Integration

How do I get my FRONT END

(for example, an HTML web page displayed by a browser)

To talk to my BACK END ?

(for example, a MySQL database)

Suppose my application must

- display a form and collect user data entry
- find and display data from the database
- Find and update data in the database

Integration

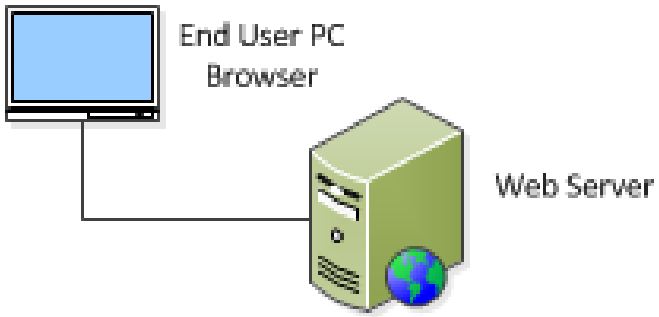
Consider the software stack we are using here.

Integration

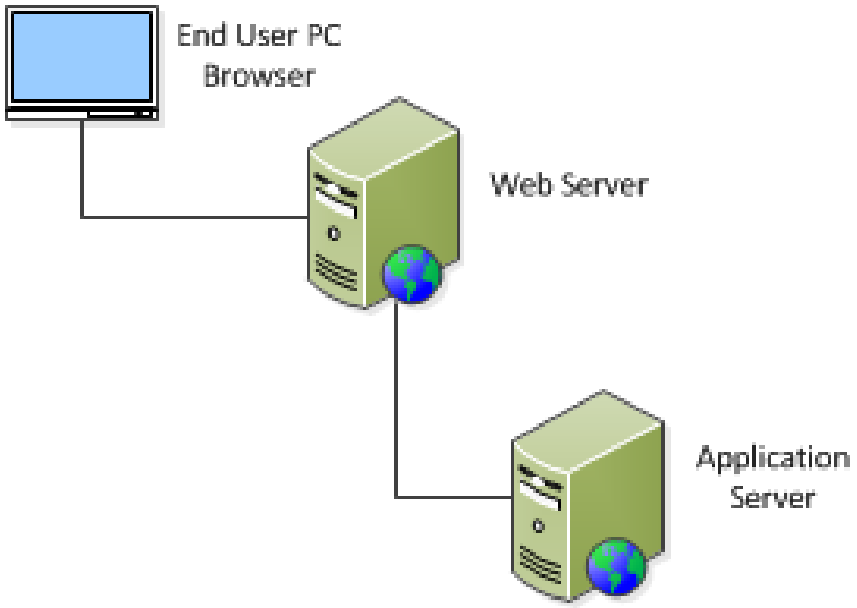
Three Components:

- **HTML –** Takes a marked up file and renders it in the browser. Runs on the PC's browser.
- **PHP –** A server-side scripting language. Runs on a web server.
- **SQL –** Communicates with the database server. The DBMS runs on a database server.

