19

- Advanced Features of PHP

19.8 Form Processing and Business Logic

- Superglobal arrays are associative arrays predefined by PHP that hold variables acquired from user input, the environment or the web server and are accessible in any variable scope.
- The arrays $_GET and $_POST retrieve information sent to the server by HTTP get and post requests, respectively.

Variable name	Description
`$_SERVER`	Data about the currently running server.
`$_ENV`	Data about the client's environment.
`$_GET`	Data sent to the server by a `get` request.
`$_POST`	Data sent to the server by a `post` request.
`$_COOKIE`	Data contained in cookies on the client's computer.
`$GLOBALS`	Array containing all global variables.

Fig. 19.12 | Some useful superglobal arrays.

19.8.2 Using PHP to Process HTML5 Forms

- Using `method = "post"` appends form data to the browser request that contains the protocol and the requested resource's URL. Scripts located on the web server's machine can access the form data sent as part of the request.

19.4 Form Processing and Business Logic (Cont.)

- We escape the normal meaning of a character in a string by preceding it with the backslash character (\).
- Function `die` *terminates* script execution. The function's optional argument is a string, which is printed as the script exits.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 19.13: form.html -->
4  <!-- HTML form for gathering user input. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Sample Form</title>
9        <style type = "text/css">
10          label  { width: 5em; float: left; }
11       </style>
12    </head>
13    <body>
14       <h1>Registration Form</h1>
15       <p>Please fill in all fields and click Register.</p>
16
17       <!-- post form data to form.php -->
18       <form method = "post" action = "form.php">
19          <h2>User Information</h2>
20
21          <!-- create four text boxes for user input -->
22          <div><label>First name:</label>
23             <input type = "text" name = "fname"></div>
24          <div><label>Last name:</label>
```

Fig. 19.13 | HTML5 form for gathering user input. (Part 1 of 4.)

```
25             <input type = "text" name = "lname"></div>
26          <div><label>Email:</label>
27             <input type = "text" name = "email"></div>
28          <div><label>Phone:</label>
29             <input type = "text" name = "phone"
30                placeholder = "(555) 555-5555"></div>
31          </div>
32
33          <h2>Publications</h2>
34          <p>Which book would you like information about?</p>
35
36          <!-- create drop-down list containing book names -->
37          <select name = "book">
38             <option>Internet and WWW How to Program</option>
39             <option>C++ How to Program</option>
40             <option>Java How to Program</option>
41             <option>Visual Basic How to Program</option>
42          </select>
43
44          <h2>Operating System</h2>
45          <p>Which operating system do you use?</p>
46
47          <!-- create five radio buttons -->
48          <p><input type = "radio" name = "os" value = "Windows"
49                checked>Windows
```

Fig. 19.13 | HTML5 form for gathering user input. (Part 2 of 4.)

