Creating an Online Storefront

Lesson 11a

In this lesson, you will create a generic online storefront. You will learn the methods for creating the relevant database tables, as well as the scripts for displaying the information to the user. The examples used in this lesson represent one of an infinite number of possibilities to complete these tasks, and are meant to provide a foundation of knowledge rather than a definitive method for completing this task.

In this lesson, you will learn how to

- Create relational tables for an online store
- Create the scripts to display store categories
- Create the scripts to display individual items

Planning and Creating the Database Tables

Before you tackle the process of creating database tables for an online store, think about the real-life shopping process. When you walk into a store, items are ordered in some fashion: The hardware and the baby clothes aren't mixed together, the electronics and the laundry detergent aren't side by side, and so on. Applying that knowledge to database normalization, already you can see you will need a table to hold categories and a table to hold items. In this simple store, items will each belong to one category.

Next, think about the items themselves. Depending on the type of store you have, your items may or may not have colours, and may or may not have sizes. But all your items will have a name, a description, and a price. Again, thinking in terms of normalization, you can see that you will have one general items table and two additional tables that relate to the general items table.

Table 1 shows sample table and field names to use for your online storefront. In a minute, you'll create the actual SQL statements, but first you should look at this information and try to see the relationships appear. Ask yourself which of the fields should be primary or unique keys.

Table 1. Storefront Table and Field Names

Table Name	**Field Names**
`store_categories`	`id, cat_title, cat_desc`
`store_items`	`id, cat_id, item_title, item_price, item_desc, item_image`
`store_item_size`	`item_id, item_size`

Table 1. Storefront Table and Field Names

Table Name	**Field Names**
`store_item_color`	`item_id, item_color`

As you can see in the following SQL statements, the `store_categories` table has two fields besides the `id` field: `cat_title` and `cat_desc`, for title and description. The `id` field is the primary key, and `cat_title` is a unique field because there's no reason you would have two identical categories.

```
mysql> create table store_categories (
    -> id int not null primary key auto_increment,
    -> cat_title varchar (50) unique,
    -> cat_desc text
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Next we tackle the `store_items` table, which has five fields besides the `id` field none of which are unique keys. The lengths specified in the field definitions are arbitrary; you should use whatever best fits your store.

The `cat_id` field relates the item to a particular category in the `store_categories` table. This field is not unique because you will want more than one item in each category. The `item_title, item_price`, and `item_desc` (for description) fields are self-explanatory. The `item_image` field will hold a filenamein this case, the file is assumed to be local to your serverwhich you will use to build an HTML `<IMG>` tag when it's time to display your item information.

```
mysql> create table store_items (
    -> id int not null primary key auto_increment,
    -> cat_id int not null,
    -> item_title varchar (75),
    -> item_price float (8,2),
    -> item_desc text,
    -> item_image varchar (50)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Both the `store_item_size` and `store_item_color` tables contain optional information: If you sell books, they won't have sizes or colors, but if you sell shirts, they will. For each of these tables, no keys are involved because you can associate as many colors and sizes with a particular item as you want.

```
mysql> create table store_item_size (
    -> item_id int not null,
    -> item_size varchar (25)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> create table store_item_color (
    -> item_id int not null,
    -> item_color varchar (25)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

These are all the tables necessary for a basic storefront that is, for displaying the items you have for sale. In the next lesson "Creating a Shopping Cart Mechanism," you will integrate the user experience into the mix. For now, just concentrate on your inventory.

In the previous lesson "Creating an Online Address Book," you learned how to use PHP forms and scripts to add or delete records in your tables. If you apply the same principles to this set of tables, you can easily create an administrative front end to your storefront. We won't go through that process in this book, but feel free to do it on your own. At this point, I am confident you know enough about PHP and MySQL to complete the tasks.

For now, you can simply issue MySQL queries, via the MySQL monitor or other interface, to add information to your tables. Following are some examples, if you want to follow along with sample data.

Inserting Records into the `store_categories` Table

The following queries create three categories in your `store_categories` table: hats, shirts, and books.

```
mysql> insert into store_categories values
    -> ('1', 'Hats', 'Funky hats in all shapes and sizes!');
Query OK, 1 row affected (0.01 sec)

mysql> insert into store_categories values ('2', 'Shirts', 'From t-
shirts to
    -> sweatshirts to polo shirts and beyond.');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_categories values ('3', 'Books', 'Paperback,
hardback,
    -> books for school or play.');
Query OK, 1 row affected (0.00 sec)
```

In the next section, we'll add some items to the categories.

Inserting Records into the `store_items` Table

The following queries add three item records to each category. Feel free to add many more.

