PHP and Database Connectivity

- PHP offers built-in support for a wide variety of database systems from Unix DBM through relational systems such as MySQL to full size commercial systems like Oracle.

- We’ll continue to use MySQL as the underlying database system so that you can easily compare the work we’ve done with MySQL using Java servlets and JSPs.

- Before you go any further in these notes you must configure PHP to access MySQL databases. Beginning with PHP 5, MySQL is not enabled by default in PHP, nor is the MySQL library bundled with PHP.
  - Versions of MySQL greater than 4.1.0 use MySQLi extensions.
  - Versions of MySQL less than 4.1.0 use MySQL extensions.
This is the MySQL library that both mysql and mysqli extensions require.
These are the MySQL extension files that will be used to link PHP to MySQL.
These two extensions will not be commented out. At loadtime, these extensions will now be included in the PHP environment, provided that the file php.ini is set.
PHP should be configured for MySQL. You can verify that the php.ini file was properly read and the MySQL extensions are loaded by running the info.php script and looking for these entries.
PHP and Database Connectivity (cont.)

• PHP contains a fairly extensive set of commands that can be used to access and manipulate MySQL databases.

• A very brief listing of some of these commands appears on the next page.

• For a complete listing see:


### Table of Contents

- `mysql_affected_rows` -- Get number of affected rows in previous MySQL operation
- `mysql_change_user` -- Change logged in user of the active connection
- `mysql_client_encoding` -- Returns the name of the character set
- `mysql_close` -- Close MySQL connection
- `mysql_connect` -- Open a connection to a MySQL Server
- `mysql_create_db` -- Create a MySQL database
- `mysql_data_seek` -- Move internal result pointer
- `mysql_db_name` -- Get result data
- `mysql_db_query` -- Send a MySQL query
- `mysql_drop_db` -- Drop (delete) a MySQL database
- `mysql_errno` -- Returns the numerical value of the error message from previous MySQL operation
- `mysql_error` -- Returns the text of the error message from previous MySQL operation
- `mysql_escape_string` -- Escapes a string for use in a `mysql_query`
- `mysql_fetch_array` -- Fetch a result row as an associative array, a numeric array, or both
- `mysql_fetch_assoc` -- Fetch a result row as an associative array
- `mysql_fetch_field` -- Get column information from a result and return as an object
- `mysql_fetch_lengths` -- Get the length of each output in a result
- `mysql_fetch_object` -- Fetch a result row as an object
- `mysql_fetch_row` -- Get a result row as an enumerated array
- `mysql_field_flags` -- Get the flags associated with the specified field in a result
- `mysql_field_len` -- Returns the length of the specified field
- `mysql_field_name` -- Get the name of the specified field in a result
- `mysql_field_seek` -- Set result pointer to a specified field offset
- `mysql_field_table` -- Get name of the table the specified field is in
- `mysql_field_type` -- Get the type of the specified field in a result
- `mysql_free_result` -- Free result memory
- `mysql_get_client_info` -- Get MySQL client info
Portion of `mysqli.dll` Extension

```plaintext
mysqli_fieldtell -- Get current field offset of a result pointer
mysqli_free_result -- Frees the memory associated with a result
mysqli_get_client_info -- Returns the MySQL client version as a string
mysqli_get_client_version -- Get MySQL client info
mysqli_get_host_info -- Returns a string representing the type of connection used
mysqli_get_metadata -- Alias for `mysqli_stmt_result_metadata()`
mysqli_get_proto_info -- Returns the version of the MySQL protocol used
mysqli_get_server_info -- Returns the version of the MySQL server
mysqli_get_server_version -- Returns the version of the MySQL server as an integer
mysqli_info -- Retrieves information about the most recently executed query
mysqli_init -- Initializes mysqli and returns an object for use with
mysqli_real_connect
mysqli_insert_id -- Returns the auto generated id used in the last query
mysqli_kill -- Asks the server to kill a MySQL thread
mysqli_master_query -- Enforce execution of a query on the master in a master/slave setup
mysqli_more_results -- Check if there is any more query results from a multi query
mysqli_multi_query -- Performs a query on the database
mysqli_next_result -- Prepare next result from multi_query
mysqli_num_fields -- Get the number of fields in a result
mysqli_num_rows -- Gets the number of rows in a result
mysqli_options -- Set options
mysqli_param_count -- Alias for `mysqli_stmt_param_count()`
mysqli_ping -- Pings a server connection, or tries to reconnect if the connection has gone down
```

© Mark Llewellyn

CNT 4714: PHP – Part 3
PHP and Database Connectivity (cont.)

