Server Side Languages

Web Design & Development

Day 1

About this Class

The Server-side Languages Course will examine the benefits of a server-side scripting language to heighten human computer interaction with web content. In this class students will learn how to take their existing knowledge of static-based web content and implement a server-side scripting language to develop a more robust web application. By implementing server-side languages, students will be able to deploy dynamic content to further the level of interaction between client and server communication.

What You Should Already Know

- **HTML/CSS** - DOM, Forms, Styling, Interface Design, Static Pages
- **Programming Basics** - JavaScript variables, functions, arrays, conditionals, loops, instantiating and using objects
- **Web Server/Client Relationship** - HTTP Requests/Responses, HTTP Headers, Basic IP/DNS, Web Servers
- **Web Design Patterns** - MVC

- **Optional**
  - Client-Side Frameworks (Bootstrap, Foundation, 960, etc.)
  - AJAX and JQuery Library
  - Python Programming Basics

Class Introductions

- Your Name
- Do You Prefer Client-side, Server-side, Design
- Your Experience

About Your Instructor

Previous Classes at Full Sail (SSL, ASL, WPF)
Bachelors in Computer Science
Masters in Cyber Crime Counter Terrorism
Technical Publications and Classes: DIgital Forensics and Cybercrime

Languages: PHP, Classic ASP, ASP.Net, Flex/ActionScript, JavaScript, ColdFusion, Python, VB

Contact Information

Refer to FSO for contact information.

Course Outline

- Server-Side Architecture Overview, HTTP, REST
- PHP - Intro, Features, Sessions, Database and DB abstraction, Objects, APIs.
- In-Class Team Challenge
- Python/Flask - Intro, Database abstraction, Templating, URL Routing,
- Student Topic Presentations
- Comprehensive In-Class Final Exam

Final Grade Breakdown

25% **Final Practical Exam**
35% Labs
10% Lab Screencasts
10% **Team Challenge**
10% Student Topic Presentations
10% Professionalism

Attendance & Class Requirements

Be On Time!

Up to 15 minutes late, Tardy.

After 15 minutes, 2 hours absent.

More than 8 hours unexcused absences - attendance fail & retake.

If you will be late or absent (excused or unexcused) email me prior to absence.

It's a professional common courtesy, can lose 10 GPS points for no call/no show.

No Food or Drinks Allowed in Class or Lab.

Labs are due at 11:59pm the date due. Late work will receive 25% per day off, after 3 days not accepted. I do not like late work.

Do not wait until the last minute to start or ask questions.

Student Topic Presentations

Most lecture meetings will have a few presentations.
Each student will present 3 topics to the class.
Give a 3-5 minute presentation on your topic.

Presentation Format:

Define and Describe the topic.
Show Examples.
Be prepared to answer questions.

Topic : You research and choose topic (with instructor approval).

Lecture & Lab Format

- Come Prepared with your Laptop.
- Participate in exercises during Lecture.
- Integrate knowledge in Lab exercises.
- Submit Screencast and Outline with Lab.
- You will have opportunity to ask questions in following class before submitting lab.
- **Successful Students:**
  - Ask Questions - Lecture and Lab. We are here to help.
  - Pair up with other students, work together (don’t copy work, but help each other)
  - Break code functionality into pieces (isolate), test, then integrate.

Questions?

Lecture Overview

- Review Web Server Architecture
- HTTP request (GET, POST)
- HTTP response, headers, status codes, redirections, errors
- MIME and multipart/data responses
- PHP Language environment and output – test development environment (MAMP, Sequel Pro)
- PHP Language basics – variables, scope, arrays, functions, loops, conditionals, array sorting, includes, superglobals, type juggling
- Forms

Some server side scripting languages

- Perl
- Ruby
- Python
- ASP
- ColdFusion
- PHP
- JavaScript (using Node.js)

Perl

- Perl is a high-level, general-purpose, interpreted, dynamic programming language.
- Originally developed in 1987 by Larry Wall as a general purpose Unix scripting language.
- Larry Wall continues to oversee development of the core language, and its newest version, Perl 6.

ColdFusion

- ColdFusion Markup Language (CFML) used by Adobe ColdFusion
- CFML enhances standard HTML files with database commands, conditional operators, high-level formatting functions, and other elements to rapidly produce easy-to-maintain web applications.
- The pages in a ColdFusion application include the server-side CFML tags in addition to HTML (XHTML) tags.

Ruby

- Ruby is a dynamic, reflective, general-purpose object-oriented programming language that combines syntax inspired by Perl.
- Originated in Japan during the mid-1990s and was initially developed and designed by Yukihiro "Matz" Matsumoto.
- Supports multiple programming paradigms, including functional, object oriented, imperative and reflection.
- Has a dynamic type system and automatic memory management.
- The standard 1.8.6 (stable) implementation is written in C, as a single-pass interpreted language.
- There is currently no specification of the Ruby language, so the original implementation is considered to be the de facto reference.

