Chapter 4

Handling User Input

PHP Programming with MySQL
2nd Edition

Objectives

In this chapter, you will:
- Learn about autoglobal variables
- Build XHTML Web forms
- Process form data
- Handle submitted form data
- Create an All-in-One form
- Display dynamic data based on a URL token

Using Autoglobals

- **Autoglobals** are predefined global arrays that provide information about server, environment, and user input

Array	Description
$_COOKIE	An array of values passed to the current script as HTTP cookies
$_ENV	An array of environment information
$_FILES	An array of information about uploaded files
$_GET	An array of values from a form submitted with the "get" method
$_POST	An array of values from a form submitted with the "post" method
$_REQUEST	An array of all the elements in the $_COOKIE, $_GET, and $_POST arrays
$_SERVER	An array of information about the Web server that served the current script
$_SESSION	An array of session variables that are available to the current script
$GLOBALS	An array of references to all variables that are defined with global scope

Table 4-1 PHP autoglobals

Using Autoglobals (continued)

- Autoglobals are associative arrays
  - To access the values in an associative array, place the element's key in single or double quotation marks inside the array brackets. (the following example displays the SCRIPT_NAME element of the $_SERVER autoglobal)

```
$_SERVER["SCRIPT_NAME"];//displays
the path and name of the current
script
```


Building XHTML Web Forms

- **Web forms** are interactive controls that allow users to enter and submit data to a processing script
- A Web form is a standard XHTML form with two required attributes in the opening `<form>` tag:
  - **Action attribute:** Identifies the program on the Web server that will process the form data when it is submitted
  - **Method attribute:** Specifies how the form data will be sent to the processing script

Adding an `action` Attribute

- The opening form tag requires an `action` attribute
- The value of the action attribute identifies the program on the Web server that will process the form data when the form is submitted

```
<form action="http://www.example.com/
      HandleFormInput.php">
```

Adding the `method` Attribute

- The value of the `method` attribute must be either "post" or "get"
  - The "post" method embeds the form data in the request message
  - The "get" method appends the form data to the URL specified in the form's action attribute
- When a Web form is submitted using the "post" method, PHP automatically creates and populates a `$_POST` array; when the "get" method is used, PHP creates and populates a `$_GET` array

Adding the `method` Attribute (continued)

- Form fields are sent to the Web server as a *name/value* pair
  - The *name* portion of the *name/value* pair becomes the key of an element in the `$_POST` or `$_GET` array, depending on which method was used to submit the data
  - The *value* portion of the name/value pair is populated by the data that the user enters in the input control on the Web form

Adding the `method` Attribute (continued)

- When submitting data using the "get" method, form data is appended to the URL specified by the action attribute
- Name/value pairs appended to the URL are called **URL tokens**

Adding the `method` Attribute (continued)

- The form data is separated from the URL by a question mark (?)
- the individual elements are separated by an ampersand (&)
- the element name is separated from the value by an equal sign (=).
- Spaces in the *name* and *value* fields are encoded as plus signs (+)

Adding the `method` Attribute (continued)

  - all other characters except letters, numbers, hyphens (-), underscores (_) and periods (.) are encoded using a percent sign (%) followed by the two-digit hexadecimal representation of the character's ASCII value
    - (the following code shows three form elements submitted to the process_Scholarship.php script)

```
http://www.example.net/process_Scholarship.php?fName=John&lName=Smith&Submit=Send+Form
```


Adding the `method` Attribute (continued)

- Limitations of the "get" method for submitting form data
  - Restricts the number of characters that can be appended to a single variable to 100
  - The form values are appended to the URL in plain text, making a URL request insecure
- Advantage of the "get" method for submitting form data
  - Passed values are visible in the Address Bar of the browser

Processing Form Data

- A **form handler** is a program or script that processes the information submitted from a Web form
- A form handler performs the following:
  - Verifies that the user entered the minimum amount of data to process the form
  - Validates form data
  - Works with the submitted data
  - Returns appropriate output as a Web page

Retrieving Submitted Data

- The PHP script that processes the user-submitted data is called a **form handler**.
- The values stored in the `$_POST` array can be accessed and displayed by the `echo` statement as shown below:

```
$firstName = $_POST['fName'];
$lastName = $_POST['lName'];
  echo "Thank you for filling out the
scholarship form, ".$firstName."
".$lastName  . ".";
```


Handling Special Characters

- **Magic Quotes** automatically add a backslash character to any single quote, double quote, or `NULL` character contained in form data that a user submits to a PHP script

Figure 4-4 Form input string with magic quotes

Handling Special Characters (continued)

