COMP4711 Lab 1 (Winter 2016)

AMP & PHP Familiarization
Excuse: Tic-Tac-Toe

Background

The purpose of this lab is to expose you to the AMP environment and to refresh your memory of PHP, before we dive too deeply into a framework and our projects. The goal for this lab is to equip you to manage an Apache server with separate folders for multiple webapps, each running as a different virtual host.

This lab will be more of a walkthrough, and this lab document may be updated based on in-lab experiences. I would not be surprised to see some FAQ entries added to the course organizer. ***Disclaimer: my gameplay logic may be flawed!***

This lab is to be completed individually.

Goal

You are to build a functioning Tic-Tac-Toe game, following the steps herein. They leave some room for your inspiration and brilliance at the end :)

Overview

In this lab, you shall:

1. make sure no servers are running already on your system
2. install AMP & verify that it works
3. make sure NetBeans is properly installed & configured
4. Create a PHP project
5. Setup a virtual host linking to your project
6. make sure it works as expected
7. build some representative PHP scripts

Note: if using the machines in lab, you may find that the drive has been rebuilt since the last time you used the machine, and you may have to repeat the first three steps. With experience, it ends up taking 5-10 minutes.

Caution & Disclaimer

This lab is provided as a PDF, to make it easier to copy/paste material. HOWEVER, some of the characters do not copy/paste nicely between PDF and your editor, specifically the double quotes and hyphens.

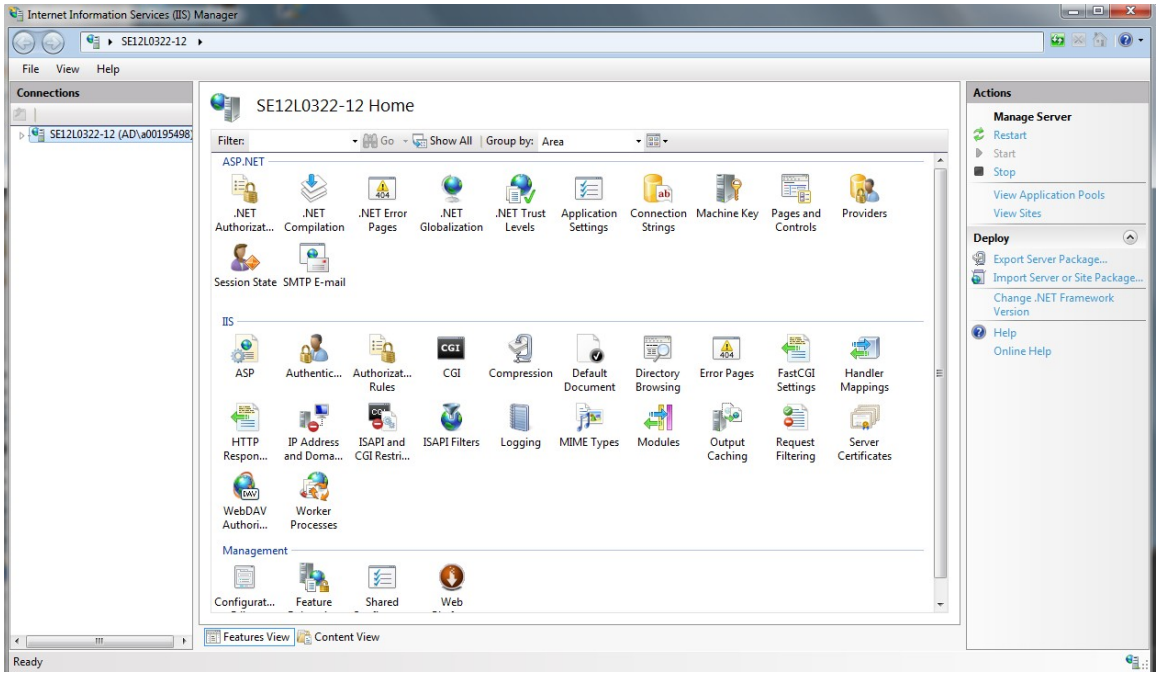
Workaround 1: retype those lines
Workaround 2: use something simple lije notepad as a buffer between the PDF and your IDE

Spoiler: Section 2 is awesome and informative, but you might want to check Section 2b first!