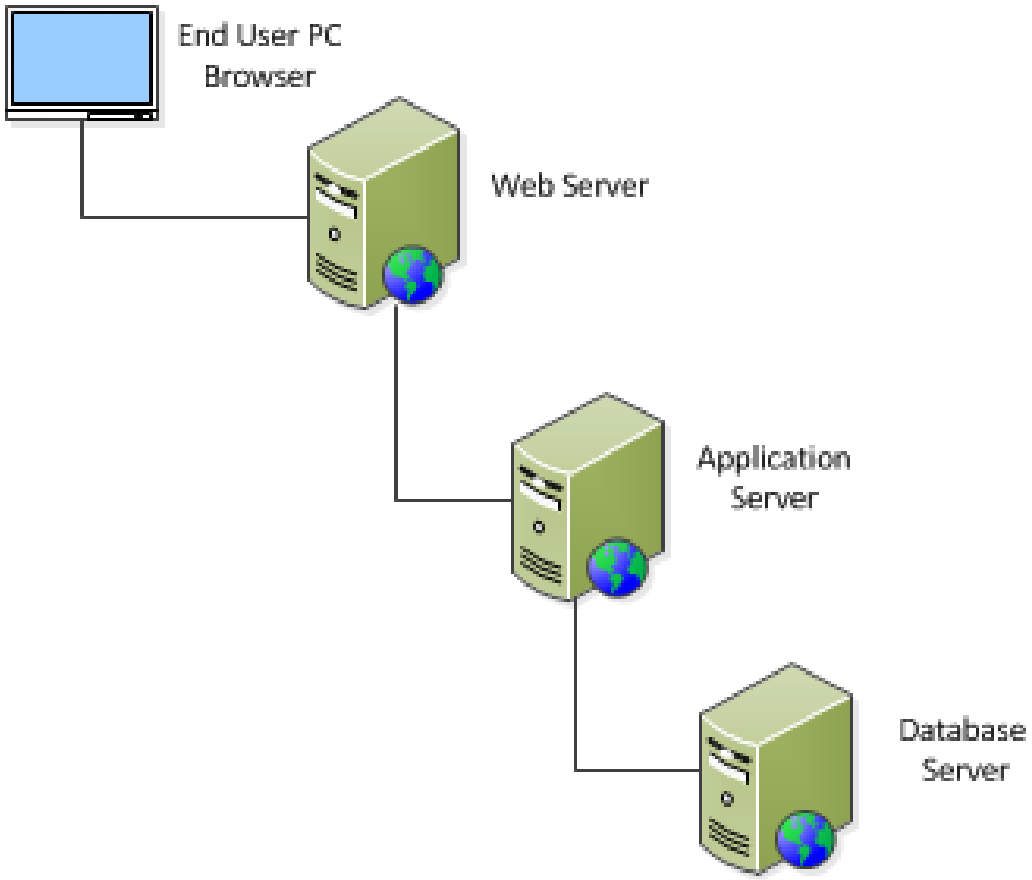
Application Architecture



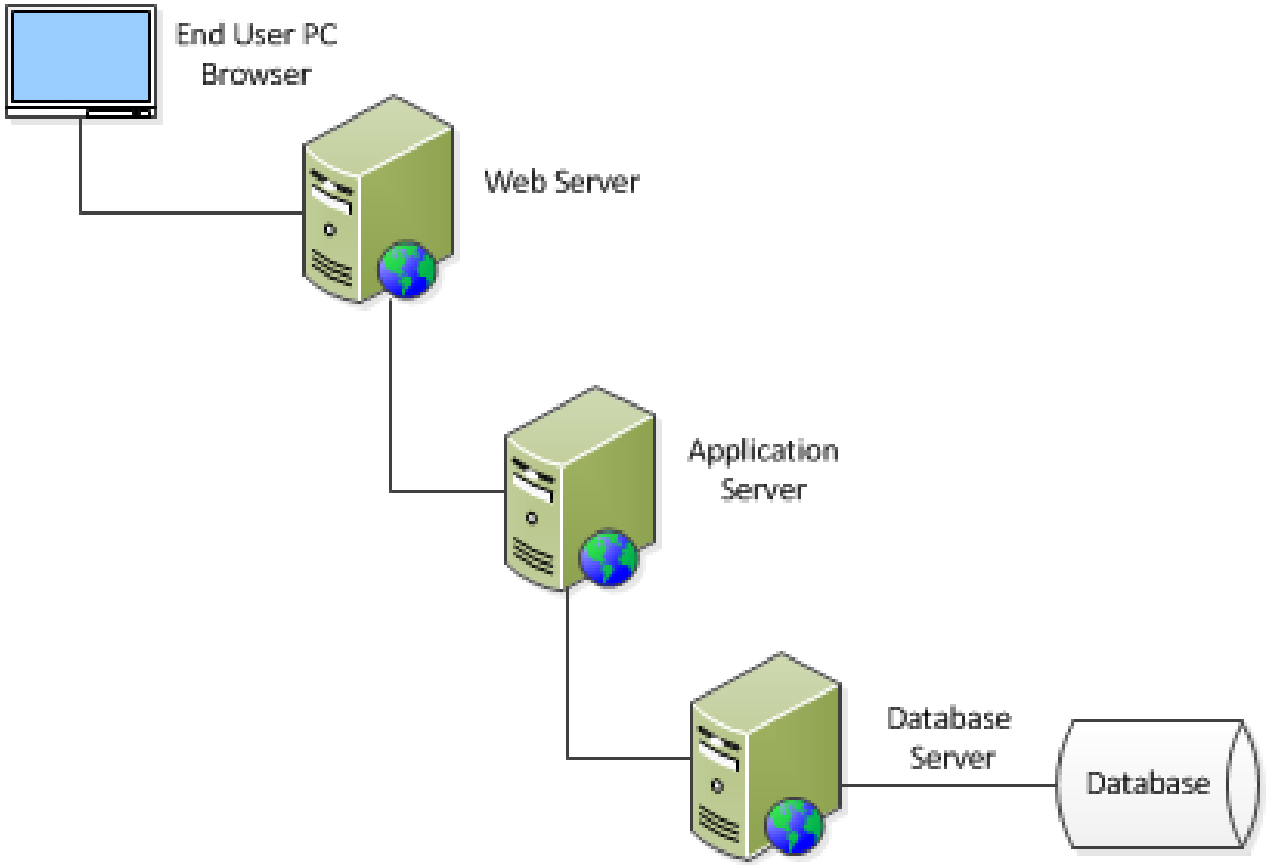
Application Architecture



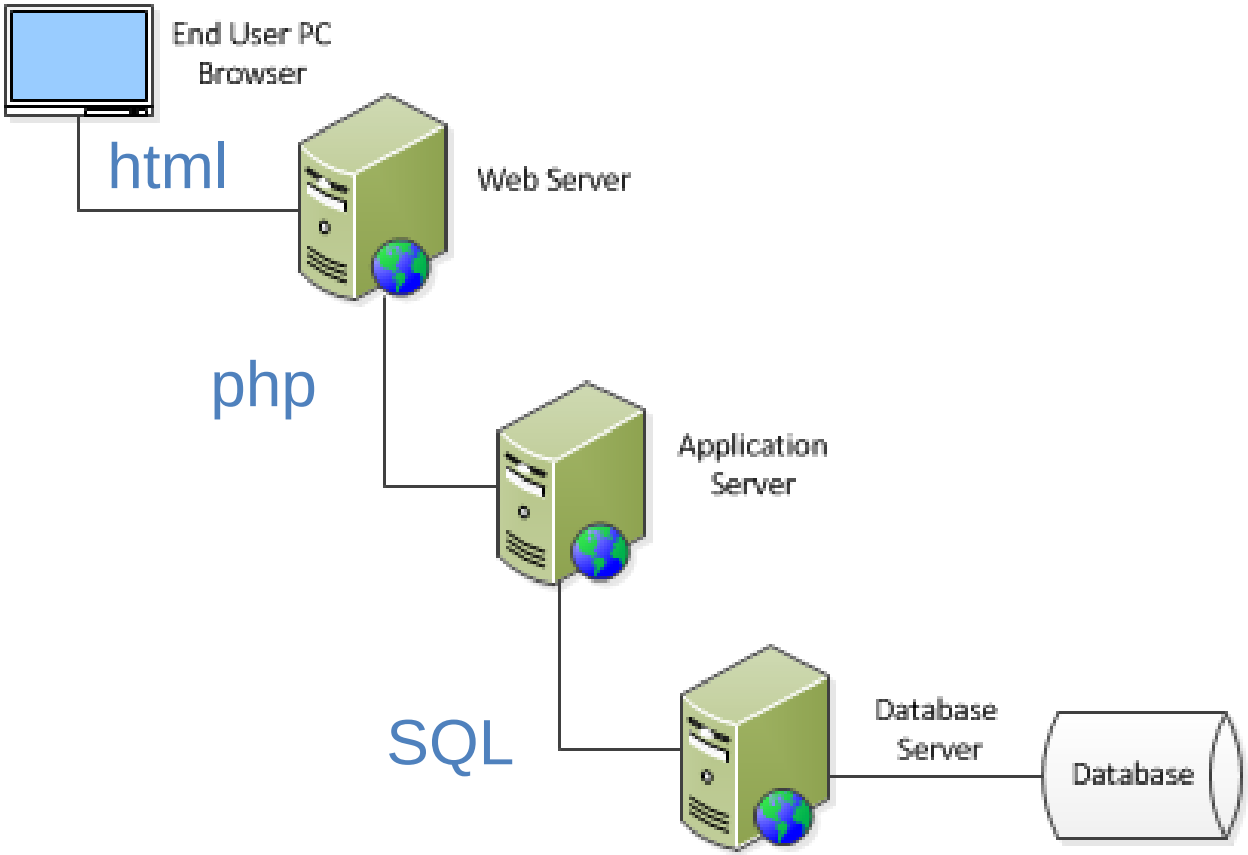
Application Architecture



Application Architecture



Application Architecture



php Basics

- php pages MUST be invoked via the URL rather than simply opening the file in your browser. Why?
  - They are ***SERVER SIDE SCRIPTS***
  - They are EXECUTED by the ***web server***
  - Not just PARSED by the ***browser***

- Your web server has a default location/path where it expects to find executable php scripts.
- Execute the php files (via the Apache web server) by entering a URL into your browser: `localhost/filename.php`

How do we simulate this multi-tiered architecture in a lab? ➔ XAMPP.

