cis20.2
design and implementation of software applications 2
spring 2010
lecture # II.4

today's topics:

- introduction to php
- on-line documentation
  - `http://www.php.net` (php home page)
  - `http://www.php.net/manual/en/` (online reference)

introduction to php

- history and background
- your first php program
- basics
- writing your own functions
- arrays
- classes
- I/O

history and background

- developed in the latter 1990's
- originally created as "Personal Home Page" tools, by Rasmus Lerdorf
- at first, was a quick tool for embedding sql queries in a web page (v1.0)
- then structured code was added (v2.0), but with a buggy language parser
- official release (v3.0) fixed parser bugs - June 1998
- by Jan 1999, 100,000 web pages were using php!!!
- php is better than CGI because:
  - it runs as part of the web server process and doesn't require forking (unlike CGI)
  - it runs faster than CGI
  - it's faster to write...
- php was designed to run with apache web server on unix
  - but also runs on windows and mac
- it's free!

- php is coded in C
  - has a well-defined API
  - extensible
- the way it runs:
  - a php engine is installed as part of a web server
  - the engine runs the php script and produces html, which gets passed back to the browser

your first program(s)

- `hello.php` (plain php)
- `hello2.php` (php embedded in html)
- `hello3.php` (uses <?php start tag)

hello.php

```
<?
print "hello world!";
?>
```


hello2.php

```
<html>
<body bgcolor=#000000 text=#ffffff>
<?
print "hello world!";
?>
</body>
</html>
```


hello3.php

```
<html>
<body bgcolor=#000000 text=#ffffff>
<?php
print "hello world!";
?>
</body>
</html>
```

basics

- php start and end tags: `<? ... ?>`
- also: `<?php ... ?>`
- semi-colon ends a statement (like C)
- string constants surrounded by quotes (") or (')
- you can embed multiple php blocks in a single html file
- variable names are preceded by dollar sign ($)
- user input is through html forms
- the language is case-sensitive, but calls to built-in functions are not (not sure if that's true for all built-in functions)
- identifiers are made of letters, numbers and underscore (_); and cannot begin with a number
- expressions are just like in C

data types

- integers
- floating-point numbers
- strings
- loosely typed (you don't have to declare a variable before you use it)
- conversion functions: `intval, doubleval, strval, settype`
- `settype( <value>, <newtype> )` where `newtype="integer", "double" or "string"`
- typecasting: `(integer), (string), (double), (array), (object)`

operators

- mathematical: `+, -, *, /, %, ++, --`
- relational: `<, >, <=, >=, ==, !=`
- logical: `AND, &&, OR, ||, XOR, !`
- bitwise: `&, |, ^ (xor), ~ (ones complement), >>, <<`
- assignment: `=`, =, -=, *=, /=,
- other:
  - `.` → concatenate
  - `->` → references a class method or property
  - `=>` → initialize array element index

conditionals (1)

- if/elseif/else:

```
if ( <expression1> ) {
  <statement(s)>
}
elseif ( <expression2> ) {
  <statement(s)>
}
else {
  <statement(s)>
}
```

conditionals (2)

- tertiary operator:

```
<conditional-expression> ?
  <true-expression> : <false-expression>;
```

- switch:

```
switch( <root-expression> ) {
  case <case-expression>:
    <statement(s)>;
    break;
  default:
    <statement(s)>;
    break;
}
```


loops

- while

```
while ( <expression> ) {
  <statement(s)>;
}
```

- do-while

```
do {
  <statement(s)>;
} while ( <expression> );
```

- for

```
for ( <initialize> ; <continue> ; <increment> ) {
  <statement(s)>;
}
```

- break:
  - execution jumps outside innermost loop or switch

other

- `exit()` function
  - halts execution, meaning that no more code (php or html) is sent to the browser
- built-in constants
  - `PHP_VERSION`
  - `__FILE__`, `__LINE__`
  - `TRUE` = 1, `FALSE` = 0
  - `M_PI` = pi (3.1415927....)

writing your own functions

- declared just like C:

```
function <name> ( args ) {
  <body>
  [return <value>]
}
```

