**Adding the Customer Data**

Once you receive the form data (either as encrypted data or plain text), you need to create a new record in the customers table. The addcust.php file does this for us. Let's build that file now.

1. Create the file *addcust.php* in the store folder of your application area.
2. Open the file with a text editor and add the following code:

<?php

session\_start();

include("mylibrary/login.php");

login();

$firstname = $\_POST['firstname'];

$lastname = $\_POST['lastname'];

$address = $\_POST['address'];

$city = $\_POST['city'];

$state = $\_POST['state'];

$zip = $\_POST['zip'];

$phone = $\_POST['phone'];

$email = $\_POST['email'];

$password1 = $\_POST['password1'];

$password2 = $\_POST['password2'];

if (get\_magic\_quotes\_gpc())

{

$firstname = stripslashes($firstname);

$lastname = stripslashes($lastname);

$address = stripslashes($address);

$city = stripslashes($city);

$state = stripslashes($state);

$zip = stripslashes($zip);

$phone = stripslashes($phone);

$email = stripslashes($email);

$password1 = stripslashes($password1);

$password2 = stripslashes($password2);

}

$firstname = mysql\_real\_escape\_string($firstname);

$lastname = mysql\_real\_escape\_string($lastname);

$address = mysql\_real\_escape\_string($address);

$city = mysql\_real\_escape\_string($city);

$state = mysql\_real\_escape\_string($state);

$zip = mysql\_real\_escape\_string($zip);

$phone = mysql\_real\_escape\_string($phone);

$email = mysql\_real\_escape\_string($email);

$password1 = mysql\_real\_escape\_string($password1);

$password2 = mysql\_real\_escape\_string($password2);

$baduser = 0;

if (trim($email) == '')

$baduser = 1;

if (trim($password1) == '')

$baduser = 2;

if ($password1 != $password2)

$baduser = 3;

$query = "SELECT \* from customers where email = '$email'";

$result = mysql\_query($query);

$rows = mysql\_num\_rows($result);

if ($rows != 0)

$baduser = 4;

if ($baduser == 0)

{

$query = "INSERT INTO customers (firstname, lastname, address, city, state, " .

"zip, phone, email, password) VALUES ('$firstname', '$lastname' , " .

" '$address', '$city', '$state', '$zip' , '$phone', '$email', " .

" PASSWORD('$password1'))";

$result=mysql\_query($query);

if ($result)

{

$query = "SELECT LAST\_INSERT\_ID() from customers";

$result = mysql\_query($query);

$row = mysql\_fetch\_array($result);

$\_SESSION['cust'] = $row[0];

header("Location: index.php?content=confirmorder");

}

else

{

echo "<h2>Sorry, I could not process your form at this time</h2>\n";

}

} else

{

switch($baduser)

{

case(1):

echo "<h2>Please enter an e-mail address</h2>\n";

break;

case(2):

echo "<h2>Please enter a password</h2>\n";

break;

case(3):

echo "<h2>Your passwords did not match!</h2>\n";

break;

case(4):

echo "<h2>I'm sorry, that e-mail address already exists.</h2>\n";

}

echo "<a href=\"index.php?content=newcust\">Try again</a>\n";

}

?>

1. Save the file and exit the text editor.

You should recognize this code. However, I threw in a couple of tricks that you may not be familiar with. Let's take a closer look at those.

**Verifying User Data**

The first thing the program does is the standard data normalization for the MySQL query. You should remember this technique from back in the back-end application. All of the data retrieved from the HTML form is checked for the proper MySQL format, and escape characters are added as necessary using the mysql\_real\_escape\_string() function.

Following that, you'll notice that I perform a few checks on the data. I create a PHP variable called *$baduser* to track if any of the checks fail. A different value is stored in the $baduser variable if the check fails. At the end of the checks, all you need to do is check the final value of the $baduser variable. If it's still zero, you know all of the checks passed. If it has a non-zero value, then you know something failed.

You could check the value of the $baduser variable using multiple if-then-else statements. However, this would take several layers of "ifs" to process and would look pretty messy.

A better way to check the value of a variable is to use the *switch-case* statement. The switch-case statement defines a variable to use as the switch and then allows you to define actions to take based on the value contained in the switch variable. The PHP preprocessor jumps to the appropriate case statement that contains the value in the variable and then processes the statements from there.

At the end of each case section, you need to include the *break* statement. The break statement tells PHP to jump out of the switch section instead of continuing with the next statements.

**Getting the New Customer ID**

Once the new customer has registered on the system, you'll need to store the new customer ID value in the *cust* session cookie, just as we did for a returning customer. However, there's a small problem with that.

When you use the INSERT statement to create a new customer record, the MySQL server uses the auto\_increment attribute of the custid data field to assign a new customer ID value. Unfortunately, it doesn't tell you what that value is. You need to go back to the database to retrieve it.

This can cause a dilemma. How do you know what customer ID value the server assigned to your record? One method would be to create a SELECT statement, retrieving the custid value for the e-mail address you have. The method I chose utilizes a little MySQL server trick.

The LAST\_INSERT\_ID() function returns the most recently generated auto\_increment value on a per-connection basis. This means that even if you have hundreds of customers registering at the same time, you're guaranteed to get the last ID that was generated by the current connection. This is a great way to retrieve the auto\_increment value quickly for an inserted record.

After retrieving the new customer's ID, addcust.php stores it in the cust session cookie. It redirects the customer to the next step in the process, which is to confirm the order using the confirmorder.inc.php file that you saw earlier for returning customers. This takes us full circle in the checkout process. The application validates both new and returning customers and is ready to process the order. We'll save that part for the next lesson.

Let's take a break from the customer checkout process for today. Chapter 5 will wrap things up and get us ready for the next lesson.