

Script 8.3 The registration script adds a record to the database by running an INSERT query.

```

1  <?php # Script 8.3 - register.php
2
3  $page_title = 'Register';
4  include ('includes/header.html');
5
6  // Check if the form has been submitted:
7  if (isset($_POST['submitted'])) {
8
9      $errors = array(); // Initialize an
      error array.
10
11     // Check for a first name:
12     if (empty($_POST['first_name'])) {
13         $errors[] = 'You forgot to enter your
         first name.';
14     } else {
15         $fn = trim($_POST['first_name']);
16     }
17
18     // Check for a last name:
19     if (empty($_POST['last_name'])) {
20         $errors[] = 'You forgot to enter your
         last name.';
21     } else {
22         $ln = trim($_POST['last_name']);
23     }
24
25     // Check for an email address:
26     if (empty($_POST['email'])) {
27         $errors[] = 'You forgot to enter your
         email address.';
28     } else {
29         $e = trim($_POST['email']);
30     }
31
32     // Check for a password and match
     against the confirmed password:
33     if (!empty($_POST['pass1'])) {
34         if ($_POST['pass1'] !=
         $_POST['pass2']) {

```

(script continues on next page)

To execute simple queries:

1. Create a new PHP script in your text editor or IDE (**Script 8.3**).

```

<?php # Script 8.3 - register.php
$page_title = 'Register';
include ('includes/header.html');

```

The fundamentals of this script—using included files, having the same page both display and handle a form, and creating a sticky form—come from Chapter 3. See that chapter if you're confused about any of these concepts.

2. Create the submission conditional and initialize the `$errors` array.

```

if (isset($_POST['submitted'])) {
    $errors = array();

```

This script will both display and handle the HTML form. This conditional will check for the presence of a hidden form element to determine whether or not to process the form. The `$errors` variable will be used to store every error message (one for each form input not properly filled out).

continues on next page

3. Validate the first name.

```

if (empty($_POST['first_name'])) {
    $errors[] = 'You forgot to enter
    → your first name.';
} else {
    $fn =
    → trim($_POST['first_name']);
}

```

As discussed in Chapter 3, the `empty()` function provides a minimal way of ensuring that a text field was filled out. If the first name field was not filled out, an error message is added to the `$errors` array. Otherwise, `$fn` is set to the submitted value, after trimming off any extraneous spaces. By using this new variable—which is obviously short for *first_name*—I make it syntactically easier to write the query later.

4. Validate the last name and email address.

```

if (empty($_POST['last_name'])) {
    $errors[] = 'You forgot to enter
    → your last name.';
} else {
    $ln = trim($_POST['last_name']);
}

if (empty($_POST['email'])) {
    $errors[] = 'You forgot to enter
    → your email address.';
} else {
    $e = trim($_POST['email']);
}

```

These lines are syntactically the same as those validating the first name field. In both cases a new variable will be created, assuming that the minimal validation was passed.

Script 8.3 *continued*

```

35     $errors[] = 'Your password did not
        match the confirmed password.';
36 } else {
37     $p = trim($_POST['pass1']);
38 }
39 } else {
40     $errors[] = 'You forgot to enter your
        password.';
41 }
42
43 if (empty($errors)) { // If everything's
    OK.
44
45     // Register the user in the
        database...
46
47     require_once
        ('../mysqli_connect.php'); // Connect
        to the db.
48
49     // Make the query:
50     $q = "INSERT INTO users (first_name,
        last_name, email, pass,
        registration_date) VALUES ('$fn',
        '$ln', '$e', SHA1('$p'), NOW())";
51     $r = @mysqli_query ($dbc, $q); // Run
        the query.
52     if ($r) { // If it ran OK.
53
54         // Print a message:
55         echo '<h1>Thank you!</h1>
56         <p>You are now registered. In Chapter
            11 you will actually be able to log
            in!</p><p><br /></p>';
57
58     } else { // If it did not run OK.
59
60         // Public message:
61         echo '<h1>System Error</h1>
62         <p class="error">You could not be
            registered due to a system error.
            We apologize for any
            inconvenience.</p>';
63

