23

PHP

Introduction

PHP Basics

Form Processing and Business Logic

Connecting to a Database

Introduction

- PHP, or PHP: Hypertext Preprocessor, has become one of the most popular server-side scripting languages for creating dynamic web pages.
- PHP is open source and platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports a large number of databases.

PHP Basics

- The power of the web resides not only in serving content to users, but also in responding to requests from users and generating web pages with dynamic content.
- PHP code is embedded directly into XHTML documents, though these script segments are interpreted by a server before being delivered to the client.
- PHP script file names end with `.php`.
- Although PHP can be used from the command line, a web server is necessary to take full advantage of the scripting language.
- In PHP, code is inserted between the scripting delimiters `<?php` and `?>`. PHP code can be placed anywhere in XHTML markup, as long as the code is enclosed in these delimiters.

PHP Basics (Cont.)

- Variables are preceded by a `$` and are created the first time they are encountered.
- PHP statements terminate with a semicolon (`;`).
- Single-line comments which begin with two forward slashes (`//`) or a pound sign (`#`). Text to the right of the delimiter is ignored by the interpreter. Multiline comments begin with delimiter `/*` and end with delimiter `*/`.
- When a variable is encountered inside a double-quoted (`""`) string, PHP interpolates the variable. In other words, PHP inserts the variable's value where the variable name appears in the string.
- All operations requiring PHP interpolation execute on the server before the XHTML document is sent to the client.
- PHP variables are loosely typed—they can contain different types of data at different times.

Outline

first.php

```
1  <?php print( '<?xml version = "1.0" encoding = "utf-8"?>' ) ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 23.1: first.php -->
6  <!-- Simple PHP program. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <?php
9     $name = "Harvey"; // declaration and initialization
10 ?><!-- end PHP script -->
11    <head>
12       <title>Using PHP document</title>
13    </head>
14    <body style = "font-size: 2em">
15       <p>
16          <strong>
17             <!-- print variable name's value -->
18             Welcome to PHP, <?php print( "$name" ); ?>!
19          </strong>
20       </p>
21    </body>
22 </html>
```

Delimiters enclosing PHP script

Declares and initializes a PHP variable

Interpolates the variable so that its value will be output to the XHTML document

Common Programming Error

Failing to precede a variable name with a $ is a syntax error.

Common Programming Error

Variable names in PHP are case sensitive. Failure to use the proper mixture of cases to refer to a variable will result in a logic error, since the script will create a new variable for any name it doesn't recognize as a previously used variable.

Common Programming Error

Forgetting to terminate a statement with a semicolon (;) is a syntax error.

Type	Description
`int`, `integer`	Whole numbers (i.e., numbers without a decimal point).
`float`, `double`, `real`	Real numbers (i.e., numbers containing a decimal point).
`string`	Text enclosed in either single (`''`) or double (`""`) quotes. [*Note:* Using double quotes allows PHP to recognize more escape sequences.]
`bool`, `boolean`	True or false.
`array`	Group of elements.
`object`	Group of associated data and methods.
`resource`	An external source—usually information from a database.
`NULL`	No value.

PHP types.

PHP Basics (Cont.)

- Type conversions can be performed using function `settype`. This function takes two arguments—a variable whose type is to be changed and the variable's new type.
- Variables are automatically converted to the type of the value they are assigned.
- Function `gettype` returns the current type of its argument.
- Calling function `settype` can result in loss of data. For example, doubles are truncated when they are converted to integers.
- When converting from a string to a number, PHP uses the value of the number that appears at the beginning of the string. If no number appears at the beginning, the string evaluates to `0`.
- Another option for conversion between types is casting (or type casting). Casting does not change a variable's content—it creates a temporary copy of a variable's value in memory.
- The concatenation operator (`.`) combines multiple strings.
- A `print` statement split over multiple lines prints all the data that is enclosed in its parentheses.

Outline

data.php

(1 of 3)

```
1  <?php print( '<?xml version = "1.0" encoding = "utf-8"?>' ) ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 23.3: data.php -->
6  <!-- Data type conversion. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Data type conversion</title>
10    </head>
11    <body>
12       <?php
13          // declare a string, double and integer
14          $testString = "3.5 seconds";
15          $testDouble = 79.2;
16          $testInteger = 12;
17       ?><!-- end PHP script -->
18
19       <!-- print each variable's value and type -->
20       <?php
21          print( "$testString is a(n) " . gettype( $testString )
22             . "<br />" );
```

