





# Mashups: Remixing the Web

## Lecture 3: Server-side Data Processing in PHP



# Preparing for Today's Exercises

- You should already have a text editor and a standards-compliant browser installed.
- Install an SFTP client from <http://software.nyu.edu>
  - Windows:  WinSCP
  - Mac:  Fetch or  Fugu
- Or, for either platform, install  Dreamweaver  
<http://www.adobe.com/products/dreamweaver/>



# Reminders

- Assignment #1 out today, due in two weeks
- Use the course mailing list for assignment questions and discussion:  
[mashups-fall09@lists.nyu.edu](mailto:mashups-fall09@lists.nyu.edu)
- Office hours: Thursday 6:30PM – 8:30PM, or by appointment
- Please feel free to give me feedback on the course, in person or anonymously via the web form:  
<http://www.webremix.org/feedback.php>



# Today's Goals

- Overview and History of PHP
- Server-side Script Parsing
- Programming in PHP
  - Variables
  - Strings
  - Arrays
  - Operators
  - Functions
  - Control Structures
  - Reading and Writing Data Files
- PHP Exercises



# Background on PHP

- **PHP is a server side scripting system**
  - PHP stands for "PHP: Hypertext Preprocessor"
  - Syntax based on Perl, Java, and C
  - Very good for creating dynamic content
  - Powerful, but somewhat risky
  - If you want to focus on one system for dynamic content, this is a good one to choose



# History

- Started as a Perl hack in 1994 by Rasmus Lerdorf (to handle his resume), developed to PHP/FI 2.0
- By 1997 up to PHP 3.0 with a new parser engine by Zeev Suraski and Andi Gutmans
- Version 5.2.4 is current version, rewritten by Zend ([www.zend.com](http://www.zend.com)) to include a number of features, such as an object model
- Current is version 5
- PHP is one of the premier examples of what an open source project can be



# PHP Scripts

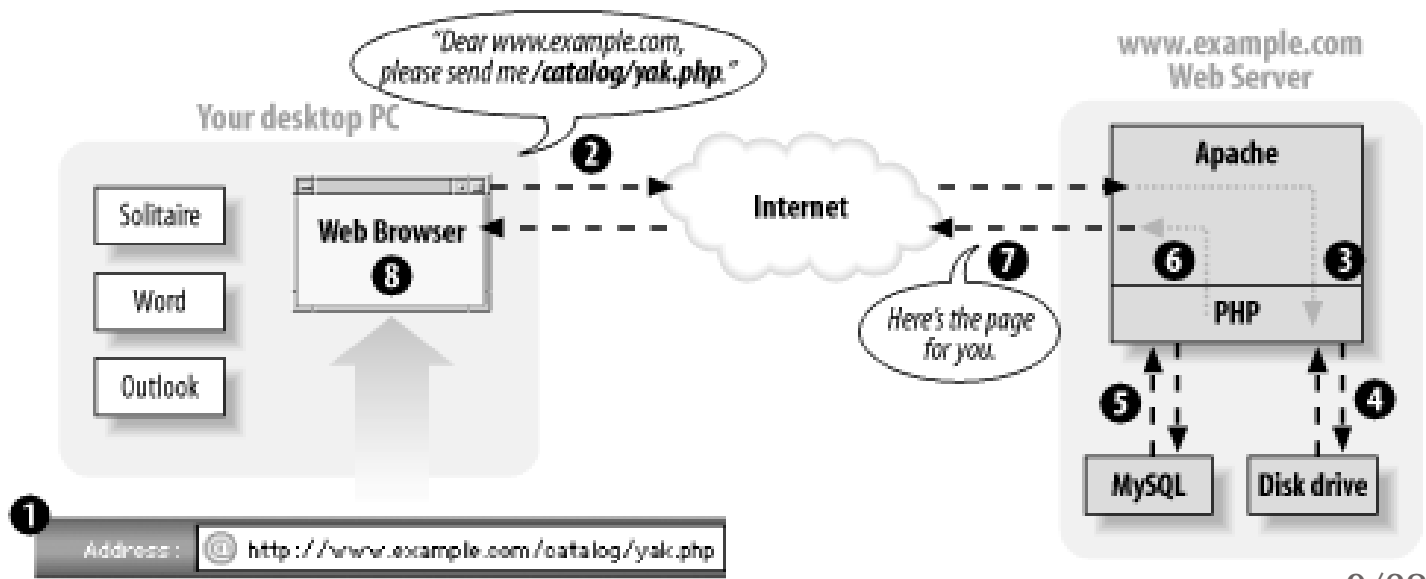
- Script files typically end in `.php` (this is set by the web server configuration).
- Sections of script are always delimited with the tags:  
`<?php ?>`
- PHP commands can make up an entire file, or can be contained in HTML.
- Program lines end in `;` or you get an error.
- The server recognizes an embedded script and executes it.
- The result is passed to the browser, so the source isn't visible.

```
<p>  
  <?php  
    $myvar = "Hello World!";  
    echo $myvar;  
  ?>  
</p>
```