- XAMPP is an open source PHP development architecture stack that provides a MySQL (MariaDB) database engine and an Apache Web server that run on your PC.
- Download XAMPP from https://www.apachefriends.org/index.html
- Runs on Windows or OS-X on mac
- Alternatives to XAMPP
  - WAMP – windows only
  - LAMP – for a linux PC

php Basics

- Demonstrate this

C:/Users/aparadise/Desktop/HTML_demo.html

C:/Users/aparadise/Desktop/demo1.php

C:/XAMPP/htdocs/demo1.php

Localhost/demo1.php

SO:

- to render a php file through the browser, you must execute the php code on the “localhost” web server.

- You must keep your files in the folder c:/XAMPP/htdocs for the apache web server to execute them

NetBeans

- To make things a bit easier: Try NetBeans
  - An integrated development environment (“IDE”) for PHP
  - Download from https://netbeans.org/downloads/

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						•
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						•
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•
	Download	Download	Download x86 Download x64	Download x86 Download x64	Download x86 Download x64	Download
	Free, 95 MB	Free, 197 MB	Free, 108 - 112 MB	Free, 108 - 112 MB	Free, 107 - 110 MB	Free, 221 MB

php Basics

- php code is typically embedded within an HTML page
- php code is embedded within special tags:

```
<?php
    php code…
    php code…
?>
```

php Basics

- You may also see:

```
<?
    php code…
    php code…
?>
```

- Or:

```
<script language="php">
    php code…
    php code…
</script>
```

- To send text to the browser

```
<?php
    echo "Hello World";
    echo 'Hello World';
    print 'Hello World';
    print "Hello World";
?>
```

Note: all statements end with " ; “

At this point, the quotes don’t matter.

- To send a quote to the browser

```
<?php
   echo "Hello Wayne's World";
?>
```

- This won't work

```
<?php
   echo 'Hello Wayne's World';
?>
```

- Use " \ " (backslash) as *escape* character

```
<?php
   echo "Hello Wayne\'s World";
?>
```

- To send text + html tags to the browser

```
<?php
    echo "<b>Hello</b><em>World</em>";
?>
```

- Multiple lines (demo)

```
<?php
    echo "this text is spread across
          multiple lines in php";
?>
```

- Yields multiple lines in HTML, but not when rendered

- Commenting your code (for the sake of human readers)
  - HTML comments `<!-- xxxxxx -->`
  - php comments
    - Use # or // for single line comments
    - Use /* through */ for multi line comments

- Comments

```
<?
# Created August 27, 2017
# Created by Alan Paradise
# This script does nothing much.
echo "<p>This is a line of text.
      <br>This is another line of text.</p>";
/*echo 'This line will not be executed.';*/
echo "<p>Now I'm done.</p>"; // End of PHP code
?>
```

- Note on debugging
  - **HTML with php can be very unforgiving and difficult to debug**
  - **Tips:**
    - **Comment out stuff to see what's not working**
    - **Use notepad++ or NetBeans**
      - **It matches tags with end tags**
      - **Color codes your HTML & PHP**

- Variables
  - Must be named
  - Name must start with a $
  - Names may contain letters, numbers, and "_"
  - First character after $ must be a letter or "_"
  - Names ARE case sensitive
    - `$name` does not equal `$Name`
  - Assigned values with "="

```
$name = "Desmond";
```

- Php comes with some pre-defined variables
  - $GLOBALS — References all variables available in global scope
  - $_SERVER — Server and execution environment information
  - $_GET — HTTP GET variables
  - $_POST — HTTP POST variables
  - $_FILES — HTTP File Upload variables
  - $_REQUEST — HTTP Request variables
  - $_SESSION — Session variables
  - $_ENV — Environment variables
  - $_COOKIE — HTTP Cookies
  - $php_errormsg — The previous error message
  - $HTTP_RAW_POST_DATA — Raw POST data
  - $http_response_header — HTTP response headers
  - $argc — The number of arguments passed to script
  - $argv — Array of arguments passed to script

php Basics

- Variables
  - Get in the habit of using a CONSISTENT naming scheme
    - Lower case, underscores, mixed caps
  - No need to initialize
  - No need to declare the variable type
  - Easy to switch types