Automatically declares a string

Automatically declares a double

Automatically declares an integer

Outputs the type of `$testString`

Outline

data.php

(2 of 3)

```
23          print( "$testDouble is a(n) " . gettype( $testDouble )
24             . "<br />" );
25          print( "$testInteger is a(n) " . gettype( $testInteger)
26             . "<br />" );
27       ?><!-- end PHP script -->
28       <br />
29       converting to other data types:<br />
30       <?php
31          // call function settype to convert variable
32          // testString to different data types
33          print( "$testString" );
34          settype( $testString, "double" );
35          print( " as a double is $testString <br />" );
36          print( "$testString" );
37          settype( $testString, "integer" );
38          print( " as an integer is $testString <br />" );
39          settype( $testString, "string" );
40          print( "converting back to a string results in
41             $testString <br /><br />" );
42
```

Modifies `$testString` to be a double

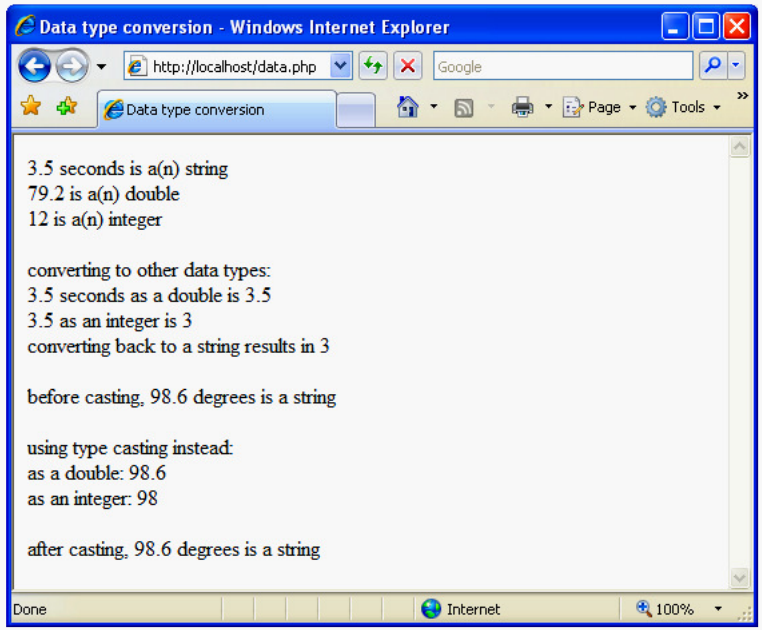
Modifies `$testString` to be an integer

Modifies `$testString` to be a string

```
43          // use type casting to cast variables to a different type
44          $data = "98.6 degrees";
45          print( "before casting, $data is a " .
46             gettype( $data ) . "<br /><br />" );
47          print( "using type casting instead: <br />
48             as a double: " . (double) $data .
49             "<br />as an integer: " . (integer) $data );
50          print( "<br /><br />after casting, $data is a " .
51              gettype( $data ) );
52       ?><!-- end PHP script -->
53    </body>
54 </html>
```

Temporarily casts `$data` as a double and an integer

Concatenation



Outline

data.php

(3 of 3)

Error-Prevention Tip

Function `print` can be used to display the value of a variable at a particular point during a program's execution. This is often helpful in debugging a script.

PHP Basics (Cont.)

- Function `define` creates a named constant. It takes two arguments—the name and value of the constant. An optional third argument accepts a boolean value that specifies whether the constant is case insensitive—constants are case sensitive by default.
- Uninitialized variables have the value `undef`, which has different values, depending on its context. In a numeric context, it evaluates to `0`. In a string context, it evaluates to an empty string (`""`).
- Keywords may not be used as identifiers.

Common Programming Error

Assigning a value to a constant after it is declared is a syntax error.

Outline

operators.php

(1 of 3)

```
1  <?php print( '<?xml version = "1.0" encoding = "utf-8"?>' ) ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 23.4: operators.php -->
6  <!-- Using arithmetic operators. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Using arithmetic operators</title>
10    </head>
11    <body>
12       <?php
13          $a = 5;
14          print( "The value of variable a is $a <br />" );
15
16          // define constant VALUE
17          define( "VALUE", 5 );
18
19          // add constant VALUE to variable $a
20          $a = $a + VALUE;
21          print( "Variable a after adding constant VALUE
22             is $a <br />" );
23
24          // multiply variable $a by 2
25          $a *= 2;
26          print( "Multiplying variable a by 2 yields $a <br />" );
27
```

