times. If you don't, you can end up with duplicate records. This can be caused by a careless user who submits a form once but then clicks reload, causing the data to be submitted a second time. Worse is the case of a spammer attacking your website by rapidly submitting the same form again and again. This not only creates an ugly cleanup problem to solve but also eats precious site resources.

Just redirecting the user to another web page after the form submission is complete can solve the accidental case; however, there is a solution that fulfills both needs. This solution uses sessions, which are explored in depth in Chapter 12, "Sessions and User Tracking." The method is fairly simple. When the user accesses the form page, a session is started in which a variable indicates that the form submission will be valid from this browser. Then on the submission page, the script checks this variable, and if it exists, the submission is allowed and the variable is cleared. If the page is loaded again, an error will be generated, because the variable stating that it was valid is not there. Listing 10.9.1 implements the form page of this pair, and Listing 10.9.2 is the page that accepts the submission.

Listing 10.9.1 **Preventing Multiple Form Submissions, the Basic Form**

```
<?php
// Begin a session, and save a 'validity' variable to it:
session_start();

// If the validsubmit session variable doesn't exist, create it to be true:
if (!isset($_SESSION['validsubmit'])) {
    $_SESSION['validsubmit'] = true;
}
?>
<!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>Data Entry Form</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<form action="submit.php" method="post" name="f1">
<p>Please enter your name: <input name="name" type="text" /></p>
<p><input type="submit" /></p>
</form>
</body>
</html>
```

Listing 10.9.2 **Preventing Multiple Form Submissions, the Submission Page—Filename: submit.php**

Listing 10.9.2 **Continued**

```
<?php
// Open the session
session_start();
?>
<!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>Data Processing Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<p>
<?php
// If this session is not valid, say so:
if (!isset($_SESSION['validsubmit']) || !$_SESSION['validsubmit']) {
    echo "ERROR:  Invalid form submission, or form already submitted!";
} else {
    // This was valid, so first of all clear the validity:
    $_SESSION['validsubmit'] = false;

    // Now echo out the data:
    echo "Your name is {$_POST['name']}?  That's a nice name.";
}
?>
</p>
</body>
</html>
```

This method can still be circumvented by a user willing to click back, reload the page, re-enter data, and submit again. It can also fall prey to a script that is smart enough to load the form page first, receive the session cookie, and then return that session cookie upon the form submission. It does, however, stop the simple and common forms of abuse. Taking it one step further would be to check against the IP address and only allow so many submissions, per IP address, per a time period. This can backfire, however, because many ISPs place all their users behind dynamic IP systems that make thousands of users look like one IP address.

This concept can be extended to attempt to only allow one submission per user by also storing a cookie on the user's machine that shows that the user already submitted that page. However, because users can edit or delete their cookies, that is not reliable. The only reliable way to make only one submission, per user, ever, is to have users sign up for accounts and restrict posting to one per account.