1. Stop Any Servers

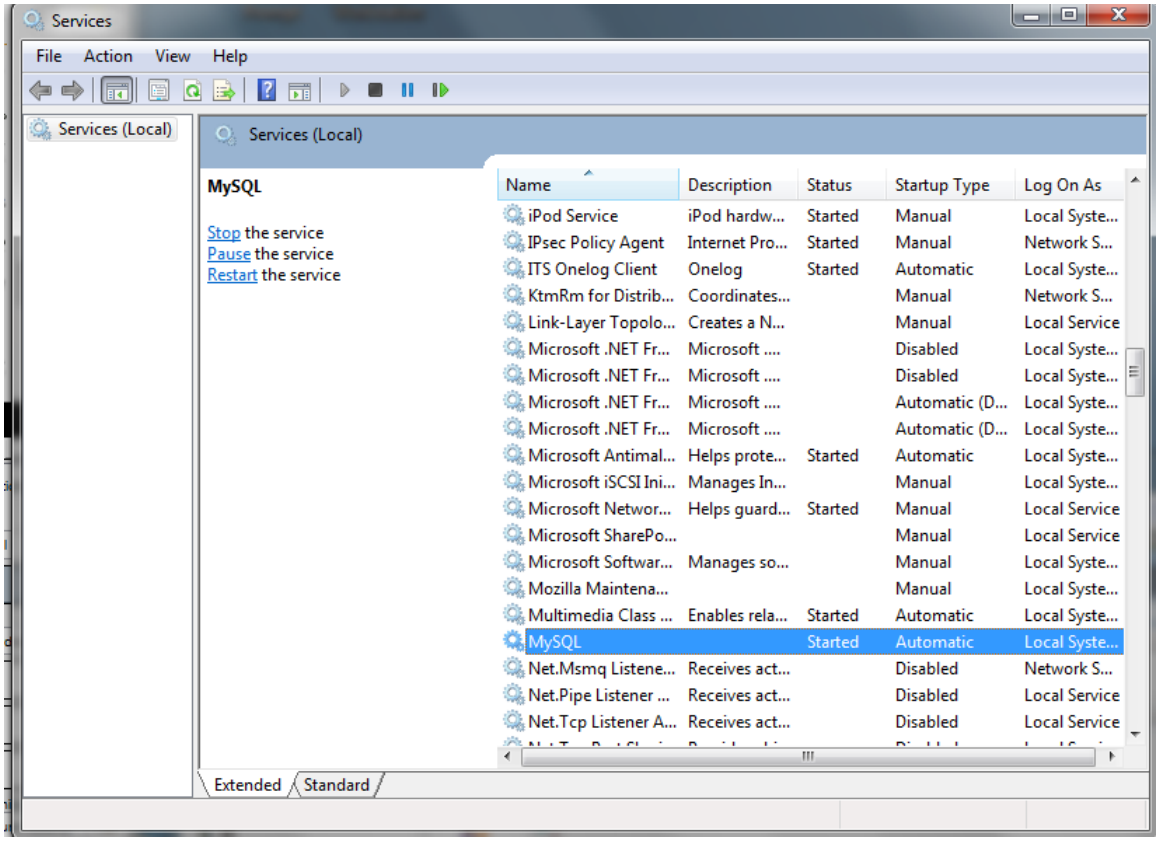
For convenience, we want to run our own web and database server, so need to stop any existing servers. In lab, that would be IIS and MYSQL.

From the Start button, search for “iis”. If nothing is found, you do not have Internet Information Server installed. If you find it, select it, and you should get a control panel something like:



In the Actions panel (top right), click “Stop” to disable IIS.

From the Start button, search for “Services”, and launch the services control panel. Scroll down to where MYSQL is, alphabetically. Your screen should look something like:



Click “Stop the service” in the action area, top left of the panel showing the list of services.