```
mysql> insert into store_items values ('1', '1', 'Baseball Hat',
'12.00',
    -> 'Fancy, low-profile baseball hat.', 'baseballhat.gif');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into store_items values ('2', '1', 'Cowboy Hat',
'52.00',
    -> '10 gallon variety', 'cowboyhat.gif');
Query OK, 1 row affected (0.01 sec)

mysql> insert into store_items values ('3', '1', 'Top Hat', '102.00',
    -> 'Good for costumes.', 'tophat.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('4', '2', 'Short-Sleeved T-
Shirt',
    -> '12.00', '100% cotton, pre-shrunk.', 'sstshirt.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('5', '2', 'Long-Sleeved T-
Shirt',
    -> '15.00', 'Just like the short-sleeved shirt, with longer
sleeves.',
    -> 'lstshirt.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('6', '2', 'Sweatshirt',
'22.00',
    -> 'Heavy and warm.', 'sweatshirt.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('7', '3', 'Jane\'s Self-Help
Book',
    -> '12.00', 'Jane gives advice.', 'selfhelpbook.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('8', '3', 'Generic Academic
Book',
    -> '35.00', 'Some required reading for school, will put you to
sleep.',
    -> 'boringbook.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('9', '3', 'Chicago Manual of
Style',
    -> '9.99', 'Good for copywriters.', 'chicagostyle.gif');
Query OK, 1 row affected (0.00 sec)
```

Inserting Records into the `store_item_size` Table

The following queries associate sizes with one of the three items in the `shirts` category and a generic "one size fits all" size to each of the hats (assume they're strange hats). On your own, insert the same set of size associations for the remaining items in the `shirts` category.

```
mysql> insert into store_item_size values (1, 'One Size Fits All');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (2, 'One Size Fits All');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (3, 'One Size Fits All');
```

```
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'S');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'M');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'L');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'XL');
Query OK, 1 row affected (0.00 sec)
```

Inserting Records into the `store_item_color` Table

The following queries associate colors with one of the three items in the `shirts` category. On your own, insert color records for the remaining shirts and hats.

```
mysql> insert into store_item_color values (1, 'red');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_color values (1, 'black');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_color values (1, 'blue');
Query OK, 1 row affected (0.00 sec)
```

Displaying Categories of Items

Believe it or not, the most difficult task in this project is now complete. Compared to thinking up categories and items, creating the scripts used to display the information is easy! The first script you will make is one that lists categories and items. Obviously, you wouldn't want to list all categories and all items all at once as soon as the user walks in the door, but you do want to give the user the option of immediately picking a category, seeing its items, and then picking another category. In other words, this script will serve two purposes: It will show the categories and then show the items in that category.

Listing 1 shows the code for `seestore.php`.

Listing 1. Script to View Categories

```
<?php
//connect to database
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn) or die(mysql_error());

$display_block = "<h1>My Categories</h1>
<P>Select a category to see its items.</p>";

//show categories first
$get_cats = "select id, cat_title, cat_desc from
    store_categories order by cat_title";
$get_cats_res = mysql_query($get_cats) or die(mysql_error());
```

```

if (mysql_num_rows($get_cats_res) < 1) {
   $display_block = "<P><em>Sorry, no categories to
browse.</em></p>";
} else {

   while ($cats = mysql_fetch_array($get_cats_res)) {
       $cat_id = $cats[id];
       $cat_title = strtoupper(stripslashes($cats[cat_title]));
       $cat_desc = stripslashes($cats[cat_desc]);

        $display_block .= "<p><strong><a
href=\"$_SERVER[PHP_SELF]?cat_id=$cat_id\">$cat_title</a></strong>
        <br>$cat_desc</p>";

       if ($_GET[cat_id] == $cat_id) {
           //get items
           $get_items = "select id, item_title, item_price
           from store_items where cat_id = $cat_id
            order by item_title";
           $get_items_res = mysql_query($get_items) or
die(mysql_error());

           if (mysql_num_rows($get_items_res) < 1) {
              $display_block = "<P><em>Sorry, no items in
               this category.</em></p>";
           } else {

               $display_block .= "<ul>";

               while ($items = mysql_fetch_array($get_items_res))
{
                   $item_id = $items[id];
                   $item_title =
stripslashes($items[item_title]);
                   $item_price = $items[item_price];

                   $display_block .= "<li><a
href=\"showitem.php?item_id=$item_id\">$item_title</a>
                    </strong> (\$$item_price)";
               }

