Exploring PHP and MySQL Using an Online Travel Agency as a Case Study

Charles R. Moen, M.S.
Morris M. Liaw, Ph.D.

October 9, 2004
Our Goal

- Look at how PHP and a MySQL database have been used to develop a Web application – an online travel agency

- Some important features of the online travel agency
  1. Lists the items for sale
  2. Supports user authentication
  3. Provides a shopping cart
  4. Provides a method for collecting billing information from clients
  5. Sends automatic email notification to the company

- We will look at the code behind the pages to see how these features were implemented with PHP
An imaginary adventure travel company
http://dcm.cl.uh.edu/moen/

Wilderness Texas Adventures

“Nature is our widest home.”
Edward Hoagland

CANOEING AND HIKING JOURNEYS

Select a Destination
Its Purpose

- Advertises the adventure travel trips available from the company

- So that potential customers can
  - View descriptions of the trips
  - Sign up for and pay for trips

- And the company staff can
  - View lists of the trips and the customers who sign up for them
  - Add or delete customers for each of the trips
  - Add, change, and delete trips that are offered
1. Listing the Items for Sale

Items for sale

- The adventure travel trips
- Stored in a MySQL database
- The trip data is used to populate a Web page
The Customer Interface

Click on “Select a Destination”

Choose your adventure!

Welcome to Wilderness Texas Adventures, the Big Bend adventure travel specialists.

The Big Bend region in west Texas is a special place in the middle of the Chihuahuan Desert. It is here that the Rio Grande makes its majestic northeasterly bend, flowing between Texas and Mexico on its long journey through the canyons.

On our trips you can enjoy panoramic views while hiking in the Sierra del Carmen or paddle through the steep-walled canyons along the Mexican border.

Select one of the trips at the left to read more about our journeys. Please join us as we travel through this unique desert and mountain wilderness.
Information about the travel company

- Static pages
- Frameset of regular HTML files

Links will send a request for a PHP page along with a parameter embedded in the URL

```html
<a href="detail.php?dest=1" target="detail">Hiking in the Sierra del Carmens</a>

<a href="detail.php?dest=2" target="detail">Canoeing in Boquillas Canyon</a>
```
detail.php

- Displays the list of trips for sale, a destination photo, and a description
- All of this information was retrieved from a MySQL database by the PHP code in detail.php
- The database query used the value for “dest” passed as part of the URL detail.php?dest=1

### Wilderness Texas Adventures

<table>
<thead>
<tr>
<th>Trip name</th>
<th>Date</th>
<th>Cost</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring Trek 2002</td>
<td>Mar 2 - 4, 2002</td>
<td>$257.00</td>
<td>Available</td>
</tr>
<tr>
<td>Fall Trek 2002</td>
<td>Nov 15 - 17, 2002</td>
<td>$265.00</td>
<td>Filled</td>
</tr>
<tr>
<td>Spring Trek 2003</td>
<td>Mar 8 - 10, 2003</td>
<td>$265.00</td>
<td>Available</td>
</tr>
</tbody>
</table>

This challenging mountain range is filled with a unique, rugged beauty that will inspire every hiker. Its remote trails climb to 3000 ft. while winding past spectacular vistas.
$_GET

- Associative array that is automatically available
- (All php variables start with $)
- The preferred way to capture query string parameter values
- Replaces the older, deprecated $HTTP_GET_VARS
- Parameters are automatically available as global variables only if register_globals is “on” in php.ini

```php
detail.php

<table border="0" cellspacing="5">
  <?
  $dest = $_GET["dest"];  

```
Query the Database

Step 1 – Connect to MySQL
Step 2 – Select a database
Step 3 – Run a query
Step 4 – Get the result
Step 5 – Display the result on the page
Query the Database

Step 1 – Connect to MySQL

**mysql_connect**
- Connects PHP to MySQL and returns a connection variable
- First parameter is the name of the host running MySQL – often the same as the Web server host
- Second parameter is the username for the account
- Third parameter is the password

```php
<?
$dest = $_GET["dest"];  
$connection =  
    mysql_connect("localhost","turing","3nigM4");

<
<table border="0" cellpadding="5">

Listing the Items for Sale
```
include