Creates the named constant VALUE with a value of 5

Equivalent to `$a = $a * 2`

Outline

operators.php

(2 of 3)

```
28          // test if variable $a is less than 50
29          if ( $a < 50 )
30             print( "Variable a is less than 50 <br />" );
31
32          // add 40 to variable $a
33          $a += 40;
34          print( "Variable a after adding 40 is $a <br />" );
35
36          // test if variable $a is 50 or less
37          if ( $a < 51 )
38             print( "Variable a is still 50 or less<br />" );
39
40          // test if variable $a is between 50 and 100, inclusive
41          elseif ( $a < 101 )
42             print( "Variable a is now between 50 and 100,
43                inclusive<br />" );
44          else
45             print( "Variable a is now greater than 100 <br />" );
46
47          // print an uninitialized variable
48          print( "Using a variable before initializing:
49             $nothing <br />" ); // nothing evaluates to ""
50
51          // add constant VALUE to an uninitialized variable
52          $test = $num + VALUE; // num evaluates to 0
```

Uses a comparison operator with a variable and an integer

Uninitialized variable `$num` evaluates to `0`

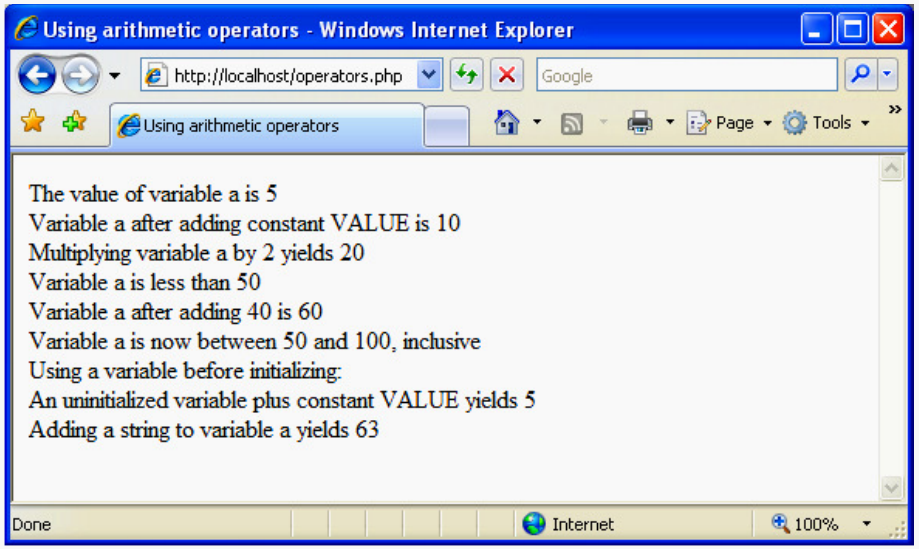
Outline

operators.php

(3 of 3)

```
53          print( "An uninitialized variable plus constant
54             VALUE yields $test <br />" );
55
56          // add a string to an integer
57          $str = "3 dollars";
58          $a += $str;
59          print( "Adding a string to variable a yields $a <br />" );
60       ?><!-- end PHP script -->
61    </body>
62 </html>
```

`$str` is converted to an integer for this operation



Error-Prevention Tip

Initialize variables before they are used to avoid subtle errors. For example, multiplying a number by an uninitialized variable results in 0.

PHP keywords

abstract	die	exit	interface	require
and	do	extends	isset	require_once
array	echo	__FILE__	__LINE__	return
as	else	file	line	static
break	elseif	final	list	switch
case	empty	for	__METHOD__	throw
catch	enddeclare	foreach	method	try
__CLASS__	endfor	__FUNCTION__	new	unset
class	endforeach	function	or	use
clone	endif	global	php_user_filter	var
const	endswitch	if	print	while
continue	endwhile	implements	private	xor
declare	eval	include	protected	
default	exception	include_once	public	

PHP keywords.

PHP Basics (Cont.)

- PHP provides the capability to store data in arrays. Arrays are divided into elements that behave as individual variables. Array names, like other variables, begin with the `$` symbol.
- Individual array elements are accessed by following the array's variable name with an index enclosed in square brackets (`[]`).
- If a value is assigned to an array that does not exist, then the array is created. Likewise, assigning a value to an element where the index is omitted appends a new element to the end of the array.
- Function `count` returns the total number of elements in the array.
- Function `array` creates an array that contains the arguments passed to it. The first item in the argument list is stored as the first array element (index `0`), the second item is stored as the second array element and so on.

PHP Basics (Cont.)