# Document Parsing

- Your *browser* parses web pages as they load
- Web *servers* with server side technologies like PHP parse web pages as they are being delivered to the browser





# Two Ways of Calling PHP

- You can embed sections of PHP inside HTML:

```
<body>
<p>
  <?php
    $myvar = "Hello World!";
    echo $myvar;
  ?>
</p>
</body>
```

- Or you can print HTML tags from PHP:

```
<?php
  echo "<html><head>";
  echo "<title>Howdy</title>";
  ...
?>
```



# What do you know already?

**PHP shares similarities with multiple other languages. Much of what you may already know about JavaScript or C will apply.**

```
$name = "monzy";  
echo "Hi, my name is $name";  
echo "What will $name be in this line?";  
echo 'What will $name be in this line?';  
echo 'What's wrong with this line?';  
if ($name == "monzy") {  
    echo "got a match!";  
}
```



# PHP Formatting Basics

- Free format – white space is ignored
- Statements are terminated by semi-colon ;
- Statements grouped by curly braces { ... }
- Comments begin with // or are delimited by /\* \*/
- Assignment is '=': \$a=6
- Relational operators are < > ==
- Control structures include if, else, while, for
- Arrays are accessed with [ ]: \$x[4] is the 5th element of the array \$x (indices start at 0)
- Associative arrays (hash array in Perl, dictionary in Java) are accessed in the same way: \$y["fred"]
- Functions are called with the name followed by arguments in a fixed order enclosed in ( ): substr("fred", 0, 2)
- Case sensitive: \$fred is a different variable from \$FRED



# Variables

- Loosely typed by context
- Begin with "\$" (unlike JavaScript!)
- Typically assigned by value
  - `$foo = "Bob"; $bar = $foo;`
- Can be assigned by reference (this links variables)
  - `$bar = &$foo;`
- Some are pre-assigned, like server and environment variables:
  - `PHP_SELF`
  - `HTTP_GET_VARS`



## phpinfo()

- The `phpinfo()` function prints out a bunch of useful information about the PHP environment
- Use this to read system and server variables, settings stored in `php.ini`, versions, and modules
- This is a good first script to write



## Variable Variables

- You can use the value of a variable as the name of a second variable:

```
$a = "hello";
```

```
$$a = "world";
```

- Thus

```
echo "$a ${$a}";
```

is the same as

```
echo "$a $hello";
```

- But `$$a` echoes as "\$hello"



# Operators

- Arithmetic (+, -, \*, /, %) and String (.)
- Assignment (=) and combined assignment
  - `$a = 3;`
  - `$a += 5; // sets $a to 8;`
  - `$b = "Hello ";`
  - `$b .= "There!"; // sets $b to "Hello There!";`
- Bitwise (&, |, ^, ~, <<, >>)
  - `$a ^ $b`  
(xor: Bits that are set in \$a or \$b but not both are set.)
  - `~ $a`  
(not: Bits that are set in \$a are not set, and vice versa.)
- Comparison (==, ===, !=, !==, <, >, <=, >=)



## Coercion (Implicit Type Conversion)

- **Just like JavaScript, PHP is loosely typed**
- **Coercion occurs the same way**
- **If you concatenate a number and string, the number becomes a string**



# More Operators

- **Error Control (@)**
  - When this precedes a command, errors generated are ignored (allows custom messages)
- **Execution (` is similar to the shell\_exec() function)**
  - You can pass a string to the shell for execution:  
`$output = `ls -al`;`  
`$output = shell_exec("ls -al");`
  - **DANGER:** be careful about user set variables!
- **Incrementing/Decrementing**
  - `++$a` (Increments by one, then returns \$a.)
  - `$a++` (Returns \$a, then increments \$a by one.)
  - `--$a` (Decrements \$a by one, then returns \$a.)
  - `$a--` (Returns \$a, then decrements \$a by one.)



