Ken's PHP5 & MySQL Notes

Contents

Switching from HTML to PHP

```
<?php
  # perl style comment
  //  c-style comment
  /* c-style comment */
…
?>
```

Printing to the Browser

```
$aVal = 5;  // dont need to convert to string if using echo
echo "<p>The value is = " . $aVal . ". Any ideas?</p>";

printf ("<p>The value is = %d. Any ideas?</p>", $aVal);
$img_name = sprintf("%02d", $img_no) . ".jpg";

<?php print $message ?> // is same as
<?=$message?>           // shortcut
```

Superglobals

```
$_COOKIE  $_REQUEST  $_SESSION $_FILES $_REQUEST $_ENV $GLOBALS
$_SERVER["PHP_SELF"];  // name of the current script

$_GET      $_POST
```

Server Side Post, Get

```
foreach ( $_POST as $key => $value ) {

foreach ( $_GET as $key => $value ) {

<?php
  echo "<p>Welcome <b>" . $_POST["user"] . "</b></p>";
  echo "<p>Your message is:<b>" . $_POST["message"] . "</b></p>";
?>
```

Client Side Forms

```
<form action="send_simpleform.php" method="POST">
…
<input type="text" name="user" />
…
<textarea name="message" …
```

Client Side Ajax

```
<input type="button" value="Add comment"
onClick="add_comment('CTR93');" />

var gURL = "http://localhost/svtc/toolsdown/toolsdownAddComment.php";

function add_comment(aTool) {
  ajaxRequest(gURL . "?tool=aTool&op='456'");  // no response wait
}

var gAjaxReq = false, gAjaxCallback;
function ajaxRequest(get_string) {
  try {
    gAjaxReq = new XMLHTTPRequest();
  } catch (error) {
        try {
        // IE5, IE6
        gAjaxReq = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (error) {
            return false;
        }
  }
  gAjaxReq.open("GET", get_string);
  //ajax.onreadystatechange = ajaxResponse; //uncomment for response
  gAjaxReq.send(null);
}

function ajaxResponse() {
var rc = true;
  if (gAjaxReq.readState != 4) {
    if (gAjaxReq.status == 200) {
          if (gAjaxCallback) {
              gAjaxCallback();
              rc = true;
            } else {
              alert("Request filed: " + gAjaxReq.statusText);
              rc = false;
            }
      }
  }
  return rc;
}
```

Variables

```
Are dynamically typed
      $aVar = 5;

Datatypes:
      Object, boolean(true, false), Integer, Float or double, String,
      Array, Resource (e.g. database handle), NULL

      is_null($aVar); is_int($aVar); is_string($aVar);
      is_double($aVar); is_bool($aVar); is_array($aVar);
      is_numeric($aVar); is_resource($aVar);
      is_empty($aVar);

settype($aVar, 'string'); settype($aVar, 'integer');
settype($aVar, 'double'); settype($aVar, 'bool');

      Casting:
            $aFloat = (double)$aVar; $aString = (string)$aVar;
            $anInt = (integer)$aVar; $aBool = (boolan)$aVar;
```

Strings

```
$length = strlen($aString);
strtoupper($aString);
strtolower($aString);

$anInt = (integer)$aString;
$aString = (string)$anInt;
$anotherString = $aString . (string)$anInt;  // concat

$aString = "pAB7";
if (strstr($aString, "AB") { … // returns AB (or false if not found)
$i = strpos($aString, "AB"); // returns 1
$piece = substr($aString, 1, 2);  // returns AB
$piece = substr($aString,1);       // returns AB7

$piece = substr_replace($aString, "CD", 1, 2);  // returns pCD7
str_replace("AB", "CD", $aString);

$delims = "?&";
$aList = strtok($aString, $delims); //list of delim sep tokens
```

Explode (like Perl split)

```
$pieces_array = explode (“/”, $filename);  // perl split
if (count($pieces) == 2) {
   $csv_buffer = $pieces[0] . "," . $pieces[1] . "\n";
}
```

Common Code:

Getting the Last Piece for the Return URL

```
if ($value == "Submit") {
  $pieces = explode ("/", $key);
  $returnlink = $pieces[sizeof($pieces)-1];     //last piece
  $returnlink = substr($returnlink,0,strlen($returnlink)-4) . ".php";
  print "(DBG) Return link: " . $returnlink . "<br />";
…
```

```
<?php
  if (strlen($returnlink) > 0) {
    print ("<input type='button' onClick=\"location.href='" .
              $returnlink . "'\" value='RETURN'>");
  }
?>
```

Implode

```
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);
echo $comma_separated; // lastname,email,phone
```

Trim

```
$text = trim($text); $text = rtrim($text); $text = ltrim($text);
```

Arrays

```
$anArray = array('red', 'green', 'blue');  // $anArray[0],[1],[2]
// for numeric arrays only:
$aList($a, $c, $c) = list($anArray); // assigns array values to $a,$b,$c
```

Associative Arrays (= Perl Hash)

```
$aPerson = array (
      "name" => "Bob",
      "occupation" => "superhero",
      "age" => 30);

echo $aPerson['occupation'];
// array within an array datastructure
$people = array (
      array(
            "name" => "Bob",
            "occupation" => "superhero",
            "age" => 30),
            array(
               "name" => "Ken",
               "occupation" => "plumber",
               "age" => 55
                  )
            );

echo $people[0]['occupation'];  // superhero
```

Array Functions

```
count($anArray);  sizeof($anArray);
$newArray = array_merge($array1, $array2);
$keysArray = array_keys($anArray);
$valuesArray = array_values($anArray);
each(), list(), reset() // for iterators-back to the array start

array_push($anArray, "element1", "element2");
$my_array[] = "foo";  // places foo at the end of my_array
$anEntry = array_pop($anArray);  // stack


array_unshift($anArray, "element1");  // puts onto the front
$anEntry = array_shift($anArray); //pops from the front (queue)

shuffle($anArray);  // randomize the elements
```

Last Array Element

```
Picking off the array end: (unlike Perl, an index of –1 didn't work)

  $pieces_array = explode ("/", $this_img_name);
```

```
  if (count($pieces) == 2) {
      $csv_buffer = $pieces[0] . "," . $pieces[1] . "\n";
  }
  $last = sizeof($pieces_array) - 1;
  $filename = "./" . $pieces_array[$last - 1] . "/" .
  $pieces_array[$last]; // eg: ./o0020/anImage.jpg
```

Getting last array element (alternate ways)

```
  echo "end($pieces_array)";                    // is same as:
  echo "$pieces_array[count($pieces_array) – 1]"; // is same as
  echo "array_slice($anArray, -1, 1)"; // don't confuse w array_splice
```

Looping through numeric [0..n] array values

```
   foreach ($pieces_array as $aValue) {
```

Looping through an associative array

```
    foreach ($pieces_array as $aKey => $aValue ) {

$merged_array = array_merge($first_array, $second_array);

$no_of_elements = array_push ($merged_array, first_array);
$no_of_elements = array_push ($merged_array, second_array);

$anElement = array_shift($anArray); // removes and returns element [0]

$elements = array_slice($anArray, 2, 3); // starting at [2] returns 3 elements
$elements = array_slice($anArray, -1, 1); // last element
```

Sorting Arrays

```
sort($anArray); // no return, sorts numeric if all numbers, alphabet otherwise

asort($anArray); // sorts assoc array values, moving the key,value pairs
arsort($anArray); // decending form of asort

ksort($anArray); // sort by assoc array keys, moving key,value pairs
// someFunc returns –1 for less than, 0 for equal, 1 for greater than
uasort($anArray, someFunc); // changes anArray using filtering
```

Filtering an Array (like Perl Map)

```
function is_less_than_120 ( $aVal ) {
  return $aVal < 120;
}

$anArray = array (4,23,67,83,546,768);
$fileteredArray = array_filter( $anArray, is_less_than_120);

$anArray = array ('bob' => 4, 'sam' => 23, 'jim' => 67);
$fileteredArray = array_filter( $anArray, is_less_than_120);
```

Constants (no preceeeding $ sign)

```
define('LOGFILE_NAME', "logfile.txt");
define('LOGFILE_NAME', "logfile.txt", true);
    // true means: lOgFILE_name is case insensitive

Some_call(LOGFILE_NAME);  // the define gets the quotes

Predefined Constants
__FILE__  filename that PHP is currently reading
__LINE__  current line number
```