- Arrays with nonnumeric indices are called associative arrays. You can create an associative array using the operator `=>`, where the value to the left of the operator is the array index and the value to the right is the element's value.
- PHP provides functions for iterating through the elements of an array. Each array has a built-in internal pointer, which points to the array element currently being referenced. Function `reset` sets the internal pointer to the first array element. Function `key` returns the index of the element currently referenced by the internal pointer, and function `next` moves the internal pointer to the next element.
- The `foreach` statement, designed for iterating through arrays, starts with the array to iterate through, followed by the keyword `as`, followed by two variables—the first is assigned the index of the element and the second is assigned the value of that index's element. (If only one variable is listed after `as`, it is assigned the value of the array element.)

Outline

arrays.php

(1 of 4)

```
1  <?php print( '<?xml version = "1.0" encoding = "utf-8"?>' ) ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 23.6: arrays.php -->
6  <!-- Array manipulation. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Array manipulation</title>
10    </head>
11    <body>
12       <?php
13          // create array first
14          print( "<strong>Creating the first array</strong><br />"
15          $first[ 0 ] = "zero";
16          $first[ 1 ] = "one";
17          $first[ 2 ] = "two";
18          $first[] = "three";
19
20          // print each element's index and value
21          for ( $i = 0; $i < count( $first ); $i++ )
22             print( "Element $i is $first[$i] <br />" );
23
```

Automatically creates array `$first`

Sets the first element of array `$first` to the string "`zero`"

"`three`" is appended to the end of array `$first`

Returns the number of elements in the array

Outline

arrays.php

(2 of 4)

```
24          print( "<br /><strong>Creating the second array
25             </strong><br />" );
26
27          // call function array to create array second
28          $second = array( "zero", "one", "two", "three" );
29
30          for ( $i = 0; $i < count( $second ); $i++ )
31             print( "Element $i is $second[$i] <br />" );
32
33          print( "<br /><strong>Creating the third array
34             </strong><br />" );
35
36          // assign values to entries using nonnumeric indices
37          $third[ "Amy" ] = 21;
38          $third[ "Bob" ] = 18;
39          $third[ "Carol" ] = 23;
40
41          // iterate through the array elements and print each
42          // element's name and value
43          for ( reset( $third ); $element = key( $third ); next( $third ) )
44             print( "$element is $third[$element] <br />" );
45
```

Function `array` creates array `$second` with its arguments as elements

Creates associative array `$third`

Sets the internal pointer to the first array element in `$third`

Returns the index of the element being pointed to

Moves the internal pointer to the next element and returns it

Outline

arrays.php

(3 of 4)

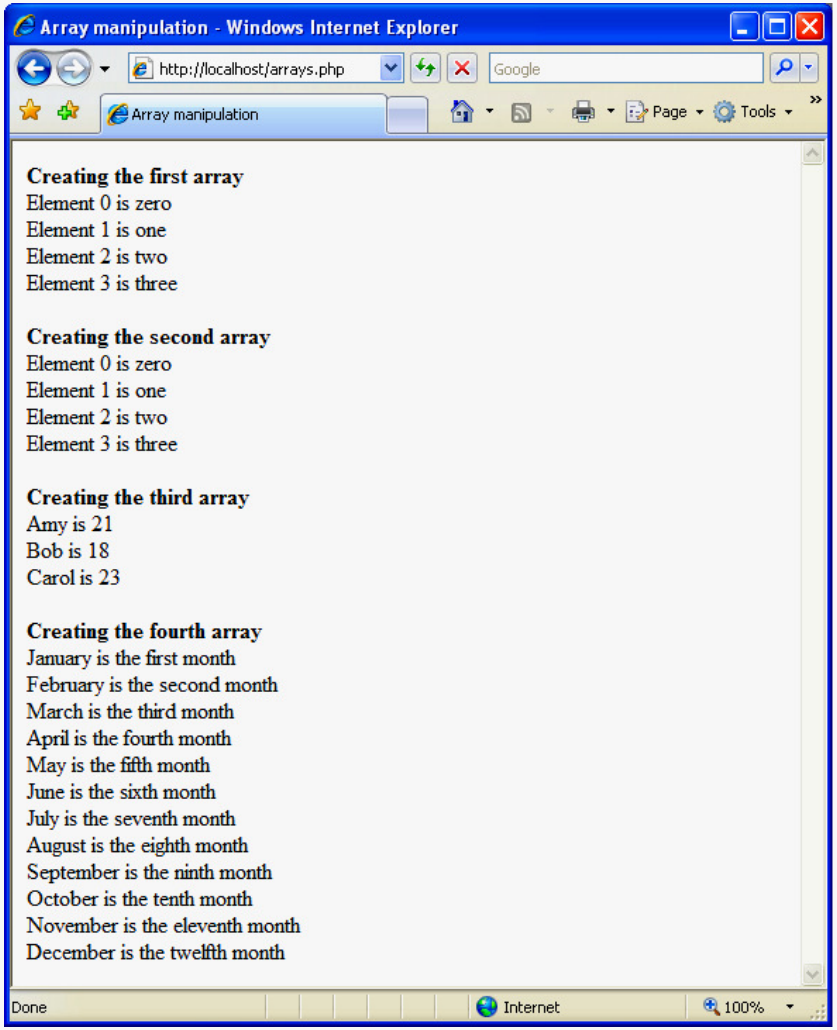
```
46          print( "<br /><strong>Creating the fourth array
47             </strong><br />" );
48
49          // call function array to create array fourth using
50          // string indices
51          $fourth = array(
52             "January"   => "first",   "February" => "second",
53             "March"     => "third",   "April"    => "fourth",
54             "May"       => "fifth",   "June"     => "sixth",
55             "July"      => "seventh", "August"   => "eighth",
56             "September" => "ninth",   "October"  => "tenth",
57             "November"  => "eleventh","December" => "twelfth"
58             );
59
60          // print each element's name and value
61          foreach ( $fourth as $element => $value )
62             print( "$element is the $value month <br />" );
63       ?><!-- end PHP script -->
64    </body>
65 </html>
```