2. Install AMP

AMP = Apache + MYSQL + PHP (& perl)

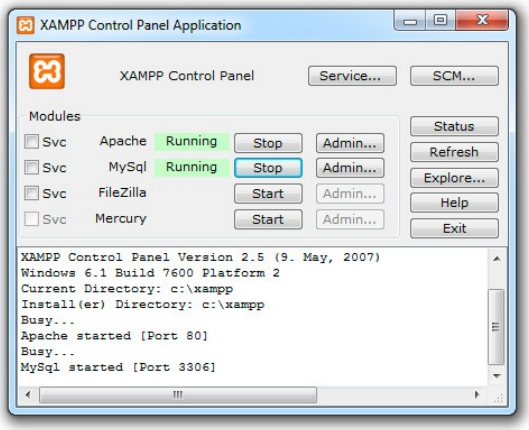
Get AMP from http://www.apachefriends.org/en/xampp.html
This site has a link in the sidebar of our organizer.

I strongly recommend using XAMPP instead of other apparently suitable packages, such as WAMPP or MAMPP. The other packages (and even some of the AMP stacks built into Linux) often have less-than-intuitive folder and configuration conventions, and we may not be able to help you with them. I have found the MacOS and Ubuntu distributions of XAMPP to be the most confusing.

Do read the installation page on apachefriends.org. It will give you some basic configuration hints, explain the operation, and tell you where the main configuration files are. You want to be very familiar with this if you are working on a machine in lab, as there are good odds you will need to setup your environment each time you start an ACIT2910 lab!

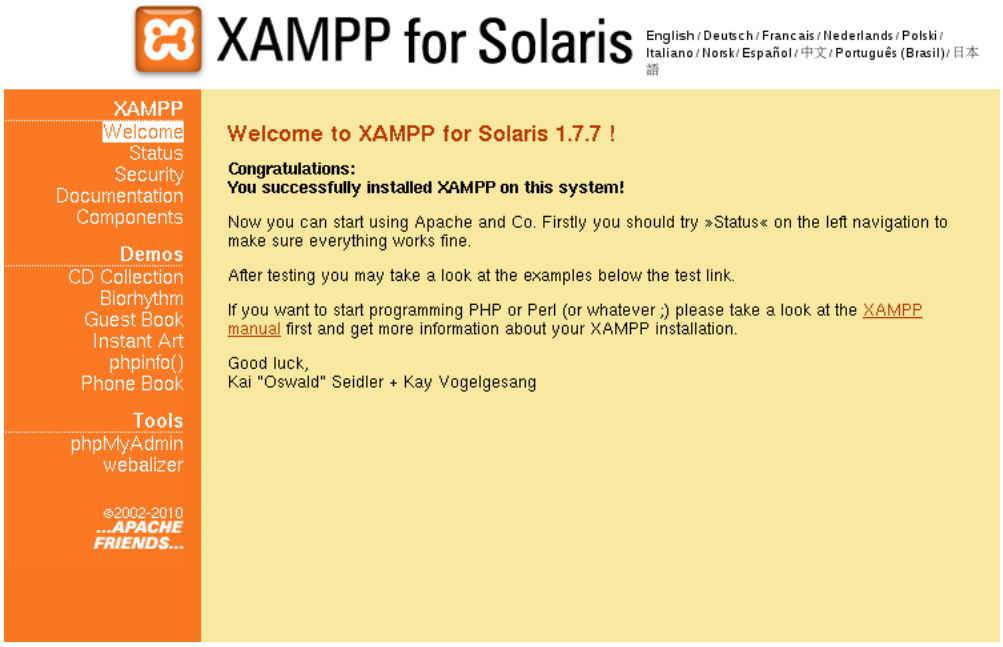
Install the flavour of AMP suited to your platform. On Windows, with standard install selections, this will end up in c:\xampp; on Unix, this will be in /opt/xampp. You can select default options when installing, but you don't need anything other than Apache, MYSQL and phpMyadmin. Unselecting the other choices will result in a smaller disk and memory footprint for your installation.

