Quick E-Commerce with PHP and PayPal

by
Peter Lavin

August 28, 2003

Overview

Finding a secure method of payment can be a problem for small businesses without merchant credit card accounts or access to a secure server. This article concerns creating an e-commerce site using your own shopping cart or a third-party cart in conjunction with *PayPal*. This solution will maximise the business benefits and minimise the inconvenience to the customer. A database and server-side scripting will be implemented using *MySQL* and *PHP*. Some knowledge of these technologies and of HTML is assumed.

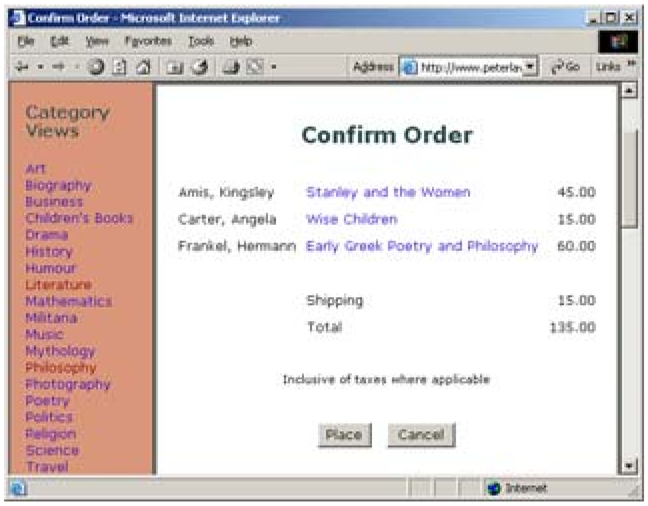
Introduction

Using *PayPal* is an inexpensive way to implement e-commerce. To find out if it suits your needs simply go to the *PayPal* site and read the information provided there. In order to use *PayPal* a business account needs to be set up. This is a fairly straightforward matter and will not be dealt with here. Once your account is set up you might want to read the article entitled “Adding PayPal to Your 3rd Party shopping Cart” found in the “How-To Articles” of “PayPal Developers Network”. It forms the basis for this discussion. It may be found at the URL, https://www.paypal.com/cgi-bin/webscr?cmd=p/pdn/howto_checkout-outside.

When using a third party shopping cart with *PayPal* you have only two options. You can pass your entire shopping cart to *PayPal* and show detailed purchase information or you can pass in only a total. You cannot pass both your shopping cart items and detailed customer information. Passing in the contents of your cart means that your customer will have to duplicate information already entered into your database. Given the precariousness of retail e-commerce transactions this is not a viable option. For this reason, we have chosen to pass in summary information. Additionally, capturing customer information in our own database rather than one that resides at *PayPal* makes better business sense. It will be more easily accessible and more easily manipulated.

The only minor downside to this option is that your customers will not receive details of their order with their *PayPal* invoice. However, this can easily be remedied with an e-mail from your site.

Let’s assume that our customer is viewing order details and is about to complete his purchase. We present him with a confirmation screen such as the following:



Confirm Order

Amis, Kingsley	Stanley and the Women	45.00
Carter, Angela	Wise Children	15.00
Frankel, Hermann	Early Greek Poetry and Philosophy	60.00
	Shipping	15.00
	Total	135.00

Inclusive of taxes where applicable

The remainder of this article discusses what happens when the “Place” button is pressed. When this button is pressed a page called “paypal.php” will be invoked and three values will be passed in a query string. These are, the number of items purchased, the total cost and the unique identifier for our current customer. The dynamically created code to invoke this page might well look like this:

```
paypal.php?quantity=3&total=120&id=1
```

This is all the information we need to complete the transaction.

Collecting the Information

This section deals in detail with all the code in our “paypal.php” page. Explanatory comments have been added. Discussion will be broken up into three parts. The first part will deal with hard-coded information that doesn’t change, next the information passed in the query string is retrieved and finally the customer data from our MySQL database.

This page will not be visible to the user. It includes a form but every input type on the page is “hidden”. This page will be invoked after the user has reviewed and confirmed his order on your website. Any information needed for order fulfillment or follow-up has already been captured to the database and presented to the user. This page merely consolidates all the information required by *PayPal* and automatically submits it using JavaScript.

Hard-Coded Information

A number of the parameters needed by *PayPal* can be hard-coded right into your page. For example, the “action” attribute of the “form” is the page at *PayPal* that is set up to accept submission of your order. At the time of writing, it is as follows:

```
<html>
<body>
<form name= "order" action="https://www.paypal.com/cgi-bin/webscr"
method="post">
```

Since we wish to send detailed customer information, the input control with the name attribute set to “cmd” must have a value of “_ext-enter”.

```
<input type="hidden" name="cmd" value="_ext-enter"><br>
```

The next line is also a requirement for using extended values. Its value must be set to “_xclick”. In this way we will be able to pass our customer information to *PayPal* and not require that it be re-entered.

```
<input type="hidden" name="redirect_cmd" value="_xclick"><br>
```