- called just like C
- arguments (and local variables) are local, and don't exist when you exit the function; but you can use "static" to declare a variable so that when you call a function again, the value is retained
- use the "global" statement to declare global variables that you want to be able to access from within a function, or the `GLOBALS` array (which is like a perl hash)
  e.g., `GLOBALS['username']`
- recursion is okay, but be careful!
- example: `colors.php`

colors.php

```
<html>
<body bgcolor=#ffffff>
<?
// modified from figure 2-4 from "introduction to php" by leon atkinson
function useColor() {
  static $colorval;
  if ( $colorval == "#ff0000" ) {
    $colorval = "#0000ff";
  }
  else {
    $colorval = "#ff0000";
  }
  return( $colorval );
}

print "<table width=\"300\">";
for ( $count=0; $count < 10; $count++ ) {
```



```
  $rowcolor = useColor();
  print "<tr><td bgcolor=\"$rowcolor\">";
  print "row number $count</td></tr>";
}
print "</table>"
?>
</body>
</html>
```


arrays

- indexed using `[...]`
- indeces can be integers or strings (like a perl hash)
- when strings are indeces, it's called an "associative array"
- `array()` function can be used to initialize an array
- e.g., `$var = array( value0, value1, value2, ... );`
- use the => operator to define the index:

```
$var = array( 1=>value1, value2, ... );
$var = array( "a"=>value1, "b"=>value2, ... );
```

- multidimensional arrays are okay (like C)
- example: `arrays.php`

arrays.php

```
<html>
<body bgcolor=#ffffff>
<?
// modified from figure 5-6 from "introduction to php" by leon atkinson
$states = array( "CA","NY" );
print "here are the states:<br>";
for ( $i=0; $i<count( $states ); $i++ ) {
  print "-- $states[$i]<br>";
}
print "<p>";
$cities = array( "CA"=>array( "san francisco","los angeles" ),
 "NY"=>array( "new york","albany","buffalo" ));
print "here are the CA cities:<br>";
for ( $i=0; $i<count( $cities["CA"] ); $i++ ) {
  print( "-- ".$cities["CA"][$i]."<br>" );
}
print "here are the NY cities:<br>";
```

```
for ( $i=0; $i<count( $cities["NY"] ); $i++ ) {
  print( "-- ".$cities["NY"][$i]."<br>" );
}
print "<p>";
$states[] = "MA";
print "now here are the states:<br>";
for ( $i=0; $i<count( $states ); $i++ ) {
  print "-- $states[$i]<br>";
}
$cities[] = "MA";
$cities["MA"][] = "boston";
print "here are the MA cities:<br>";
for ( $i=0; $i<count( $cities["MA"] ); $i++ ) {
  print( "-- ".$cities["MA"][$i]."<br>" );
}


?>
</body>
</html>
```


classes

- defining a class:

```
class <class-name> {
  // declare properties
  // declare methods
}
```

- use just like java and c++
- example: `myclass.php` and `userclass.php`
- note use of `include` statement

myclass.php

```
<html>
<body>
<?
// figure 6-2 from "introduction to php" by leon atkinson
include "userclass.php";

$currentuser = new user;
$currentuser->init( "yoda","jedi" );

print( "name = ".$currentuser->name."<br>" );
print( "last login = ".$currentuser->getLastLogin() );

?>
</body>
</html>
```


userclass.php

```
<?
// figure 6-2 from "introduction to php" by leon atkinson
class user {

  // properties
  var $name;
  var $password;
  var $last_login;

  // methods
  function init( $inputname, $inputpassword ) {
    $this->name = $inputname;
    $this->password = $inputpassword;
    $this->last_login = time();
  }

  function getLastLogin() {
```

```
    return( date( "M d Y", $this->last_login ));
  }

}
```


I/O

- get input from html forms using

  ```
  $_POST['<name>']
  $_GET['<name>']
  $_REQUEST['<name>']
  ```

- file I/O
  - basically just like C:

    ```
    $fp = fopen( "filename","w" );
    fwrite( $fp,"stuff" );
    fclose( $fp );
    ```

  - note that `fopen` second argument mode is like C)

lunch.html

```
<!-- figure 1-4 from "introduction to php" by leon atkinson -->
<html>
<body>
<form action="lunch.php" method="post">
your name:
<input type="text" name="yourname"><br>
cost of lunch:
<input type="text" name="cost"><br>
days buying lunch:
<input type="text" name="numdays"><br>
<input type="submit">
</form>
</body>
</html>
```


lunch.php

```
<?
// figure 1-5 from "introduction to php" by leon atkinson

$today = date("l F d, Y");
$yourname = $_POST['yourname'];
$cost     = doubleval( $_POST['cost'] );
$numdays  = intval( $_POST['numdays'] );

?>

<html>
<body>
today is:
<?
print( "$today<br>" );
print( "$yourname, you will be out \$" );
print( doubleval( $cost * $numdays ));
```

```
print( " for buying lunch this week!" );
?>
</body>
</html>
```