- Inserts a file
- The username and password can be defined in another file, then included when necessary
- Easier to change them
- Slightly better security

settings.php

```php
<?
/host = "localhost";
/user = "turing";
/pass = "3nigM4";
/database = "wildtx";
?>
```

detail.php

```php
<
/dest = $_GET["dest"];
/include "settings.php";
/connection =
/mysql_connect($host,$user,$pass);
```
die
• Outputs a message
• Ends the script
• Message can include HTML formatting
@
• Prefix to a PHP function
• Suppresses the PHP error messages
• Allows us to write more “user friendly” custom error messages

```
<table border="0" cellpadding="5">
  <?
  $dest = $_GET["dest"];
  include "settings.php";
  $connection =
  @mysql_connect($host,$user,$pass)
  or die("Sorry, no connection to MySQL.");
</?
```

Query the Database
Step 1 – Connect to MySQL

Listing the Items for Sale
Query the Database
Step 1 – Connect to MySQL

mysql_pconnect
• Connects to MySQL with a persistent connection
• Persistent connections are not closed in the script
• Maintained in a pool
• Better performance

detail.php

```php
<table border="0" cellpadding="5">
<?
$dest = $_GET["dest"];
include "settings.php";
$connection = 
    @mysql_pconnect($host,$user,$pass)
or die("Sorry, no connection to MySQL.");
```
Query the Database
Step 2 – Select the Database

detail.php

```php
<table border="0" cellpadding="5">
<?
$dest = $_GET["dest"]; include "settings.php";
$connection = @mysql_pconnect($host,$user,$pass)
or die("Sorry, no connection to MySQL.");
mysql_select_db($database,$connection)
or die("Sorry, the database is down.");
```
Query the Database

Step 3 – Run a query

- Create a SQL query
- Assign it to a variable
- Do not add “;” at the end of the SQL statement

detail.php

```php
<?
$dest = $_GET['dest'];
include "settings.php";
$conn =
@mysql_pconnect($host,$user,$pass)
or die("Sorry, no connection to MySQL.");
@mysql_select_db($database,$conn)
or die("Sorry, the database is down.");
$sql =
"SELECT destName, image, imageText, description
FROM Destination
WHERE destinationID = $dest";
```
Listing the Items for Sale

Query the Database
Step 3 – Run a query

detail.php

```php
<table border="0" cellspacing="5">
<?
$dest = $_GET["dest"];
include "settings.php";
$connection =
    @mysql_pconnect($host,$user,$pass)
    or die("Sorry, no connection to MySQL.");
@mysql_select_db($database,$connection)
    or die("Sorry, the database is down.");
$sql =
    "SELECT destName, image, imageText, description
        FROM Destination
        WHERE destinationID = $dest";
$sql_result = @mysql_query($sql,$connection)
    or die("No trips to that destination.");
```
Query the Database
Step 4 – Get the result

detail.php

```php
<?
$dest = $_GET['dest'];
include "settings.php";
$connection =
@mysql_pconnect($host,$user,$pass)
or die("Sorry, no connection to MySQL.");
@mysql_select_db($database,$connection)
or die("Sorry, the database is down.");
$sql =
"SELECT destName, image, imageText, description
FROM Destination
WHERE destinationID = $dest";
$sql_result = @mysql_query($sql,$connection)
or die("No trips to that destination.");
$row = mysql_fetch_array($sql_result);
```
• Each field in the result set row can be retrieved by using the field name as an index.
• Additional queries can be run in the same PHP file
• When we retrieve the result set containing the list of trips to a destination, there may be more than one row
• A while loop is used to retrieve each row
• The field number can also be used as an index

```
detail.php
$sql =
    "SELECT tripID, name, date, cost, filled, max
    FROM Trip
    WHERE destinationID = $dest";
$sql_result = mysql_query($sql,$connection)
or die("Could not execute query 2.");
while( $row = mysql_fetch_array($sql_result) ){
    $tripID = $row[0];
    $name = $row[1];
    $date = $row[2];
    $cost = $row[3];
    $cost = sprintf("$%0.2f",$cost);
    $filled = $row[4];
    $maxPeople = $row[5];
```
• One row in the Web page table is printed in each iteration of the while loop.

detail.php

```php
while( $row = mysql_fetch_array($sql_result) ){
    $tripID = $row[0];
    $name = $row[1];
    $date = $row[2];
    $cost = $row[3];
    $cost = sprintf("$%0.2f", $cost);
    $filled = $row[4];
    $maxPeople = $row[5];
    echo "<tr valign="top">\n    $ filled ){
        $status = "Filled";
        echo "$status";
    } else{
        $status = "Available";
        echo "<a href="/reservations.php?trip=$tripID" target="_parent">Sign up</a>";
    }
    echo "</td><td>$name</td><td>$date</td><td>$cost";  
    <td>$status</td></tr>"; }
```
2. User Authentication

