# Ders # 8

## Web DB Programming: PHP

*From Elmasri/Navathe textbook Ch9,26*

*Sciore textbook, Ch 9-10*

# Web Programming w/ PHP

- Overview
- PHP (PHP Hypertext Preprocessor {recursive acronym})
- PHP Examples
- Basic features of PHP
  - PHP Arrays
  - PHP Functions
- PHP Database programming
  - Connect
  - Query
  - Retrieval Queries

# PHP:

- for programming dynamic features into Web
- Open source scripting language
- **Interpreter** engine in C;
  - free of charge
  - Can be used on nearly all computer types
- Particularly suited for manipulation of text pages
- Manipulates at the Web *server*
  - Conversely, JavaScript is downloaded and executed on the client
- **dynamic web pages**: part of info is extracted from databases
- Has libraries of functions for accessing databases

# Where is PHP

- DBMS: **Bottom-tier database server**
- PHP: **Middle-tier Web server**
- HTML: **Client tier**
- A simple PHP Example:
  - Suppose the file P1 is stored at www.myserver.com/example/greeting.php

(a)

```
    //Program Segment P1:
 0) <?php
 1) // Printing a welcome message if the user submitted their name
    // through the HTML form
 2) if ($_POST['user_name']) {
 3)   print("Welcome,  ") ;
 4)   print($_POST['user_name']);
 5) }
 6) else {
 7)   // Printing the form to enter the user name since no name has
      // been entered yet
 8)   print <<<_HTML_
 9)   <FORM method="post" action="$_SERVER['PHP_SELF']">
10)   Enter your name: <input type="text" name="user_name">
11)   <BR/>
12)   <INPUT type="submit" value="SUBMIT NAME">
13)   </FORM>
14)   _HTML_;
15) }
16) ?>
```

(b)


(c)


(d)


**Figure 14.1**
(a) PHP program segment for entering a greeting, (b) Initial form displayed by PHP program segment, (c) User enters name *John Smith*, (d) Form prints welcome message for *John Smith*.

# A Simple PHP Example (cont'd.)

- `<?php`
  - PHP start tag
- `?>`
  - PHP end tag
- Comments: `//` or `/* */`
- `$_POST`
  - **Auto-global** predefined PHP variable
  - Array that holds all the values entered through **form parameters**
- Arrays are dynamic
- **Long text strings:**
  - Between opening `<<<_HTML_` and closing `_HTML_;`
- PHP **variable names :** Start with `$` sign

# Basic features of PHP

- PHP variables, data types, and constructs
  - Variable names: characters, letters, numbers, and _.
    - No other special characters are permitted
    - Are case sensitive and Can't start with a number after $
  - Variables: no types (PHP is typeless lang)
    - Values assigned to variables determine their type
    - Assignments can change the type
  - Assignment op is =
- Main ways to express strings and text
  - **Single-quoted strings**
  - **Double-quoted strings**
  - **Here documents**
  - **Single and double quotes**

# Basic features of PHP

- Main ways to express strings
  - Single-quoted strings (lines 0, 1, 2): Literal strings that contain no PHP program variables. \' represents a quote in a string
  - Double-quoted strings (line 7): Values in var names are interpolated
  - Here documents (line 8-11)
    - Enclose between <<<DOCNAME and end it with a single line containing the doc name DOCNAME
    - Values from variables need to be interpolated into string
- String operations: (.) is concatenate as in Line 6
    - (strtolower()) converts string into lower case;

```
 0) print 'Welcome to my Web site.';
 1) print 'I said to him, "Welcome Home"';
 2) print 'We\'ll now visit the next Web site';
 3) printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
 4) print strtolower('AbCdE');
 5) print ucwords(strtolower('JOHN smith'));
 6) print 'abc' . 'efg'
 7) print "send your email reply to: $email_address"
 8) print <<<FORM_HTML
 9) <FORM method="post" action="$_SERVER['PHP_SELF']">
10) Enter your name: <input type="text" name="user_name">
11) FORM_HTML
```

**Figure 26.4**
Illustrating basic PHP string and text values.

# Programming Constructs

- Numeric data types: Integers and floating points; follow C rules
- Programming language constructs
  - For-loops, while-loops, and conditional if-statements
- Boolean expressions
- Comparison operators
  - == (equal), != (not equal), > (greater than), >= (greater than or equal), < (less than), and <= (less than or equal)