After submission to *PayPal*, the user is returned to your site and the success or failure of the transaction is confirmed. This will be handled in a page called “notify.php”, but feel free to name it whatever you like. A query string called “status” will be examined in order to output an appropriate message to the user.

```
<input type="hidden" name="return"
      value = "http://<your site>/notify.php?status=T"><br>
<input type="hidden" name="cancel_return"
      value = "http://<your site>/notify.php?status=F"><br>
```

Enter the appropriate values for the next set of items. Remember, the value of the input “business” will be the same as the e-mail address of your *Paypal* account. Likewise with “item_name” and “shipping”. The first shipping value represents the cost of shipping the first item and the second the cost of each additional item.

```
<input type="hidden" name="business" value ="mybusiness@myisp.com">
<input type="hidden" name="item_name" value="your product">
<input type="hidden" name="shipping" value="5.00">
<input type="hidden" name="shipping2" value="5.00">
```

If handling shipping costs in this way does not meet your requirements *PayPal* provides other ways of doing this. These are described in the article mentioned earlier.

Information Retrieved from Query String

As shown in our graphic above, detailed order information was presented to the user in the previous page and will not be re-presented at the *PayPal* site. *Paypal* still needs summary information in order to bill the correct amount. This will now be retrieved from the query string passed to this page. The number of items is also retrieved as this will affect shipping costs.

```
<?php
//get values from previous page
$quantity = $HTTP_GET_VARS['quantity'];
$total = $HTTP_GET_VARS['total'];
echo "<input type=\"hidden\" name=\"quantity\"
value=\"$quantity\"><br>\n";
echo "<input type=\"hidden\" name=\"amount\" value=\"$total\"><br>\n";
```

The “customerid” parameter was also passed in to this page but will be retrieved when needed to create a database query.

Information From the Database

First a note about the database. Assume a customer table with the following structure:

Field	Type	Key
id	int(11)	Primary Key
email	varchar(50)	
lastname	varchar(50)	
firstname	varchar(50)	
streetaddress1	varchar(50)	
streetaddress2	varchar(50)	
city	varchar(50)	
stateprov	varchar(50)	
pcode	varchar(50)	
country	varchar(50)	
password	varchar(50)	
dateadded	timestamp(14)	

The customer information needs to be retrieved from the database. Let's create a connection and the SQL query to retrieve the information we wish to pass to *PayPal*.

```
//include database password information etc.
$hostname = "myhost.com";
$username = "user";
$password = "password";
if(!($link = mysql_connect($hostname, $username,$password)))
      die("Could not connect to database.");
$databasename = "mydatabase";
if(!(mysql_select_db($databasename,$link)))
      die("Could not open table.");
```

Retrieve the information from the database using the primary key.

```
//now get customer info from database
$customerid = $HTTP_GET_VARS['id'];
$strsql="SELECT email, firstname, lastname, streetaddress1, ".
      "streetaddress2, city, stateprov, pcode FROM ".
      "tblcustomer WHERE id = '$customerid'";
if(!($rs= mysql_query($strsql, $link)))
      die("Could not open table.");
//only one row should be returned
$row = @ mysql_fetch_array($rs);
```

The form is now completed with the information retrieved from the database.

```
// now complete the form
echo "<input type=\"hidden\" name=\"email\"
value=\"$row[email]\"><br>\n";
echo "<input type=\"hidden\" name=\"first_name\"
value=\"$row[firstname]\"><br>\n";
echo "<input type=\"hidden\" name=\"last_name\"
value=\"$row[lastname]\"><br>\n";
echo "<input type=\"hidden\" name=\"address1\"
      value = \"$row[streetaddress1]\"><br>\n";
echo "<input type=\"hidden\" name=\"address2\" value =
      \"$row[streetaddress2]\"><br>\n";
echo "<input type=\"hidden\" name=\"city\" value=\"$row[city]\"><br>\n";
echo "<input type=\"hidden\" name=\"state\"
value=\"$row[stateprov]\"><br>\n";
echo "<input type=\"hidden\" name=\"zip\" value=\"$row[pcode]\"><br>\n";

?>
<!-- end of form  -->
</form>
```

You might have noticed that there is no “submit” button associated with our “paypal.php” page. Because our customer has already confirmed his order, code is used to submit this form to *PayPal*. The following JavaScript code will execute after the page has loaded and your customer will not see the “paypal.php” page at all.

```
<script type="text/javascript" language="JavaScript">
      //submit form
      document.order.submit();
</script>
</body>
<html>
```

Your customer will now be looking at a summary of his order on the *PayPal* site. All personal information will already be filled in.

Conclusion

An economical e-commerce solution was created using *PayPal*. A secure e-commerce transaction can be conducted in this way at minimal cost. Business benefits were maximised because we were able to retain full control of customer information in our own database and the customer was not inconvenienced by having to enter the same information twice.

About the Author

Peter Lavin runs a Web Design/Development firm in Toronto, Canada. For more information visit http://www.softcoded.com/. Peter may be reached at peter@softcoded.com.