```
50              <input type = "radio" name = "os" value = "Mac OS X">Mac OS X
51              <input type = "radio" name = "os" value = "Linux">Linux
52              <input type = "radio" name = "os" value = "Other">Other</p>
53
54           <!-- create a submit button -->
55           <p><input type = "submit" name = "submit" value = "Register"></p>
56        </form>
57     </body>
58  </html>
```

Fig. 19.13 | HTML5 form for gathering user input. (Part 3 of 4.)

The form is filled out with an incorrect phone number

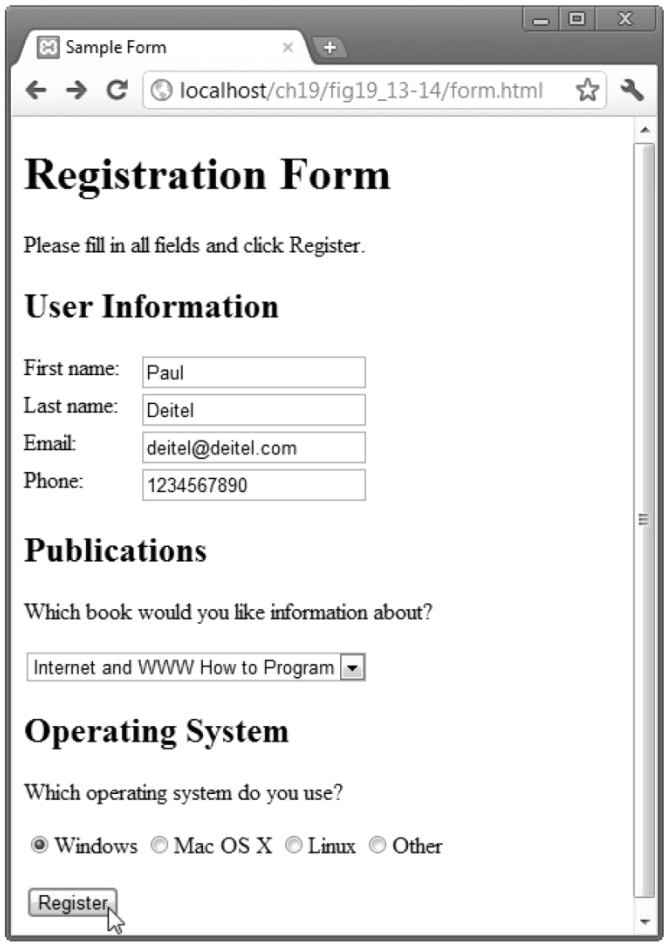


Fig. 19.13 | HTML5 form for gathering user input. (Part 4 of 4.)

Good Programming Practice 19.1

Use meaningful HTML5 object names for `input` fields. This makes PHP scripts that retrieve `form` data easier to understand.

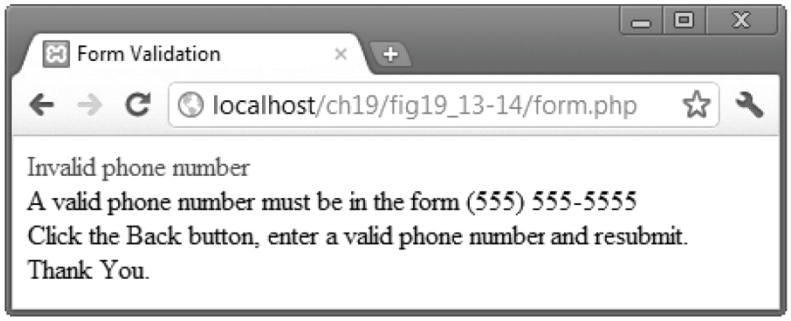
```
1  <!DOCTYPE html>
2
3  <!-- Fig. 19.14: form.php -->
4  <!-- Process information sent from form.html. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Form Validation</title>
9        <style type = "text/css">
10          p       { margin: 0px; }
11          .error  { color: red }
12          p.head  { font-weight: bold; margin-top: 10px; }
13       </style>
14    </head>
15    <body>
16       <?php
17          // determine whether phone number is valid and print
18          // an error message if not
19          if (!preg_match( "/^\([0-9]{3}\) [0-9]{3}-[0-9]{4}$/",
20             $_POST["phone"]))
21          {
```

Fig. 19.14 | Process information sent from `form.html`. (Part 1 of 3.)