## Even More Operators

- Logical

`$a and $b` And True if both `$a` and `$b` are true.

`$a or $b` Or True if either `$a` or `$b` is true.

`$a xor $b` Xor True if either `$a` or `$b` is true,  
but not both.

`! $a` Not True if `$a` is not true.

`$a && $b` And True if both `$a` and `$b` are true.

`$a || $b` Or True if either `$a` or `$b` is true.

- The two ands and ors have different precedence rules:  
"and" and "or" are lower precedence than "&&" and  
"||"
- Use parentheses to resolve precedence problems or  
just to be clearer



# Control Structures

- Generally what you would expect if you are familiar with other languages:
  - `if`, `else`, `elseif`
  - `while`, `do-while`
  - `for`, `foreach`
  - `break`, `continue`, `switch`
  - `require`, `include`, `require_once`, `include_once`



# Switch

- Switch often comes in handy
- These two do the same thing:

```
if ($i == 0) {  
    echo "i equals 0";  
} elseif ($i == 1) {  
    echo "i equals 1";  
} elseif ($i == 2) {  
    echo "i equals 2";  
}
```

```
switch ($i) {  
    case 0:  
        echo "i equals 0";  
        break;  
    case 1:  
        echo "i equals 1";  
        break;  
    case 2:  
        echo "i equals 2";  
        break;  
}
```



# Nesting Files

- `require()`, `include()`, `include_once()`, and `require_once()` can bring in external files
- This lets you use the same chunk of code in a number of pages, or read other kinds of files into your program
- Be **VERY** careful of using these anywhere close to user input.



# String Handling

- String literals (constants) can be enclosed in double quotes " " or single quotes ' '
- Within " ", variables are replaced by their value (called *variable interpolation*) "My name is \$name"
- Within single quoted strings, interpolation doesn't occur

```
$age = 10;
```

```
echo 'I am $age years old'; // output: I am $age years old
```

```
echo "I am $age years old"; // output: I am 10 years old
```

- Strings are concatenated with the dot operator "key"."board" == "keyboard"
- Many standard string functions: strlen(), substr(), etc.
- Values of other types can be easily converted to and from strings – numbers are implicitly converted to strings in a string context
- Regular expressions be used for complex pattern matching



# Arrays

- You can create an array with the `array()` function, or use the `explode()` function (very useful when reading files into programs)
  - `$my_array = array(1, 2, 3, 4, 5);`
  - `$pizza = "piece1 piece2 piece3 piece4";`
  - `$pieces = explode(" ", $pizza);`
- An array is really a list of *key-value pairs*.
  - Each key can be a number or a text label
  - Each value can be a number, a string, or even another array



# Arrays

- Arrays can be multi-dimensional (lists of lists)
- Array elements can be addressed by either by number or by key name
- If you want to see the structure of an array, use the `print_r()` function to recursively print an array inside of `<pre>` tags

```
$my_test_array = array('first'=>1, 'second'=>2, 'third'=>3);  
echo "<pre>";  
print_r($my_test_array);  
echo "</pre>";
```



# Walking Arrays

- **foreach** is a handy way to walk through an array and access its keys and values
- **for** and **while** loops also work for arrays with numeric keys

```
$colors = array('red', 'blue', 'green', 'yellow');
```

```
foreach ($colors as $color) {  
    echo "Do you like $color?\n";  
}
```



# Printing Arrays

- You can't echo an array directly, but...
  - You can walk through the array and echo or `print()` line by line
  - You can use `print_r()`, which will show you the structure of complex arrays – handy for learning array structure

```
Array(  
  [1] => Array(  
    [sku] => A13412  
    [quantity] => 10  
    [item] => Whirly Widgets  
    [price] => .50  
  )  
  [2] => Array(  
    [sku] => A43214  
    [quantity] => 142  
    [item] => Widget Nuts  
    [price] => .05  
  )  
)
```



# Multidimensional Arrays

- You can make an array of arrays:

```
$multiD = array(  
  "fruits" => array(  
    "myfavorite" => "orange",  
    "yuck" => "banana",  
    "yum" => "apple"),  
  "numbers" => array(1, 2, 3, 4, 5, 6),  
  "holes" => array(  
    "first",  
    5 => "second",  
    "third")  
);
```

