PHP Crash Course

tutorial #tutorial01c

James L. Parry
B.C. Institute of Technology

Tutorial Goals

This tutorial is not so much a walkthrough as a comparison between PHP and Java/C, for those unfamiliar with PHP and intimidated by the thought of learning a new programming language.

If you are familiar with PHP, skip this tutorial.

If you are unfamiliar with PHP, I suggest a quick run through the tutorial, and remember it is here for reference if you get hung up on PHP syntax or constructs later in the course!

Suggestion: you may want to skim the slideshow first, before working your way through it.

PHP Scripts

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.

Basic PHP syntax

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

Read on, and you shall see :)

PHP Variables and Data Types

Variables in PHP have conventional names, but start with a dollar sign ($name). They are variably 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 itself inside a string (\' or \").

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.

PHP supports the ternary operator too:
$answer = (some test) ? Value if true : value if false;

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

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.

Force the data typing through casting, same as in Java.

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

Functions are referenced similarly
Java: customer.distance("Chicago")
PHP: $customer->distance('Chicago')

Objects can be made on the fly, without a formal class definition.
Example 1: $object = new stdClass();
Example 2: $object->something = 1234;

PHP Object Serialization

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.

PHP Arrays

Arrays in PHP are simultaneously arrays as we think of them in Java, as well as java.util.Map kind of things.
Examples:
$grades [4] = 123;
$grades['fall'] = 123;
$grades[] = 123; // adds to the end

If using the associative array technique, the keys are strings.

Arrays do not have to hold the same data type.
Example:
$grades[4] = 123;
$grades['fall'] = 'fail';

PHP Array Declaration

Arrays can be declared ahead of time, but don't have to be...
$grades = array();

Arrays can be declared and initialized at the same time...
$grades = array(50, 40, 123, 50);
with a special syntax for key/value pairs...
$grades = array('first' => 50, 'second' => 40, 'last' => 'passed!');

PHP Resources

"Resources" in PHP, are native implementations of something, that can be referenced like a variable. Examples include streams and XML documents.

Resources cannot be serialized the same way as regular variables.
Specifically, you cannot store them inside a session :(

PHP Scope

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

Variables defined inside a block are only in scope inside that block.

PHP Selection

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 Iteration

PHP has the same iteration constructs as in Java.

```
for ($i=0; $i < size($array); $i)) {
do something useful
}

while (condition) {
do something
}

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

PHP for-each

The for-each construct has slightly different syntax..

```
foreach ($arr as $value) {
... do something with each value in the array
}

foreach ($arr as $kery => $value) {
... same thing, except that we have access to the keys too
}
```

PHP vs Java

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

The biggest differences, in my opinion, are the variable naming (dollar signs) and qualifying (->), 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 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 I 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

There are lots of conflicting naming and brace conventions in the PHP world, and lots of PHP developers adamant that "their" convention is the only right one and that everyone else is wrong.

Similar battles have raged for years with other programming languages too; PHP is not unique.

You will need to understand the class, method and source file naming conventions used by a framework, to succeed with it.

I am going to attempt to defer that as long as possible :)

PHP Case Enforcement

Huge caution:

You can get away with naming convention violations on Windows, because it is forgiving 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!

Moral of this caution: test your webapp on both Windows and Linux before considering it "done". If I get a PHP error because something can't be found on my platform, you get "fired"!

Further Reading

We have only scratched the surface of PHP ... there is tons more that can or will need to be learned, but we can defer that for the moment.

Where can you find out more? Here are a few suggestions:

- 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.

Congratulations!

You have completed tutorial #tutorial01c: PHP Crash Course

If you would take a minute to provide some feedback, we would appreciate it!

The next activity in sequence is: lab01 Development Setup 2015.01.11 17:30

You can use your browser's back button to return to the page you were on before starting this activity, or you can jump directly to the course homepage, organizer, or reference page.