```
22              print( "<p class = 'error'>Invalid phone number</p>
23                 <p>A valid phone number must be in the form
24                 (555) 555-5555</p><p>Click the Back button,
25                 enter a valid phone number and resubmit.</p>
26                 <p>Thank You.</p></body></html>" );
27              die(); // terminate script execution
28           }
29        ?><!-- end PHP script -->
30        <p>Hi <?php print( $_POST["fname"] ); ?>. Thank you for
31           completing the survey. You have been added to the
32           <?php print( $_POST["book"] ); ?>mailing list.</p>
33        <p class = "head">The following information has been saved
34           in our database:</p>
35        <p>Name: <?php print( $_POST["fname"] );
36           print( $_POST["lname"] ); ?></p>
37        <p>Email: <?php print( "$email" ); ?></p>
38        <p>Phone: <?php print( "$phone" ); ?></p>
39        <p>OS: <?php print( $_POST["os"] ); ?></p>
40        <p class = "head">This is only a sample form.
41           You have not been added to a mailing list.</p>
42     </body>
43  </html>
```

Fig. 19.14 | Process information sent from `form.html`. (Part 2 of 3.)

a) Submitting the form in Fig. 19.13 redirects the user to `form.php`, which gives appropriate instructions if the phone number is in an incorrect format



b) The results of form.php after the user submits the form in Fig. 19.13 with a phone number in a valid format

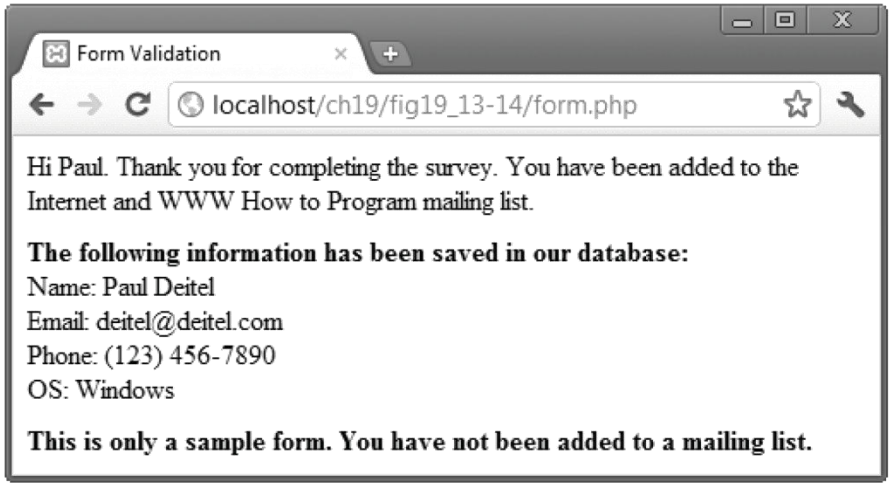


Fig. 19.14 | Process information sent from `form.html`. (Part 3 of 3.)

Software Engineering Observation 19.1

Use business logic to ensure that invalid information is not stored in databases. Validate important or sensitive form data on the server, since JavaScript may be disabled by the client. Some data, such as passwords, must always be validated on the server side.

19.10 Using Cookies

- A cookie is a piece of information that's stored by a server in a text file on a client's computer to maintain information about the client during and between browsing sessions.
- *A server can access only the cookies that it has placed on the client.*
- Function `setcookie` takes the name of the cookie to be set as the first argument, followed by the value to be stored in the cookie.
- The optional third argument indicates the expiration date of the cookie.
- *If no expiration date is specified, the cookie lasts only until the end of the current session*—that is, when the user closes the browser. This type of cookie is known as a session cookie, while one with an expiration date is a persistent cookie.

19.10 Using Cookies (Cont.)

- If only the name argument is passed to function `setcookie`, the cookie is deleted from the client's computer.
- Cookies defined in function `setcookie` are sent to the client at the same time as the information in the HTTP header; therefore, `setcookie` needs to be called *before* any other output
- PHP creates the superglobal array `$_COOKIE`, which contains all the cookie values indexed by their names, similar to the values stored in array `$_POST` when an HTML5 form is posted

```
<!DOCTYPE html>

<!-- Fig. 19.17: cookies.html -->
<!-- Gathering data to be written as a cookie. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>Writing a cookie to the client computer</title>
      <style type = "text/css">
         label { width: 7em; float: left; }
      </style>
   </head>
   <body>
      <h2>Click Write Cookie to save your cookie data.</h2>
      <form method = "post" action = "cookies.php">
         <div><label>Name:</label>
            <input type = "text" name = "name"><div>
         <div><label>Height:</label>
            <input type = "text" name = "height"></div>
         <div><label>Favorite Color:</label>
            <input type = "text" name = "Color"></div>
         <p><input type = "submit" value = "Write Cookie">
      </form>
   </body>
</html>
```

Fig. 19.17 | Gathering data to be written as a cookie. (Part 1 of 2.)

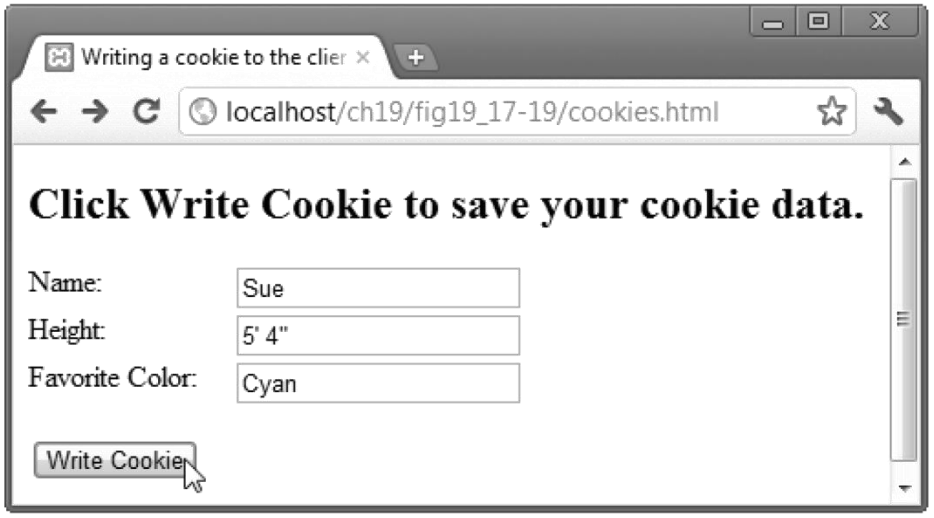


Fig. 19.17 | Gathering data to be written as a cookie. (Part 2 of 2.)

Software Engineering Observation 19.2

Some clients do not accept cookies. When a client declines a cookie, the browser application normally informs the user that the site may not function correctly without cookies enabled.

Software Engineering Observation 19.3

Cookies should not be used to store e-mail addresses, passwords or private data on a client's computer.