Uses operator => to initialize the element with index "January" to have value "first"

Iterates through each element in array $fourth

Stores the index of the element

Stores the value of the element



Outline

`arrays.php`

(4 of 4)

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.
- 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.

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.

Some useful superglobal arrays.

Outline

form.html

(1 of 4)

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5
6  <!-- XHTML form for gathering user input. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Sample form to take user input in XHTML</title>
10       <style type = "text/css">
11          .prompt { color: blue;
12                    font-family: sans-serif;
13                    font-size: smaller }
14       </style>
15    </head>
16    <body>
17       <h1>Sample Registration Form</h1>
18       <p>Please fill in all fields and click Register.</p>
19
20       <!-- post form data to form.php -->
21       <form method = "post" action = "form.php">
22          <div>
23             <img src = "images/user.gif" alt = "User" /><br />
24             <span class = "prompt">
25                Please fill out the fields below.<br />
26             </span>
27
```

Appends form data to the browser request that contains the protocol and the URL of the requested resource

Form data is posted to `form.php` to be processed

Outline

form.html

(2 of 4)

```
28          <!-- create four text boxes for user input -->
29          <img src = "images/fname.gif" alt = "First Name" />
30          <input type = "text" name = "fname" /><br />
31
32          <img src = "images/lname.gif" alt = "Last Name" />
33          <input type = "text" name = "lname" /><br />
34
35          <img src = "images/email.gif" alt = "Email" />
36          <input type = "text" name = "email" /><br />
37
38          <img src = "images/phone.gif" alt = "Phone" />
39          <input type = "text" name = "phone" /><br />
40
41          <span style = "font-size: 10pt">
42             Must be in the form (555)555-5555</span>
43          <br /><br />
44
45          <img src = "images/downloads.gif"
46          alt = "Publications" /><br />
47
48          <span class = "prompt">
49             Which book would you like information about?
50          </span><br />
51
52          <!-- create drop-down list containing book names -->
53          <select name = "book">
```

Creates form fields

Creates drop-down list with book names

Outline

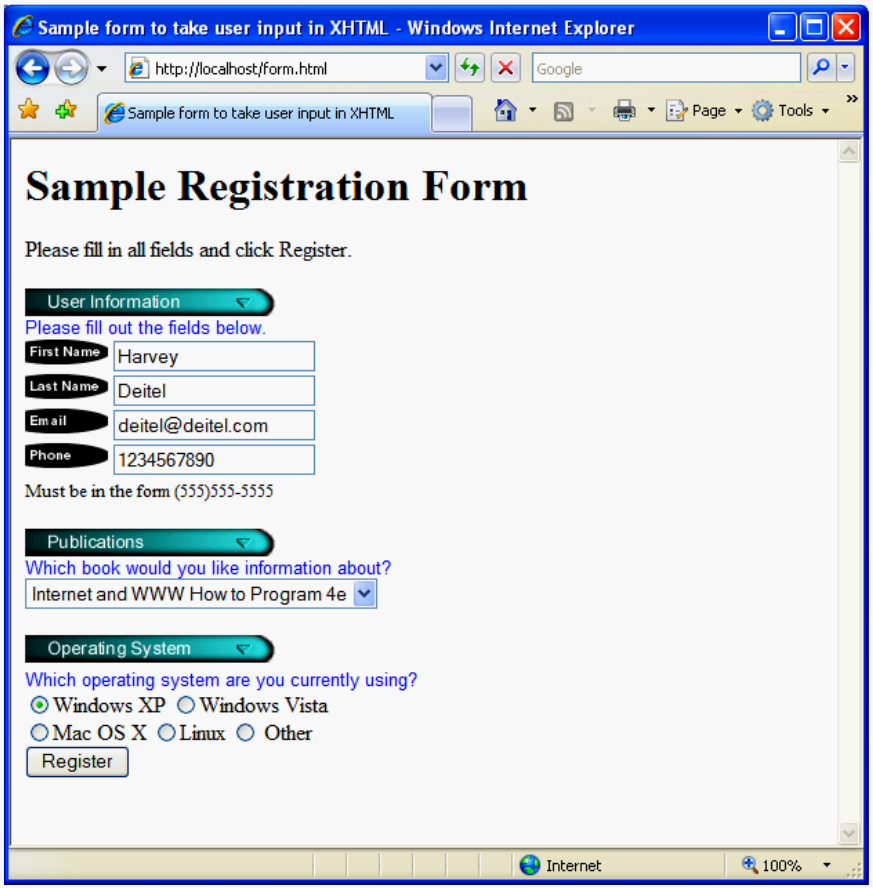
form.html

(3 of 4)

