How to manage a session

Figure 12-7 describes several functions you can use to manage a session. You typically won't need to use these functions, but they are useful in certain situations. In the next figure, for instance, you'll see an example that uses the session_name() function.

You can use the session_name() function to get the name of the session cookie as shown in the first example. By default, this cookie has a name of "PHPSESSID". In addition, you can use the session_id() function to get or set the session ID. Here, the second example gets the session ID, and the third example sets the session ID.

Since the session cookie is often unencrypted, it's possible for an attacker on the network to view the session cookie and to use that data to impersonate the user. This is called *session hijacking*. To help prevent this, you can call the session_regenerate_id() function at key points in your code to change the value of the session ID. For example, you may want to call this function after a user logs in or changes a password. Then, if someone has stolen an old session ID, that ID can't be used to impersonate the user.

In most cases, PHP automatically saves session data when the script ends. However, if you redirect the browser by using the header() function followed by the exit() function, PHP doesn't always save the session data. As a result, when using the header() function to redirect users, you should call the session_write_close() function to force PHP to save the session data. In addition, you must call the session_write_close() function before you call the session_destroy() function that you'll learn about in the next figure.

Functions to manage sessions

Function	Description
`session_name()`	Gets the name of the session cookie. The default is PHPSESSID.
`session_id([$id])`	If the parameter isn't specified, this function gets the current session ID. If no session exists, this function gets an empty string. If the parameter is specified, this function sets the session ID to the specified value.
`session_regenerate_id()`	Creates a new session ID for the current session. Returns TRUE if successful and FALSE otherwise. This function can be used to help prevent session hijacking.
`session_write_close()`	Ends the current session and saves session data. This function is only needed in special cases like redirects.

Get the name of the session cookie

```
$name = session_name();      // By default, PHPSESSID
```

Get the value of the session ID

```
$id = session_id();          // For example, l1jef1foi1g8u6qnui4f8b6e14
```

Set the session ID

```
session_id('abc123');
```

How to end a session

By default, a session ends after 24 minutes without a request. As a result, you often don't need to write any code to end a session. However, in some cases, you may need to write code that ends a session. For example, if a user logs out of your application, you typically want to end the session.

The first example in figure 12-8 shows how to end a session. To start, the first statement removes all session variables from memory by setting the $_SESSION array to an empty array. Then, the second statement cleans up the session ID by calling the session_destroy() function. However, this doesn't delete the session cookie from the user's browser.

As a result, if you want to completely remove the session data from both the client and the server, you need to delete the session cookie as shown in the second example. This code begins by using the session_name() function to get the name of the session cookie. Then, it sets an expiration date of one year in the past. Next, it uses the session_get_cookie_params() function to get an associative array of the values used when creating the session cookie, and it copies four elements from this array to individual variables. Finally, it calls the setcookie() function with these values. Since this call uses an empty string for the $value parameter and a date in the past for the $expire parameter, this deletes the session cookie.

When you use the session_destroy() function, you must call it after the session_start() function. In addition, you shouldn't use it before the session_write_close() function.

A function to end a session

Function	Description
`session_destroy()`	Ends a session. Returns TRUE if successful and FALSE otherwise.

End a session

```
$_SESSION = array();                    // Clear session data from memory
session_destroy();                      // Clean up the session ID
```

Delete the session cookie from the browser

```
$name = session_name();                 // Get name of session cookie
$expire = strtotime('-1 year');         // Create expire date in the past
$params = session_get_cookie_params();  // Get session params
$path = $params['path'];
$domain = $params['domain'];
$secure = $params['secure'];
$httponly = $params['httponly'];
setcookie($name, '', $expire, $path, $domain, $secure, $httponly);
```

Description

- A session ends when the user closes the browser, when a specified amount of time elapses without a request, or when the code calls the session_destroy() function.
- To remove all data associated with the session from the client and the server, you can clear the session data from memory, call the session_destroy() function, and use the setcookie() function to delete the session cookie.
- The session_name() function gets the name of the session cookie. By default, the session cookie has a name of "PHPSESSID".
- The session_get_cookie_params() function gets an associative array that contains all of the parameters for the session cookie. For a list of these parameters, see figure 12-5.

Figure 12-8 How to end a session

The Shopping Cart application

This topic presents an application that illustrates the use of session variables. Without session tracking, the code for this application would be much more complicated.

The user interface

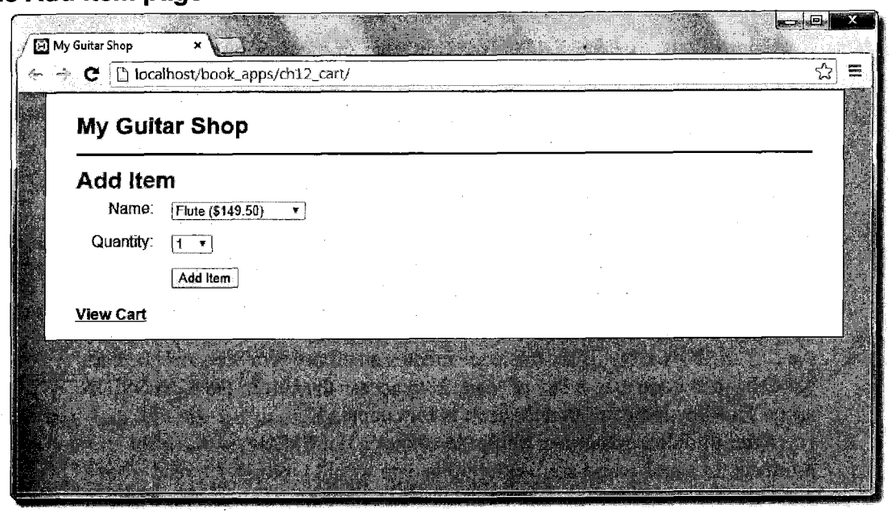
Figure 12-9 shows the user interface for the Shopping Cart application. This application consists of two pages: the Add Item page and the Cart page.

The Add Item page allows the user to add an item to the cart. In addition, it includes a link that allows the user to view the cart without adding an item.

The Cart page displays all items in the user's cart along with a subtotal. From this page, the user can update the quantities for the items in the cart by changing the quantity and clicking the Update Cart button. Or, the user can remove an item by changing the quantity to 0 and clicking the Update Cart button. To add more items, the user can click on the Add Item link to return to the Add Item page. Or, to remove all items from the cart, the user can click on the Empty Cart link.

Since the point of this application is to illustrate the use of cookies and sessions, the user interface isn't realistic. Also, this application doesn't get the product data from a database or store the session in a database. These, of course, are improvements that you would make in a real-world application.

The Add Item page



The Cart page