```
1  <!-- Fig. 19.18: cookies.php -->
2  <!-- Writing a cookie to the client. -->
3  <?php
4     define( "FIVE_DAYS", 60 * 60 * 24 * 5 ); // define constant
5
6     // write each form field's value to a cookie and set the
7     // cookie's expiration date
8     setcookie( "name", $_POST["name"], time() + FIVE_DAYS );
9     setcookie( "height", $_POST["height"], time() + FIVE_DAYS );
10    setcookie( "color", $_POST["color"], time() + FIVE_DAYS );
11 ?><!-- end PHP script -->
12
13 <!DOCTYPE html>
14
15 <html>
16    <head>
17       <meta charset = "utf-8">
18       <title>Cookie Saved</title>
19       <style type = "text/css">
20          p { margin: 0px; }
21       </style>
22    </head>
```

Fig. 19.18 | Writing a cookie to the client. (Part 1 of 3.)

```
23      <body>
24         <p>The cookie has been set with the following data:</p>
25
26         <!-- print each form field's value -->
27         <p>Name: <?php print( $Name ) ?></p>
28         <p>Height: <?php print( $Height ) ?></p>
29         <p>Favorite Color:
30            <span style = "color: <?php print( "$Color" ) ?> ">
31            <?php print( "$Color" ) ?></span></p>
32         <p>Click <a href = "readCookies.php">here</a>
33            to read the saved cookie.</p>
34      </body>
35   </html>
```

Fig. 19.18 | Writing a cookie to the client. (Part 2 of 3.)

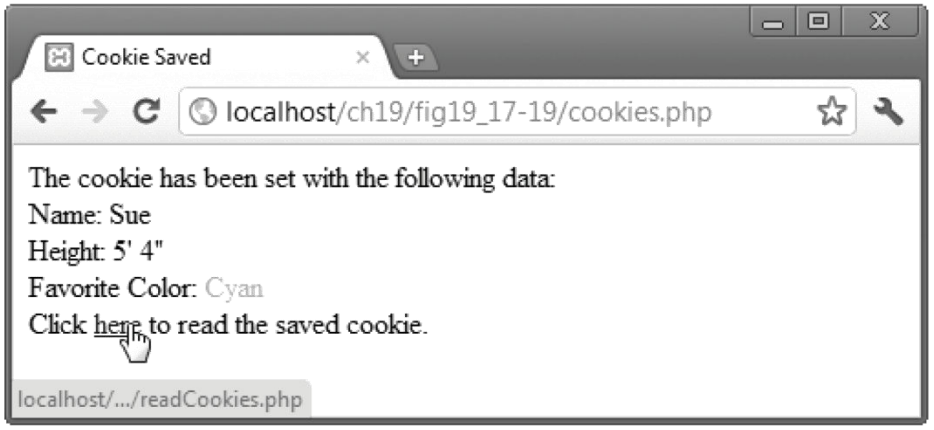


Fig. 19.18 | Writing a cookie to the client. (Part 3 of 3.)

19.11 Dynamic Content

- The `isset` function determines whether the `$_POST` array contains keys representing the various form fields.
- The notation *`$$variable`* specifies a variable variable, which allows the code to reference variables dynamically.
- You can use this expression to obtain the value of the variable whose name is equal to the value of `$variable`.
- The function `mysql_real_escape_string` inserts a backslash (\) before any special characters in the passed string.

