**Tracking Products**

Okay, so now you have a customer browsing through the online store with products in his or her shopping cart. However, you'll need to address a small challenge.

Have you ever been in a grocery store and seen a cart full of groceries sitting off to the side with no one attending it? The same thing can happen in our online store. A customer can fill up a shopping cart with products and then walk away from the PC for a while.

This can cause problems with your inventory control, depending on how you handle ordering products. Trying to track product inventory is a notoriously difficult thing for online retailers. Let's take some time out from coding to look at how to deal with this.

**Handling Session Cookies**

When you produced the addtocart.inc.php code, you checked the quantity in stock of a product before allowing a customer to put the items in the shopping cart. The scenario has one small problem that will need your attention.

The Food Store application attempts to keep track of the entire inventory in the store. The products table tracks the quantity in stock value for each product. Before allowing a customer to purchase a product, you check to make sure there are enough in stock to satisfy the purchase. In the addtocart.inc.php example, you'll notice that we don't do anything to the quantity value in the products table once we allow the customer to place the item in the shopping cart. There's a reason for this.

As customers browse around in the online store, they may place products in the shopping cart and then remove them. If you modified the products table every time a customer places or removes a product, you'd have tons of database updates.

To reduce this problem, the Food Store application uses a slightly different method of tracking inventory:

* It checks to see if an item is in inventory before allowing a customer to place it in the shopping cart, but it doesn't update the database.
* It waits until the customer checks out before reducing the products table inventory by what's in the shopping cart.

This process helps reduce the number of transactions in the database, as you only need to update the product quantity when the customer finally checks out from the store. By reducing the number of transactions, you greatly increase the performance of the application. However, it also creates a small window for trouble.

During the time between when a customer places an item in the shopping cart and checks out, the inventory of the product in the products table is inaccurate. The store application doesn't account for products sitting in a customer's shopping cart. Those items are in limbo, as the official store inventory is concerned.

To get a better understanding of what can happen, let's walk through an example:

1. The products table shows there are three apples in inventory for the store.
2. Customer A places three apples in her cart. The addtocart.inc.php program checks to see that there are three apples in stock according to the products table. So far, so good.
3. Customer B browses through the store and places three apples in his cart. Since customer A hasn't checked out yet, the products table still thinks there are three apples in stock. So customer B is also allowed to place three apples in his cart.
4. Customer B checks out of the store and purchases the items in his shopping cart. At this time, the program reduces the products table quantity for apples by three, leaving no more apples in stock.
5. Customer A finally decides to check out of the store and purchases the items in her shopping cart. Now the products table quantity for apples is zero, and the program realizes that there are no more apples to purchase.

This can be somewhat confusing for the customer and embarrassing for the store manager. There are a couple of ways around this issue:

* Don't track specific quantities when customers order items.
* Minimize the amount of time between ordering and purchasing.

The first method has become a common technique for most online stores. Commercial Web stores warn you if supplies are getting low (mostly as a marketing technique) but will allow you to order products even if they're not in stock. All that changes is an estimated shipping date when the products become available.

If you want to track inventory quantities, you can't take an order for an item that's out of stock. In this case, you should try to minimize the amount of time a customer can spend between placing an item in the shopping cart and checking out. The last thing you want is for a customer to place a product in the shopping cart, walk away from the PC, and come back an hour later trying to purchase it.

To prevent this problem, you need a way for the shopping cart to expire at a preset interval. This way, if a customer leaves the PC with items in the shopping cart, when he or she comes back, the cart will have reset, and the customer will need to select the items again.

You can accomplish this with a special feature of session cookies. I mentioned that a session cookie is active for as long as the browser window session remains active. This isn't 100% true.

**Altering the Session Cookie Lifetime**

This is the default behavior of session cookies, but you can modify it. Some sites (especially ones that process sensitive data) use the timeout feature for the session cookie. If the customer hasn't finished the Web transaction within a preset amount of time, the session cookie expires, even if the browser window is still open.

To force session cookies to expire prematurely, you must make a setting change in the PHP configuration file. The setting is called *session.cookie\_lifetime*.

This entry controls how many seconds the session cookie remains active while the Web browser is active. The default value of 0 indicates that the session remains active for the duration of the time the Web browser is active. If you set this value to anything other than zero, the session cookie expires when the time limit is expired, even if the Web browser is still active.

This value is set in the php.ini configuration file. You can modify this value in WampServer by following these steps:

1. Start the WampServer services on the PC.
2. Click the **WampServer** icon in the system tray, and select **PHP**.
3. Select the **php.ini** configuration file.
4. In the Notepad editor, search for the term *lifetime*. This will produce the line:

session.cookie\_lifetime = 0

1. Change the zero value to 300 seconds.
2. Save the php.ini configuration file and exit the editor.

Now your customers must check out within five minutes of starting their session with the Web store. After five minutes, their shopping cart will disappear without warning, and a new one will start.

Now let's continue on to Chapter 4 and discuss modifying the contents of the shopping cart.