**The Storefront**

The main Web page in a store application is called the *storefront*. The storefront Web page must do several jobs for your customer:

* Provide basic information about the store.
* Provide an easy method for customers to navigate through the store.
* Provide information on what products are available for purchase.
* Provide a way for customers to select products for purchase.
* Provide an easy method for checking out.
* Provide contact information in case a customer has problems with the store.
* Provide specialized information, such as sales and news events.

There are lots of ways to present this information in the storefront. I'm sure you've seen (and probably even used) many of the thousands of online shopping Web sites and have seen different techniques used to implement these features.

The Food Store uses a common Web page approach to present the storefront. By using a common Web page, the general layout of each page in the store is the same. The only thing that changes is the main content in the page. This allows you to guide the customer through different parts of the store using a common interface.

The storefront uses the same template we used to create the back-end application. This provides for a Web page with five sections:

* A *header* section for displaying standard store information.
* A *footer* section for displaying contact information.
* A *navigation* section for allowing the customer to easily jump to any location in the store.
* A *shopping cart* section to provide instant information on items the customer selects for purchase.
* A *main* section where customers can browse and select items from a catalog of products, then checkout when they're done shopping.

The Food Store storefront

Before we get too deep in the application code required to run the main section of the storefront, let's take a quick look at the template code we'll use to create the common Web page interface.

**Creating the Storefront**

The basic storefront template copies from the *admin.php* code we used for the back-end application. Follow these steps to create the storefront template:

1. Create a file called *index.php* in the store folder.
2. Enter the following code into the new file:

<?php

session\_start();

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css" />

<title>The Food Store</title>

</head>

<?php

include("mylibrary/login.php");

include("mylibrary/showproducts.php");

login();

?>

<body>

<table width="100%" border="0">

<tr>

<td id="header" height="90" colspan="3">

<?php include("header.inc.php"); ?></td>

</tr>

<tr>

<td id="nav" width="20%" valign="top">

<?php include("nav.inc.php"); ?></td>

<td id="main" width="50%" valign="top">

<?php

if (!isset($\_REQUEST['content']))

include("main.inc.php");

else

{

$content = $\_REQUEST['content'];

$nextpage = $content . ".inc.php";

include($nextpage);

}

?></td>

<td id="status" width="30%" valign="top">

<?php include("cart.inc.php"); ?></td>

</tr>

<tr>

<td id="footer" colspan="3">

<div align="center">

<?php include("footer.inc.php"); ?>

</div></td>

</tr>

</table>

</body>

</html>

1. Save the file and exit the editor.
2. The index.php file uses the same *header.inc.php*, *footer.inc.php*, and *mystyle.css* files that the back-end application uses. Copy each of these files from the admin folder to the store folder.

The storefront index.php file starts with a small piece of PHP code:

<?php  
session\_start();  
?>

This code is crucial to using session cookies, which we'll discuss later in this lesson.

You now have the core template complete for your storefront. Now, you can move on to creating the navigation section.

**Creating the Navigation Section**

Your navigation section should provide simple links for the customer to find his or her way around the store Web pages. One common technique for storefronts is to provide links to separate categories within the catalog and indicate how many items are available in each category. You can do this easily with PHP code. Let's build the nav.inc.php file, which provides this feature:

1. Create the file *nav.inc.php* in the store folder.
2. Enter the following code into the file:

<table width="100%" cellpadding="2">

<tr>

<td><h3>Welcome to the store!</h3></td>

</tr>

<tr>

<td><a href="index.php"><strong>Home</strong></a></td>

</tr>

<tr>

<td><hr size="1" noshade="noshade" /></td>

</tr>

<tr>

<td>

<label><h3>Browse Products:<br><br></h3> </label>

<?php

$query="SELECT catid,name from categories";

$result=mysql\_query($query);

while($row=mysql\_fetch\_array($result,MYSQL\_ASSOC))

{

$catid = $row['catid'];

$name = $row['name'];

$query2="SELECT count(prodid) FROM products WHERE catid = $catid";

$result2 = mysql\_query($query2);

$row=mysql\_fetch\_array($result2);

$total = $row[0];

echo "<a href=\"index.php?content=buyproducts&cat;=$catid\">$name</a> ($total)<br>\n";

}

?>

</td>

</tr>

<tr>

<td><hr size="1" noshade="noshade" /></td>

</tr>

<tr>

<td>

<form action="index.php" method="get">

<label><font color="#663300" size="-1">search for product:</font> </label>

<input name="searchFor" type="text" size="14" />

<input name="goButton" type="submit" value="find" />

<input name="content" type="hidden" value="search" />

</form> </td>

</tr>

<tr>

<td><hr size="1" noshade="noshade" /></td>

</tr>

<tr>

<td><a href="index.php?content=reviewcart"><strong>Review shopping cart</strong></a></td>

</tr>

<tr>

<td><hr size="1" noshade="noshade" /></td>

</tr>

<tr>

<td bgcolor=#FFFF99><a href="index.php?content=checkout"><strong>Check out</strong></a></td>

</tr>

<tr>

<td><hr size="1" noshade="noshade" /></td>

</tr>

<tr>

<td> </td>

</tr>

</table>

1. Save the file and exit the editor.

The nav.inc.php code provides several features in the navigation section:

* A *Home* link to return to the storefront Web page.
* A *Browse Products* area with links to go to a specific location in the catalog.
* A *Search* textbox and button to allow searching products.
* A *Review Shopping Cart* link, allowing the customer to view and modify the current shopping cart contents.
* A *Check Out* link that allows the customer to purchase the items contained in the shopping cart.

The PHP code that creates the Browse Products area performs one SQL query to retrieve all of the categories in the categories table, and then it performs a second query to tally the number of products in each category. This is a common practice in storefront applications.

That's the core of the storefront template code. In the next chapter, you'll work on getting the PHP code for the storefront main section working.