- String Variables
  - Variable is assigned any value in quotes (single or double)
  - Any new value assigned overwrites the old value
  - No strict limit on length
  - Concatenated with " . "

```
$first_name = "Elon";
$last_name = "Musk";
$full_name = $first_name." ".$last_name;
echo "$full_name";
```

  - Execute concat_demo.php

- Numeric Variables
  - Variable is assigned any numeric value without quotes
  - Any new value assigned overwrites the old value
  - Don’t use commas for thousands
  - Assumed positive
  - Arithmetic operators

    `+  -  *  /  ++ (increment) -- (decrement)`
  - Arithmetic Functions
    - `round(xxx, yyy)`
    - `number_format(xxx, yyy)`

      Where xxx is the number and yyy is number of decimal place

- A little math demo
- http://localhost/math_demo.php

Using Quotes

- Double quoted strings resolve values
- Single quoted strings do not resolve values (http://localhost/quote_demo.php)
- `$var = 'test';`

  `echo "var equals $var";` //yields `var equals test`

  `echo 'var equals $var';` //yields `var equals $var`

  `echo "\$var equals $var";` //yields `$var equals test`

  `echo '\$var equals $var';` //yields `\$var equals $var`

- Use double quotes to echo (or print) the value of a variable
- Use single quotes to echo (or print) HTML

Php basics

- Handy tool = var_dump
- Shows you the value of a variable

Using HTML Forms

- HTML forms
- How are they used?
  - Use the browser's window as a data entry screen
  - Collect information from the user
  - Pass it to the web server via http
  - Invoke a server-side script
  - Passes ***form data*** as input to the script

Using HTML Forms

- `<form>` tag has several attributes – two are required
- ACTION
  - `<form action="http://URL">` name of a program on the web server
    - URL specifies the location of the executable file on the web server
  - `<form action="mailto:mailrecipient">` sends an email
- METHOD
  - `<form method="POST" >` or `<form method="GET">`
    - **POST** when you have large amount of data being sent, encryption available, a two-step process
    - **GET** for small amounts, no security – all in one step

    `<form enctype=`
      - » multipart/form-data (default)
      - » text/plain (used only for mailto)

Using HTML Forms

- the <input> tag
  - Specifies an input field on a form
- type attribute – tells us what kind of control
  - **text**
  - **radio**
  - **checkbox**
  - **submit button**
  - **reset button**

Using HTML Forms

- <form> examples
- Text Box

```
<input type="text" name="Name" size="20" maxlength="30">
```

- Radio Button(s)

```
<input type="radio" name="Gender" value="M" /> Male
<input type="radio" name="Gender" value="F" /> Female
```

- Check Box(es)

```
<input type="checkbox" name= "size" value="S"
   checked="checked" />Small
<input type="checkbox" name="size" value="M" />Medium
<input type="checkbox" name="size" value="L" />Large
<input type="checkbox" name="size" value="XL" />X-Large
```

- List Box

```
<select name="Grade" size="3">
        <option>A
        <option>B
        <option>C
        <option>D
        <option>F
</select>
```

- List Box via <select> tag
  - **Size** attribute
    - When absent: you get a "drop down list", first item selected by default
    - When present: indicates the number of items in the list
  - **Selected** attribute: specifies selected item
  - **Multiple** attribute: when "yes", can click > 1

Using Forms

```
<input type="submit" />
<input type="reset" />

<textarea name="comments" cols="40" rows="8">
```

Using Forms

- Sending FORM data to a PHP program Requires TWO files
  - An **HTML** page with a FORM and a SUBMIT button
  - A **PHP** program invoked when the FORM is submitted, specified in ACTION attribute
  - Must be specified via **URL** in the ACTION attribute

(In the HTML Form page)

```
<form method="post"
   action="http://localhost/handleform.php"
  enctype="multipart/form-data"
  onsubmit="window.alert('Form is being posted')">
```