- Now that you have PHP set to accept MySQL extensions, let’s connect to the bike database that we used for examples with Java servlets and JSPs.

- The following example is a simple database connection process in PHP where the client interacts with the database from an XHTML form that simply asks them to select which attributes from the bikes table that they would like to display. This is done through the data.html file.

- When the client clicks the submit query button, the database.php script executes by connecting to the database, posting the query, retrieving the results, and displaying them to the client.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- data.html -->
<!-- Querying a MySQL Database From a PHP Script -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head> 
<title>Sample Database Query From PHP</title> 
</head>
<body style = "background-color: #545454" background=image1.jpg >
<h2 style = "font-family: arial color: blue"> Querying a MySQL database from a PHP Script. </h2>
<form method = "post" action = "database.php">
<p>Select a field to display:

<!-- add a select box containing options for SELECT query -->

<select name = "select">

<option selected = "selected">*</option>
<option>bikename</option>
<option>size</option>
<option>color</option>
<option>cost</option>
<option>purchased</option>
<option>mileage</option>
</select>
</p>
<input type = "submit" value = "Send Query" style = "background-color: blue; color: yellow; font-weight: bold" />

</form>

</body> 
</html>
<?php

extract( $_POST );

// build SELECT query
$query = "SELECT "$select" FROM bikes";

// Connect to MySQL
if (mysqli_connect("localhost", "root", "root"))
    die("Could not connect to database");

// Connect to MySQL database.
// URL, username, password, and database all specified.

Default query is to select the attributes chosen by the client for use in a SELECT query.
```php
// query bikedb database
if ( !( $result = mysql_query( $database, $query ) ) ) {
    print( "Could not execute query! <br/>");
    die( mysql_error() );
}
?>

<h3 style = "color: blue"> Database Search Results </h3>
<table border = "1" cellpadding = "3" cellspacing = "3"
 style = "background-color: #00FFFF">
<?php
// fetch meta-data
$metadata = mysqli_fetch_fields( $result);
print("<tr>");
for ($i=0; $i<count($metadata); $i++){
    print("<td>");
    printf("%s",$metadata[$i]-name);
    print("</td>");
}
print("</tr>");
```

Get metadata for the query

Display metadata in the top row of the table
// fetch each record in result set
for ( $counter = 0;
    $row = mysql_fetch_row( $result );
    $counter++ ){ // build table to display results
    print( "<tr>" );
    foreach ( $row as $key => $value )
        print( "<td>$value</td>" );
    print( "</tr>" );
}
mysql_close( $database );
?>
</table>
<br />
Your search yielded <strong>
<?php print( "$counter" ) ?> results.<br />
</strong>
<h5>Please email comments to
<a href = "mailto:markl@cs.ucf.edu">
    markl@cs.ucf.edu
</a>
</h5>
</body></html>
Execution of data.html – Client side

Execution of data.html (client side of the application) showing the drop-down menu for the client to select the attributes for the query.

When the selection is made and the **Send Query** button is clicked the results on the following page will be displayed.
Results of query `SELECT * FROM bikes`. Display indicates that 10 rows were included in the result.

