**Registering a New Customer**

When you have a brand-new customer, you'll need to get their basic information, such as first and last names and shipping address. Besides this information, you'll also want to create a login account for them so they can be a returning customer the next time they shop on your site. Often sites use an e-mail address as a login ID. This not only makes it easier for the customer to remember, but it gives you an easy way of acquiring customer contact information for later advertisements!

When the customer clicks the button to register in the checkout.inc.php Web page, it sends him or her to the main index.php page, setting the content variable to *newcust*. This link uses the *newcust.inc.php* include file in the main section of the Web page. Let's build that file and then take a look at what it does.

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

<h2>Welcome, new customer!</h2><br><br>

<p>Please fill out this form so we can send you your products<p>

<form action="addcust.php" method="post">

<table width="50%" cellpadding="1" border="1">

<tr>

<td>First name</td>

<td><input type="text" size="40" name="firstname"></td>

</tr>

<tr>

<td>Last name</td>

<td><input type="text" size="40" name="lastname"></td>

</tr>

<tr>

<td>Address</td>

<td><input type="text" size="60" name="address"></td>

</tr>

<tr>

<td>City</td>

<td><input type="text" size="30" name="city"></td>

</tr>

<tr>

<td>State</td>

<td><input type="text" size="2" name="state"></td>

</tr>

<tr>

<td>Zip</td>

<td><input type="text" size="5" name="zip"></td>

</tr>

<tr>

<td>Phone</td>

<td><input type="text" size="15" name="phone"></td>

</tr>

<tr>

<td>e-mail address</td>

<td><input type="text" size="60" name="email"></td>

</tr>

<tr>

<td>password</td>

<td><input type="password" size="15" name="password1"></td>

</tr>

<tr>

<td>Confirm password</td>

<td><input type="password" size="15" name="password2"></td>

</tr>

</table>

<input type="submit" name="button" value="Submit form">

</form>

1. Save the file and exit the text editor.

That's a lot of code, but it's all pretty basic. It just creates a single HTML form, with textboxes for all of the data you need to collect from the customer. Note the use of the password input types again to help hide the passwords from view.

The new customer registration page

After the customer fills out the necessary information and clicks the submit button, the information is sent to the Web page defined in the *action* attribute of the form. Here's another place that may need some security in your application.

**HTML Form Security**

Your new customer registration form requests personal information from the customer. These days it's not always a good idea to send that information across the Internet as plain text. Instead, you should use encryption to hide the data as the client sends it.

HTTP is a text-based protocol. The transfer of all data that the server sends to the client, as well as data the client returns to the server is in plain text format. This provides a window of opportunity for someone to monitor and steal your customer's personal information.

However, there's a way to encrypt the data before sending it across an HTTP connection. After receiving the encrypted data on the other end, it must be decrypted before the client or the Web server can process it.

A secure HTTP connection (called *HTTPS*) provides a method for using an encryption key on the server for encrypting data sent by the client to the server. This is called a *public encryption key*. The server shares its encryption key with the client (the reason it's called *public*), which uses the key to encrypt the data. When the server receives the encrypted data, it uses a *private encryption key* to decrypt the data. Only the private encryption key can decrypt data encrypted using the public encryption key.

Using public and private encryption keys

There are two types of private and public encryption keys:

* A self-signed encryption key
* An authority-signed encryption key

Just because a site sends a public encryption key, it doesn't mean you can trust it. There's a method of authenticating encryption keys in which a trusted security company certifies that an encryption key is valid. This is called *signing*.

When a trusted security company signs an encryption key, it validates that the encryption key belongs to the issuing company. When your browser receives a signed public encryption key, it can verify the signature on the key and, thus, verify the authenticity of the encryption key.

Most ISPs provide encrypted hosting, but usually for an additional fee. With encrypted hosting, you receive the public and private encryption keys to encrypt and decrypt data for your Web site. Unfortunately, the WampServer doesn't support this by default, so you won't be able to test it on your development server.

**Posting Data Using HTTPS**

When you connect to a secure Web site, you must use a different protocol specification in the URL. For a normal HTTP Web site, you usually create a URL like:

http://localhost/store/

For the customer's browser to send data using the encryption key, they must specify the *HTTPS* protocol name in the URL:

https://localhost/store/

However, you don't need to use a secure Web connection for the entire shopping session. All you need to secure is the personal information that the customer sends to your host.

If your Web site supports encryption, you can easily do this by modifying the action attribute of the HTML form:

<form action="https://servername/addcust.php" method="post">

By specifying the HTTPS form of the Web page in the form action attribute, you force the client's browser to send the data using the public encryption key your server provides, even if they used the HTTP format to access the original Web page for the form. Note that when you use this format you need to include the full servername in your URL.

After the addcust.php file does its thing, it can continue the Web session using the normal HTTP method. Let's continue on to Chapter 4 and look at the addcust.php program.