CS3157: Advanced Programming

Lecture #14

May 1st

Shlomo Hershkop
shlomo@cs.columbia.edu

Outline

- Welcome to the last class!
- Advanced topics
  - PHP Overview
  - AJAX
- Review for final
- Final!

Php.net

- developed 1994-1995, first as collection of perl scripts and then own interpreter
- originally created as “Personal Home Page” tools, by Rasmus Lerdorf
- 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 introduced object oriented ideas
- V4 more object, and passing variables in the system modified
- V5 new engine, many fixes etc
- Early as Jan 1999, 100,000 web pages were using php!!! Much higher now!

Some say..

- 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...
  - Tons of libraries supported
- php was designed to run with apache web server on unix
  - but also runs on windows and mac
- Did I mention…it’s free!
  - One important way of getting something adopted….don’t underestimate the power of a ‘free lunch’

LAMP

- Linux
- Apache
- Mysql
- Perl/PHP/Python

- 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
  - So user never sees the php code (if done right)

3 different ways !

- 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
  - Any ideas why ???????
- identifiers are made of letters, numbers and underscore (_); and cannot begin with a number
- expressions are similar to 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: &, |, ˆ (xor), ˜ (ones complement), >>, <<
- assignment: =, =, -=, *=, /=,
- other:
  - . concatenate
  - -> references a class method or property
  - => initialize array element index

Conditionals

- if/elseif/else:

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

Conditional II

- 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!

Simple 1

```
<?
$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>
```

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)

code

```
<html>
<body bgcolor=#ffffff>
<?
$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>" );
}
```

Code II

```
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

userclass.php

```
<?
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 ));
  }

}
```

myclass.php

```
<html>
<body>
<?

include "userclass.php";

$currentuser = new user;
$currentuser->init( "yaddi","cat" );

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

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

```
<?php
class Car {
 public $miles; // variable that can be accessed outside the class
 private $mpg; // variable that can only be accessed within the class
 protected $mph; // variable that can only be accessed from within the class, and
 // from any inherited child classes

 public function __construct($param) { // constructor is called when object "Car" is
    created
 $this->miles = $param;
 }

 public function start() {
 // starts the car...
 }
a
 public function stop() {
 // stops the car...
 }

 public function getMpg() {
 return $this->mpg;
 }
}

$car = new Car($param);
echo $car->miles; // echos the value of the property "miles" of the class "Car"
?>
```

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)

More than just reacting

- We have been working with perl/c/cpp in a static context
- Some information is presented to the user
- React to user input

- Is this how google maps works ?

Ajax

- **A**synchronous **J**avaScript **A**nd **X**ML
- technique for developing interactive applications over the web
- Style
- Platform
- Format
- XMLHttpRequest
- Objects

Basic HTML

- Specific set of tags (depends on version)
- Up to user to set things correctly
- Most browsers will attempt to figure out what you want
  - Example not putting end body end html, will still work

Advanced HTML

- CSS
  - Cascading style sheets
    - Define format of the WebPages
    - Central location of style
    - With a few clicks can completely change thousands of WebPages.
- DOM
  - Document object model
    - Formal data structure to represent web based documents/information
- Client side scripting

DOM problems

- Different browsers supported things differently

```
if (document.getElementById &&
   document.getElementsByTagName) {
 // as the key methods getElementById and getElementsByTagName
 // are available it is relatively safe to assume W3CDOM support.

    obj = document.getElementById("navigation")
    // other code which uses the W3CDOM.
    // .....
 }
```

Examples

- http://www.dynamicdrive.com/dynamicindex12/pong/pong.htm
- http://www.dynamicdrive.com/dynamicindex4/butterfly.htm

javascript

- Client side
  - PHP & CGI were both server side
- Developed under Netscape as LiveScript
  - Currently 1.5
- Developed under IE as Jscript
- Object oriented approach
- Syntax like c
  - No input/output support native
  - Keywords
  - DOM for interfacing between server and client
- Can evaluate reg expressions (eval)

Javascript

- Heavy use of defined functions
  - Example: MouseOver
- Need to adopt to specific browser if doing anything fancy
- Adobe
  - Support javascript in pdf
- MAC
  - Dashboard widgets

Programming

- You need to learn on your own
- Many good books/websites
- Most of the time .js file if not in html
- Powerful example:
  - Thunderbird/firefox
- Get good debugger

How to do research?