Tricky Stuff with Define

```
// The value 'veggie' is assigned to a constant named fruit.
$arr = array('fruit' => 'apple', 'veggie' => 'carrot');
print $arr['fruit']; // apple
print $arr['veggie']; // carrot

define('fruit', 'veggie');  // fruit defined as 'veggie'

// Notice the difference now
print $arr['fruit']; // apple, no constants not referenced with strings
print $arr[fruit]; // carrot

// The following is okay, as it's inside a string.
// Constants are not looked for within strings,
// so no E_NOTICE occurs here

print "Hello $arr[fruit]"; // Hello apple, inside string NOT a define

// With one exception:
//   braces surrounding arrays within strings allows constants
//   to be interpreted
print "Hello {$arr[fruit]}"; // Hello carrot
print "Hello {$arr['fruit']}"; // Hello apple


If you want the last key of an array regardless of how it is indexed or
what sequence (if any) the keys are in:

end($array); // lasy key in the associative array
$maxIndex = key($array);
```

Curly Braces

```
<?php
  $username = $_POST['username'];
  $password = $_POST['password'];
  echo "{username}:{$password};  // since no spaces, need curly braces
?>

<?php
  $beer = 'Heineken';
```

```
echo “He drank some $beers”;  // wont work, tailing s throws off the var
echo “He drank some ${beer}s”; // works
echo “He drank some {$beer}s”; // works
```

Decisions

C-like

```
      if ( $a != $b) {   // === type and value equivalence

      } elseif {      // both forms of elseif will work, eg

      } else if {  // another form of elseif (but not elsif)

      } else {

      }

      while ( ) {

      }

      do {

      } while ( );
```

Looping, Collections

```
for ($i = 0; $i < $gMAX; $i++ ) {
   ...
}

foreach ($aPerson as $anEntry) {
      list ($aKey, $aValue) = $anEntry;
      echo "$aKey = $aValue";
}
```

Do While, Until

```
$i = 0;

while ($i < 99) {
  echo $i++;
}

do {
  echo $i++;
} while ($i < 99);
```

Functions

```
<?php
      $ghFile = 0;

      // parm3 is optional, and has a default
      function some_function ($parm1, &$parm2ref, $parm3 = 5) {
        global $ghFile;  // else will be scoped outside of function
        static $count = 0; // same as C static
        $parm2ref = 12;  // note – dont have to dereference
        $rc = 0;
      …
        return $rc;
      }

// passing by ref
<?php
function foo(&$var)
{
$var++;
}

$a=5;
foo($a);  // $a becomes 6 here
?>

if ( function_exists(some_function) ) { …

// must always use parenthesis in a function call
some_function ($num,2);

// will also work:
$my_function = $some_function;
$my_function($num,2);

// Anonymous function is macro-like
$anon_function = create_function('$a, $b', 'return $a + $b;');
      $anon_function = create_function("\$a, \$b", "return \$a + \$b;");

// Returning multiple variables
function my_function() {
…
  return array($parm1, $parm2…);
}

//Caller
list($parm1, $parm2,…) = my_function();
```

Date Time

```
$now = getdate();  // returns an associative array
$now['seconds']
$now['minutes']
$now['hours']
$now['mday'];  // day of the month 1-31
$now['wday']; // days of the week 0-6
$now['mon'];  // 1-12
$now['year']; // 4 digits 2008
$now['yday']; // 0-366
$now['weekday'];  // Sunday
$now['month'];  // February
$now['0'];  // unix timestamp

$now=time();    // returns Unix datetime
$aDateString = date("Y-m-d h:i:s", $now); //2013-19-30 17:28:01
$aDateString = date("l m/d h:ia", $now);  //Monday 9/30 5:28pm
// make a unix timestamp for Feb 3 2008 at 2:13pm
$then = mktime(2,13,0,2,3,2008);

$is_valid_date = checkdate(4,4,1066);
```