```
54             <option>Internet and WWW How to Program 4e</option>
55             <option>C++ How to Program 6e</option>
56             <option>Java How to Program 7e</option>
57             <option>Visual Basic 2005 How to Program 3e</option>
58          </select>
59          <br /><br />
60
61          <img src = "images/os.gif" alt = "Operating System" />
62          <br /><span class = "prompt">
63             Which operating system are you currently using?
64          <br /></span>
65
66          <!-- create five radio buttons -->
67          <input type = "radio" name = "os" value = "Windows XP"
68             checked = "checked" /> Windows XP
69          <input type = "radio" name = "os" value =
70             "Windows Vista" /> Windows Vista<br />
71          <input type = "radio" name = "os" value =
72             "Mac OS X" /> Mac OS X
73          <input type = "radio" name = "os" value = "Linux" /> Linux
74          <input type = "radio" name = "os" value = "Other" />
75             Other<br />
76
```

Creates radio buttons with "Windows XP" initially selected

```
77             <!-- create a submit button -->
78             <input type = "submit" value = "Register" />
79          </div>
80       </form>
81    </body>
82 </html>
```



Outline

form.html

(4 of 4)

Good Programming Practice

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

Form Processing and Business Logic (Cont.)

- Function `extract` creates a variable/value pair corresponding to each key/value pair in the associative array passed as an argument.
- Business logic, or business rules, ensures that only valid information is stored in databases.
- 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.

Outline

form.php

(1 of 5)

```
1  <?php print( '<?xml version = "1.0" encoding = "utf-8"?>' ) ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 23.13: form.php -->
6  <!-- Process information sent from form.html. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Form Validation</title>
10       <style type = "text/css">
11          body      { font-family: arial, sans-serif }
12          div       { font-size: 10pt;
13                      text-align: center }
14          table     { border: 0 }
15          td        { padding-top: 2px;
16                      padding-bottom: 2px;
17                      padding-left: 10px;
18                      padding-right: 10px }
19          .error    { color: red }
20          .distinct { color: blue }
21          .name     { background-color: #ffffaa }
22          .email    { background-color: #ffffbb }
23          .phone    { background-color: #ffffcc }
24          .os       { background-color: #ffffdd }
25       </style>
26    </head>
27    <body>
```

```
28    <?php
29       extract( $_POST );
30
31       // determine whether phone number is valid and print
32       // an error message if not
33       if ( !ereg( "^\([0-9]{3}\)[0-9]{3}-[0-9]{4}$", $phone ) )
34       {
35          print( "<p><span class = 'error'>
36             Invalid phone number</span><br />
37             A valid phone number must be in the form
38             <strong>(555)555-5555</strong><br />
39             <span class = 'distinct'>
40             Click the Back button, enter a valid phone
41             number and resubmit.<br /><br />
42             Thank You.</span></p>" );
43          die( "</body></html>" ); // terminate script execution
44       }
45    ?><!-- end PHP script -->
46    <p>Hi
47       <span class = "distinct">
48          <strong><?php print( "$fname" ); ?></strong>
49       </span>.
50       Thank you for completing the survey.<br />
51       You have been added to the
52       <span class = "distinct">
53          <strong><?php print( "$book " ); ?></strong>
54       </span>
55       mailing list.
```

Creates a variable/value pair for each key/value pair in `$_POST`

Ensures that phone number is in proper format

Terminates execution and closes the document properly

form.php

(2 of 5)

Outline

form.php

(3 of 5)