```

(script continues on next page)

Script 8.3 *continued*

```

64      // Debugging message:
65      echo '<p>' . mysqli_error($dbc) .
        '<br /><br />Query: ' . $q .
        '</p>';
66
67      } // End of if ($r) IF.
68
69      mysqli_close($dbc); // Close the
        database connection.
70
71      // Include the footer and quit the
        script:
72      include ('includes/footer.html');
73      exit();
74
75      } else { // Report the errors.
76
77      echo '<h1>Error!</h1>
78      <p class="error">The following
        error(s) occurred:<br />';
79      foreach ($errors as $msg) { // Print
        each error.
80          echo " - $msg<br />\n";
81      }
82      echo '</p><p>Please try
        again.</p><p><br /></p>';
83
84      } // End of if (empty($errors)) IF.
85
86      } // End of the main Submit conditional.
87      ?>
88      <h1>Register</h1>
89      <form action="register.php" method="post">
90          <p>First Name: <input type="text"
            name="first_name" size="15"
            maxlength="20" value="<?php if
            (isset($_POST['first_name'])) echo
            $_POST['first_name']; ?>" /></p>
91          <p>Last Name: <input type="text"
            name="last_name" size="15"
            maxlength="40" value="<?php if
            (isset($_POST['last_name'])) echo
            $_POST['last_name']; ?>" /></p>

```

*(script continues on next page)***5.** Validate the password.

```

if (!empty($_POST['pass1'])) {
    if ($_POST['pass1'] !=
        → $_POST['pass2']) {
        $errors[] = 'Your password
        → did not match the
        → confirmed password.';
    } else {
        $p = trim($_POST['pass1']);
    }
} else {
    $errors[] = 'You forgot to enter
    → your password.';
}

```

To validate the password, the script needs to check the *pass1* input for a value and then confirm that the *pass1* value matches the *pass2* value (so the password and confirmed password are the same).

6. Check if it's OK to register the user.

```

if (empty($errors)) {
    If the submitted data passed all of the
    conditions, the $errors array will have
    no values in it (it will be empty), so this
    condition will be TRUE and it's safe to add
    the record to the database. If the $errors
    array is not empty, then the appropriate
    error messages should be printed (see
    Step 10) and the user given another
    opportunity to register.

```

continues on next page

7. Add the user to the database.

```
require_once
→ ('../mysqli_connect.php');

$q = "INSERT INTO users (first_name,
→ last_name, email, pass,
→ registration_date) VALUES ('$fn',
→ '$ln', '$e', SHA1('$p'), NOW() )";

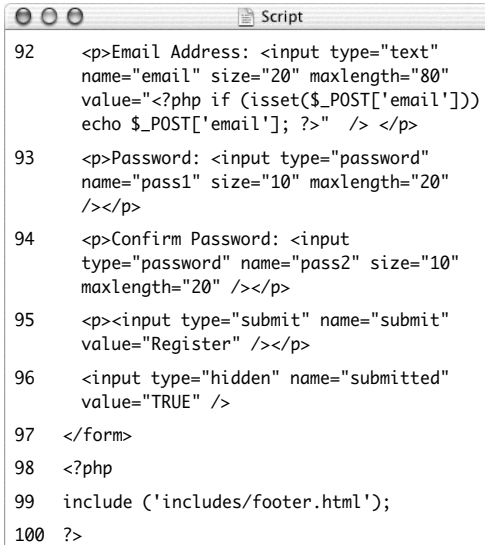
$r = @mysqli_query ($dbc, $q);
```

The first line of code will insert the contents of the `mysqli_connect.php` file into this script, thereby creating a connection to MySQL and selecting the database. You may need to change the reference to the location of the file as it is on your server (as written, this line assumes that `mysqli_connect.php` is in the parent folder of the current folder).

The query itself is similar to those demonstrated in Chapter 5. The `SHA1()` function is used to encrypt the password, and `NOW()` is used to set the registration date as this moment.