<table>
<thead>
<tr>
<th>bikename</th>
<th>size</th>
<th>color</th>
<th>cost</th>
<th>purchased</th>
<th>mileage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battaglin Carrera</td>
<td>60</td>
<td>red/white</td>
<td>4000</td>
<td>2001-03-10</td>
<td>11200</td>
</tr>
<tr>
<td>Bianchi Corse Evo 4</td>
<td>58</td>
<td>celeste</td>
<td>5700</td>
<td>2004-12-02</td>
<td>300</td>
</tr>
<tr>
<td>Bianchi Evolution 3</td>
<td>58</td>
<td>celeste</td>
<td>4800</td>
<td>2003-11-12</td>
<td>2000</td>
</tr>
<tr>
<td>Colnago Dream Rabobank</td>
<td>60</td>
<td>blue/orange</td>
<td>5500</td>
<td>2002-07-07</td>
<td>4300</td>
</tr>
<tr>
<td>Colnago Superissimo</td>
<td>59</td>
<td>red</td>
<td>3800</td>
<td>1996-03-01</td>
<td>13000</td>
</tr>
<tr>
<td>Eddy Merckx Domo</td>
<td>58</td>
<td>blue/black</td>
<td>5300</td>
<td>2004-02-02</td>
<td>0</td>
</tr>
<tr>
<td>Eddy Merckx Molteni</td>
<td>58</td>
<td>orange</td>
<td>5100</td>
<td>2004-08-12</td>
<td>0</td>
</tr>
<tr>
<td>Gianni Motta Personal</td>
<td>59</td>
<td>red/green</td>
<td>4400</td>
<td>2000-05-01</td>
<td>8700</td>
</tr>
<tr>
<td>Gios Torino Super</td>
<td>60</td>
<td>blue</td>
<td>2000</td>
<td>1998-11-08</td>
<td>9000</td>
</tr>
<tr>
<td>Schwinn Paramount P14</td>
<td>60</td>
<td>blue</td>
<td>1800</td>
<td>1992-03-01</td>
<td>200</td>
</tr>
</tbody>
</table>
Cookies

- A **cookie** is a text file that a Web site stores on a client’s computer to maintain information about the client during and between browsing sessions.

- A Web site can store a cookie on a client’s computer to record user preferences and other information that the Web site can retrieve during the client’s subsequent visits. For example, many Web sites use cookies to store client’s zipcodes. The Web site can retrieve the zipcode from the cookie and provide weather reports and news updates tailored to the user’s region.

- Web sites also use cookies to track information about client activity. Analysis of information collected via cookies can reveal the popularity of Web sites or products.
Cookies (cont.)

- Marketers use cookies to determine the effectiveness of advertising campaigns.

- Web sites store cookies on users’ hard drives, which raises issues regarding security and privacy. Web sites should not store critical information, such as credit-card numbers or passwords, in cookies, because cookies are just text files that anyone can read.

- Several cookie features address security and privacy concerns. A server can access only the cookies that it has placed on the client.

- A cookies has an expiration date, after which the Web browser deletes it.
Cookies (cont.)

• Users who are concerned about the privacy and security implications of cookies can disable them in their Web browsers. However, the disabling of cookies can make it impossible for the user to interact with Web sites that rely on cookies to function properly.

• Information stored in the cookie is sent to the Web server from which it originated whenever the user requests a Web page from that particular server. The Web server can send the client XHTML output that reflects the preferences or information that is stored in the cookie.

• The location of the cookie file varies from browser to browser. Internet Explorer places cookies in the Cookies directory located at `C:\Documents and Settings\...\Cookies`
Cookies (cont.)

- After a cookie is created, a text file is added to this directory. While the name of the file will vary from user to user a typical example is shown below.

- The contents of a cookie are shown on page 74.
Cookies (cont.)

• Now let’s create the code necessary to create our own cookie.

• In this example, a PHP script is invoked from a client-side HTML document. The HTML document creates a form for the user to enter the information that will be stored in the cookie. (Often the information that is stored in a cookie will be extracted from several different areas and may involved tracking the client’s actions at the Web site.)

• Once the user has entered their information, when they click the Write Cookie button, the cookies.php script executes.

• The XHTML document and the PHP script are shown on the next pages. The XHTML document cookies.html is on page 36 and the PHP script cookies.php appears on page 37.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- cookies.html -->
<!-- Writing a Cookie -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head> <title>Writing a cookie to the client computer</title>   </head>

<body style = "font-family: arial, sans-serif; background-color: #856363" background=image1.jpg>
<h2>Click Write Cookie to save your cookie data.</h2>
<form method = "post" action = "cookies.php" style = "font-size: 10pt"
    background-color: #856363">
    <strong>Name:</strong><br />
    <input type = "text" name = "NAME" /><br />
    <strong>Height:</strong><br />
    <input type = "text" name = "HEIGHT" /><br />
    <strong>Favorite Color:</strong><br />
    <input type = "text" name = "COLOR" /><br />
    <p>
    <input type = "submit" value = "Write Cookie" style = "background-color: #0000FF; color: yellow; font-weight: bold" /></p>
    </form>