Directive	Description
`magic_quotes_gpc`	Applies magic quotes to any user-submitted data
`magic_quotes_runtime`	Applies magic quotes to runtime-generated data, such as data received from a database
`magic_quotes_sybase`	Applies Sybase-style magic quotes, which escape special characters with a single quote (') instead of a backslash (\)

Table 4-2 Magic quote directives

Handling Special Characters (continued)

- The `addslashes()` function adds a backslash before a single or double quote or a `NULL` character in user input (if magic quotes is disabled, this is the alternative to escape a character before saving to a text file or database)
- The `stripslashes()` function removes a backslash before a single or double quote or `NULL` character in user input (if magic quotes is enabled, this is required before outputting a string with the `echo` statement)

Handling Submitted Form Data

- It is necessary to validate Web form data to ensure PHP can use the data
- The optimal way to ensure valid form data is only allow the user to enter an acceptable response
- Examples of data validation include verifying that
  - the user did not leave any required fields blank
  - an e-mail address was entered in the correct format
  - the user did not exceed the word limit in a comment box

Determining if Form Variables Contain Values

- When form data is posted using the "post" or "get" method, all controls except unchecked radio buttons and checkboxes get sent to the server even if they do not contain data
- The `empty()` function is used to determine if a variable contains a value
- The `empty()` function returns `FALSE` if the variable being checked has a nonempty and nonzero value, and a value of `TRUE` if the variable has an empty or zero value

Validating Entered Data

- Validating form data refers to verifying that the value entered in a field is appropriate for the data type that should have been entered
- The best way to ensure valid form data is to build the Web form with controls (such as check boxes, radio buttons, and selection lists) that only allow the user to select valid responses
- Unique information, such as user name, password, or e-mail must be validated

Validating Numeric Data

- All data in a Web form is string data and PHP automatically converts string data to numeric data if the string is a number
  - The `is_numeric()` function is used to determine if a variable contains a number
  - The `round()` function can be used to a numeric variable with an appropriate number of decimal places

Validating String Data

- Regular expression functions are some of the best tools for verifying that string data meets the strict formatting required for e-mail addresses, Web page URLs, or date values
  - The `stripslashes()` function removes the leading slashes for escape sequences
  - The `trim()` function removes any leading or trailing white space from a string

Handling Multiple Errors

- When processing a Web form, it is best to track any errors on the form during processing and then redisplay the form for the user to correct all the errors at one time

Redisplaying the Web Form

- A **sticky form** is used to redisplay the form with the controls set to the values the user entered the last time the form was submitted
- The following syntax illustrates how to use the value attribute to display previous submitted values in sticky form:

```
<p>First Name: <input type="text"
name="fName" value="<?php echo $firstName;
?>" /></p>
```

Redisplaying the Web Form

- The following syntax illustrates how to use the value attribute to display previous submitted values in sticky form:

```
<p>First Name: <input type="text"
name="fName" value="<?php echo $firstName;
?>" /></p>
```


Emailing the Web Form

- The `mail()` function is used to send an e-mail message containing form data in PHP
- The basic syntax for this function is

  `mail(`*`recipient(s), subject, message`*`)`
- The **Address Specifier** defines the format of the e-mail addresses that can be entered as the recipient argument
  - Plain e-mail address: `jdoe@example.net`
  - Recipients name and e-mail address: `Mary Smith <mary.smith@example.com>`

Emailing the Web Form (continued)

- The `subject` argument of the `mail()` function must include only plain text with no XHTML tags or character entities unless a special MIME format is used
- The `message` argument of the `mail()` function is a text string that must also be in plain text
- A fourth, optional `additional_headers` argument can include headers that are standard in most e-mail editors – From, Cc, Bcc and Date.

Emailing the Web Form (continued)

With the `additional_headers` argument

- Each header must be on its own line
- Each line must start with the header name, followed by a colon, a space, and the value of the header element

```
Date: Fri, 03 Apr 2009 16:05:50 -0400
From: Linda M. Jones
  linda@jones.example.com
CC: Mary R. Jones <mary@jones.example.com>
```

- A successful e-mail message returns a value of `TRUE`

Creating an All-in-One Form

- A **two-part form** has one page that displays the form and one page that processes the form data
- For simple forms that require only minimal processing, it's often easier to use an **All-in-One form**—a single script used display a Web form and process its data

Validating an All-in-One Form

- It uses a conditional to determine if the form as been submitted or if it is being viewed for the first time
  - The `isset()` function is used to determine if the $Submit variable has been set

```
if (isset($Submit)) {
// Validate the data
}
```