Staff members must be authenticated to

- View the trip guest lists
- Add or delete customers to the guest lists
- Add, change, or delete trips
Form-Based Authentication

- Usernames and passwords are stored in the database
- Users are required to enter a username and password in a Web form
- When the form is submitted, the data entered by the user is compared to the usernames and passwords stored in the database
- If the data matches a username and password in the database, the script sets a session variable that shows that the user has logged in
User Authentication in the Wilderness Texas Web Site

- Uses form-based authentication

Login form

A Web page used by the staff after logging in
User Authentication

Define the Login Form

- The login dialog is a Web form

```
<html><head>
<link rel=stylesheet href="wildtx.css" type="text/css">
<title>Login</title>
</head>
<body>
<center>
<form action="login.php" method="POST">
<table border="0" cellpadding="1" bgcolor="#000000">
<tr><td align="left">
<span style="color:#ffffff">Staff Login</span></td></tr>
<tr><td>
<table border="0" cellpadding="5" bgcolor="#d2d2cc">
<tr><td colspan="2" align="left">
<em>Enter your user name and password:</em>
</td></tr>
<tr><td align="left">User Name:</td><td align="left">
<input type="text" name="usr" size="35">
</td></tr>
<tr><td align="left">Password:</td><td align="left">
<input type="password" name="pw" size="35">
</td></tr>
<tr><td colspan="2" align="center">
<input type="submit" name="btnOk" value="OK">
</td></tr>
</table>
</td></tr>
</table>
</form>
</center>
```

login.php
Define a Function to Display the Login Form

• The HTML for the entire Web page that displays the form is in the “showForm” function defined in login.php

• Three parameters
  $msg = A message for the user
  $u = a value for the field used to get the user name
  $p = a value for the field used to get the password

```php
function showForm($msg="&nbsp;", $u="", $p="") {
    ?><!-- ... -->
    <?=$msg?>
    <form action="login.php" method="POST">
      <table border="0" cellpadding="1" bgcolor="#000000">
        <tr><td align="left">
          <span style="color:#ffffff">Staff Login</span></td></tr>
        <tr><td>
          <table border="0" cellpadding="5" bgcolor="#d2d2cc">
            <tr><td colspan="2" align="left">
              <em>Enter your user name and password:</em>
            </td></tr>
            <tr><td align="left">User Name:</td><td align="left">
              <input type="text" name="usr" value="<?=$u?>" size="35">
            </td></tr>
            <tr><td align="left">Password:</td><td align="left">
              <input type="password" name="pw" value="<?=$p?>" size="35">
            </td></tr>
            <tr><td colspan="2" align="center">
              <input type="submit" name="btnOk" value="OK">
            </td></tr>
          </table>
        </td></tr></table></form></center>
    ?></?
}
```
User Authentication

Using showForm()

1. login.php contains both the login dialog form inside showForm() and the PHP code to handle the data entered in the form.
2. The data-handling code examines the input from the form, then if it is not acceptable, it calls showForm() to display the login dialog again.
3. By putting the HTML in a function, we can call it from multiple branches.

/*
If this is the first request for login.php then
   Call showForm() to show the login dialog

Else the user clicked on OK in the dialog
   If the user did not fill in both fields then
      Display an error message
      Call showForm() to show the login dialog

   Else
      Query the database for the username and password
      If there is no match in the database then
         Display an error message
         Call showForm() to show the login dialog

      Else
         Set the "logged in" session variable to true
         Redirect the request to the staff page

*/
Show an Empty Form the First Time the Page is Requested

- When the file is requested, `isset()` checks whether the `_POST` array has a variable corresponding to one of the form fields.
- If `_POST["user"]` is not set, then this is the first time the form was requested, so the form is displayed and no data processing is required.

```php
login.php

if( !isset($_POST["usr"])) {
    showForm();
}
```
Check the User Input