</body>  </html>
<?php

// cookies.php
// Program to write a cookie to a client's machine
extract( $_POST );
// write each form field's value to a cookie and set the
// cookie's expiration date
setcookie( "Name", $NAME, time() + 60 * 60 * 24 * 5 );
setcookie( "Height", $HEIGHT, time() + 60 * 60 * 24 * 5 );
setcookie( "Color", $COLOR, time() + 60 * 60 * 24 * 5 );
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>      <title>Cookie Saved</title>   </head>
<body style = "font-family: arial, sans-serif", background=image1.jpg>
<p><b>The cookie has been set with the following data:</b></p>
<br /><span style = "color: blue">Name:</span><?php print( $NAME ) ?><br />
<span style = "color: blue">Height:</span><?php print( $HEIGHT ) ?><br />
<span style = "color: blue">Favorite Color:</span><?php print( "$COLOR" ) ?><br />
<p>Click <a href = "readCookies.php">here</a>  to read the saved cookie.</p>
</body>  </html>
Cookies (cont.)

HTML form generated by cookies.html
Cookies (cont.)

Output from cookies.php script showing the values in the newly created cookie.

The cookie has been set with the following data:

Name: Mark Llewellyn
Height: 6' 3"
Favorite Color: blue

Click here to read the saved cookie.
Cookies (cont.)

- Once the cookie has been created, the cookies.php script gives the user the chance to view the newly created cookie by invoking the readCookies.php script from within the cookies.php script by clicking on the link.

- The readCookies.php script code is illustrated on the next page followed by the output from the execution of this PHP script.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- readCookies.php -->
<!-- Program to read cookies from the client's computer -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head><title>Read Cookies</title></head>
<body style = "font-family: arial, sans-serif" background=image1.jpg>

<p>
   <strong>The following data is saved in a cookie on your computer.</strong>
</p>
<table border = "5" cellspacing = "0" cellpadding = "10">
 <?php
   // iterate through array $_COOKIE and print name and value of each cookie
   foreach ( $_COOKIE as $key => $value )
     print( "<tr>
         <td bgcolor="#F0E68C">$key</td>
         <td bgcolor="#FFA500">$value</td>
     </tr>" );

?>
</table>
</body>  </html>

Superglobal array holding cookie.
Cookies (cont.)

The following data is saved in a cookie on your computer.

<table>
<thead>
<tr>
<th>Name</th>
<th>Mark Llewellyn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>6' 3&quot;</td>
</tr>
<tr>
<td>Color</td>
<td>blue</td>
</tr>
</tbody>
</table>
Contents of the cookie stored on the client machine.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP_ACCEPT</td>
<td><em>/</em></td>
</tr>
<tr>
<td>HTTP_ACCEPT_LANGUAGE</td>
<td>en-us</td>
</tr>
<tr>
<td>HTTP_UA_CPU</td>
<td>x86</td>
</tr>
<tr>
<td>HTTP_ACCEPT_ENCODING</td>
<td>gzip, deflate</td>
</tr>
<tr>
<td>HTTP_USER_AGENT</td>
<td>Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.04506; InfoPath.2)</td>
</tr>
<tr>
<td>HTTP_HOST</td>
<td>localhost:8081</td>
</tr>
<tr>
<td>HTTP_CONNECTION</td>
<td>Keep-Alive</td>
</tr>
<tr>
<td>HTTP_COOKIE</td>
<td>Name=Mark+Llewellyn; Height=6%27+3%22; Color=blue</td>
</tr>
<tr>
<td>PATH</td>
<td>C:\Windows\System32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files\MySQL\MySQ...</td>
</tr>
</tbody>
</table>

Apache Environment
Cookies (cont.)

Actual text file holding cookie data for the cookie that was created in this example.
Dynamic Content in PHP

- Of all the strengths PHP exhibits as a server-side scripting language, perhaps its greatest strength lies in its ability to dynamically change XHTML output based on user input.

