Listing 12.4.2 **Continued**

```
// Open a database connection (Update these with your own DB server info)
$db = mysql_connect('localhost', 'my_user', 'my_password')
    or die('DB Failure: ' . mysql_error());
mysql_select_db('my_database');

// Do the insert:
$sid = session_id();
mysql_query("
    insert into web_tracker
        (session, addid,
         url, refer)
    values
        ('{$sid}', '{$_SESSION['addid']}',
         '{$_SERVER['REQUEST_URI']}', {$refer})
    ");
?>
```

After you have collected all this data, you can manipulate it however you want. You can look at individual users' paths, average length of stay on the website, or much more. A few things should be noted about this, however. First, unless you force your pages to never be cached by a local browser, a user can click on the Back button, and you will not notice this activity because the PHP does not get run again. Second, you never know when the user has left your website nor where the user went. If you want to know these two things, you need to make sure that all external links actually bounce through a web page on your server, which will allow you to track the access. This can have an additional benefit of allowing you to give disclaimers on the page you are bouncing through that the user is leaving the website, perhaps even with a timer to give the user a few seconds to change his mind.

12.5 Implementing a Simple Shopping Cart

The Web is filled with commercial websites selling their wares. All these websites share the concept of a shopping cart. As you browse the website you can add items to your shopping cart, and when you are ready to check out it will remember them.

A simple shopping cart is no more than a PHP session that keeps track of all the items that someone has clicked on to buy. It is then useful to have a page allowing you to review your shopping cart, make changes, and, eventually, check out. Listings 12.5.1, 12.5.2, and 12.5.3 implement a simple shopping cart.

Listing 12.5.1 **Shopping Cart—Product Information Database, Filename: products.php**

```
<?php
// Just an included file with our product data.  Normally you would read
//  this data in from a database.  You would probably also store much more
```

Listing 12.5.1 **Continued**

```
//  data such as pictures and full product specifications.  For this
//  example we are going much simpler:
$products = array(
    'x563942' => array('desc' => 'Xcast gaming console', 'price' => 674.99),
    'x583954' => array('desc' => 'Xcast controller', 'price' => 24.99),
    'g7' => array('desc' => 'Xcast game: Deaf Fury', 'price' => 45.98),
    'g9' => array('desc' => 'Xcast game: C the Armadillo', 'price' => 19.95),
    's23' => array('desc' => 'Store brand Logo T-Shirt', 'price' => 12.97),
    'c997' => array('desc' => 'Gamerz Rool! Basecall Cap', 'price' => 5),
    );
?>
```

Listing 12.5.2 **Shopping Cart—Product View Page, Filename: view.php**

```
<?php
// Our products page
//  We will display all the products and allow people to choose them:
session_start();

// Include the products library
require 'products.php';

// If we were given an item to add to our cart, do so:
if (isset($_GET['add'])) {
    @$_SESSION['cart'][$_GET['add']]++;
}

// Calculate the total items, and total value, of our cart:
$total_num = 0;
$total_value = 0;
foreach (@$_SESSION['cart'] as $id => $count) {
    $total_value += $products[$id]['price'] * $count;
    $total_num += $count;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>View Products</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<style>
table { border-collapse: collapse; }
.num { text-align: right; }
#cart { float: right; text-align: center; }
td, th, div { border: 1px solid black; padding: 4px;}
```

Listing 12.5.2 **Continued**

```
.button {
    display: block; padding: 2px 5px; background-color: #0000FF;
    color: white; text-decoration: none;
    border-style: solid; border-width: 4px;
    border-bottom-color: #000099; border-right-color: #000099;
    border-top-color: #0066FF; border-left-color: #0066FF;
}
</style>
</head>
<body>
<div id="cart">Items in your cart: <?= $total_num ?>
<br />Total value of cart: <?= $total_value ?>
<br /><a href="cart.php">View your cart</a>
</div>
<p>Please choose what products you may desire below:</p>
<table>
 <tr><th scope="col">Item</th><th scope="col">Name</th>
 <th scope="col">Price</th><th scope="col">Buy it!</th></tr>
<?php
// Loop through all the products and give the options:
$counter = 0;
foreach ($products as $id => $p) {
    // Echo out a table row with this data:
    $counter++;
    echo "<tr><td>{$counter}</td><td>{$p['desc']}</td><td class=\"num\">",
        number_format($p['price'],2), "</td>";

    // Finish off the row by creating a 'Add to Cart' button ...
    echo "<td><a class=\"button\"
href=\"{$_SERVER['PHP_SELF']}?add={$id}\">Add to Cart</a></td></tr>\n";
}
?>
</table>
</body>
</html>
```