  - The argument of the `isset()` function is the name assigned to the Submit button in the Web form

Redisplaying the Web Form

- If the submitted data did not pass all validation checks or no data has been entered, the All-in-One form will display the Web form, for the user to enter data for the first time or re-enter data that did not pass validation

```
if (isset ($_POST['Submit'])) {
// Process the data
}
else {
// Display the Web form
}
```


Displaying Dynamic Content Based on a URL Token

- By passing URL tokens to a PHP script, many different types of information can be displayed from the same script
- By using a Web page template with static sections and a dynamic content section, a single PHP script can produce the same content as multiple static XHTML pages

Using a Web Page Template

- A **Web template** is a single Web page that is divided into separate sections such as
  - Header
  - Button Navigation
  - Dynamic Content
  - Footer
- The contents of the individual sections are populated using include files

Using Text Hyperlinks for Navigation

- When the user clicks on a text hyperlink the contents that display in the dynamic data section of the index.htm (home page) are replaced by the contents referenced by the `href` attribute
- A *name/value* pair is appended to the index URL (this attribute and value will be referenced in the dynamic data section of the index.php file)
  - The name is user defined
  - The value is user defined

```
<a href = "index.php?page=home_page">Home</a>
```


Using Form Image Buttons for Navigation

- Buttons must be enclosed by a opening and closing `<form>` tag

```
<input type = "image" src = "home.jpg" name =
"home" style = "border:0"  alt= "Home" />
```

- x- and y- coordinates are sent in the form *"Button.x"* and "*Button.y*" where "Button" is the value of the name attribute (home)
- In PHP, the periods are replaced by underscores for the `$_GET` or `$_POST` array indexes
- The `$_GET` and `$_POST` array would have two elements "home_x" and "home_y"

Displaying the Dynamic Content

- The `$_REQUEST` autoglobal can be used to access the results from form data sent using either the "get" or "post" methods
  - The syntax to save the value of the page attribute to a variable is shown below:

```
$displayContents = $_REQUEST["page"];
```

- The dynamic content section of the index.php file will contain the code to determine which content page to display

Displaying the Dynamic Content (continued)

```
if (isset($_GET['page'])) {
  switch ($_GET['page']) {
      case 'About Me':
            include('inc_about.html');
            break;
      case 'home'://display the default page
            include('inc_home.html');
            break;
      default:
            include('inc_home.html');
            break;
    }
}
```


Summary

- PHP includes various predefined global arrays, called **autoglobals** or **superglobals**, which contain client, server, and environment information that you can use in your scripts
- **Web forms** are standard XHTML Web pages with interactive controls that allow users to enter data

Summary (continued)

- The `<form>` tag requires an **action attribute** to identify the script that will process the submitted data and a method attribute to identify whether the data will be sent using the “get” or “post” method
- The `$_POST` autoglobal contains data submitted from a form using the “post” method; the `$_GET` autoglobal contains data submitted from a form using the “get” method or through a hyperlink

Summary (continued)

- Web forms may have two components: the data entry form page and the data processing script
- If **Magic Quotes** is enabled, the PHP scripting engine inserts an escape character before a single quotation mark, double quotation mark, or `NULL` character in any submitted form data
- Magic quotes may be enabled for a PHP server

Summary (continued)

- The `addslashes()` function inserts an escape character before a single quotation mark, double quotation mark, or `NULL` character in a string
- The `stripslashes()` function removes the escape character before a single quotation mark, double quotation mark, or `NULL` character in a string
- The first step in processing form data is to validate the input

Summary (continued)

- The `empty()` function determines if the entered value has an empty or zero value
- The `is_*()` family of functions determines if the entered value is of the required data type
- **Regular expressions** determine if an entered string value is formatted correctly for the required type of entry
- The user should be notified of all errors in the values entered into the form

Summary (continued)

- **Sticky forms** are forms that redisplay after an error has been found
- The fields in a sticky form are populated with the values the user entered previously.
- **Advanced escaping from XHTML** is a convenient way to display XHTML code within a PHP code block

Summary (continued)

- The `mail()` function is used to send mail from PHP; it can be used to send form data via e-mail when the form has been successfully completed and validated
- **All-in-One Web forms** combine the data entry form page and the data processing script into a single script
- The `isset()` function determines if the entered value has been initialized (or set)

Summary (continued)

- **URL tokens** use the "get" method and additional data appended to the URL to submit information to a PHP script
- **Web templates** combine static elements and a dynamic content section within a Web page
- Web templates can use the `include()` function within a conditional or switch statement to display dynamic content from different include files within the same section of the template