After assigning the query to a variable, it is run through the `mysqli_query()` function, which sends the SQL command to the MySQL database. As in the `mysqli_connect.php` script, the `mysqli_query()` call is preceded by `@` in order to suppress any ugly errors. If a problem occurs, the error will be handled more directly in the next step.

Script 8.3 continued



```
92  <p>Email Address: <input type="text"
    name="email" size="20" maxlength="80"
    value="<?php if (isset($_POST['email']))
    echo $_POST['email']; ?>" /> </p>
93  <p>Password: <input type="password"
    name="pass1" size="10" maxlength="20"
    /></p>
94  <p>Confirm Password: <input
    type="password" name="pass2" size="10"
    maxlength="20" /></p>
95  <p><input type="submit" name="submit"
    value="Register" /></p>
96  <input type="hidden" name="submitted"
    value="TRUE" />
97  </form>
98  <?php
99  include ('includes/footer.html');
100 ?>
```

8. Report on the success of the registration.

```

if ($r) {
    echo '<h1>Thank you!</h1>
    <p>You are now registered. In
    → Chapter 11 you will actually be
    → able to log in!</p><p><br
    → /></p>';
} else {
    echo '<h1>System Error</h1>
    <p class="error">You could not be
    → registered due to a system
    → error. We apologize for any
    → inconvenience.</p>';
    echo '<p>' . mysqli_error($dbc) .
    → '<br /><br />Query: ' . $q .
    → '</p>';
}

```

The `$r` variable, which is assigned the value returned by `mysqli_query()`, can be used in a conditional to indicate the successful operation of the query.

If `$r` is TRUE, then a *Thank you!* message is displayed (**Figure 8.8**). If `$r` is FALSE, error messages are printed. For debugging purposes, the error messages will include both the error spit out by MySQL (thanks to the `mysqli_error()` function) and the query that was run (**Figure 8.9**). This information is critical to debugging the problem.

continues on next page



Figure 8.8 If the user could be registered in the database, this message is displayed.

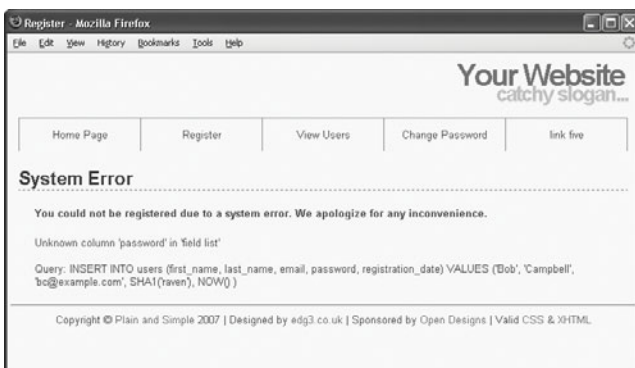


Figure 8.9 Any MySQL errors caused by the query will be printed, as will the query that was being run.

9. Close the database connection and complete the HTML template.

```
mysqli_close();
include ('includes/footer.html');
exit();
```

Closing the connection isn't required but is a good policy. Then the footer is included and the script terminated (thanks to the `exit()` function). If those two lines weren't here, then the registration form would be displayed again (which isn't necessary after a successful registration).

10. Print out any error messages and close the submit conditional.

```
    } else {
        echo '<h1>Error!</h1>
        <p class="error">The
        → following >>error(s)
        → occurred:<br />';
        foreach ($errors as
$msg) {
            echo " - $msg<br
/>\n";
        }
        echo '</p><p>Please try
        → >again.</p><p><br
        /></p>';
    }
}
```

The `else` clause is invoked if there were any errors. In that case, all of the errors are displayed using a `foreach` loop (**Figure 8.10**).

The final closing curly brace closes the main submit conditional. The main conditional is a simple `IF`, not an `if-else`, so that the form can be made sticky (again, see Chapter 3).

Error!

The following error(s) occurred:

- You forgot to enter your last name.
- You forgot to enter your email address.
- Your password did not match the confirmed password.