- Practical research
  - Know a programming language
  - Have an inquisitive mind
  - Keep an open mind to new ideas
  - Try to solve an open research problem ☺
- Theory research
  - Learn some math
  - Learn some theory
  - Relearn the math
  - Solve something ☺

Where to start?

1. Need an idea
2. See if anyone's done it or tried it in your way
   1. Citeseer (citeseer.ist.psu.edu)
   2. Google
   3. Appropriate Faculty/Researcher
   4. Google groups

Sketch out the idea on small scale

- Design a small experiment which can validate your idea
- Data, data, data, and Data
  - Make or break you
  - Will help your research
    - Make sure it isn't a circular relationship
- Evaluate results
  - Don't fake them
  - Even bad results are results
  - Can learn of what not to do
- Write up results

Write up options

- Word vs Latex
- gnuplot
- cvs
- Element of Style

In the real world

1. Keep it simple
   1. Don’t re-invent the wheel
   2. Design first
   3. Even with fancy blinking lights, a bad idea is still a bad idea (but with bad taste)
2. Incremental testing
   1. Recognize when the bug is your fault
   2. See if others have faced it too
3. Make sure version 1 works on most popular browsers

Question

- What is this designed with?
- Can you do a better job?

- Theyrule.net

Bottom line

- We've covered a lot this semester
  - Some of it was fun
  - Some of it was hard work (ok most)
  - Some of it was frustrating.
- BUT
  - You have lots of tools
  - Have an idea of where to start when dealing with programming projects

Important lessons for learning new languages

- CS is not meant to be a trade school
- Language isn't important…things change
- Ideas and design are more important

- Lessons:
  - Choose correct environment
  - Choose correct tools
  - Make sure to test out ideas…might be someone else's fault (program think)
  - Enjoy what you are doing

Important

- To get the most out of a language find comfortable programming environment

- Emacs – color files
- Eclipse
- Others , see
  - www.freebyte.com/programming/cpp/

Review time

- Focus on C
- Focus on CPP
- Shell programming stuff
- Idea of PHP
- Perl
- Review the labs

Word list

- Compiling
- Linking
- Reference parameter
- Variable scope
- Stdio.h
- Stdlib.h
- cout
- cast
- Inline
- Linked list
- Preprocessor
- Typedef
- Struct
- Pointer
- Void pointer
- . Vs ->
- Function pointer
- Reference
- const
- malloc

Word list II

- Huffman
- getopt
- constructor
- destructor
- iostream
- overloading
- extern
- private
- Public
- GDB
- Cgi
- GET/POST
- overload
- overriding
- Template
- This
- Friend class
- New/delete
- virtual

c

- Basic constructs
- Basic type
- Advanced types
- (review labs and class examples)
- Memory stuff – understand what is happening
- Arrays
- Functions
- Pointers
- Debuggers

C

- Working with CGI
- Working on different platforms
- Makefiles
- How we built libraries

C++

- Basic language
- Difference to c
- Classes
- Permissions
- new/free memory allocations
- Inheritance and polymorphism
- Keywords
- Working with files….

Sample exam

- You've done most of the work for the course, the exam is just to make sure you remember the important concepts
- Will be posted online

- Couple Definitions
- 2 code checking question
- Shell code question
- C++ class manip question
- Small CGI question

Thinking question

- Say you are writing code which uses a random number generator….
- What is important to know about it ?
- How can your code be affected ?
- If you crash, how to reconstruct events, since based on random numbers ??

Closing Remarks

- If you like this…..just the beginning
- If you didn't ….. You now know how complicated it is….never trust a program ☺
- Hope you had a fun semester..
- Extra office hours this week, will email….