- If this request is the result of the user clicking on “OK”, then 
  \$\_POST will have variables for both form fields
- \texttt{empty()} checks whether the user has entered a value in these fields
- If either field was empty, then \texttt{showForm} is called to redisplay the login dialog
- Arguments include the values already entered

```php
if( !isset($_POST["usr"])) {
    showForm();
}
else if( empty($_POST["usr"] || empty($_POST["pw"])) ){
    showForm("Please fill in both fields.",
              $_POST["usr"], $_POST["pw"]
            );
}
```
User Authentication

If Both Fields Have Values
Check the Database

- When both fields have values we can check the database
- Look for a match with the username and password entered by the user
- If no match, call showForm()
- If there is a match, set the “wildtxloggedin” session variable and redirect

```php
//...
else{
    $usr = $_POST["usr"];  
    $pw = $_POST["pw"]; 
    $connection = @mysql_pconnect($host,$user,$pass) 
        or die ("Sorry, no connection to MySQL."); 
    @mysql_select_db($database, $connection) 
        or die("Sorry, the database is not available."); 
    $sql = "SELECT Count(*) FROM Staff 
        WHERE staffID = '$usr' 
            AND password = '$pw'"; 
    $sql_result = @mysql_query($sql,$connection) 
        or die("Could not execute query 1."); 
    $row = mysql_fetch_array($sql_result); 
    if( $row["Count(*)"] == 0 ){
        show_form(); 
    }
    else{
        $wildtxloggedin = TRUE; 
        header("Location: admin.php"); 
        exit; 
    }
}
```
For better security
- Store the passwords in encrypted form inside the database by encrypting them with PHP before they are stored
  $salt = substr($usr,0,2);
  $pw = crypt($pw,$salt);
- In login.php, encrypt the user entry for the password before running the query that looks for a match

```php
//...
else{
    $usr = $_POST["usr"];  
    $pw = $_POST["pw"];  
    $salt = substr($usr,0,2);  
    $pw = crypt($pw,$salt);
    $connection = @mysql_pconnect($host,$user,$pass)
        or die ("Sorry, no connection to MySQL.");
    @mysql_select_db($database, $connection)
        or die("Sorry, the database is not available.");
    $sql = "SELECT Count(*) FROM Staff
           WHERE staffID = '$usr'
           AND password = '$pw'";
    $sql_result = @mysql_query($sql,$connection)
        or die("Could not execute query 1.");
    $row = mysql_fetch_array($sql_result);
    if( $row["Count(*)"] == 0 ){
        show_form();
    }
    else{
        $wildtxloggedin = TRUE;
        header("Location: admin.php");
        exit;
    }
```
3. Shopping Cart

When a customer signs up for a trip

- The trip is added to a shopping cart
- Multiple trips can be added to the cart
- Trips can be removed from the cart
1. To sign up for a trip, click “Sign up” on detail.php

<table>
<thead>
<tr>
<th>Trip name</th>
<th>Date</th>
<th>Cost</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign up</td>
<td>Spring Trek 2002</td>
<td>Mar 2 - 4, 2002</td>
<td>$257.00</td>
</tr>
<tr>
<td>Filled</td>
<td>Fall Trek 2002</td>
<td>Nov 15 - 17, 2002</td>
<td>$265.00</td>
</tr>
<tr>
<td>Sign up</td>
<td>Spring Trek 2003</td>
<td>Mar 8 - 10, 2003</td>
<td>$265.00</td>
</tr>
</tbody>
</table>

2. reservations.php asks for the number of people that you are taking on this trip

Select the number ➔
Click ➔

Wilderness Texas Adventures

You have selected the following trip:

Trip name: Spring Trek 2003
Date: Mar 8 - 10, 2003
Cost per person: $265.00

Please tell us how many people you wish to make reservations for:

Number of people: 1

Submit Number of People
3. `cart.php` adds this trip to the shopping cart, and then shows the entire content of the cart.

The Wilderness Texas Adventures

Your shopping cart has the following trips:

<table>
<thead>
<tr>
<th>Trip name</th>
<th>Date</th>
<th>Cost per Person</th>
<th>People</th>
<th>Trip cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring Trek 2003</td>
<td>Mar 8 - 10, 2003</td>
<td>$265.00</td>
<td>1</td>
<td>$265.00</td>
</tr>
</tbody>
</table>

**TOTAL**  $265.00

(There is no sales tax required for our trips.)

You can [confirm your reservations and checkout](#) at any time.