On Windows, you have the option of installing the different servers as a service or as programs managed from a control panel. I suggest the control panel management, so you don't pay for the server overhead when you are not working on course stuff.



Test if your installation worked...

Fire up Apache and MYSQL, using your control panel for instance. In your browser, ask for “localhost” or “http://localhost” if your browser needs the protocol. You should get the XAMPP homepage...



If you don't see this, there are good odds that another program is using port 80, which Apache wants to own by default. Such programs include IIS, Oracle, **Skype**, and probably others. Step 1 of this lab showed how to stop IIS; the others would be handled similarly.

In such a case, you have two choices – kill them or use a different port.

It is possible to configure Apache to use a different server port, by tailoring its configuration file (httpd.conf), but we want to avoid that if possible.

If still not getting the XAMPP homepage, see me for help.

2b. Alternate AMP Install

I did warn you to skim the writeup first, no?

An easier AMP install is to use the appropriate AMP stack for your platforn, from https://bitnami.com/stacks/infrastructure

Choose the one with the most recent PHP5.
Select the Apache and MySQL ports during installation, so they don't conflict with anything else running on your system :)

Suggested settings to avoid conflict:

- run Apache on port 4711
- run MySQL on port 4706
- use “squeal” for your mySQL root password

3. PHP-aware IDE

Make sure you have a PHP-aware IDE installed. It is *possible* to do everything with notepad or its equivalent, but you will find an IDE more productive for refactoring, GIT integration, and debugging support.

Your IDE should have GIT support built-in and enabled. Your IDE may have optional components to make working in this environment easier – a GIT toolbar, for instance.

Suitable IDEs include NetBeans and phpStorm. I will be doing marking using NetBeans, just as an FYI.

4. Start a new PHP Project

Create an account on github.com, if you do not already have one.

Create a new repository to use for this lab, suitably named.

Clone that repository locally, inside your XAMPP htdocs folder. We will be working inside that project for now.

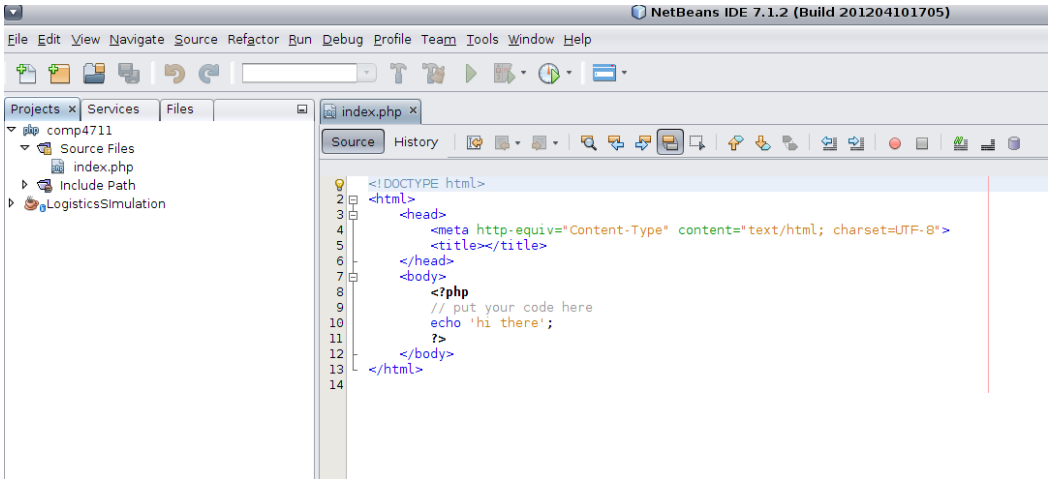
You can create an IDE project for this, for instance by creating a new PHP project with existing sources in NetBeans.

Your IDE may provide “support” for multiple frameworks – we don't need any of those.