               $display_block .= "</ul>";
            }
        }
    }
}
?>
<HTML>
<HEAD>
<TITLE>My Categories</TITLE>
</HEAD>
<BODY>
<?php echo $display_block; ?>
</BODY>
</HTML>
```

Given the length of scripts you saw in the previous lesson, these 65 fully functional lines should be a welcome change. In lines 35, the database connection is opened because regardless of which action the script is taking showing categories or showing items in categoriesthe database is necessary.

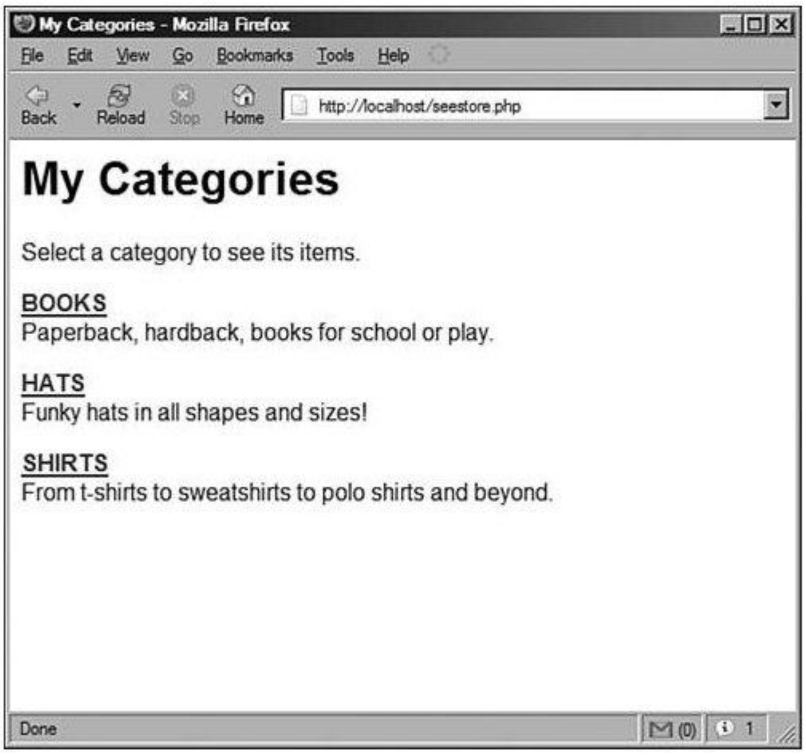
In lines 78, the `$display_block` string is started, with some basic page title information. Lines 11- 13 create and issue the query to retrieve the category information. Line 15 checks for categories; if none are in the table, a message is printed to the user and that's all this script does. However, if categories are found, the script moves on to line 19, which begins a `while` loop to extract the information.

In the `while` loop, lines 20 - 22 retrieve the ID, title, and description of the category. String operations are performed to ensure that no slashes are in the text and that the category title is in uppercase for display purposes. Lines 24 - 26 place the category information, including a self-referential page link, in the `$display_block` string. If a user clicks the link, she will return to this same script, except with a category ID passed in the query string. The script checks for this value in line 28.

If a `$cat_id` value has been passed to the script because the user clicked on a category link in hopes of seeing listed items, the script builds and issues another query (lines 3033) to retrieve the items in the category. Lines 4253 check for items and then build an item string as part of `$display_block`. Part of the information in the string is a link to a script called `showitem.php`, which you'll create in the next section.

After reaching that point, the script has nothing left to do, so it prints the HTML and value of `$display_block`. Figure 1 shows the outcome of the script when accessed directly; only the category information shows.

Figure 1. Categories in the store.


My Categories - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Back Reload Stop Home
http://localhost/seestore.php
Done

My Categories

Select a category to see its items.

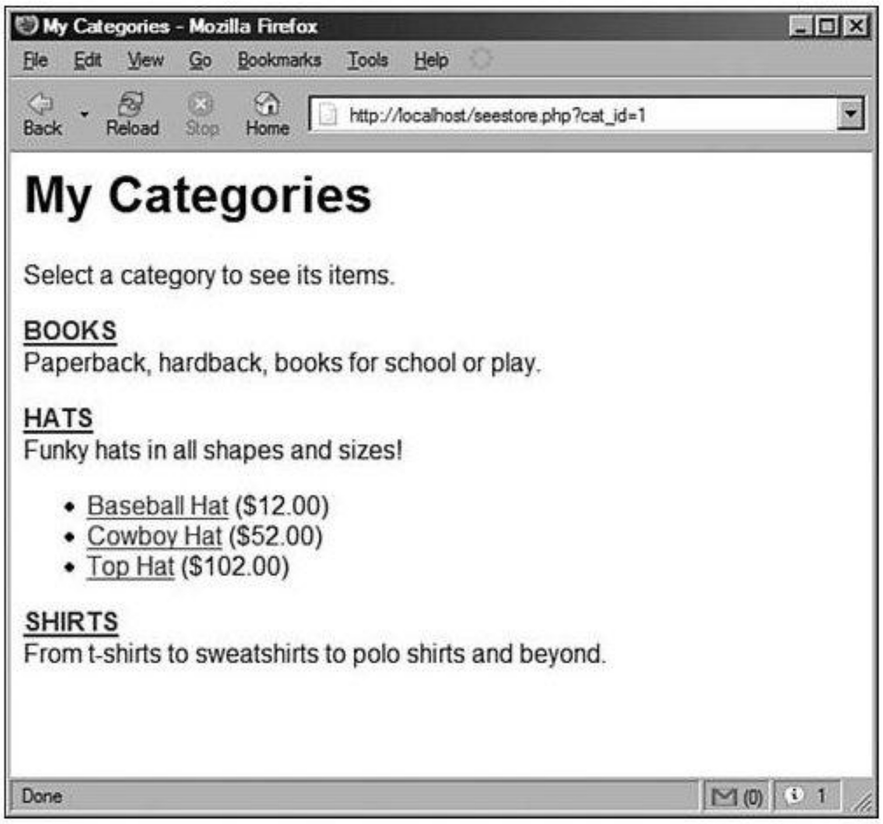
BOOKS
Paperback, hardback, books for school or play.

HATS
Funky hats in all shapes and sizes!

SHIRTS
From t-shirts to sweatshirts to polo shirts and beyond.

In Figure 2, you see what happens when the user clicked on the `HATS` link: The script gathered all the items associated with the category and printed them on the screen. The user can still jump to another category on this same page, and it will gather the items for that category.

Figure 2. Items within a category in the store.


My Categories - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Back Reload Stop Home
http://localhost/seestore.php?cat_id=1
Done

My Categories

Select a category to see its items.

BOOKS
Paperback, hardback, books for school or play.

HATS
Funky hats in all shapes and sizes!

- Baseball Hat ($12.00)
- Cowboy Hat ($52.00)
- Top Hat ($102.00)

SHIRTS
From t-shirts to sweatshirts to polo shirts and beyond.

The last piece of the puzzle for this lesson is the creation of the item display page.

Displaying Items

The item display page in this lesson will simply show all the item information. In the next lesson, you'll add a few lines to it to make it function with an "add to cart" button. So for now, just assume this is a paper catalogue.

Listing 2 shows the code for `showitem.php`.

Listing 2. Script to View Item Information

