Web Database Application

CIT414 (2 Units)

Dr. Bilkisu Muhammad-Bello

Objectives

At the end of this unit, you should be able to:

- Explain what PHP can do and how it handles HTML forms
- Describe the PHP basic syntax and different ways of escaping from HTML.
- Describe the different comment styles supported by PHP.
- Identify the eight primitive types supported by PHP.
- Describe some of the predefined variables
- Elucidate variable scoping in PHP
- Explain how variables from external sources are used.

What is PHP?

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source general-purpose scripting language
- PHP is especially suited for web development and can be embedded into HTML.
- PHP scripts are executed on the server
- PHP costs nothing, it is free to download and use

Example #1 An introductory example

- `<!DOCTYPE HTML PUBLIC`

```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

- **Note: PHP statements end with a semicolon** (**;**).

Client-side Scripting vs Server-side Scripting

- What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client.
- The client would receive the results of running that script, but would not know what the underlying code was.
- You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

What can PHP do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is a widely-used, and efficient alternative to competitors such as Microsoft's ASP.
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Example #2 Get system information from PHP

- `<?php phpinfo(); ?>`

- It makes a call to the phpinfo() function
- It displays a lot of useful information about your system and setup such as
  - available predefined variables
  - loaded PHP modules
  - configuration settings

Take some time and review this important information

Something Useful: Browser details

- To check what sort of browser the visitor is using, the user agent string which the browser sends as part of the HTTP request is checked.
- This information is stored in a variable. Variables always start with a dollar-sign $ in PHP.
- The variable we are interested in right now is $_SERVER['HTTP_USER_AGENT'].
- *$_SERVER* is a special reserved PHP variable that contains all web server information. It is known as a superglobal.
- Example 3 shows how to display this variable.

Example #3 Printing a variable

```
<?php
echo $_SERVER['HTTP_USER_AGENT'];
?>
```

- There are many types of variables available in PHP.
- In the above example we printed an Array element. Arrays can be very useful.
- *$_SERVER* is just one variable that PHP automatically makes available to you. It is known as a SUPERGLOBAL
- Superglobals are built-in variables that are always available in all scopes
- You can get a complete list of them by looking at the output of the phpinfo() function used in the previous example.

Basic Syntax

- When PHP parses a file, it looks for opening and closing tags, which tell PHP to start and stop interpreting the code between them.
- Parsing in this manner allows PHP to be embedded in all sorts of different documents, as everything outside of a pair of opening and closing tags is ignored by the PHP parser.
- There are ***four*** different pairs of opening and closing tags which can be used in PHP.
  - <?php ?> and
  - <script language="php"> </script>, are always available.
  - The other two are short tags and ASP style tags, they are less portable, and generally not recommended.

PHP Opening and Closing Tags

1.
   ```
   <?php echo 'if you want to serve XHTML or XML
   documents,  do it like this'; ?>
   ```

2.
   ```
   <script language="php">
           echo 'some editors (like FrontPage) do
   n\'t like processing instructions';
   </script>
   ```

3.
   ```
   <? echo 'this is the simplest, an SGML process
   ing instruction'; ?>
   <?= expression ?> This is a shortcut for "<? e
   cho expression ?>"
   ```

4.
   ```
   <% echo 'You may optionally use ASP-
   style tags'; %>
   <%= $variable; # This is a shortcut for "<% ec
   ho . . ." %>
   ```

Some Basic Stuff about PHP

- **Comments in PHP**
  - // This is a single-line comment
  - # This is also a single-line comment
  - /*
    This is a multiple-lines comment block
    that spans over multiple
    lines
    */
- **PHP Case Sensitivity**
  - In PHP, **NO keywords** (e.g. if, else, while, echo, etc.), **classes, functions, and user-defined functions** are case-sensitive.
  - However; all **variable names** are case-sensitive.

Types

- PHP supports eight primitive types.
  - Four scalar types:
    - boolean
    - integer
    - float (floating-point number, aka double)
    - string
  - Two compound types:
    - array
    - object
  - And finally two special types:
    - resource
    - NULL

Rules for PHP Variables

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

PHP Variables

- For examples:

  -
    ```
    <?php
    $var = 'Bob';
    $Var = 'Joe';
    echo "$var, $Var";      // outputs "Bob, Joe"
    $4site = 'not yet';    // invalid; starts with a number
    $_4site = 'not yet';  // valid; starts with an underscore ?>
    ```

  -
    ```
    <?php
    $txt = "Hello world!";
    $x = 5;
    $y = 10.5;
    ?>
    ```
- By default, variables are always assigned by value i.e. when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable's value to another, changing one of those variables will have no effect on the other.

Assign by reference

- PHP also offers another way to assign values to variables: assign by reference.
- This means that the new variable simply references (or "points to") the original variable. Changes to the new variable affect the original, and vice versa.
- To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable). For example:

  -
    ```
    <?php
    $foo = 'Bob';                // Assign the value 'Bob' to $foo
    $bar = &$foo;                // Reference $foo via $bar.
    $bar = "My name is $bar";    // Alter $bar...
    echo $bar;
    echo $foo;                   // $foo is altered too.
    ?>
    ```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

Global Scope

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:
- Example

  -
    ```
    <?php
    $x = 5; // global scope

    function myTest() {
        // using x inside this function will generate
    an error
        echo "<p>Variable x inside function is:
       $x</p>";
    }
    myTest();

    echo "<p>Variable x outside function is: $x</p>";
    ?>
    ```

Local Scope

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:
- Example:

  -
    ```
    <?php
    function myTest() {
        $x = 5; // local scope
        echo "<p>Variable x inside function is: $x</p>";
    }
    myTest();

    // using x outside the function will generate an error
    echo "<p>Variable x outside function is: $x</p>";
    ?>
    ```

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

PHP The global Keyword

- The *global* keyword is used to access a global variable from within a function.
- To do this, use the *global* keyword before the variables (inside the function):

```
<?php
$x = 5;
$y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

- PHP also stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

Using static variables

- Another important feature of variable scoping is the *static* variable.
- A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope. Consider the following example:

-
  ```
  <?php
  function test()
  {
      $a = 0;
      echo $a;
      $a++;
  }
  ?>
  ```
- This function is quite useless since every time it is called it sets *$a* to *0* and prints *0*. The *$a++* which increments the variable serves no purpose since as soon as the function exits the *$a* variable disappears.
- To make a useful counting function which will not lose track of the current count, the *$a* variable is declared static as in the next example.

Example use of static variables

-
  ```
  <?php
  function test()
  {
      static $a = 0;
      echo $a;
      $a++;
  }
  ?>
  ```
- Now, *$a* is initialized only in first call of function and every time the *test()* function is called it will print the value of *$a* and increment it.
- Static variables also provide one way to deal with recursive functions.
- A recursive function is one which calls itself. Care must be taken when writing a recursive function because it is possible to make it recurse indefinitely. You must make sure you have an adequate way of terminating the recursion.

Variable variables

- Sometimes it is convenient to be able to have variable variable names. That is, a variable name which can be set and used dynamically.
- A normal variable is set with a statement such as:

-
  ```
  <?php
       $a = 'hello';
  ?>
  ```
- A variable variable takes the value of a variable and treats that as the name of a variable. In the above example, *hello*, can be used as the name of a variable by using two dollar signs. i.e.

-
  ```
  <?php
       $$a = 'world';
  ?>
  ```
- At this point two variables have been defined and stored in the PHP symbol tree: *$a* with contents "hello" and *$hello* with contents "world".
- `<?php        echo "$a ${$a}";   ?>`
- `<?php        echo "$a $hello";   ?>`

Variables From External Sources

- **HTML Forms (GET and POST)**
  - When a form is submitted to a PHP script, the information from that form is automatically made available to the script. There are many ways to access this information, using form variables such as:
    - $_POST
    - $_GET
    - $_REQUEST
  - Using a GET form is similar except you'll use the appropriate GET predefined variable instead. GET also applies to the *QUERY_STRING* (the information after the '?' in a URL).
  - The $_REQUEST is a HTTP Request variable. It is an associative array that by default contains the contents of *$_GET*, *$_POST* and *$_COOKIE*.

Variables From External Sources cont.

- **IMAGE SUBMIT variable names**
  - When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:
  - `<input type="image" src="image.gif" name="sub" />`
  - When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables, `sub_x` and `sub_y`. These contain the coordinates of the user click within the image.
- **HTTP Cookies**

  PHP transparently supports HTTP cookies. Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the setcookie() function.

  Cookies are part of the HTTP header, so the SetCookie function must be called before any output is sent to the browser. Cookie data is then available in the appropriate cookie data arrays, such as `$_COOKIE, $HTTP_COOKIE_VARS` as well as in `$_REQUEST`.