# PHP Arrays

- Database query results
  - Two-dimensional arrays
  - First dimension representing rows of a table
  - Second dimension representing columns (attributes) within a row
- Main types of arrays:
  - **Numeric** and **associative**
- **Numeric**: Associates a numeric index with each element in the array
  - Indexes are integer numbers
    - Start at zero
    - Grow incrementally
- **Associative**: Provides pairs of (key => value) elements

# PHP Arrays (cont'd.)

**Figure 14.3**
Illustrating basic PHP array processing.

```
0)  $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                      'Graphics' => 'Kam');
1)  $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
2)  sort($teaching);
3)  foreach ($teaching as $key => $value) {
4)    print " $key : $value\n";}
5)  $courses = array('Database', 'OS', 'Graphics', 'Data Mining');
6)  $alt_row_color = array('blue', 'yellow');
7)  for ($i = 0, $num = count($courses); i < $num; $i++) {
8)    print '<TR bgcolor="' . $alt_row_color[$i % 2] . '">';
9)    print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
10) }
```

# PHP Arrays (cont'd.)

- Techniques for looping through arrays in PHP
- Count function
  - Returns current number of elements in array
- Sort function
  - Sorts array based on element values in it

# programming constructs

- Other programming constructs similar to C language constructs
  - for-loops
  - while-loops
  - if-statements
- Boolean logic
  - True/false is equivalent no non-zero/zero
- PHP Arrays
  - Allow a list of elements
  - Can be 1-dimensional or multi-dimensional
  - Can be **numeric** or **associative**
    - Numeric array is based on a numeric index
    - Associative array is based on a key => value pair
  - A**ssociative**
    - Line 0: $teaching is a associative array
    - Line 1 shows how the array can be updated/accessed
  - **Numeric**
    - Line 5: $courses is a numeric array
    - No key is provided =>; numeric array
  - There are several ways of looping through arrays
    - Line 3 and 4 show "**for each**" construct for looping through each and every element in the array
    - Line 7 and 10 show a traditional "**for loop**" construct for iterating through an array

**Figure 26.5**
Illustrating basic PHP array processing.

```
0)  $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                      'Graphics' => 'Kam');
1)  $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam'
2)  sort($teaching);
3)  foreach ($teaching as $key => $value) {
4)    print " $key : $value\n";}
5)  $courses = array('Database', 'OS', 'Graphics', 'Data Mining');
6)  $alt_row_color = array('blue', 'yellow');
7)  for ($i = 0, $num = count($courses); i < $num; $i++) {
8)    print '<TR bgcolor="' . $alt_row_color[$i % 2] . '">';
9)    print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
10) }
```

# PHP Functions

- to structure complex program and share common sections of code
- Arguments passed by value
- Code P1' has now two functions: display_welcome() and display_empty_form()
- Line 14-18 show how these functions can be called
- Fcn course_instructor(): A function with arguments and return value

```
//Program Segment P1':
 0) function display_welcome() {
 1)      print("Welcome,  ") ;
 2)      print($_POST['user_name']);
 3) }
 4)
 5) function display_empty_form(); {
 6) print <<<_HTML_
 7) <FORM method="post" action="$_SERVER['PHP_SELF']">
 8) Enter your name: <INPUT type="text" name="user_name">
 9) <BR/>
10) <INPUT type="submit" value="Submit name">
11) </FORM>
12) _HTML_;
13) }
14) if ($_POST['user_name']) {
15)    display_welcome();
16) }
17) else {
18)    display_empty_form();
19) }
```

```
 0) function course_instructor ($course, $teaching_assignments) {
 1)   if (array_key_exists($course, $teaching_assignments)) {
 2)     $instructor = $teaching_assignments[$course];
 3)     RETURN "$instructor is teaching $course";
 4)   }
 5)   else {
 6)     RETURN "there is no $course course";
 7)   }
 8) }
 9) $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                      'Graphics' => 'Kam');
10) $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
11) $x = course_instructor('Database', $teaching);
12) print($x);
13) $x = course_instructor('Computer Architecture', $teaching);
14) print($x);
```