```
<?php
//connect to database
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn) or die(mysql_error());

$display_block = "<h1>My Store - Item Detail</h1>";

//validate item
$get_item = "select c.id as cat_id, c.cat_title, si.item_title,
si.item_price, si.item_desc, si.item_image from store_items as si left join
store_categories as c on c.id = si.cat_id where si.id = $_GET[item_id]";
```

```

$get_item_res = mysql_query($get_item) or die (mysql_error());

if (mysql_num_rows($get_item_res) < 1) {
   //invalid item
   $display_block .= "<P><em>Invalid item selection.</em></p>";
} else {
   //valid item, get info
   $cat_id = mysql_result($get_item_res,0,'cat_id');
   $cat_title = strtoupper(stripslashes(
       mysql_result($get_item_res,0,'cat_title')));
   $item_title = stripslashes(mysql_result($get_item_res,0,'item_title'));
   $item_price = mysql_result($get_item_res,0,'item_price');
   $item_desc = stripslashes(mysql_result($get_item_res,0,'item_desc'));
   $item_image = mysql_result($get_item_res,0,'item_image');

   //make breadcrumb trail
   $display_block .= "<P><strong><em>You are viewing:</em><br>
   <a href=\"seestore.php?cat_id=$cat_id\">$cat_title</a>
    &gt; $item_title</strong></p>

   <table cellpadding=3 cellspacing=3>
   <tr>
   <td valign=middle align=center><img src=\"$item_image\"></td>
   <td valign=middle><P><strong>Description:</strong><br>$item_desc</p>
   <P><strong>Price:</strong> \$$item_price</p>";

   //get colors
   $get_colors = "select item_color from store_item_color where
    item_id = $item_id order by item_color";
   $get_colors_res = mysql_query($get_colors) or die(mysql_error());

   if (mysql_num_rows($get_colors_res) > 0) {
       $display_block .= "<P><strong>Available Colors:</strong><br>";

       while ($colors = mysql_fetch_array($get_colors_res)) {
           $item_color = $colors['item_color'];
           $display_block .= "$item_color<br>";
       }
   }

