Listing 12.1.2 **Continued**

```
if (isset($_REQUEST['firstname']) && isset($_REQUEST['zipcode'])) {
    // Update or create the cookies, all set to expire in 1 hour:
    setcookie('firstname', $_REQUEST['firstname'], time()+3600);
    setcookie('zipcode', $_REQUEST['zipcode'], time()+3600);

    // Now, also, just print the standard output:
    echo "
<p>Welcome back {$_REQUEST['firstname']}!  Perhaps you would like to see
 things happening near your home zipcode ({$_REQUEST['zipcode']}) ?</p>
";
} else {
    // We have nothing, so make a form so they can provide data.
    echo "
<form action=\"{$_SERVER['PHP_SELF']}\" method=\"POST\">
<p>What is your first name? <input type=\"text\" name=\"firstname\" /></p>
<p>What is your zipcode? <input type=\"text\" name=\"zipcode\" /></p>
<p><input type=\"submit\" /></p>
</form>
";
}
?>
```

12.2 Saving User Data with Sessions

The major drawbacks of using cookies for data storage are that you rely on the person allowing the cookie in the first place, and the amount of data you can store is somewhat limited. PHP sessions are a solution to those problems and many more. Sessions are a way of storing data locally within the web server and tying it to a specific user of the website. They are not meant to be permanent by their nature and instead are meant to persist through a "session"—that is, as long as a web browser is open.

They work by storing PHP variables on the back end and then presenting the client with a session id. This is usually via a cookie but can also be via a `GET` parameter in the URL. By calling `session_start()`, PHP takes care of all this for you. After doing that, you can reference session variables by the superglobal `$_SESSION`, and any changes you make to `$_SESSION` will be automatically updated when the page is finished.

Sessions are an excellent way, for example, to track a user who has used a username/password to log in to your website. You can start a session for the user, store data that she has successfully logged in, and therefore not prompt the user for a password again.

Listings 12.2.1, 12.2.2, and 12.2.3 present an example of a mini website with logins. Three separate web pages make up the site: A login page, which redirects you if you are already logged in; a "view" page that shows you some information, including session information; and finally an "edit" page that allows you to create this session data.

Listing 12.2.1 **Website Login Example—The Login Page, Filename: login.php**

```
<?php
// The login page -- First check if the user has requested to log-off
$error = '';
session_start();
if (isset($_GET['logoff'])) {
    // We need to completely destroy the session.  First the data:
    $_SESSION = array();

    // If a session cookie exists, tell the browser to destroy it
    //  (give it a time in the past)
    if (isset($_COOKIE[session_name()])) {
        setcookie(session_name(), '', time()-1000, '/');
    }

    // Finally, finialize the session destruction:
    session_destroy();
}
// Secondly, see if the user is already logged in.
elseif (isset($_SESSION['valid']) && $_SESSION['valid']) {
    // Bounce them to the view page:
    header('Location: view.php');
    exit();
}
// 3rd, see if they attempted to log in:
elseif (isset($_POST['user']) || isset($_POST['pass'])) {
    // Just allowing a couple of fake users here, in real life you would
    //  probably be reading this information from a database.
    $user = isset($_POST['user']) ? $_POST['user'] : '';
    $pass = isset($_POST['pass']) ? $_POST['pass'] : '';
    if ( ( ($user == 'tanderson') && ($pass == 'Z1ON0101') ) ||
         ( ($user == 'asmith') && ($pass == 'brawl') ) ) {
        // user/pass is good, Store this fact in the session
        $_SESSION['valid'] = 1;
        $_SESSION['user'] = $_POST['user'];

        // and direct them to the main page
        header('Location: view.php');
        exit();
    } else {
        // Prepare an error statement:
        $error = 'Username & Password do not match, please try again';
    }
}
// Otherwise, let's ask them to log in:
?>
```

Listing 12.2.1 **Continued**

```
<!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>Login</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<style>
form { border: 1px solid black; padding: 10px; }
#error { color: #FF0000; font-weight: bold; }
</style>
</head>
<body>
<?php // If we had an error:
if ($error) { echo "<p id=\"error\">{$error}</p>\n"; }
?>
<form action="<?= $_SERVER['PHP_SELF'] ?>" method="post">
<p>Welcome to our website, you need to log in:</p>
<p>Username: <input type="text" name="user" value="<?= @$user ?>"><br />
Password: <input type="password" name="pass" value="<?= @$pass ?>"></p>
<p><input type="submit" value="Login" /></p>
</form>
</body>
</html>
```

Listing 12.2.2 **Website Login Example—The Information View Page, Filename: view.php**

```
<?php
// The View page
//  First we need to check if they are logged in, and if not, make them:
session_start();
if (!(isset($_SESSION['valid']) && $_SESSION['valid'])) {
    // Back to login for you
    header('Location: login.php');
    exit();
}

// Now, set some default values in case they haven't told us anything yet:
$fname = isset($_SESSION['name']) ? $_SESSION['name'] : '[Unknown]';
$fcolor = isset($_SESSION['color']) ? $_SESSION['color'] : '[Unknown]';
?>
<!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</title>
```