- You can reference individual elements by nesting:

```
echo "<p>Yes, we have no " . $multiD["fruits"]["yuck"] . "</p>"
```



# Getting Data into Arrays

- You can directly read data into individual array slots via a direct assignment:  
`$pieces[5] = "pepperoni";`
- You can load a file into an array:
  - Use the `file()` function to read a delimited file (the delimiter can be any unique character):  
`$pizza = file("./our_pizzas.txt")`
  - Use `explode()` to create an array from a line:  
`$pieces = explode(" ", $pizza);`



# The Power of PHP

- The power of PHP lies partly in the wealth of available functions – for example, the 40+ array functions:
  - `array_flip()` swaps keys for values
  - `array_count_values()` returns an associative array of all values in an array, and their frequency
  - `array_rand()` pulls a random element
  - `array_unique()` removes duplicate elements
  - `array_walk()` applies a user defined function to each element of an array
  - `count()` returns the number of elements in an array
  - `array_search()` returns the key for the first match in an array



## Using External Data

- PHP scripts make websites dynamic...
- But where PHP shines is in building pages out of external data sources, so that the web pages change when the data does
- Most of the time, people think of a database like MySQL as the backend, but you can also use text or data files, or query an external API



## Standard Data Files

- Comma- or tab-delimited files are easy to parse, but you can use anything as a delimiter
- Files are usually read as arrays, one line per slot
- Remember that each line ends in `\n`, so you should clean this up, and be careful about white space
- Once the file is read, you can use `explode()` to break the lines into fields, one at a time, in a loop



# Standard Data Files

- You can use `trim()` to clean white space and carriage returns instead of `str_replace()`
- Notice that this is building an array of arrays

```
$items=file("./mydata.txt");
foreach ($items as $line) {
    $line = str_replace("\n", "", $line);
    $line = explode("\t", $line);
    // do something with $line array
}
```



# Useful string functions

- `str_replace()`
- `trim()`, `ltrim()`, `rtrim()`
- `implode()`, `explode()`
- `addslashes()`, `stripslashes()`
- `htmlentities()`, `html_entity_decode()`,  
`htmlspecialchars()`
- `striptags()`



# Alternative syntax

- Applies to `if`, `while`, `for`, `foreach`, and `switch`
- Change the opening brace to a colon
- Change the closing brace to an end statement

```
<?php if ($x == 5): ?>
    x is equal to 5.
<?php else: ?>
    x is not equal to 5.
<?php endif; ?>
```

```
<?php
if ($x == 5):
    echo "x is equal to 5.";
else:
    echo "x is not equal to 5.";
endif;
?>
```



## A Note on Forms

- Forms are parts of an HTML document that users can fill in (buttons, checkboxes, text areas, etc.)
- Forms are a block-level element denoted by the `<form>` tag, with all form inputs as children
- Form fields are submitted to PHP in the form of variables. Each control in the HTML form becomes an array element in PHP.
- `<form>` has a `method=` attribute. This attribute determines the HTTP method by which the form is submitted to the script. There are two choices:
  - `method="get"`
  - `method="post"`
- When the form is submitted, the http request line that follows will have the method GET or POST.



# GET versus POST

- GET: form data is transmitted by appending it to the URL of the script  
<http://www.google.com/search?q=llamas>
  - You can bookmark the form parameters
  - There is a limit of 1024 chars for the URL, so only limited information can be transmitted
  - In PHP, accessed using predefined `$_GET` array  
`$_GET["q"] == "llamas "`
- POST: user agent sends the form as a POST message to the server, in body of HTTP request
  - Form request can't be bookmarked
  - No size limit
  - In PHP, accessed using predefined `$_POST` array



# Input Attributes

- The `type=` attribute of an `<input/>` defines input type, for example:
  - 'text'            textbox
  - 'password'        textbox with masked input
  - 'checkbox'           checkbox
  - 'radio'           radio button
  - 'submit'          button to submit form
- The `name=` attribute gives the input a unique identifier, so it can be accessed from a script

```
<input type="text" name="city" />
```



```
<?php echo $_GET["city"]; ?>
```



# PHP Activity

- **Download the lab files, and we'll start experimenting!**