```
56        </p>
57        <p><strong>The following information has been saved
58            in our database:</strong></p>
59        <table>
60           <tr>
61              <td class = "name">Name </td>
62              <td class = "email">Email</td>
63              <td class = "phone">Phone</td>
64              <td class = "os">OS</td>
65           </tr>
66           <tr>
67              <?php
68                 // print each form field's value
69                 print( "<td>$fname $lname</td>
70                    <td>$email</td>
71                    <td>$phone</td>
72                    <td>$os</td>" );
73              ?><!-- end PHP script -->
74           </tr>
75        </table>
76        <br /><br /><br />
77        <div>This is only a sample form.
78           You have not been added to a mailing list.</div>
79     </body>
80  </html>
```

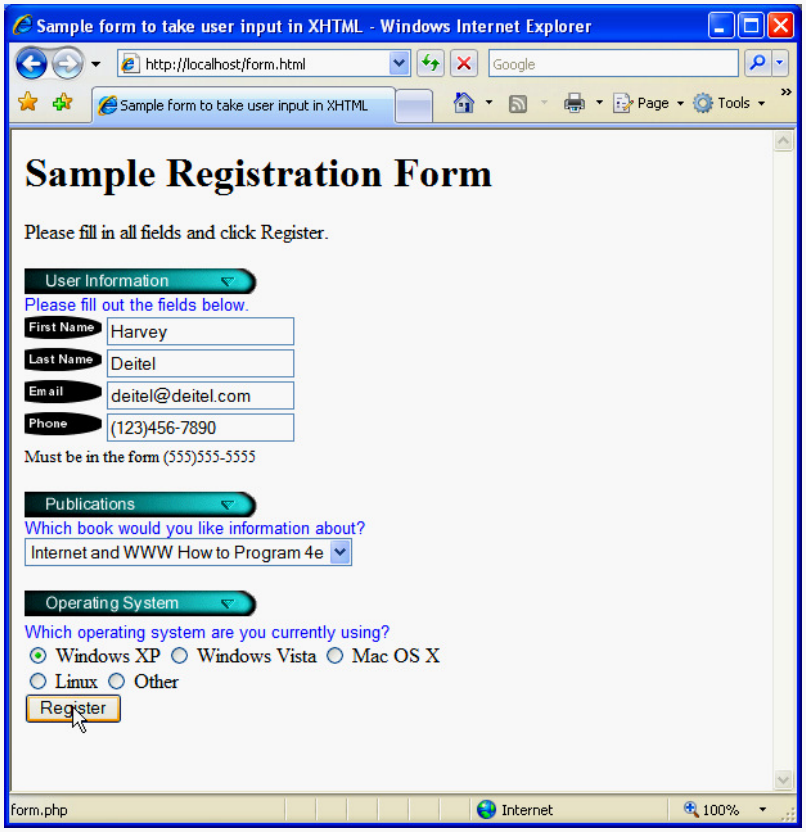
Prints the value entered in the `email` field in `form.html`

Outline

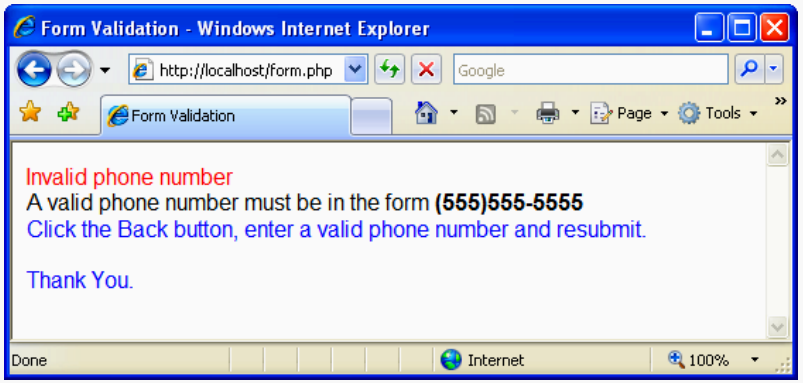
`form.php`

(4 of 5)

a) The form in `form.html` is filled out with an incorrect phone number.



b) The user is redirected to `form.php`, which gives appropriate instructions.

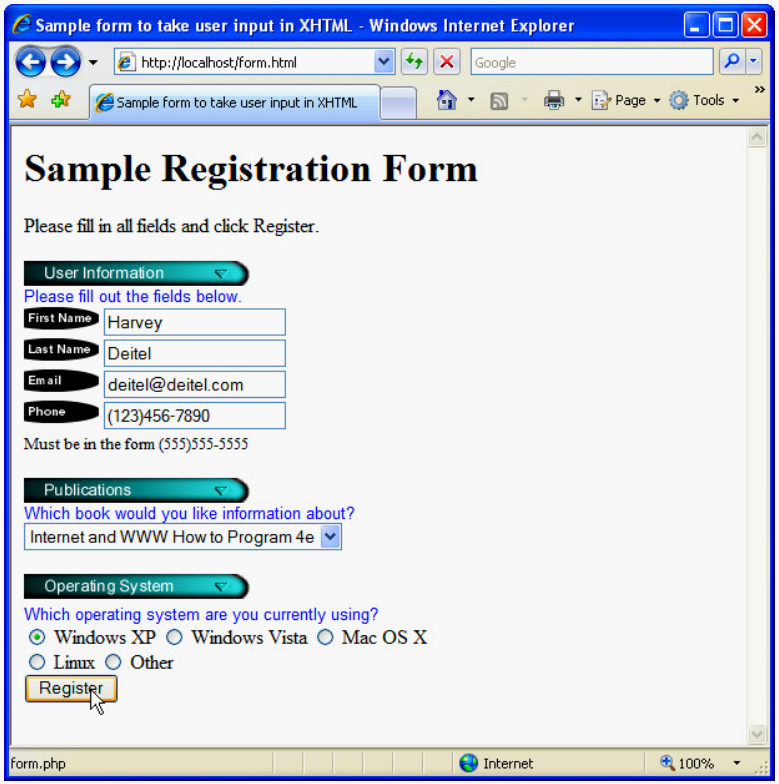


Outline

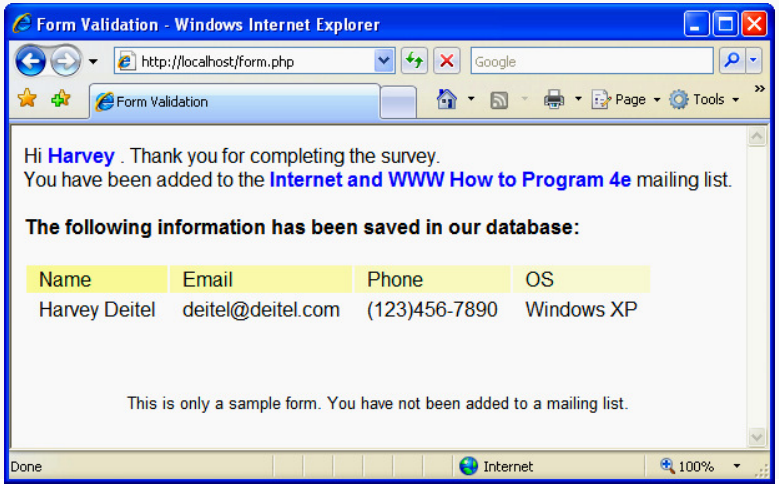
form.php

(5 of 5)

c) The form is now filled out correctly.



d) The user is directed to an acceptance page, which displays the entered information.



Software Engineering Observation

Use business logic to ensure that invalid information is not stored in databases. When possible, 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.

Error-Prevention Tip

Be sure to close any open XHTML tags when calling function `die`. Not doing so can produce invalid XHTML output that will not display properly in the client browser. Function `die` has an optional parameter that specifies a message to output when exiting, so one technique for closing tags is to close all open tags using `die`, as in `die("</body></html>")`.

Connecting to a Database

- Function `mysqli_connect` connects to the MySQL database. It takes three arguments—the server's hostname, a username and a password, and returns a database handle—a representation of PHP's connection to the database, or `false` if the connection fails.
- Function `mysqli_select_db` specifies the database to be queried, and returns a bool indicating whether or not it was successful.
- To query the database, we call function `mysqli_query`, specifying the query string and the database to query. This returns a resource containing the result of the query, or false if the query fails. It can also execute SQL statements such as `INSERT` or `DELETE` that do not return results.
- Function `mysqli_error` returns any error strings from the database.
- `mysqli_close` closes the connection to the database specified in its argument.