Using Forms

- Values passed from the HTML page to the PHP program appear in a SYSTEM VARIABLE ARRAY called "$_REQUEST"
- Entries in the "$_REQUEST" array are referenced by their HTML "name= " attribute
- Note that the CHECKBOX input type comes through as an array (multiple values)

Using Forms

- Use the var_dump() method to see all variable information
- We execute “formdemo.html” to display the form
- Hitting the “submit” button invokes “formhandler.php”

Code Examples: Forms

```
<html>
<head>
    <title>Form Demo</title>
</head>
<body>

<form enctype="multipart/form-data"
      action="http://localhost/handleform.php">

    <h2>Name:</h2>

    <input type="text" name="Name" size="20" maxlength="30" />
    <br><hr>

    <h2>Please Specify Gender:</h2>
    <input type="radio" name="Gender" value="M" /> Male
    <input type="radio" name="Gender" value="F" /> Female
    <br><hr>

    <h2>Please Select One or More Sizes:</h2>
    <input type="checkbox" name="Size" value="S" checked="checked" />Small
    <input type="checkbox" name="Size" value="M" />Medium
    <input type="checkbox" name="Size" value="L" />Large
    <input type="checkbox" name="Size" value="XL" />X-Large
    <br><hr>

    <h2>Please Select Your Grade</h2>
    <select name="Grade" size="5" multiple="yes" />
        <option />A
        <option />B
        <option />C
        <option />D

        <option selected="yes" />F
    </select>
    <br><hr><br>

    Comments:<br>
    <textarea name="Comments" cols="40" rows="4"></textarea>
    <br><hr><br>

    <input type="submit" value="Send Form" />
    <input type="reset" />

</form>

</body>
</html>
```

Code Examples: Forms

```
<html>
<head>
    <title>Form Handler php</title>
</head>
<body>

<?php

# this program handles form data

echo 'the variable $_REQUEST =';
var_dump($_REQUEST);

$Name = $_REQUEST['Name'];
$Gender = $_REQUEST['Gender'];
$Grade = $_REQUEST['Grade'];
$Size = $_REQUEST['Size'];
$Comments = $_REQUEST['Comments'];

echo "<br> <br>";

echo "The following values were passed from the HTML form: <br><br>";
echo "\$Name = $Name <br> <br>";
echo "\$Gender = $Gender <br> <br>";
echo "\$Grade = $Grade <br> <br>";
echo "\$Size = $Size <br> <br>";
echo "\$Comments = $Comments <br> <br>";

?>

</body>
</html>
```

- Use the `isset()` function
  - To determine whether a variable has been assigned a value
- Example:

```
<?php
    If (isset($_REQUEST['Gender'])) {
          $Gender = $_REQUEST['Gender'];
    } else {
          $Gender = NULL;
    }
</select>
```

- Validating FORM data
  - You don't want to let any bad data get into your database
  - You must assume that if users are allowed to enter bad data, they will
  - **`isset()`** will test FALSE for an empty string
  - **`empty()`** will test TRUE for an empty string
  1. Did they enter ANYTHING?
  2. Is it VALID for the variable type?
  3. Is it a valid VALUE for the field?

- Example

```
<?php
// Validate the name:
if (!empty($_REQUEST['name'])) {
    $name = $_REQUEST['name'];
} else {
    $name = NULL;
    echo '<p>You forgot to enter your name!</p>';
}
?>
```

- Tips on Validating FORM fields
  - `is_numeric()` tests if a field contains valid numbers
  - It is a good idea (courtesy) to inform your users whether or not form fields are REQUIRED or OPTIONAL

- Handling ARRAYS
  - Two types:
    - Index Keys, like $_REQUEST[1]
    - String Keys, like $_REQUEST['Name']
  - Indexes begin at 0 (default)
  - Wrap array references with string keys in { } when using `echo()` or `print()` to avoid parse errors

```
<?php
        echo "{$_REQUEST['Name']}";
?>
```

- Creating ARRAYS
  - Declare and initialize with `array()` function

```
<?php
   $cars = array();
   $cars[0] = 'Acura';
   $states = array('AL','AK','AR',…,…,'WY');
   $days = array('M'=>'Monday','T'=>'Tuesday');
   $months = array('Jan','Feb','Mar',… …);
?>
```

  - Load them one entry at a time or all at once

- Accessing ARRAYS using `foreach()` loop

```
<?php
   foreach ($array as $value) {
       // do something with $value
   }
?>
```