We are not going to actually do anything with the project quite yet, but we want the project/folder to exist so that the next step is easier.

If your project doesn't have an index.php, make a dummy `index.php` inside your project, which is all we need for now. Notice the PHP delimiters `<?php` and `?>`. Anything inside such delimiters is interpreted as PHP.

Add a “hello world” echo statement to it, so you can recognize your “app” when it is run.



We won't be able to serve & test the webapp until the next step.

Note the similarity with Java: strings inside quotes, semi-colon to terminate a statement, documentation style comments.

5. The meat of our lab!

Time to learn (or remember) a bit about PHP.

After each successful step, remember to commit your changes and "push" them to your repository!

PHP can be used for "merely" scripting (snippets of logic embedded in an HTML document and interpreted server-side), or it can be used for full-fledged O-O programming. Confusingly, it can be used for both at the same time :-/

PHP documents have a .php extension, and can contain HTML with PHP embedded inside appropriate tags

```
<?php yadayadayada ?>
```

or it can have an opening PHP tag and then contain class and function definitions as well as scripting, but would not then contain a closing tag. We will do that later.

PHP statement syntax is very similar to Java, for assignment statements, unqualified method calls, block structure, and so on.

PHP Variables and Data Types

Variables in PHP have conventional names, but start with a dollar sign ($name). They are loosely typed, i.e. According to the last type of data assigned to them, and they do not have to be declared in advance.

Built-in data types include the "standards" (integer, float, string, boolean), as well as object, array, resource and null. These last will be discussed separately.

PHP has a number of built-in variables, such as `$DOCUMENT_ROOT`, `$SERVER` array, `$_POST` for HTTP post parameters, and so on.

PHP Strings

Strings can be delimited by either single or double quotes. If double quotes, additional escape sequences are enabled (\n, \r, \t, \\, \$). You can escape the delimiter inside a string (\' or \").

Try it!

In your index.php, replace your "hello world" statement with something like...

```
$temp = 'Jim';
echo 'Hi, my name is';
echo $temp;
$temp = 'geek';
echo "I am a";
echo $temp;
$temp = 10;
echo 'My level is';
echo $temp;
```

The above has horrible style and multiple bad smells, but it illustrates the loose typing. You will probably have to add a space or two to get it to look right.

Note: If you copy and paste into Netbeans, you have to make sure that the quotes are the proper characters, just like you did in Java.

PHP Operators

The same arithmetics operators are found in PHP as in Java ... +, -, *, /, %.
It supports increment and decrement, both pre and postfix notation.
It has the same comparison operators, ==, !=, <, >, <=, >=. It also has ===, which tests if the two sides have the same data type as well as values considered equal.

PHP has logical operators, !, &&, ||, `and` and `or`. The && and || do short-circuit evaluation, as in Java.

The only string operator is the dot ('.') for concatenation.

Try it!

Replace your statement block with something like...

```
$name = 'Jim';
$what = 'geek';
$level = 10;
echo 'Hi, my name is '.$name,'. and I am a level '.$level.'
'.$what;
```

This illustrates better naming practice, and the dynamic typing.

PHP Expressions

Same as Java ... the result of an expression can be assigned to a variable.
The variable takes on the data type of the expression.

Try it!

Add something like the following to your block of code from above.

```
$hoursworked = 10;
$rate = 12;
$total = $hoursworked * $rate;
echo 'You owe me '.$total;
```

The output runs together on the same line :(

PHP is meant to produce HTML, however, so we can just “echo” some to fix this. Insert another line, `echo '<br/>';` so that your output looks something like...

```
Hi, my name is Jim, and I am a level 10 geek
You owe me 120
```

PHP Selection

PHP has blocks, denoted by braces, as in Java.

PHP has similar conditional structures to Java...

```
if (logical expression) {
     block to execute if true
} else {
     block to execute if false
}

switch (name) {
 case 'Jim': $answer = 'great'; break;
 case 'George': $answer = 'unknown'; break;
 default: $answer = 'unknown';
}
```

PHP supports the ternary operator too:

```
     $answer = (some test) ? Value if true : value if false;
```

Try it!