```
<!DOCTYPE html>

<!-- Fig. 19.19: readCookies.php -->
<!-- Displaying the cookie's contents. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>Read Cookies</title>
      <style type = "text/css">
         p { margin: 0px; }
      </style>
   </head>
   <body>
      <p>The following data is saved in a cookie on your computer.</p>
      <?php
         // iterate through array $_COOKIE and print
         // name and value of each cookie
         foreach ($_COOKIE as $key => $value )
            print( "<p>$key: $value</p>" );
      ?><!-- end PHP script -->
   </body>
</html>
```

Fig. 19.19 | Displaying the cookie's contents. (Part 1 of 2.)

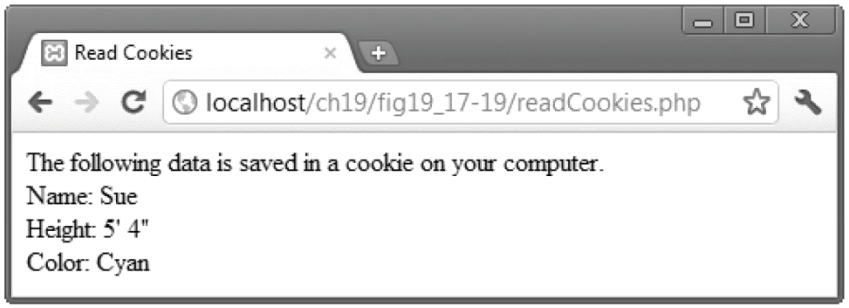


Fig. 19.19 | Displaying the cookie's contents. (Part 2 of 2.)

```
<!DOCTYPE html>

<!-- Fig. 19.20: dynamicForm.php -->
<!-- Dynamic form. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>Registration Form</title>
      <style type = "text/css">
         p       { margin: 0px; }
         .error  { color: red }
         p.head  { font-weight: bold; margin-top: 10px; }
         label   { width: 5em; float: left; }
      </style>
   </head>
   <body>
      <?php
         // variables used in script
         $fname = isset($_POST[ "fname" ]) ? $_POST[ "fname" ] : "";
         $lname = isset($_POST[ "lname" ]) ? $_POST[ "lname" ] : "";
         $email = isset($_POST[ "email" ]) ? $_POST[ "email" ] : "";
         $phone = isset($_POST[ "phone" ]) ? $_POST[ "phone" ] : "";
         $book = isset($_POST[ "book" ]) ? $_POST[ "book" ] : "";
         $os = isset($_POST[ "os" ]) ? $_POST[ "os" ] : "";
```

Fig. 19.20 | Dynamic form. (Part 1 of 10.)

```
25          $iserror = false;
26          $formerrors =
27             array( "fnameerror" => false, "lnameerror" => false,
28                "emailerror" => false, "phoneerror" => false );
29
30          // array of book titles
31          $booklist = array( "Internet and WWW How to Program",
32             "C++ How to Program", "Java How to Program",
33             "Visual Basic How to Program" );
34
35          // array of possible operating systems
36          $systemlist = array( "Windows", "Mac OS X", "Linux", "Other" );
37
38          // array of name values for the text input fields
39          $inputlist = array( "fname" => "First Name",
40             "lname" => "Last Name", "email" => "Email",
41             "phone" => "Phone" );
42
```

Fig. 19.20 | Dynamic form. (Part 2 of 10.)

```
43          // ensure that all fields have been filled in correctly
44          if ( isset( $_POST["submit"] ) )
45          {
46             if ( $fname == "" )
47             {
48                $formerrors[ "fnameerror" ] = true;
49                $iserror = true;
50             } // end if
51
52             if ( $lname == "" )
53             {
54                $formerrors[ "lnameerror" ] = true;
55                $iserror = true;
56             } // end if
57
58             if ( $email == "" )
59             {
60                $formerrors[ "emailerror" ] = true;
61                $iserror = true;
62             } // end if
63
```

Fig. 19.20 | Dynamic form. (Part 3 of 10.)