- This command loops through the array `$array` and with each iteration sets `$value` equal to the value of each successive entry

- Getting KEYS and VALUES

```
<?php
   foreach ($array as $key => $value) {
       // do something with $key and $value
   }
?>
```

The symbol => maps the key to the value

- Setting initial KEY

```
<?php
   $states = array( 1 => 'AL','AK',…,…,'WY');
?>
```

- To fill an array with numeric values use the `range()` function

```
<?php
   $months = range(1, 12);
?>
```

(Demo = arraydemo1 & 2)

- While Loop

```
<?php
   while (condition) {
       // do something
   }
?>
```

- Checks the condition FIRST
- If the condition is TRUE, it executes "something"

- For Loop

```
<?php
   for ($i=1, $i <=10, $i++) {
       // do something
   }
?>
```

- Sets $i to 1, checks the condition ($i <= 10)
- Then if the condition is true, it will do something
- Then it will increment the counter, check the condition, etc.
- "Do While" versus "Do Until"
- Beware infinite loops

Combining PHP and MySQL

- Three steps
  - Connect to the database
  - Run the query
  - Parse query output

Combining PHP and MySQL

- Connecting to the database
  - "mysqli" is a class that represents the connection between a php program and a database
  - We use the `mysqli_connect()` function to connect
  - Syntax:

```
$dbc = mysqli_connect(
        hostname,username,pw,db_name)
```

  - `$dbc` is used as a variable in subsequent MySQL functions

Combining PHP and MySQL

- Define the four connection parameters as CONSTANTS – prohibits them from being changed (not very secure)
- Use the "or DIE" option on the call to the function

Combining PHP and MySQL

- Running a Simple Query
  - Uses the `mysqli_query()` function
  - Syntax:

    ```
    $r = mysqli_query($dbc,$q)
    ```

    where:

    `$dbc =` database connection placeholder

    `$q =` text string containing your SQL query

    `$r =` result of query, can be used as a flag indicating success/failure

Combining PHP and MySQL

- Handling Query Output
  - `msyqli_assoc` – associates the column name to the array index
  - `mysqli_num` – associates a number to the column array index
    - These are “constants”
  - Uses the `mysqli_fetch_array()` function
  - Returns ONE answer set ROW at a time
  - Typically embedded in a "while" loop
  - Syntax:

    ```
    $row = mysqli_fetch_array($r)
    ```

    where:

    `$row =` an array holding the contents of one row of the answer set

    `$r =` the placeholder variable for the query answer

Code Samples - SELECT

```php
<?php # simple_select.php

include ('header.html');

// Set the database access information as constants:
DEFINE ('DB_USER', 'root');
DEFINE ('DB_PASSWORD', '');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'northwinds');

// Make the connection:
$dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME)
       OR die ('Could not connect to MySQL: ' . mysqli_connect_error() );  // Connect to the db.

$q = "SELECT FirstName, LastName, Country from nwemployees;";   // Define the query.

$r =  mysqli_query($dbc,$q);  // Run the query.

// Count the number of returned rows:
$num = mysqli_num_rows($r);

if ($num > 0) { // If it ran OK, display the records.

    // Print how many rows were returned:
echo "<p>This query returned $num rows.</p>\n"; }

    // Fetch and print all the records:
    while ($row = mysqli_fetch_array($r, MYSQLI_NUM)) {
        echo $row[0]." ".$row[1]." ".$row[2]."<br>";

    }
mysqli_close($dbc); // Close the database connection.

include ('footer.html');
?>
```

Code Samples - UPDATE

```php
<?php # update.php

include ('header.html');

// Set the database access information as constants:
DEFINE ('DB_USER', 'root');
DEFINE ('DB_PASSWORD', '');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'northwinds');

// Make the connection:
$dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME)
       OR die ('Could not connect to MySQL: ' . mysqli_connect_error() );  // Connect to the db.

//$q = "SELECT FirstName, LastName, Country from nwemployees;"; // Define the query.

$q = "UPDATE nwemployees set FirstName = 'Alan' where EmployeeID = 3;"; // Define update query.

$r =  mysqli_query($dbc,$q);  // Run the UPDATE query.

$q = "SELECT FirstName, LastName, Country from nwemployees;";   // Define select query to show updates.

$r =  mysqli_query($dbc,$q);  // Run the SELECT query.

        // Fetch and print all the records:
    while ($row = mysqli_fetch_array($r, MYSQLI_NUM)) {

        echo $row[0]." ".$row[1]." ".$row[2]."<br>";

    }

mysqli_close($dbc); // Close the database connection.

include ('footer.html');
?>
```