[Sign up and add another trip.](#)

[Remove a trip from the cart.](#)
Wilderness Texas shopping cart data is stored in a MySQL table

- Database storage is more secure than hidden fields or file storage
- The unique PHP session ID is stored as the “userID” so that each customer’s purchases are easily identifiable

The table definition

```
CREATE TABLE Cart(
    itemID    INT AUTO_INCREMENT,
    userID    VARCHAR(50),
    tripID    VARCHAR(10),
    tripName  VARCHAR(50),
    numPeople VARCHAR(10),
    tripDate  VARCHAR(50),
    tripCost  VARCHAR(50),
    totalCost VARCHAR(50),
    dateAdded VARCHAR(50),
    PRIMARY KEY(itemID)
);
```
First Create a Session

Session

• Stores the variables that should be associated with a particular user

• These variables can be accessible on all the application’s pages that the user visits – an easy way to pass values from page to page

session_start()

• Starts the session or uses one that is already started

• Gives the client a unique session identifier: $PHPSESSID

reservations.php

```php
include "settings.php";
session_start();
```
Register Session Variables

**session_register()**
- Stores a variable in the session, but does not set its value
- `isset()` can be used to check whether a variable already has a value before initializing it
- Use the assignment operator to set the value

```php
include "settings.php";
session_start();
session_register("sess_tripID");
session_register("sess_tripName");
session_register("sess_tripDate");
session_register("sess_tripCost");
session_register("sess_totalPeople");
if(!isset($sess_totalPeople)) $sess_totalPeople = 0;
```
$sess_tripID = $_GET["trip"];
• Gets its value from the query string parameter
The trip's data, such as its cost, comes from a MySQL query

Advantage
• Using session variables for values that must be passed between pages is more convenient than URL re-writing or using hidden form fields
• More secure
The data for the trip is collected by the next page, `cart.php`:

The session variables that were set from client input:
- Trip ID
- Number of people

The session variables set by database query results:
- Trip name
- Date of the trip
- Cost per person

Then additional data is calculated:
- Total cost
- Sign up date

```php
session_start();
session_register("sess_numPeople");
session_register("totalCost");
session_register("dateAdded");

if(isset($_POST["numPeople"]))
    $sess_numPeople = $_POST["numPeople"];

if( isset($sess_tripID) && isset($numPeople) ){
    $tripInCart = true;
    $tripCostPrint = sprintf("$%0.2f",$sess_tripCost);
    $totalCost = $sess_numPeople * $sess_tripCost;
    $totalCostPrint = sprintf("$%0.2f",$totalCost);
    $dateAdded = date("Y-m-d");
}

//...
```
Before adding a trip to the database, cart.php checks whether the trip is already there – a problem that is sometimes caused when the user presses the “Back” button.

```php
//...
if( isset($sess_tripID) ) {
    $sql="SELECT tripID
        FROM Cart
        WHERE userID='$PHPSESSID';
    $sql_result = mysql_query($sql,$connection)
        or die("Could not execute query 1.");
    $duplicate = false;
    while( $row = mysql_fetch_array($sql_result) ) {
        $testTripID = $row[0];
        if( $testTripID == $sess_tripID ) {
            $duplicate = true;
        }
    }
}
//...
```
The trip is stored in the shopping cart by inserting it into the Cart table of the database.
In the final step before displaying the page, the Cart table is queried to retrieve its entire content so that it can be displayed in the shopping cart table. But if the table has no trips, a message is displayed.
4. Collecting Information from Users

When the customer is ready to pay

- A Web form is used to collect the billing information
- The form changes, based on the cart contents
The Billing Form

When the client clicks on the link to check out, the billing form is displayed by checkout.php.
Define a Function to Display the Billing Form

- The HTML that displays the form is in a new “show_form” function defined in checkout.php.
- Its arguments are used for the default values of the form fields.

```
checkout.php

//...
function show_form( $fName="", $mName="", $lName="", $address="", $city="", $state="", $zip="", $phone="", $email="", $cardNumber="", $cardExpirationMonth="", $cardExpirationYear="", $cardType="") {

%>
<!--HTML code that defines the form is here-->
<td>First name:</td>
<td>
<input type="text" name="fName" size="27"
value="<? echo $fName ?>">

<% } %>
```
Common Problem – A Required Field Is Blank

When checkout.php is first requested, show_form is invoked with no arguments.

When the client submits the form, the request is sent to checkout.php again, and the variable for each required field is checked to verify that it was filled in.

