**Searching**

Searching has become one of the hottest topics in the Web world. Thanks to sites such as Google and Yahoo!, Web users have come to expect some pretty advanced features in the searching abilities of Web sites. In order to provide a professional environment for your shoppers, you need to provide some form of searching.

Most store sites allow their customers to search through the catalog of products to filter only the items they're interested in viewing. To do that, you must provide a textbox for them to enter a search string. We've already done that in the Food Store navigation bar area. Now we just need to produce the PHP code for implementing the searching.

The search textbox on the main Web page

**Searching for Data in MySQL**

The MySQL database engine provides multiple ways for you to search data fields for values. You've already seen the WHERE clause in action. This feature allows you to display only records where a data field is equal to a specific value. This is the most basic form of searching for data. However, this doesn't allow you to search for a string contained within the total data field.

Instead of looking for a specific value in a data field, you'll want to produce data fields that are similar to the search string. For example, if the customer enters the word *juice*in the search textbox, you'll want to return records where the product description contains the word *juice* anywhere in the data field, not just the single word *juice*.

This is called *pattern matching*. You provide a pattern, which the MySQL database engine then matches against all of the records in a table. If a record matches the pattern, it's returned in the result set.

When you're searching for data patterns within data fields, MySQL provides two methods for matching data:

* SQL pattern matching
* Regular expression pattern matching

In *SQL pattern matching*, MySQL uses the standard SQL LIKE clause format. The LIKE clause allows you to combine characters with wildcard characters to create a pattern. This method uses an underscore to match a single character in the pattern and a percent symbol to match zero or more characters in the pattern. As a result, the query

SELECT \* FROM products WHERE description LIKE '%juice';

matches any data record that contains the word *juice*, preceded by any number of characters (including none). This would return records containing *apple juice*, *grape juice*, or just *juice*.

If you want to match records that contain the word *juice* anywhere in the data field, you can use two wildcard characters:

SELECT \* FROM products WHERE description LIKE '%juice%';

This would return records like *apple juice concentrate*.

In *regular expression pattern matching*, your customer provides you with a coded pattern used to search for text. Regular expressions use their own language to define how to match data. The regular expression language specifies exactly how to filter data results using tons of search characters. It's amazing how detailed you can get using regular expressions.

MySQL supports regular expression queries using the REGEXP clause. You specify the matching pattern similar to how you did with the LIKE clause:

SELECT \* FROM products WHERE description REGEXP 'juice';

In a regular expression, by default, any text you enter is matched anywhere in the data field. Thus, this query automatically returns any data record that contains the word *juice* anywhere in the description. No wildcard characters are required!

You can define exactly where in the string the text pattern should match by using two special characters. The caret character (^) indicates that the data pattern must be at the start of a string. Therefore, the query

SELECT \* FROM products WHERE description REGEXP '^App';

only returns records where the description starts with the text *App* (such as Apple juice). Similarly, the dollar sign ($) indicates the data must be at the end of a string. Consequently the query

SELECT \* FROM products WHERE description REGEXP 'juice$';

only returns records that end with the word *juice*. Thus, the data field value *apple juice concentrate* wouldn't match, but *apple juice* would.

The MySQL regular expression language uses lots of special characters to define the search. If you've ever used regular expressions in programming languages such as Perl or Python, you'll notice that MySQL uses many of the same characters. You'll find some of the more popular ones below:

|  |  |
| --- | --- |
| **MySQL Regular Expressions** | |
| **Character(s)** | **Description** |
| ^*string* | Match the specified text only at the beginning of the string. |
| *string*$ | Match the specified text only at the end of the string. |
| . | Match any single character (including special characters). |
| *a*\* | Match a sequence of zero or more of the specified character. |
| *a*+ | Match a sequence of one or more of the specified characters. |
| *a*? | Match zero or one occurrence of the specified character. |
| *abc*|*def* | Match either one of the specified character sequences. |
| [*abc*] | Match any one of the specified characters. |

Another example of a regular expression would be:

SELECT \* FROM products WHERE description REGEXP 'apple|grape';

This simple query returns any record that contains either the word *apple* or *grape*anywhere in the data field. You can start to see the power of simple regular expressions. Trying to create that search using the LIKE clause would be somewhat complicated.

**Coding the Search**

The search.inc.php file handles the searching for the Food Store application. Go ahead and create that file. Then we'll talk about how it works. Here are the steps for creating the file:

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

<?php

$search = $\_GET['searchFor'];

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

{

$search = stripslashes($search);

}

$searchsql = mysql\_real\_escape\_string($search);

echo "<h2>Results of searching '$search'<br><br></h2>\n";

$query = "SELECT \* from products WHERE description REGEXP '$searchsql'";

$result = mysql\_query($query);

if (!$result)

{

echo "<h2>Sorry, unable to process search string.</h2>\n";

} else

{

echo "<table width=\"100%\" border=\"0\">\n";

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

{

$prodid = $row['prodid'];

$description = $row['description'];

$price = $row['price'];

$quantity = $row['quantity'];

echo "<tr><td>\n";

echo "<img src=\"showimage.php?id=$prodid\" width=\"80\" height=\"60\">";

echo "</td><td>\n";

if ($quantity == 0)

echo "<font size=\"3\">$description</font>\n";

else

{

echo "<a href=\"index.php?&content;=updatecart&id;=$prodid\">";

echo "<font size=\"3\"><b><u>$description</u></b></font></a>\n";

}

echo "</td><td>\n";

printf("$%.2f\n", $price);

echo "</td><td>\n";

if ($quantity == 0)

echo "<font color=\"red\">Sorry, this item out of stock</font>\n";

else if ($quantity < 5)

echo "Hurry, only $quantity left in stock!\n";

else

echo " \n";

echo "</td></tr>\n";

}

echo "</table>\n";

}

?>

1. Save the file, and exit the text editor.

The search code retrieves the text the customer enters into the search textbox on the main Web page, uses our common method of scrubbing data for MySQL, and then sends it to the SELECT query using the standard REGEXP clause.

If the query returns any data, the search.inc.php code builds a table to display the results using links to allow the customer to view the product details and add it to the shopping cart.

The results of a simple search

This search technique allows our customers to perform searches for specific items, just like in some of the more advanced search tools. That's pretty fancy for a small amount of coding!

That completes the code for our Food Store storefront application. You've covered a lot of ground with this project! Let's move on to Chapter 3 and see what we can do for the manager in the back-end application.