Sanitizing User Input

```
if ( (array_key_exists('userid',   $_POST)) &&
     (array_key_exists('password', $_POST)) ) {

      $theUserId   = escapeshellcmd($_POST['userid']);
      $thePassword = escapeshellcmd($_POST['password']);

      $theUserId   = escapeshellarg($theUserId);
      $thePassword = escapeshellarg($thePassword);

      $theUserId   = htmlentities($theUserId);
      $thePassword = htmlentities($thePassword);

      $theUserId   = trim($theUserId);
      $thePassword = trim($thePassword);

      $theUserId   = strip_tags($theUserId);
      $thePassword = strip_tags($thePassword);

 //   $theUserId   = str_replace(' ', '', $theUserId); // strip spaces
 //   $thePassword = str_replace(' ', '', $thePassword);

   $theUserId   = preg_replace('/\s+/','',$theUserId);  // strip all whitespace
   $thePassword = preg_replace('/\s+/','',$thePassword);
```

Url Decoding

What if the URL parms have embedded ?, &, = characters in the key,value names?

http://www.someadr.com?parm1=12&parm2=14

To keep from interfereing with the URL we do:

<?php echo **urlencode** ("ken&"); ?> &id=43

where & becomes %26

Likewise:

$name = **urldecode**($_GET['name']);

Note: $_POST does not need urldecoding, only $_GET

Raw Url Encoding

Use **rawurlencode** for everything BEFORE the ?
Use **urlencode** for everything AFTER the ?

and there's a 3rd special case when constructing links:

<click>& you'll see

in PHP:

```
<?php
  $linkText = "<click>& you'll see";
  echo htmlspecialchars($linktText);
?>
```

Objects

```
<?php
class SomeClass {
  var $aVar = 5;

  function SomeClass ( $initValue ) { // constructor
    $this->aVar = $initValue;
  }
}

?>

<?php
$anInstance = new SomeClass(5);
…
?>

<?php
class AnotherClass extends SomeClass {
  function sayHello($msg=”hello”) {
    print “\$msg”;
  }
}
```

Try-Catch

```
try {

} catch (Exception $e) {
  echo “Error:” . $e;
}
```

```
//show nothing
error_reporting(0);

//show everything
error_reporting(E_ALL);

//using php.ini and ini_set()
ini_set('error_reporting', E_ALL);

//show warnings and errors
error_reporting(E_ERROR | ERROR_WARNING);

//show all types but notices
error_reporting(E_ALL ^ E_NOTICE);
```

1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
6143	E_ALL
2048	E_STRICT
4096	E_RECOVERABLE_ERROR

Files

```
// creates file, all writing from file start
if ($hFile = fopen("test.txt", "w")) {
    // … good file open
```

Writing

```
fwrite($hFile, "some info");
fputs($hFile, "some other info);  // same as fwrite

$hFile = fopen("test.txt", "r");  // "w", "a" are other options
if ( $hFile == false ) { // error!
} else {
  fclose($hFile);
}

fseek($hFile, 0);
```

Reading

```
while (! feof($hFile) ) {
  // reads a line OR 1024 bytes, whichever comes 1st
  $aLine = fgets($hFile, 1024);// reads up to 1024 bytes
  $aLine = fread($hFile, 16);  // reads in 16 bytes, ignoring line endings
  $aChar = fgetc($hFile);      // reads one character
}

$hDir = opendir("/tmp");
while (! (($aFile = readdir($hDir)) == false) ) {
  if (is_dir(…
```

File Functions

```
file_exists("text.txt");
is_file("test.txt");
is_dir("/tmp");
is_readable("file.txt");
is_writable("file.txt");
is_executable("file.pl");
$size = filesize("file.txt");
$atime = fileatime("test.txt");  // accessed,unix datetime
$mtime = filemtime("test.txt");  //modified,seconds since 1/1/1970
$ctime = filectime("test.txt");  // created

    // Only list files modified within the last 30 days
    $now = time();  //unix datetime = seconds since 1/1/1970
    foreach (glob($glob_pattern) as $filename) {

        $modifiedOn = filemtime($filename);
        $daysOld = (integer)(($now - $modifiedOn) /  86400);

        if ($daysOld <= 30 ) { …

touch("test.txt");  // creates the file if it doesn't exist
unlink("test.txt");  // deletes file
```

Include

```
include(“file.php”);
include_path .:/home/wwwroot/htdocs/project4/lib/ in php.ini
include_once(“file.php”);  // incase the same code is sourced in again

<?php
if (file_exists('c:/inetpub/wwwroot/svtc/toolsdown/ash93.htm')) {
  include 'c:/inetpub/wwwroot/svtc/toolsdown/ash93.htm';
} ?>
```