**Figure 26.7**
Illustrating a function with arguments and return value.

# PHP Server Variables and Forms

- Provides useful information about server where the PHP interpreter is running
- Number of built-in entries in PHP function: **$_SERVER**
  - $_SERVER['SERVER_NAME‘]
    - This provides the Website name of the server computer where PHP interpreter is running
  - $_SERVER['REMOTE_ADDRESS']
    - IP address of client computer that is accessing the server
  - $_SERVER['REMOTE_HOST']
    - Website name of the client user computer
  - $_SERVER['PATH_INFO']
    - The part of the URL address that comes after backslash (/) at the end of the URL
  - $_SERVER['QUERY_STRING']
    - The string that holds the parameters in the IRL after ?.
  - $_SERVER['DOCUMENT_ROOT']
    - The root directory that holds the files on the Web server
- **$_POST**
  - Provides input values submitted by the user through HTML forms specified in `<INPUT>` tag

# PHP Database Programming

- PEAR DB library
  - PHP Extension and Application Repository (PEAR)
  - Provides functions for database access
- Library module DB.php must be loaded
- DB library functions accessed using `DB::<function_name>`
- `DB::connect('string')`
  - Function for connecting to a database
  - Format for 'string' is: `<DBMS software>://<user account>:<password>@<database server>`

```
 0) require 'DB.php';
 1) $d = DB::connect('oci8://acct1:pass12@www.host.com/db1');
 2) if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage());}
    ...
 3) $q = $d->query("CREATE TABLE EMPLOYEE
 4)   (Emp_id INT,
 5)   Name VARCHAR(15),
 6)   Job VARCHAR(10),
 7)   Dno INT)" );
 8) if (DB::isError($q)) { die("table creation not successful – " .
                           $q->getMessage()); }
      ...
 9) $d->setErrorHandling(PEAR_ERROR_DIE);
    ...
10) $eid = $d->nextID('EMPLOYEE');
11) $q = $d->query("INSERT INTO EMPLOYEE VALUES
12)   ($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'])" );
    ...
13) $eid = $d->nextID('EMPLOYEE');
14) $q = $d->query('INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)',
15) array($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno']) );
```

**Figure 14.6**
Connecting to a database, creating a table, and inserting a record.

# Connecting to a Database (cont'd.)

- Query function
  - `$d->query` takes an SQL command as its string argument
  - Sends query to database server for execution
- `$d->setErrorHandling(PEAR_ERROR_DIE)`
  - Terminate program and print default error messages if any subsequent errors occur

# Collecting Data from Forms and Inserting Records

- Collect information through HTML or other types of Web forms
- Create unique record identifier for each new record inserted into the database
- PHP has a function `$d->nextID` to create a sequence of unique values for a particular table
- **Placeholders**
  - Specified by `?` symbol

# Querying DB

- Query function
  - `$d->query` takes an SQL command as its string argument
  - Sends query to database server for execution
- `$d->setErrorHandling(PEAR_ERROR_DIE)`
  - Terminate program and print default error messages if any subsequent errors occur
- **Query result**
  - `$q->fetchRow()` retrieve next record in query result and control loop
- `$d=>getAll`
  - Holds all the records in a query result in a single variable called `$allresult`

```
 0) require 'DB.php';
 1) $d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
 2) if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage()); }
 3) $d->setErrorHandling(PEAR_ERROR_DIE);
    ...
 4) $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
 5) while ($r = $q->fetchRow()) {
 6)   print "employee $r[0] works for department $r[1] \n" ;
 7) }
    ...
 8) $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
 9)   array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
      $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)   print "employee $r[0] \n" ;
13) }
    ...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)   print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
    ...
```

**Figure 14.7**
Illustrating database retrieval queries.

# Retrieval queries

- Retrieval queries and Database tables
  - Lines 4-7 retrieves name and department number of all employee records
  - Lines 8-13 is a dynamic query (conditions based on user selection)
    - Values for these are entered through forms
  - Lines 14-17 is an alternative way of specifying a query and looping over its records
    - Function $d->getAll holds all the records in $allresult

```
0)  require 'DB.php';
1)  $d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
2)  if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage()); }
3)  $d->setErrorHandling(PEAR_ERROR_DIE);
    ...
4)  $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
5)  while ($r = $q->fetchRow()) {
6)    print "employee $r[0] works for department $r[1] \n" ;
7)  }
    ...
8)  $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
9)    array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
      $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)   print "employee $r[0] \n" ;
13) }
    ...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)   print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
    ...
```

**Figure 26.9**
Illustrating database retrieval queries.

# Summary

- PHP scripting language
  - Very popular for Web database programming
- PHP basics for Web programming
- Data types
- Database commands include:
  - Creating tables, inserting new records, and retrieving database records