That's Bargain! Books Developer Guide

Thank you for choosing That's Bargain! Books. This guide is intended for developers, who are interested in either extending or using some of the existing features that make this website run. We will walk through how certain aspects of the project are implemented, which will help if you plan on extending any of this project. This software was developed using the LAMP stack, using PHP and MySQL to power the backend, and using HTML / Javascript to power the front end display.

Database Architecture

This software is run using a MySQL database which contains the following tables:

- proj_customer - stores information about the customer.
- proj_admin - stores the email address of users that have administrative powers.
- proj_author - stores information about all of the author's whose book is in the DB.
- proj_book - stores information about the books being sold.
- proj_stock - stores information about the stock levels of the books being sold.
- proj_login - stores login information about the customers.
- proj_orders - stores information about orders such as date, time, and orderID.
- proj_order_items - maps a product and a quantity to an orderID.

The email address in customer is a foreign key for both the login and and admin tables. The ISBN in the book tables is a foreign key for both the stock and author tables. The order ID field in the orders table is used as a foreign key in the order items table.

User Types

There are three types of people that are using this system at the same time: guests, users, and admins. Guests are people that have not yet registered an account, and therefore, can not actually checkout an order. They can, however, search and add items to their shopping cart. They will be asked to register on checkout, and then will be able to purchase their order. Users are people that have registered for an account. They can buy books, look at their past orders, change their account information, and leave comments. Admins can do all of the same things that normal users can do, but they can also add books, update stock levels, make another user admin, and other activities that help with daily upkeep of the web store. There are certain pages that only admin should have access to, so developers should make sure that these pages check that the user has admin rights.

User Passwords

All of the user's passwords are stored in the database as an md5, and salted with their username. In PHP, this is equivalent to md5(pass . $username). This way, even if someone manages to get ahold of the database, a rainbow table attack would not be as fast, and the passwords are still not being stored in plain text. If the user forgets his password, a new one is randomly generated, and sent to them via email.

Shopping Cart

The shopping cart is used to hold the items that the user has selected that they may want to purchase. The shopping cart is internally represented as an associative array that maps an ISBN number to an array that contains information about the book. This method is advantageous, because there is no need to query the database every time that the user wishes to view the items in their cart. The information about the book is also stored as an associative array that has the fields "ISBN" => the book's ISBN, "qty" => the number of books the user wants to buy , and "price" => the price of the book. If the shopping cart is empty, internally it is just unset, which allows for easy testing to see if the cart has any items or not.

Session Variables

Any information that is stored about a user when they connect to the website is stored in the global associative array $_SESSION. This way, we have all of the personal information saved about the user so we don't have to keep on querying the database, and it lets us know if the user is a user or an admin, We can also check to see if the user is a quest, because personal information stored in $_SESSION will not be set. Here are a list of things stored about the user, and how to access them:

$_SESSION['email'] - The email address of the user. This address is unique, and can be used to check if the user is a quest or not by seeing if this i$_
$_SESSION['firstname'] - The firstname of the user.
$_SESSION['lastname'] - The lastname of the user.
$_SESSION['address_city'] - The user's city.
$_SESSION['address_street'] - The user's street.
$_SESSION['address_state'] - The user's state.
$_SESSION['address_zip'] - The user's 5 digit zip code.
$_SESSION['phone'] - The user's phone number.

It should be noted that in order to use any of the information stored in $_SESSION, the program must include the function session_start() somewhere before the reference. Also, $_SESSION will be destroyed when the user either closes the browser, or hits the logout button at the top of the screen.

Sending Book Information Via POST

Whenever the user is given a selection of books to choose from, we must have a way to store the information about the book in the web page, so that it can be passed on to the php script that will be doing the processing. In this case, whenever a list of books are shown with corresponding checkboxes, the information about that book is stored inside the actual input tag. The information is stored internally as an associative array that maps "title" => title of the book, "ISBN" => ISBN of the book, "author" => author of the book, "pages" => number of pages in the book, "year" => the publish year of the book, and "price" => price of the book. We then serialize this array, and take that string and make it the value of the checkbox. Therefore, when the form is submitted via POST, the script receiving the information can see which checkboxes were checked, and can get the information about the book by unserializing the value

associated with that checkbox. The standard idiom for this application is that any drop down select tag corresponding to a certain book will have that book's ISBN has a value, so we can also get the information that way very easily. While this method may seem a little confusing at first, it allows for us to store the information about the book directly into the HTML itself, which means that time is not wasted querying the database. Currently, this method is used to send what books have been added to the cart, and to edit items that are currently in the cart. This can be extended to any situation where the user has to select a book and / or a quantity for some reason, and that information needs to be passed onto another PHP script.

