form>
```

The script displays a form consisting of a textarea field. The user is asked to enter their address within the field. If the submit button has been pressed, but no text entered in the textarea then a message is displayed prompting the user to enter an address. Figure 18.10 illustrates the output from the above script.


http://localhost/phpbook/example18-9.php - Microsoft Internet...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-9.php Go Links
Done
Local intranet

Please enter your personal details:

Address: Please enter an Address

Submit Query

Figure 18.10 Textarea fields.

Image Fields

Forms can process images as input fields. While, this may at first seem strange consider the image illustrated in Figure 18.11. This image consists of three button shapes.

The image form field is illustrated in the following script. The program does not include a submit button because when the user clicks on the image the x and y coordinates of where

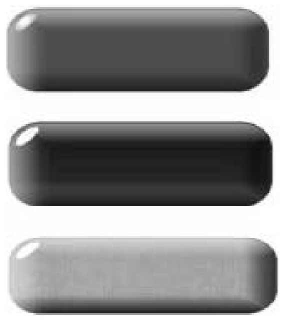
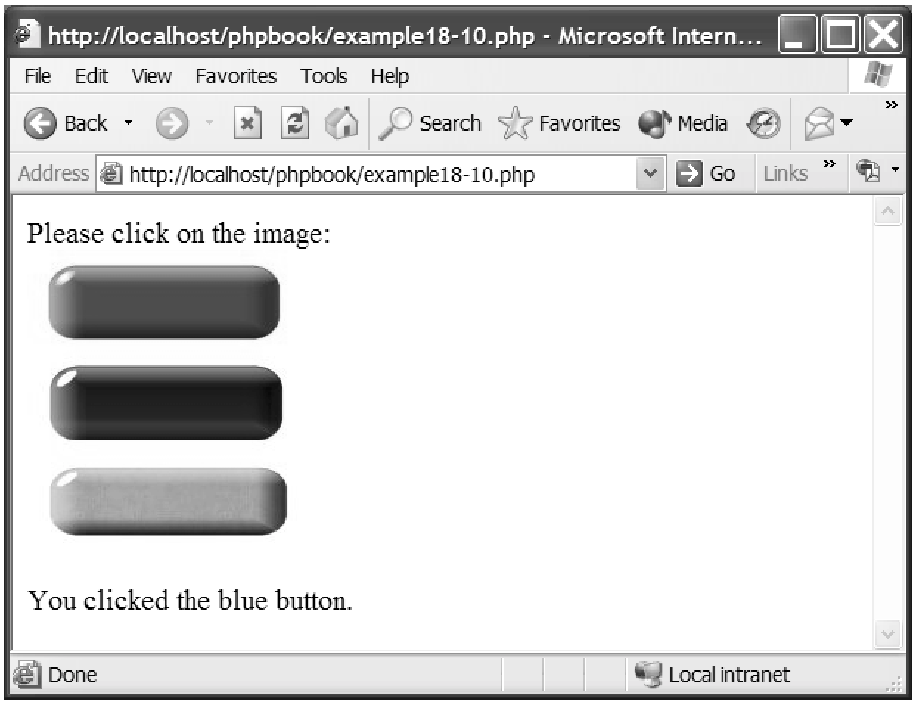


Figure 18.11 Button image.

the user clicked on the image are transmitted to the script. These values are stored in two variables, which take the name button_x and button_y because the field has been given the name “button”:

```
    <form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>
    Please click on the image:<br>
    <input type='image' src='graphics/buttons.jpg' name='button'>
    </form>

<?php

// Interacting with the user - Example 18-10
//----------------------------------


if(isset($_POST["button_x"])){
    if($_POST["button_x"] > 22 && $_POST["button_x"] < 210 &&
$_POST["button_y"] < 155 && $_POST["button_y"] > 100)
          echo("You clicked the blue button.");
    else
          echo("You clicked the image.");

}
?>
```

The above script displays the image as a form image. As there is no submit button the only action that will submit the form is when the user clicks on the image. When this happens the script checks the X and Y values passed as the variables button_x and button_y to determine if the user has clicked on the blue button (the middle one) and displays a message to that effect, as illustrated in Figure 18.12. We can determine if the user has clicked the middle button because we know that the top left-hand corner of the button is at position 22 × 100 and the bottom right-hand corner is at position 210 × 155 of the image. If the values stored in button_x and button_y are within these ranges then the button has been clicked.


http://localhost/phpbook/example18-10.php - Microsoft Intern...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example18-10.php Go Links
Done
Local intranet

Please click on the image:

You clicked the blue button.

Figure 18.12 File field output.

File Uploads

Forms can employ an element that allows the user to select a file from the local computer and upload this to the server for access by the PHP script. This is a complex process and is covered in some detail in Chapter 28.

Summary

This chapter has shown how PHP can be used to interact with the user via HTML forms. Each of the different form elements has been presented and the method in which the data it contains can be extracted has been shown. However, there is more to user interaction than simply displaying a form and accessing the data that it contains. In the following chapter we shall examine the role of form data validation and form data redisplay.