Enhance your earlier logic

```
...
if ($hoursworked > 40) {
   $total = $hoursworked * $rate * 1.5;
} else {
   $total = $hoursworked * $rate;
}
echo ($total > 0) ? 'You owe me '.$total : “You're welcome”;
```

PHP Input

PHP is meant for a browser. This means that input has to come from the browser as well. One way is to pass parameters through the URL.

You can access such parameters through the built-in variable, $_GET, which behaves like a Java array except that the index is a string.

This looks like
http://normalreference?hours=25&rate=10&name=Henry

Try it!

Modify your browser location field to have just “?hours=25” as the query string.

We are passing the “hours” parameter.

Your output should be unchanged so far...

```
Hi, my name is Jim, and I am a level 10 geek
You owe me 120
```

Replace your hoursworked assignment..

```
$hoursworked = $_GET['hours'];
```

And your output should reflect the parameter...

```
Hi, my name is Jim, and I am a level 10 geek
You owe me 300
```

PHP Functions

PHP has globally accessible functions, unlike Java.

There are hundreds built-in to the language runtime, and we can write our own.

The reason I thought of these is because we want our program to play tic-tac-toe, and our code is going to get messy if we don't start separating some functionality!

Let's represent the state of a tic-tac-toe board by a nine character string, for instance 'xx-oo—xo', where the first three letters represent the first row, the next three the second row and the last three the third row of the board. Each character is 'x' if the user has placed a mark there, an 'o' if we have placed our mark there, and a '-' if the space is open.

We want to enhance our program so that it will detect a winner for the board passed as a parameter.

We can use the built-in `str_split` function to break the game position data into individual characters...

```
$position = $_GET['board'];
$squares = str_split($position);
```

Want more information about str_split? Where do you look?
Right – the API, which is the link at the top of this page :)

And we can then look for winning patterns (three in a row horizontally, three in a row vertically, and then the corner-to-corner possibilities). That could look like

```
$theywon = false;
if (($squares[0] == 'x') && ($squares[1] == 'x') && ($squares[2] == 'x')
{
  $theywon = true;
} else if (($squares[3] == 'x') && ($squares[4] == 'x') && ($squares[5]
== 'x') {
...
$wewon = false;
if (($squares[0] == 'o') && ($squares[1] == 'o') && ($squares[2] == 'o')
{
  $wewon = true;
} else if (($squares[3] == 'o') && ($squares[4] == 'o') && ($squares[5]
== 'o') {
...
```

You see where this is going – horribly repetitive.

Let's make a function, winner, which takes a token parameter and returns true if it can find three of that token in a row.

```
function winner($token,$position) {
  $won = false;
  if (($position[0] == $token) &&
      ($position[1] == $token) &&
      ($position[2] == $token)) {
    $won = true;
  } else if (($position[3] == $token) &&
      ($position[4] == $token) &&
      ($position[5] == $token)) {
...
  }
  return $won;
}
```

Still ugly, but better.

We can use this in our logic by including the function anywhere inside the PHP delimiters – it isn't executed until it is called. A good practice would be to separate it from the “body”, so place it at the end of your `index.php`, after the closing `html` tag.

Use it inside the body by something like...

```
if (winner('x',$squares)) echo 'You win.';
else if (winner('o',$squares)) echo 'I win.';
else echo 'No winner yet.';
```

Try it!

Make the above changes, and flesh out the winner function. You have eight combinations of token placement to test. You should get proper results. You don't need to keep any of the previous PHP code!

http://mysite/?board=xxxooo--- would choose whoever you checked first

http://mysite/?board=xoxoxo--- would not find a winner

http://mysite/?board=x--xoox-- would pick “x” as winner

http://mysite/?board=x--ooox-- would pick “o” as winner

If you want to see if a board parameter was passed in the first place, you can use the builtin `isset` function...

```
if (!isset($_GET['board'])) // no board parameter given
```

PHP Iteration

PHP has the same iteration constructs as in Java.

```
for ($i=0; $i < $somelimit; $i++)) {
  do something useful
}

while (condition) {
 do something
}

do {
  do something useful
} while (condition);
```

We can apply this to our winner function, so that we don't have such an ugly if-else construct.