Listing 12.5.3 **Shopping Cart—View and Modify the Cart, Filename: cart.php**

```
<?php
// The shopping cart page itself
session_start();

// Include the products library
require 'products.php';
```

Listing 12.5.3 **Continued**

```
// If we were asked to clear the cart, do so immediately.
if (@$_POST['submit'] == 'Clear Cart') {
    unset($_SESSION['cart']);
}
// Otherwise if we have been presented with an update, handle it:
elseif (isset($_POST['update'])) {
    // Loop over all the updates
    foreach ($_POST['update'] as $id => $val) {
        // Only update if the value is numeric or blank - Otherwise ignore.
        $val = trim($val);
        if (preg_match('/^[0-9]*$/', $val)) {
            // If the value is 0 or blank, remove it:
            if ($val == 0) {
                unset($_SESSION['cart'][$id]);
            } else {
                // Otherwise, just reset to the new number
                $_SESSION['cart'][$id] = $val;
            }
        }
    }
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Shopping Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<style>
table { border-collapse: collapse; }
.num { text-align: right; }
td, th, div { border: 1px solid black; padding: 4px;}
</style>
</head>
<body>
<p>The following is in your shopping cart:</p>
<form action="<?= $_SERVER['PHP_SELF'] ?>" method="post">
<table>
 <tr><th scope="col">Item</th><th scope="col">Name</th>
 <th scope="col">Price</th><th scope="col">Qty</th></tr>
<?php
// Loop through all of the current cart:
$counter = 0;
$total = 0;
if (@is_array($_SESSION['cart'])) {
    foreach ($_SESSION['cart'] as $id => $c) {
```

Listing 12.5.3 **Continued**

```
        // Echo out a table row with this data:
        $counter++;
        echo "<tr><td>{$counter}</td><td>{$products[$id]['desc']}</td>",
            "<td class=\"num\">", number_format($products[$id]['price'],2),
            "</td><td><input type=\"text\" size=\"3\" class=\"num\" ",
            "name=\"update[{$id}]\" value=\"{$c}\" /></td></tr>\n";

        // Update our total
        $total += $products[$id]['price'] * $c;
    }
}

// Now echo out the 'total' line, and the 'update/buy options'.
// NOTE:  As it currently stands this Buy button doesn't do anything.
//  That will need implemented to your own personal system.
?>
<tr class="num"><td colspan="3">Total:</td>
 <td><?= number_format($total,2) ?></td></tr>
<tr class="num"><td colspan="4">
<input type="button" value="Keep Shopping"
 onclick="javascript:window.location.href='view.php'" />
<input type="submit" name="submit" value="Update Quantities" />
<input type="submit" name="submit" value="Clear Cart"
 onclick="return confirm('Are you sure you wish to empty your cart?')" />
<input type="button" value="Buy" />
</td></tr>
</table>
</form>
</body>
</html>
```

This example is complete except for the actual purchasing process, which would need to be custom made to your own system. The entire shopping cart process relies completely on session data in this example, and therefore there is no need for the user to have an account in order to shop.

12.6 Passing Session Data Between Two Servers

One intrinsic limitation of PHP sessions is that they are unique to that particular web server. If you have reason to use more than one web server and yet need to share data between them since they are related, sessions will let you down.

There is, however, a trick you can use to accomplish this, though it should not be done if you are storing any sensitive data in the session. This will also only work if both machines share the same subdomain. You can convert all the session data from one server into a cookie that you tell to be presented to all machines in your subdomain. Then the