Please try again.

Register

First Name:

Figure 8.10 Each form validation error is reported to the user so that they may try registering again.

11. Close the PHP section and begin the HTML form.

```
?>
<h1>Register</h1>
<form action="register.php"
→ method="post">
    <p>First Name: <input
    → type="text" name="first_name"
    → size="15" maxlength="20"
    → value="<?php if
    → (isset($_POST['first_name']))
    → echo $_POST['first_name']; ?>"
    → /></p>
    <p>Last Name: <input type="text"
    → name="last_name" size="15"
    → maxlength="40" value="<?php if
    → (isset($_POST['last_name']))
    → echo $_POST['last_name']; ?>"
    → /></p>
```

The form is really simple, with one text input for each field in the *users* table (except for the *user_id* column, which will automatically be populated). Each input is made sticky, using the code

```
value="<?php if
→ (isset($_POST['first_name']))
echo
→ $_POST['first_name']; ?>"
```

Also, I would strongly recommend that you use the same name for your form inputs as the corresponding column in the database where that value will be stored. Further, you should set the maximum input length in the form equal to the maximum column length in the database. Both of these habits help to minimize errors.

12. Complete the HTML form.

```
<p>Email Address: <input
→ type="text" name="email"
→ size="20" maxlength="80"
→ value="<?php if
→ (isset($_POST['email'])) echo
→ $_POST['email']; ?>" /> </p>
<p>Password: <input
→ type="password" name="pass1"
→ size="10" maxlength="20" /></p>
<p>Confirm Password: <input
→ type="password" name="pass2"
→ size="10" maxlength="20" /></p>
<p><input type="submit"
→ name="submit" value="Register"
→ /></p>
<input type="hidden"
→ name="submitted" value="TRUE"
/>
</form>
```

This is all much like that in Step 11. A submit button and a hidden input are in the form as well. The hidden input trick is discussed in (you guessed it...Chapter 3).

As a side note, I don't need to follow my `maxlength` recommendation (from Step 11) with the password inputs, because they will be encrypted with `SHA1()`, which always creates a string 40 characters long. And since there are two of them, they can't both use the same name as the column in the database.

13. Complete the template.

```
<?php
include ('includes/footer.html');
?>
```

continues on next page

14. Save the file as `register.php`, place it in your Web directory, and test it in your Web browser.

Note that if you use an apostrophe in one of the form values, it will likely break the query (**Figure 8.11**). The section “Ensuring Secure SQL” later in this chapter will show how to protect against this.

✓ Tips

- After running the script, you can always ensure that it worked by using the `mysql` client or `phpMyAdmin` to view the values in the `users` table.
- You should not end your queries with a semicolon in PHP, as you did when using the `mysql` client. When working with MySQL, this is a common, albeit harmless, mistake to make. When working with other database applications (Oracle, for one), doing so will make your queries unusable.
- As a reminder, the `mysqli_query()` function returns `TRUE` if the query could be executed on the database without error. This does not necessarily mean that the result of the query is what you were expecting. Later scripts will demonstrate how to more accurately gauge the success of a query.

- You are not obligated to create a `$q` variable as I tend to do (you could directly insert your query text into `mysqli_query()`). However, as the construction of your queries becomes more complex, using a variable will be the only option.
- Practically any query you would run in the `mysql` client can also be executed using `mysqli_query()`.
- Another benefit of the Improved MySQL Extension over the standard extension is that the `mysqli_multi_query()` function lets you execute multiple queries at one time. The syntax for doing so, particularly if the queries return results, is a bit more complicated, so see the PHP manual if you have this need.

System Error

You could not be registered due to a system error. We apologize for any inconvenience.

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Toole', 'pete@example.com', SHA1('venus'), NOW())' at line 1

Query: INSERT INTO users (first_name, last_name, email, password, registration_date) VALUES ('Peter', 'OToole', 'pete@example.com', SHA1('venus'), NOW())

Figure 8.11 Apostrophes in form values (like the last name here) will conflict with the apostrophes used to delineate values in the query.