- In this final section of notes, we’ll build on the examples we’ve constructed in the previous two sets of notes by combining form.html and form.php into one dynamic PHP document named dynamicForm2.php.

- We’ll add error checking to the user input fields and inform the user of invalid entries on the form itself, rather than on an error page. If an error exists, the script maintains the previously submitted values in each form element.

- Finally, after the form has been successfully completed, we’ll store the input from the user in a MySQL database.
Basically, the same registration form that was used in a previous example.
User fills in the form and clicks the Register button.
Hi Mark. Thank you for completing the survey.
You have been added to the RadSport mailing list.

The following information has been saved in our database:

Name          Email         Phone      OS
Mark Llewellyn markl@cs.ucf.edu (407)823-2790 Windows Vista

Click here to view entire database.
Screen the user sees after clicking to see the entire database.
Dynamic nature of the PHP form is illustrated when the user fails to enter proper information into the form. In this case, the user forgot to enter their first name. Error checking is in place on each user input location and the page is dynamically updated to reflect the error processing and correction capabilities. The database will not be updated until the user has correctly filled in all required fields.
Screen shot from MySQL of the contacts relation after the inclusion of several users. Note that the values in the table are the same as those returned to the PHP document in the previous slide.
<?php
extract ( $_POST );
$iserror = false;
// array of magazine titles
$maglist = array( "Velo-News",
  "Cycling Weekly",
  "Pro Cycling",
  "Cycle Sport",
  "RadSport",
  "Mirror du Cyclisme" );
// array of possible operating systems
$systemlist = array( "Windows XP",
  "Windows 2000",
  "Windows 98",
  "Linux",
  "Other" );
// array of name and alt values for the text input fields
$inputlist = array( "fname" => "First Name",
    "lname" => "Last Name",
    "email" => "Email",
    "phone" => "Phone" );

if ( isset ( $submit ) ) {
    if ( $fname == "" ) {
        $formerrors[ "fnameerror" ] = true;
        $iserror = true;
    }
}

if ( $lname == "" ) {
    $formerrors[ "lnameerror" ] = true;
    $iserror = true;
}

if ( $email == "" ) {
    $formerrors[ "emailerror" ] = true;
    $iserror = true;
}

if ( !ereg( "\^\([0-9]{3}\\)[0-9]{3}-[0-9]{4}$", $phone ) ) {
    $formerrors[ "phoneerror" ] = true;
    $iserror = true;
}

if ( !$iserror ) {
    // build INSERT query
    $query = "INSERT INTO contacts " .
    "(ID, LastName, FirstName, Email, Phone, Magazine, OS ) " .
    "VALUES (null, '$lname', '$fname', '$email', " . "'" . quotemeta( $phone ) . ",'" . $mag . ",'$os')";
}
// Connect to MySQL
if ( !( $database = mysql_connect( "localhost", "root", "root" ) ) )
    die( "Could not connect to database" );

// open MailingList database
if ( !mysql_select_db( "MailingList", $database ) )
    die( "Could not open MailingList database" );

// execute query in MailingList database
if ( !( $result = mysql_query( $query, $database ) ) ) {
    print( "Could not execute query! <br />" );
    die( mysql_error() );
}
print( "<p>Hi
    <span style = 'color: blue'> <strong>$fname</strong></span>. Thank you for completing the survey.<br />
    You have been added to the <span style = 'color: blue'> <strong>$mag</strong></span> mailing list.               </p>

    The following information has been saved in our database:<br />
<table border = '0' cellpadding = '0' cellspacing = '10'>
<tr>
    <td bgcolor = '#ffffaa'>Name </td>
    <td bgcolor = '#ffffbb'>Email</td>
    <td bgcolor = '#ffffcc'>Phone</td>
    <td bgcolor = '#ffffdd'>OS</td>
</tr>
<!-- print each form field's value -->
<td>$fname $lname</td>
<td>$email</td>
<td>$phone</td>
<td>$os</td>
</tr></table>
</div><br /><br /><br />
<div style = 'font-size : 10pt; text-align: center'>
<div style = 'font-size : 18pt'>
<a href = 'formDatabase2.php'>
Click here to view entire database.</a>
</div>
</div>

</div></body></html>" );
die();
}

print("<h1>This is a sample registration form.</h1>
Please fill in all fields and click Register." );
if ( $iserror ) {
print("<br /><span style = 'color : red'>
Fields with * need to be filled in properly.</span>"");
}
print("<br />
<form method = 'post' action = 'dynamicForm2.php'>
<img src = 'images/user.gif' alt = 'User' />
Please fill out the fields below.<br />
</form>

Invoke PHP script to see contents of entire database if user clicks this link. Code begins on page 14.