```
64          if ( !preg_match( "/^\([0-9]{3}\) [0-9]{3}-[0-9]{4}$/",
65             $phone ) )
66          {
67             $formerrors[ "phoneerror" ] = true;
68             $iserror = true;
69          } // end if
70
71          if ( !$iserror )
72          {
73             // build INSERT query
74             $query = "INSERT INTO contacts " .
75                "( LastName, FirstName, Email, Phone, Book, OS ) " .
76                "VALUES ( '$lname', '$fname', '$email', " .
77                "'" . mysql_real_escape_string( $phone ) .
78                "', '$book', '$os' )";
79
80             // Connect to MySQL
81             if ( !( $database = mysql_connect( "localhost",
82                "iw3htp", "password" ) ) )
83                die( "<p>Could not connect to database</p>" );
84
85             // open MailingList database
86             if ( !mysql_select_db( "MailingList", $database ) )
87                die( "<p>Could not open MailingList database</p>" );
88
```

Fig. 19.20 | Dynamic form. (Part 4 of 10.)

```
89                // execute query in MailingList database
90                if ( !( $result = mysql_query( $query, $database ) ) )
91                {
92                   print( "<p>Could not execute query!</p>" );
93                   die( mysql_error() );
94                } // end if
95
96                mysql_close( $database );
97
98                print( "<p>Hi $fname. Thank you for completing the survey.
99                      You have been added to the $book mailing list.</p>
100                  <p class = 'head'>The following information has been
101                     saved in our database:</p>
102                  <p>Name: $fname $lname</p>
103                  <p>Email: $email</p>
104                  <p>Phone: $phone</p>
105                  <p>OS: $os</p>
106                  <p><a href = 'formDatabase.php'>Click here to view
107                     entire database.</a></p>
108                  <p class = 'head'>This is only a sample form.
109                     You have not been added to a mailing list.</p>
110                  </body></html>" );
111               die(); // finish the page
112            } // end if
113         } // end if
```

Fig. 19.20 | Dynamic form. (Part 5 of 10.)

```
114
115         print( "<h1>Sample Registration Form</h1>
116            <p>Please fill in all fields and click Register.</p>" );
117
118         if ( $iserror )
119         {
120            print( "<p class = 'error'>Fields with * need to be filled
121               in properly.</p>" );
122         } // end if
123
124         print( "<!-- post form data to dynamicForm.php -->
125            <form method = 'post' action = 'dynamicForm.php'>
126            <h2>User Information</h2>
127
128            <!-- create four text boxes for user input -->" );
129         foreach ( $inputlist as $inputname => $inputalt )
130         {
131            print( "<div><label>$inputalt:</label><input type = 'text'
132               name = '$inputname' value = '" . $$inputname . "'>" );
133
134            if ( $formerrors[ ( $inputname )."error" ] == true )
135               print( "<span class = 'error'>*</span>" );
136
137            print( "</div>" );
138         } // end foreach
```

Fig. 19.20 | Dynamic form. (Part 6 of 10.)

```
139
140          if ( $formerrors[ "phoneerror" ] )
141             print( "<p class = 'error'>Must be in the form
142                (555)555-5555" );
143
144          print( "<h2>Publications</h2>
145             <p>Which book would you like information about?</p>
146
147             <!-- create drop-down list containing book names -->
148             <select name = 'book'>" );
149
150          foreach ( $booklist as $currbook )
151          {
152             print( "<option" .
153                ($currbook == $book ? " selected>" : ">") .
154                $currbook . "</option>" );
155          } // end foreach
156
157          print( "</select>
158             <h2>Operating System</h2>
159             <p>Which operating system do you use?</p>
160
161             <!-- create five radio buttons -->" );
162
163          $counter = 0;
```

Fig. 19.20 | Dynamic form. (Part 7 of 10.)