Code Samples – Insert – HTML Form

```
<html>
<head>
    <title>SQL Insert Demo</title>
</head>
<body>

<h2>Insert a new row into the "nwcustomers" table</h2>

    <form enctype="multipart/form-data"
     action="http://localhost/SQLInsertHandler.php">

    <p>EmployeeID:  <input type="text" name="EmployeeID" size="10" maxlength="11" /></p>
    <p>LastName:    <input type="text" name="LastName" size="10" maxlength="20" /></p>
    <p>FirstName:   <input type="text" name="FirstName" size="10" maxlength="30" /></p>
    <p>Title:   <input type="text" name="Title" size="10" maxlength="10" /></p>
    <p>BirthDate(YYYY-MM-DD):   <input type="text" name="BirthDate" size="10" maxlength="10" /></p>
    <p>HireDate(YYYY-MM-DD):    <input type="text" name="HireDate" size="10" maxlength="10" /></p>
    <p>Address:     <input type="text" name="Address" size="15" maxlength="60" /></p>
    <p>City:    <input type="text" name="City" size="15" maxlength="15" /></p>
    <p>Region:  <input type="text" name="Region" size="15" maxlength="15" /></p>
    <p>PostalCode:  <input type="text" name="PostalCode" size="15" maxlength="10" /></p>
    <p>Country:     <input type="text" name="Country" size="15" maxlength="10" /></p>
    <br>
    <input type="submit" value="Send Form" />  
    <input type="reset" />

</form>

</body>
</html>
```

http://localhost/SQLInsertDemo.html

Code Samples – Insert – Form Handler

```
<html>
<head>
    <title>SQL Insert Handler php</title>
</head>
<body>

<?php

# this program receives form data, formats passed data into a SQL Insert, and updates the database

#echo 'the variable $_REQUEST =';
#var_dump($_REQUEST);

$EmployeeID = $_REQUEST['EmployeeID'];
$LastName = $_REQUEST['LastName'];
$FirstName = $_REQUEST['FirstName'];
$Title = $_REQUEST['Title'];
$BirthDate = $_REQUEST['BirthDate'];
$HireDate = $_REQUEST['HireDate'];
$Address = $_REQUEST['Address'];
$City = $_REQUEST['City'];
$Region = $_REQUEST['Region'];
$Country = $_REQUEST['Country'];
$PostalCode = $_REQUEST['PostalCode'];

// Set the database access information as constants:
DEFINE ('DB_USER', 'root');
DEFINE ('DB_PASSWORD', '');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'northwinds');

// Make the connection:
$dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME)
    OR die ('Could not connect to MySQL: ' . mysqli_connect_error() );  // Connect to the db.
```

Code Samples – Insert – Form Handler

```
$iq = "INSERT into nwemployees (EmployeeID,LastName,FirstName,Title,Birthdate,HireDate,
                                Address,City,Region,PostalCode,Country)
       values ('$EmployeeID','$LastName','$FirstName','$Title','$BirthDate','$HireDate',
               '$Address','$City','$Region','$PostalCode','$Country')";  // Define INSERT query.

$insert_row =  mysqli_query($dbc,$iq);  // Run the INSERT query.

if($insert_row){
print 'Row Inserted !' .'<br />';
}else{
    echo "<br>ERROR: Could not execute sql. Error code = " . mysqli_error($dbc)."<br>";
    die('Error on insert');
}

$sq = "SELECT EmployeeID, FirstName, LastName, Title, BirthDate, HireDate, Address,
           City, Region, PostalCode, Country from nwemployees;"; // Define select query to show inserts.

$r =  mysqli_query($dbc,$sq);  // Run the SELECT query.

       // Fetch and print all the records:
    while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {

        echo $row["EmployeeID"]." ".$row["FirstName"]." ".$row["LastName"]." ".$row["Title"]." ".
        $row["BirthDate"]." ".$row["HireDate"]." ".$row["Address"]." ".$row["City"]." ".
        $row["Region"]." ".$row["Country"]."<br>";

    }

mysqli_close($dbc);  // Close the database connection.

?>

</body>
</html>
```