The form created is self-submitting (i.e., it posts to itself). This is done by setting the action to dynamicForm2.php
<!-- create four text boxes for user input -->

foreach ($inputlist as $inputname => $inputalt)
{
    $inputtext = $inputvalues[$inputname];

    print("<img src='images/$inputname.gif' alt='$inputalt'/><input type='text' name='$inputname' value='$inputtext'/>");
    if ($formerrors[( $inputname )."error"] == true)
    {
        print("<span style='color: red'>*</span>");
    }
}
print("<br/>");
print("<span style='font-size: 10pt'>");
if ($formerrors["phoneerror"])
    print("; color: red");
print("">Must be in the form (555)555-5555<br/></span>
    <img src='images/downloads.gif' alt='Publications'/><br/>
    <span style='color: blue'>Which magazine would you like information about?</span><br/>
    <!-- create drop-down list containing magazine names -->
    <select name='mag'>
    foreach ($maglist as $currmag)
    {
        print("<option>");
        if ( ($currmag == $mag) )
        {
            print(" selected = 'true'");
        }
        print(">$currmag</option>");
    }
}
print( "</select><br /><br />
<img src = 'images/os.gif' alt = 'Operating System' />
<br /><span style = 'color: blue'>
Which operating system are you currently using?
</span></br>

<!-- create five radio buttons -->"
);

$counter = 0;

foreach ( $systemlist as $currsystem ) {
    print( "<input type = 'radio' name = 'os'
            value = '$currsystem'" );
    if ( $currsystem == $os ) print( "checked = 'checked'" );
    if ( iserror && $counter == 0 ) print( "checked = 'checked'" );

    print( "/>$currsystem" );

    if ( $counter == 2 ) print( "<br />");
    $counter++;
}

print( "<!-- create a submit button -->
<br />
<input type = 'submit' name = 'submit' value = 'Register' />
</form></body></html>" );

?>
<?php
extract( $_POST );
// build SELECT query
$query = "SELECT * FROM contacts";
// Connect to MySQL
if ( !( $database = mysqli_connect( "localhost", "root", "root", MailingList ) ) )
  die( "Could not connect to database" );
// query MailingList database
if ( !( $result = mysqli_query( $database, $query ) ) ) {
  print( "Could not execute query! <br />" );
  die( mysqli_error() );
}
<h3 style = "color: blue">
Mailing List Contacts</h3>
<table border="1" cellpadding="3" cellspacing="2" style="background-color: #ADD8E6">
<tr>
<td>ID</td>
<td>Last Name</td>
<td>First Name</td>
<td>E-mail Address</td>
<td>Phone Number</td>
<td>Magazine</td>
<td>Operating System</td>
</tr>
<?php
// fetch each record in result set
for ( $counter = 0; $row = mysqli_fetch_row( $result ); $counter++ ){
// build table to display results
print( "<tr>" );
foreach ( $row as $key => $value )
    print( "<td>$value</td>" );
print( "</tr>" );
}
mysqli_close( $database );
?>
</table>
</body>
</html>
Schema of the MailingList database table contacts required for the PHP database example to work. Script is available on the code page and shown on the next page.
# SQL commands to create and populate the MySQL database for
# CNT 4714 - Fall 2009

# delete the database if it already exists
drop database if exists mailinglist;

# create a new database named mailinglist
create database mailinglist;

# switch to the new database
use mailinglist;

# create the schemas for the four relations in this database
create table contacts (ID integer unsigned zerofill auto_increment not null,
LastName varchar(30), FirstName varchar(30), Email varchar(30), Phone varchar(14),
Magazine varchar(60), OS varchar(30),
primary key (ID)