   //get sizes
   $get_sizes = "select item_size from store_item_size where
        item_id = $item_id order by item_size";
   $get_sizes_res = mysql_query($get_sizes) or die(mysql_error());

   if (mysql_num_rows($get_sizes_res) > 0) {
       $display_block .= "<P><strong>Available Sizes:</strong><br>";

       while ($sizes = mysql_fetch_array($get_sizes_res)) {
           $item_size = $sizes['item_size'];
           $display_block .= "$item_size<br>";
       }
   }
67:
68:    $display_block .= "
69:    </td>
70:    </tr>
71:    </table>";
72: }
73: ?>
74: <HTML>
75: <HEAD>
76: <TITLE>My Store</TITLE>
77: </HEAD>
78: <BODY>
79: <?php echo $display_block; ?>
80: </BODY>
81: </HTML>
```

In lines 35, the database connection is opened because information in the database forms all the content of this page. In line 7, the `$display_block` string is started, with some basic page title information.

Lines 10 - 4 create and issue the query to retrieve the category and item information. This particular query is a table join. Instead of selecting the item information from one table and then issuing a second query to find the name of the category, this query simply joins the table on the category ID to find the category name.

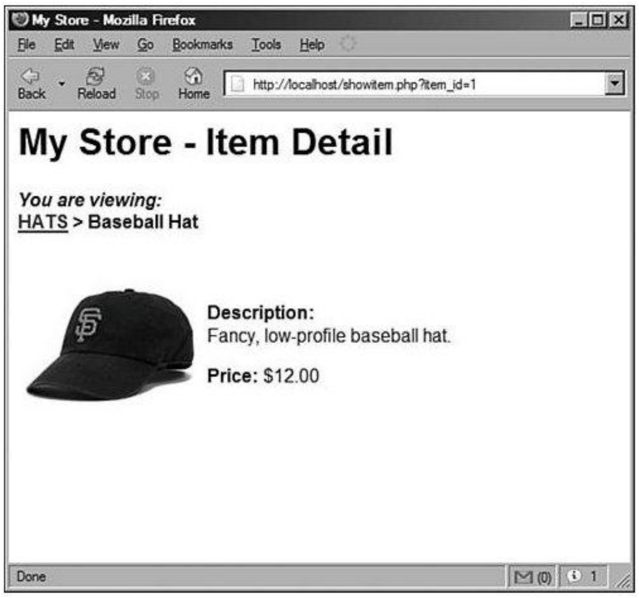
Line 16 checks for a result; if there is no matching item in the table, a message is printed to the user and that's all this script does. However, if item information is found, the script moves on and gathers the information in lines 21- 27.

In lines 30 - 32, you create what's known as a "breadcrumb trail." This is simply a navigational device used to get back to the top-level item in the architecture. Those are fancy words that mean "print a link so that you can get back to the category." The category ID, retrieved from the master query in this script, is appended to the link in the breadcrumb trail.

In lines 34 - 38, you continue to add to the `$display_block`, setting up a table for information about the item. You use the values gathered in lines 21 - 27 to create an image link, print the description, and print the price. What's missing are the colours and sizes, so lines 40 - 52 select and print any colors associated with this item, and lines 54 - 66 gather the sizes associated with the item.

Lines 68 - 72 wrap up the `$display_block` string and the master `if...else` statement, and because the script has nothing left to do, it prints the HTML (lines 74 - 81) including the value of `$display_block`. Figure 3 shows the outcome of the script when selecting the baseball hat from the hats category. Of course, your display will differ from mine, but you get the idea.

Figure 3. The baseball hat item page.


My Store - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Back Reload Stop Home
http://localhost/showitem.php?item_id=1
Done

My Store - Item Detail

You are viewing:
HATS > Baseball Hat

Description:
Fancy, low-profile baseball hat.

Price: $12.00

That's all there is to creating a simple item display. In the next lesson, you'll modify this script so that it can add the item to a shopping cart.

Workshop

The workshop is designed to help you anticipate possible questions, review what you've learned, and begin learning how to put your knowledge into practice.

Quiz

1. Which PHP function was used to uppercase the category title strings?
2. Why don't the `store_item_size` and `store_item_color` tables contain any primary or unique keys?

Answers

1. `strtoupper()`
2. Presumably, you will have items with more than one colour and more than one size. Therefore, item_id is not a primary or unique key. Also, items may have the same colours or sizes, so the item_color and item_size fields must not be primary or unique either.