Globals Values and Functions

Any Global variables and functions which are used more than once throughout the program are stored in the file globals.php. Global constants include class that act like enumeration types, which make the code more readable. For example, the actual value for searchType::AUTHOR is stored in globals.php. The other global stored in this file is the root address of all pages, that need to have the header bar stripped at the top. These are pages like the login screen, new user registration, and password reset.

Some of the functions included in globals.php are displayBookTable, which takes a title, an ISBN, an author, the number of pages, the publish year, the price, a quantity, and a select box. This function properly formats the book for display as a row in a table. The programmer should make a table, and then loop through the results and call this function on all of the results in order to properly align the books that are being displayed. Another important function defined in globals.php is selectBox, which takes a name, a number maxNum, the name of a function to be called, a number to be selected by default, and a class, which is null as default. This function will set up an HTML tag for a drop down select box with the proper default values and an option that allows the programmer to set the javascript function that is called when the value in the select box is changed. The function redirectNotAdmin check to make sure the user is an administrator, and if they aren't, then they are redirected to the home page.

Also contained in this project is a file called "header.php". This file contains all of the information for displaying the navigation bar at the top of the screen. It changes the navigation bar to adjust accordingly based on if the user is an administrator, normal user, or guest.

MySQL Connection and mysqli Object

All of the information about the database connection, which includes the location of the DB, the username, and login information is located in the file myslqi.php. This file creates a global variable called $db_obj that interfaces with the database manager, and therefore any script that interacts with the database in any way should include this file. $db_obj also contains some useful functions that are used often throughout the program. $db_obj->real_escape_string() is used to sanitize user input, $db_obj->query() is used to actually send a query to the database, and $db_obj->affected_rows returns the number of affected rows by the database. Always use real_escape_string when trying to use any input from the user. This is the best way to protect against SQL injections.

Testing

Because, this is a web-based application, most of the results can actually be seen, and it

becomes quite obvious when something is not acting in a way that it shouldn't. However, this software comes with some php scripts that perform unit testing and make sure that certain backend procedures are working the way that they are supposed to. These tests are located in the directory /tests, and can be run by typing the command php test.php. These tests should be run whenever a major change to the backend or architecture is run in order to verify that everything is in working order. The values being tested for are located in DML.sql, so it is important that file is imported before the unit tests are run.

Displaying Results Per Page

Whenever we display books or other items to the user that are dynamically generated based on user input (search results, orders, etc.), we want the user to be able to specify the number of results shown with links to the next and previous page if possible. Several variables are passed via GET to the script that actually does the displaying. They Are:
$max_results - The maximum number of results in the query.
$page - The current page that the user is on
$results - The number of results per page.
Using these variables, we can test to see if there are more results that need to be displayed, and if so, then we can generate a link to display more results. There is also a select box that lets the user choose how many results per page they would like to see. When this box is changed, and javascript function is called that changes the URL to set results to the new value, and refreshes the page so the correct number of results are shown per page.

Ideas For Improvement

Listed below are a few suggestion for how one could expand upon and add features to the site:

1.) Extending the database to support sale items

This feature should not be too hard to implement. A sale_book table can be implemented which has fields for an ISBN (the primary key), a date field that represents the start of the sale, a date field that represents the end of the sale, and a field that represents the sale price. A web page can easily be constructed with a form that allows an administrative user to enter this information. Once this relation is set up and has data enter it, only the search script has to be modified, because that is the only time that information about a book is retrieved from the database. One can check to see if the book is in the sale interval, and adjust the price accordingly.

2.) Hiding select boxes until a checkbox is chosen.

This feature is more on the design side of the project, but is still something that could make the user experience a little better. Currently when the search results are returned to the user, both a check box and select box are shown. The user clock on a check box, which signifies that the user would like to add all of the items checked into their cart. A more user friendly experience would be to hide the select boxes until the actual check box is checked. One way to

implement this would be to use the selectBox() function to give each select box a unique id attribute, such as the ISBN of the book that the book corresponds to. Using jquery, one can make it so that the select box becomes visible when the corresponding check box is clicked.

3.) View books with the most number of books during a certain duration.

This feature would be part of the admin tools, that allows an administrator to view the most popular books during a selected time period. This would be pretty easy to implement, as the orders items table in the database already records the date and time that the order took place. Therefore one would just have to use a MySQL aggregating function to count the number of copies of each book sold during the selected time, and return the max. You could also implement this feature to suggest titles that have been bought more often to the customer.