Checking for a horizontal line could be done in a loop

```
for($row=0; $row<3; $row++)
   if (($position[3*$row] == $token) && ($position[3*$row+1]
== $token) && ($position[3*$row+2] == $token)) $won = true;
```

This is still ugly ... why don't we try a loop within a loop

```
for($row=0; $row<3; $row++) {
  $result = true;
  for($col=0; $col<3; $col++)
   if ($position[3*$row+$col] != $token) $result = false; //
note the negative test
}
```

The nested loop is “elegant” but wordy, and we have to do negative testing. I am inclined to have two loops, one for rows and one for columns, and then test the two diagonals. This might result in a reduction of lines of code, for this logic – decide once done.

Try It :)

The program output should be the same as in the previous step.

PHP Objects

Objects in PHP are entities with properties and functions.
They support inheritance and over-riding, but not over-loading.
The properties of an object are referenced with the "->" notation.

Java: `customer.name`
PHP: `$customer->name`

Methods are referenced similarly

Java: `customer.distance("Chicago")`
PHP: `$customer->distance('Chicago')`

PHP classes have constructors too

Java: `public customer() {...}`
PHP: `function __construct() {...}`

Objects can be made on the fly, without a formal class definition.

Example 1: `$object = new stdClass();`
Example 2: `$object->something = 1234;`

Objects can be serialized, as either arrays or JSON, and there are a multitude of functions to convert between them. The easiest way is to cast...

Examples:

```
$object_representation = (array) $object; or
$object_representation = (json) $object;
```

You can go the other way too :)

Examples:

```
$object = (object) $an_array;
$object = (object) $some_json_string;
```

If using a formal class definition, make sure it is interpreted (included, for instance) before you instantiate one of them.

Try It!

Let's make a Game class, to model a Tic-Tac-Toe board/game.
It should have one property: the board position. It should have the winner function that we wrote earlier, as a method.

Start your class definition:

```
class Game {
}
```

Add the board position property:

```
var $position;
```

Add a constructor, taking a position parameter:

```
function __construct($squares) {
  $this->position = str_split($squares);
}
```

Note: object properties are always referenced using the -> notation.

Add the winner method. It will be a copy of the winner function from earlier, except that we don't have to pass the position (it is a property already), and the position property needs to be referenced appropriately.

So, inside the class definition:

```
function winner($token) {
...
if ($this->position[...] == $token) ...
...
}
```

Finally, we can replace the earlier main logic with

```
$game = new Game($squares);
if ($game->winner('x'))
    echo 'You win. Lucky guesses!';
else if ($game->winner('o'))
    echo 'I win. Muahahahaha';
else
    echo 'No winner yet, but you are losing.';
```

and we can then delete the stand-alone winner function.

Your program should work the same as before :)

Gameplay - Presentation

You may have noticed that we have to pass the state of the board with each request. PHP is stateless, which means that no state is maintained between successive requests. Our strategy is tacky, and easily broken. We would normally store state in a file or database, but that is being left until another day.

For now, let's add some presentation, so we can show the state of the tic-tac-toe board, with suitable links to pass requested moves to our game logic. This will turn what we have into an “interactive” game, hehe.

Instead of (or in addition to) showing the winner, if any, let's actually present the current board state in a somewhat accessible fashion.

I would like to see three rows of three squares, each having a blank face or else an X or an O, depending on the board position. We can do this in a display() method in our Game class, and call it before showing any winner.

Show a square by displaying the appropriate letter in a large font.

We will start by showing an empty board, without changing any other functionality. We will then add the appropriate letter for each spot, and finally links to reflect choosing to place a token in a spot.

We should make one more change: allow for a new game, by not passing a board parameter to our page. In that case, the Game object should be initialized with a string of dashes.