Mysql

```
mysql –u <username> -p   // login
show databases;
use aDatabase;
desc aTable;

<?php
$mysqli = new mysqli("localhost", "joeuser", "apassword", "testDB");
if (mysqli_connect_errno() ) {
  printf("Cannection failed: %s\n", mysqli_connect_error() );
} else {
  printf("Host information:%s\n", mysql_get_host_information($mysqli) );
  $theSql = "CREATE TABLE testTable
             (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
              testField VARCHAR(75))";

  $rc = mysqli_query($mysqli, $theSql);

  if ($rc == TRUE) {
    echo "Table created successfully";
  } else {
    printf ("ERROR could not create table: %s\n", mysqli_error($mysqli));
  }

  mysqli_close($mysqli);
}
?>
```

Some Common Queries

```
CREATE DATABASE aDataBase;

CREATE TABLE testTable (id INT NOT NULL AUTO_INCRMENT,
PRIMARY_KEY(id), testField VARCHAR(50);
GRANT insert,update,select on *.* to aTable@localhost identified
by 'aPassWord';

INSERT INTO testTable (testField) VALUES ('first message');
DELETE FROM testTable WHERE id=2;
UPDATE members set expire_ts='2009-09-09 12:00:00' where id=26;

ALTER TABLE some_table ADD expired TYPE boolean;  // add a column
ALTER TABLE (aTableName) ADD <colname> <coltype>;

show tables;
show tables like 'journal%'
describe service_attr;
```

Mysql Data Types

```
Numeric: int, tinyint(-128 to 127), smallint(-32768 to 32767),
mediumint, bigint, float(m,d) 6.2 is xxx.xx ; double(m,d);
decimal(m,d)=numeric(m,d)

date: yyyy-mm-dd
datetime: yyyy-mm-ss hh:mm:ss
timestamp: unix timestamp
```

```
time: hh:mm:ss
year(m) : 2 digit or 4 digit year

char(m): fixed length between 1 to 255 characters in length
varchar(m): variable length between 1 to 255 characters in length
blob or text: binary large object up to 65,535 characters
tinyblob or tinytext: up to 255 characters
mediumblob or mediumtext:
longblob or longtext:     enum:
```

Mysql Datetime

```
SELECT DATE_ADD(NOW(), INTERVAL 30 DAY);   // not dayS
SELECT DATE_SUB(NOW(), INTERVAL 30 DAY);
SELECT "2007-12-31" + INTERVAL 30 DAY;  // same as DATE_ADD
SELECT "2007-12-31" – INTERVAL 30 DAY;

SELECT CURDATE(), CURRENT_DATE();  // same
SELECT CURTIME(), CURRENT_TIME();   // same
SELECT NOW(), SYSDATE(), CURRENT_TIMESTAMP();  // same
+-------------------------
| NOW()
| --------------------------
| 2008-02-03 15:23:53

SELECT UNIX_TIMESTAMP();   // now unix timestamp
SELECT UNIX_TIMESTAMP('1973-12-30');

SELECT FROM_UNIXTIME('1202073856');
+-------------------------
| FROM_UNIXTIME('1202073856')
| --------------------------
| 2008-02-03 15:23:53
```

Mysql Stored Procedures

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE some_procedure1 () SELECT * FROM aTable //
mysql> CALL some_procedure1 () //
```

Mysql Exporting, Importing

```
select * from some_table order by id into outputfile 'out.file'
fields terminated by ',';

load data local infile '/import.csv' into table some_table
fields terminated by ','
lines terminated by '\n'
(aField1, aField2, aField3,…);
```

Database Normalization

1st normal form:

No repeated data. Create separate tables

2nd normal form:

Data depends on the WHOLE key (e.g., in case you concatenate two attributes together to make a unique key

e.g.: person,skill <= address *violates since address has nothing to do with skill*

3rd normal form:

Attributes don't depend on an attribute which depends on the whole key,

e.g.: tournament,year <= winner <= winner's birthday

Referential Integrity:

A foreign key in one table refers back to at least one non-null primary key in another –OR- the foreign key is NULLed out.