```
164
165          foreach ( $systemlist as $currsystem )
166          {
167             print( "<input type = 'radio' name = 'os'
168                value = '$currsystem' " );
169
170             if ( ( !$os && $counter == 0 ) || ( $currsystem == $os ) )
171                print( "checked" );
172
173             print( ">$currsystem" );
174             ++$counter;
175          } // end foreach
176
177          print( "<!-- create a submit button -->
178             <p class = 'head'><input type = 'submit' name = 'submit'
179             value = 'Register'></p></form></body></html>" );
180    ?><!-- end PHP script -->
```

Fig. 19.20 | Dynamic form. (Part 8 of 10.)

a) Registration form after it was submitted with a missing field and an incorrectly formatted phone number

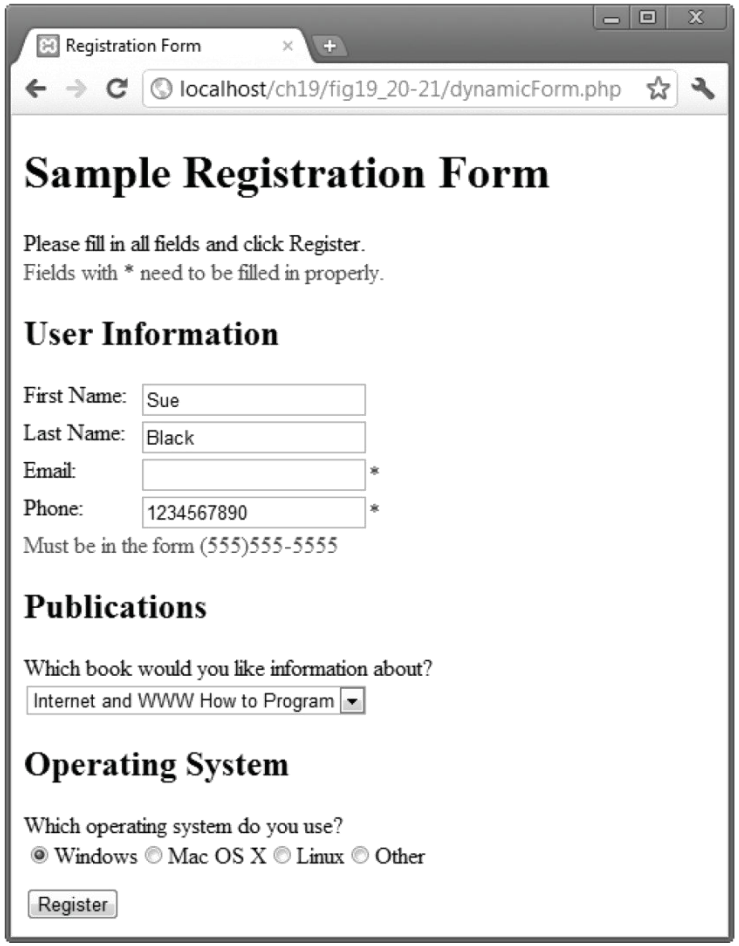


Fig. 19.20 | Dynamic form. (Part 9 of 10.)

b) Confirmation page displayed after the user properly fills in the form and the information is stored in the database

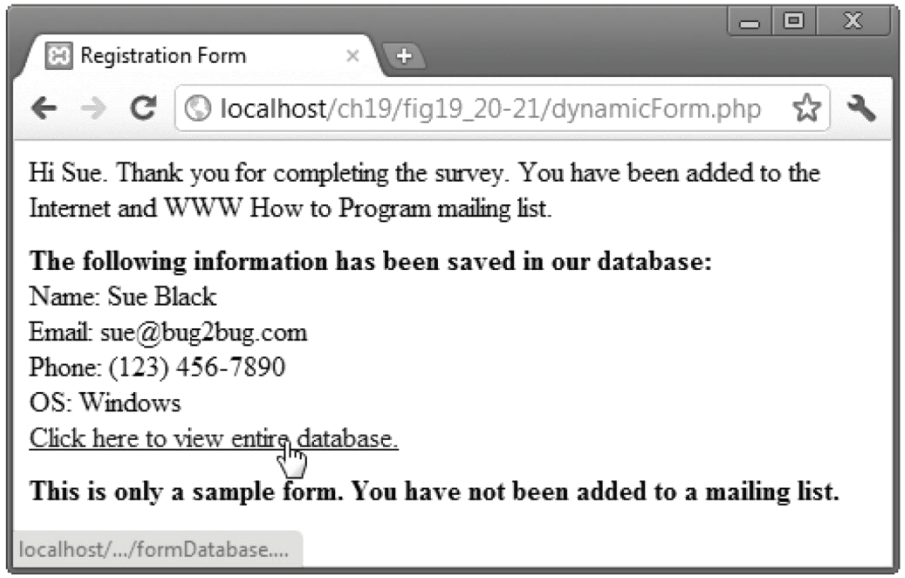


Fig. 19.20 | Dynamic form. (Part 10 of 10.)

