**Displaying Administration Information**

Are you still with me? Great! Let's get back to it. The manager has just entered in a username and password, the application has verified that information from the admins database table, then passed the manager back to the admin.php file. Remember, the admin.php file creates the main part of the Web page based on PHP code:

if (!isset($\_SESSION['store\_admin']))

include("adminlogin.html");

else

include("adminmain.inc.php");

Since the session cookie is set, the manager will see the full navigation menu in the left side of the Web page, and the main section will use the contents of the adminmain.inc.php file. Let's create that file and walk through what it does.

1. Create a new file called *adminmain.inc.php* in the admin folder.
2. Enter the following code in the file:

<?php

$userid = $\_SESSION['store\_admin'];

$query = "SELECT name from admins WHERE userid = '$userid'";

$result=mysql\_query($query);

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

$name = $row['name'];

echo "<h2>Welcome, $name</h2><br>\n";

$date = date("l, F j, Y");

echo "<h2>Today's date: $date</h2><br>\n";

echo "<h2>Admin messages:</h2>\n";

if (is\_readable("/mylibrary/dailymessages.txt"))

{

$message = file\_get\_contents("/mylibrary/dailymessages.txt");

$message = nl2br($message);

echo $message;

}

else

{

echo "No messages for today.\n";

}

echo "<h2><br>Products currently on sale:</h2>\n";

$query = "SELECT prodid,description,price,quantity from products where onsale = 1";

$result = mysql\_query($query);

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

{

$prodid = $row['prodid'];

$description = $row['description'];

$price = $row['price'];

$quantity = $row['quantity'];

printf("<a href=\"admin.php?content=updateproduct&id;=$prodid\">%s</a> - $%.2lf\n", $description, $price);

if ($quantity == 0)

echo " <font color=\"ff0000\">OUT OF STOCK</font><br>\n";

else

echo "<br>";

}

?>

The first part of the adminmain.inc.php should look familiar to you. It extracts the manager's username from the session cookie, performs an SQL query to find the name value in the database table, displays a personalized welcome message, and shows the current date using the PHP date() function. It uses a few exotic parameters for the date() function that show the full text day, the full text month, then the numeric day and the numeric year.

The last section should also look familiar. It sends an SQL query to the MySQL database looking for products that have the *onsale* data field set. It then displays basic information, showing which products are on sale, their current price, and if they're out of stock. One thing to note here is the way the code displays the information:

printf("<a href=\"admin.php?content=updateproduct&id;=$prodid\">%s</a> - $%.2lf\n", $description, $price);

The printf() function works similar to the echo statement, but it allows us to format data when displaying it. We'll look more closely at this function in a later lesson.

The middle part of the code is something you may not be familiar with from your PHP coding experiences. It reads data from a text file and displays it on the Web page.

**Reading Data From Text Files**

The middle section of the adminmain.inc.php file uses a feature of PHP that allows us to retrieve data from simple text files. This is a great tool for posting quick information without messing with the database or importing data from another application into your application. In our case, we'll allow a store senior manager (or possibly the owner) to create a simple text file to hold messages the managers will see when they log into the administration application.

The main section of the Web page

The PHP application can access and process any file it has privileges to on the server. The file access is controlled by the user account restrictions that started the Apache Web server. On a Windows server, that shouldn't be much of a problem, as the Apache Web server often starts as an administrator service. You should have access to any file on the Windows server.

On a Unix, Linux, or Macintosh server, this may get more tricky. Usually these servers run the Apache Web server from a separate user ID (usually called apache). To be able to read data from a file, the apache user ID must have privileges to read the file.

On our WampServer, all should be well, so we can play with this feature. Here's the code from the adminmain.inc.php file that we're using to process the text file:

if (is\_readable("/mylibrary/dailymessages.txt"))

{

$message = file\_get\_contents("/mylibrary/dailymessages.txt");

$message = nl2br($message);

echo $message;

}

else

{

echo "No messages for today.\n";

}

First, the code uses the *is\_readable()* PHP function. This function checks to see if the file exists on the system. It also makes sure that the user ID the Web server is running under has read access to it. For our test, we'll just put a text file called *dailymessages.txt* in the mylibrary folder. If this test fails, the code just prints out a message indicating that there aren't any messages.

If the test passes, the code uses the *file\_get\_contents()* PHP function to read the entire file into a single PHP variable. This is a great feature, because once the variable contains the file contents, you can step through the file just by processing the variable. This example uses the *nl2br()* PHP function to convert newline characters to HTML tags in the string. This makes the message appear on the Web page just as it appears in the text file.

**Creating a Status Section**

Another feature you'll find in many modern Web applications is a single status area, often called a *dashboard*. Much like the dashboard on a car, a Web application dashboard provides a single place to display important information about the application.

The Food Store dashboard

The dashboard in our administration application provides the following information:

* The number of products currently available in the store.
* The quantity of products that are out of stock.
* The amount of customer orders waiting to be processed.

Obviously you can provide as much or as little information in your dashboard area as you want. The idea is to provide crucial information to your manager immediately without making him or her search through the data manually.

The dashboard information is a result of multiple SQL queries. Often you can provide a quick SQL query to produce information that would take the manager extra time to calculate. For example, we can easily find the number of products that are out of stock with the simple query:

SELECT prodid FROM products WHERE quantity = 0;

Without this simple query, the manager would have to look through all of the individual products to find the ones that show a zero quantity value.

Let's create the dashboard file for our administration application. The admin.php file includes the file adminstatus.inc.php in the status section, so that's what we'll build:

1. Create the file *adminstatus.inc.php* in the admin folder.
2. Enter the following code:

<?php

if (isset($\_SESSION['store\_admin']))

{

echo "<h2>Current store status:</h2>\n";

echo "<table width=\"100%\" cellpadding=\"1\" border=\"1\">\n";

$query = "SELECT prodid from products";

$result = mysql\_query($query);

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

echo "<tr><td>Products in store</td><td>$totprods</td></tr>\n";

$query="SELECT prodid from products where quantity = 0";

$result = mysql\_query($query);

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

echo "<tr><td>Products out of stock</td><td>$totout</td></tr>\n";

$query = "SELECT orderid from orders where status = 'pending'";

$result = mysql\_query($query);

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

echo "<tr><td>Orders Pending</td><td>$totpending</td</tr>\n";

echo "</table>\n";

}

?>

1. Save the file and exit.

Notice that the dashboard is hidden from sight unless the visitor has the *store\_admin* session cookie set.

The power of the dashboard is in the SQL queries, not necessarily the PHP or HTML code. You'll see that the code presents the dashboard as a simple HTML table. It provides the basic information in an easy-to-read format. The SQL queries do all of the hard work. The PHP *mysql\_num\_rows()* function counts the number of records returned in the result set without us having to iterate through all of the records!

|  |
| --- |
| **Note:** Be careful with how complex you make the SQL queries in your dashboard. You don't want to use a query that takes a long time to complete, as it will slow down your entire Web page. |

That should be enough for today. Let's move on to Chapter 5 and recap what we've learned.