Try It!

Follow along with the steps below.

The easiest way to start would be to generate an HTML table, with three rows. Here's an idea for you...

```
function display() {
  echo '<table cols=”3” style=”font-size:large; font-weight:bold”>';
  echo '<tr>'; // open the first row
  for ($pos=0; $pos<9;$pos++) {
    echo '<td>-</td>';
    if ($pos %3 == 2) echo '</tr><tr>'; // start a new row for the next
square
  }
  echo '</tr>'; // close the last row
  echo '</table>';
}
```

You should see something like

```
Welcome to George, the evil Tic-Tac-Toe Game.
-                   -                   -
-                   -                   -
-                   -                   -
```

This just shows an empty board, while we want to show the token in a spot, or else a hyphen which is a link to reflect choosing that spot.

Instead of echoing a cell with a hyphen, let's make a method to show a specific cell appropriately. The first line of the for loop above would change from

```
echo '<td>-</td>';
```

to

```
echo $this->show_cell($pos);
```

And we need to add the new method inside our Game class...

```
function show_cell($which) {
  $token = $this->position[$which];
  // deal with the easy case
  if ($token <> '-') return '<td>'.$token.'</td>';
  // now the hard case
  $this->newposition = $this->position; // copy the original
  $this->newposition[$which] = 'o'; // this would be their move
  $move = implode($this->newposition); // make a string from the board
array
  $link = '/?board='.$move; // this is what we want the link to be
  // so return a cell containing an anchor and showing a hyphen
  return '<td><a href="'.$link.'">-</a></td>';
}
```

Gameplay - Interactivity

We are almost ready to unleash our killer game to the world, and start raking in the big bucks by charging to play the game!

We need to add some logic to pick our move.

Try it!

Add a pick_move method to the game class. Invoke it before displaying the game board, if there is no winner so far.

I am leaving this one up to you!

You could play “legal” tic-tac-toe by choosing the next empty slot, but your program is likely to be crushed by four-year old users.

You could see if there is a winning move that you can make.

Or you could build some advanced AI into your move selection.

Your call.

I suggest that the main logic be something like...
has the user won? Congratulate them
if not, have we won?
If so, gloat
otherwise
pick our next move
have we won now? Gloat
display the board so they can pick a move

Your game should play legal tic-tac-toe!

There is loads more, but this is a good start.

The biggest differences, in my opinion, are the variable naming (dollar signs) and typing, as well as the array treatment.

Pay careful attention to the arrays, as that is how we will be passing parameters around, between controllers and views, for instance.

We haven't talked about arrays (properly), for-each loops, exceptions, error handling, class loaders, or the hundreds of packages built-in. Hmm – one thing different from Java: the packages contain both class definitions as well as "global" functions. You'll see!

PHP Comments

I am sad to report that PHP supports comments, just like Java.

In fact, PHP can even use Javadoc style comments (there is a tool corresponding to javadoc) and implementation comments.

Unfortunately, that means that **we will expect you to comment your code**, so that you understand what you did when you come back to something, and to give me a warm fuzzy feeling that you know what you are doing, for marks!

PHP Conventions

Use lowercase names for everything except class names.
Use lowercase names for all files.
Use underscore separated naming instead of camel case (eg `$my_name` instead of `myName`).

Huge caution: you can get away with naming convention violations on Windows, because it is sloppy with case enforcement. Unix and Linux tend to be extremely case-sensitive. Translation: playing loose with naming conventions may find your webapp breaking when you deploy it. Ohoh.

Submission

Submit a readme to your D2L dropbox, with a link to your repository.

Due three days after your lab, 17:30.
For instance, a Thursday lab group would have their submission due Sunday by 17:30.

Marking Guideline

This lab will be marked out of 10 (no specific breakdown). It is more a matter of docking marks for booboos.

Further Reading

tizag.com has a reasonable comprehensive PHP tutorial, though light on O-O.
Php.net is the official home, and has the most comprehensive documentation.
Even w3schools gets in on the act.