```
<!DOCTYPE html>

<!-- Fig. 19.21: formDatabase.php -->
<!-- Displaying the MailingList database. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>Search Results</title>
      <style type = "text/css">
         table  { background-color: lightblue;
                  border: 1px solid gray;
                  border-collapse: collapse; }
         th, td { padding: 5px; border: 1px solid gray; }
         tr:nth-child(even) { background-color: white; }
         tr:first-child { background-color: lightgreen; }
      </style>
   </head>
   <body>
      <?php
         // build SELECT query
         $query = "SELECT * FROM contacts";

```

Fig. 19.21 | Displaying the `MailingList` database. (Part 1 of 4.)

```
23          // Connect to MySQL
24          if ( !( $database = mysql_connect( "localhost",
25             "iw3htp", "password" ) ) )
26             die( "<p>Could not connect to database</p></body></html>" );
27
28          // open MailingList database
29          if ( !mysql_select_db( "MailingList", $database ) )
30             die( "<p>Could not open MailingList database</p>
31                </body></html>" );
32
33          // query MailingList database
34          if ( !( $result = mysql_query( $query, $database ) ) )
35          {
36             print( "<p>Could not execute query!</p>" );
37             die( mysql_error() . "</body></html>" );
38          } // end if
39       ?><!-- end PHP script -->
40
41       <h1>Mailing List Contacts</h1>
42       <table>
43          <caption>Contacts stored in the database</caption>
44          <tr>
45             <th>ID</th>
46             <th>Last Name</th>
47             <th>First Name</th>
```

Fig. 19.21 | Displaying the `MailingList` database. (Part 2 of 4.)

```
48             <th>E-mail Address</th>
49             <th>Phone Number</th>
50             <th>Book</th>
51             <th>Operating System</th>
52          </tr>
53          <?php
54             // fetch each record in result set
55             for ( $counter = 0; $row = mysql_fetch_row( $result );
56                ++$counter )
57             {
58                // build table to display results
59                print( "<tr>" );
60
61                foreach ( $row as $key => $value )
62                   print( "<td>$value</td>" );
63
64                print( "</tr>" );
65             } // end for
66
67             mysql_close( $database );
68          ?><!-- end PHP script -->
69       </table>
70    </body>
71 </html>
```

Fig. 19.21 | Displaying the `MailingList` database. (Part 3 of 4.)

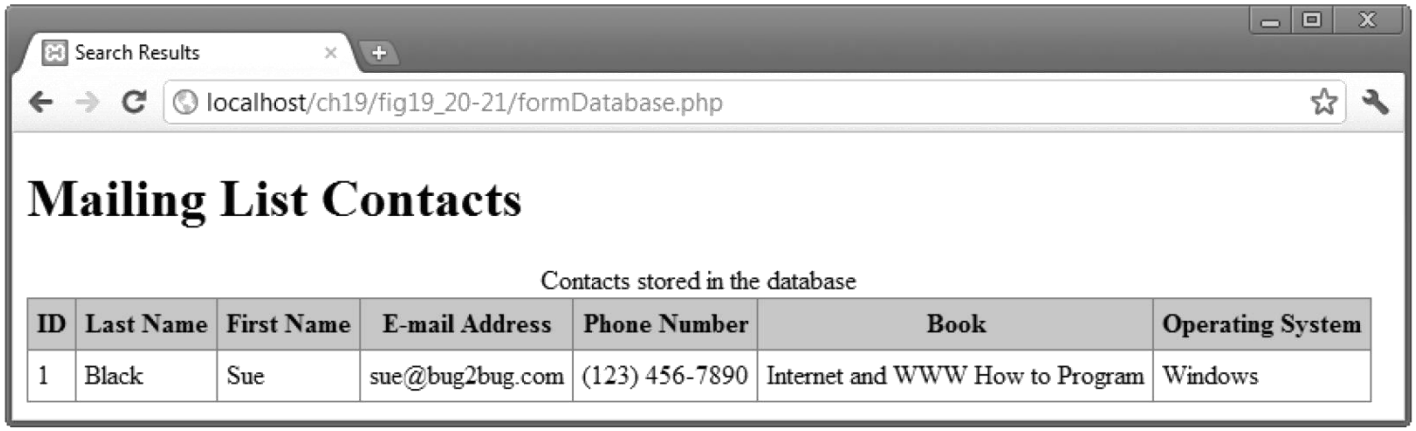


Fig. 19.21 | Displaying the `MailingList` database. (Part 4 of 4.)