If a required field is blank, then show_form is called again, and it displays a message asking the client to fill in all the fields.

```php
//...
if ( !isset( $fName ) ) {
    show_form();
} else {
    if ( empty( $fName ) || empty( $lName ) || empty( $address ) || empty( $city ) || empty( $state ) || empty( $zip ) || empty( $phone ) || empty( $email ) || empty( $cardNumber ) || $cardExpirationMonth == "--" || $cardExpirationYear == "--" ) {
        echo "<span style="color:#ff0000; font-style:italic">";
        <p>You did not fill in all the fields, please try again</p>
        echo "</span></p>";
        show_form( $fName, $mName, $lName, $address, $city, $state, $zip, $phone, $email, $cardNumber, $cardExpirationMonth, $cardExpirationYear, $cardType );
    }
} //...
```
Common Problem – Client Clicks the “Back” Button

• If the client clicks the browser’s “Back” button after filling in the billing form
  – The value of each field is saved in a session variable
  – Nothing is lost

• The next time the client selects “checkout,” the data will be passed to show_form
By using an array to hold the values displayed in an option list:

- The values are easier to change when the programmer has to update the page
- Code for the option list is more compact

```php
//...
$years = array("--","2004","2005","2006","2007",
               "2008","2009","2010","2011","2012");

?>
</select>
<em>Year</em>
<select name="cardExpirationYear">
<?
for( $i=0; $i<count($years); $i++ ) {
    echo "<option";
    if( $cardExpirationYear==$years[$i] ) {
        echo " selected ";
    }
    echo ">$years[$i]\n";
}
?>
</select>
```
5. Sending Email Notification

After the customer has checked out and sent in the billing data

- Notification is sent to the company
- An email message is sent automatically
Final Step

When the client is finished and clicks on “Send This Order!”

- A final summary is displayed to the client on the last page, send.php
- An email notification is sent to the company by the PHP code in send.php

Wilderness Texas Adventures

Reservation confirmed!

Billing information:
Name: Charle R. Moen
Address: 724 W. Temple
         Houston, TX 77009
Phone: (713) 802-1445
Email: crmoen@juno.com
Credit card number: 1234567890123456
Expiration date: Visa, Expires 1 - 2003

Your guests:
Spring Trek 2002 Guest 1: Renee van der Vennet

Cart contents:

<table>
<thead>
<tr>
<th>Trip name</th>
<th>Date</th>
<th>Cost per Person</th>
<th>People</th>
<th>Trip cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring Trek 2002</td>
<td>Mar 2 - 4, 2002</td>
<td>$257.00</td>
<td>2</td>
<td>$514.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TOTAL</td>
</tr>
</tbody>
</table>

THANKS FOR SIGNING UP WITH WILDERNESS TEXAS!
We will send you a detailed itinerary in the mail by the next business day.
Sending Email Notification

Sending the Email Message

send.php

```php
//...
$msg = "WILDERNESS TEXAS TRIP RESERVATION FORM\n";
$msg .= "-----------------------------------------\n"
$msg .= "Order date:.......$dateAdded\n"
$msg .= "BUYER INFORMATION\n"
$msg .= "First name:.......$sess_fName\n"
$msg .= "Middle name:.......$sess_mName\n"
$msg .= "Last name:.......$sess_lName\n"
$msg .= "Address:..........$sess_address\n"
$msg .= "                  $sess_city, $sess_state\n"
$msg .= "                              $sess_zip\n"
$msg .= "Phone:............$sess_phone\n"
$msg .= "Email:............$sess_email\n"
//...

$recipient = "crmoen@juno.com"
$subject = "WILDERNESS TEXAS ADVENTURES ORDER"
$mailheaders = "From: WILDERNESS TEXAS ADVENTURES\n"
$mailheaders .= "Reply-To: $sess_email\n"

mail("$recipient", "$subject", $msg, $mailheaders);
```

• First, all the data for the purchase is concatenated in a string, $msg

• The email recipient, subject, and the “from” and “to” headers are also added to strings

• The email message is sent when mail() is invoked with these strings as arguments

Sending Email Notification
Conclusion

PHP and MySQL

• Important features for e-commerce
  1. Listing the items for sale
  2. Supporting user authentication
  3. Providing a shopping cart
  4. Collecting billing information from clients
  5. Sending automatic email notification

• Powerful, freely available, and easy to learn and use
Any questions?