Python

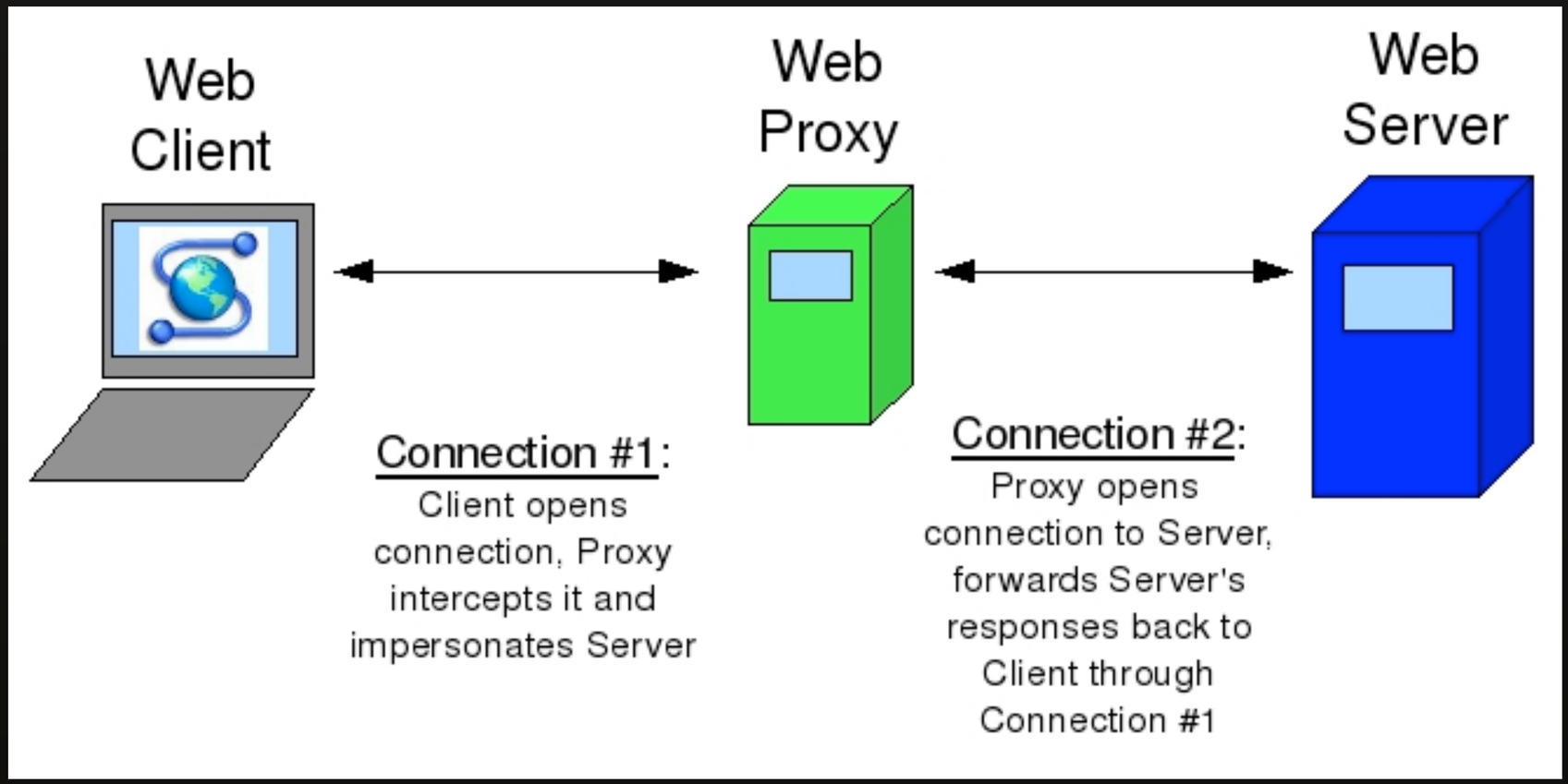
- Python is a general-purpose, high-level programming language.
- Its core syntax and semantics are minimalistic, while the standard library is large and comprehensive.
- Use of whitespace as block delimiters is unusual among popular programming languages.
- Supports multiple programming paradigm (primarily object oriented, imperative, and functional) and features a fully dynamic type system and automatic memory management, similar to Perl, Ruby.
- First released by Guido van Rossum in 1991.
- The language has an open, community-based development model managed by the non-profit Python Software Foundation.

PHP

- PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.
- PHP is a recursive acronym that stands for: PHP: Hypertext Preprocessor
- It began in 1994 as a set of Common Gateway Interface binaries written in the C programming language by the Danish/Greenlandic programmer Rasmus Lerdorf.
- On May 22, 2000, PHP 4, powered by the Zend Engine 1.0, was released. On July 13, 2004, PHP 5 was released, powered by the new Zend Engine II.

Web Server Architecture

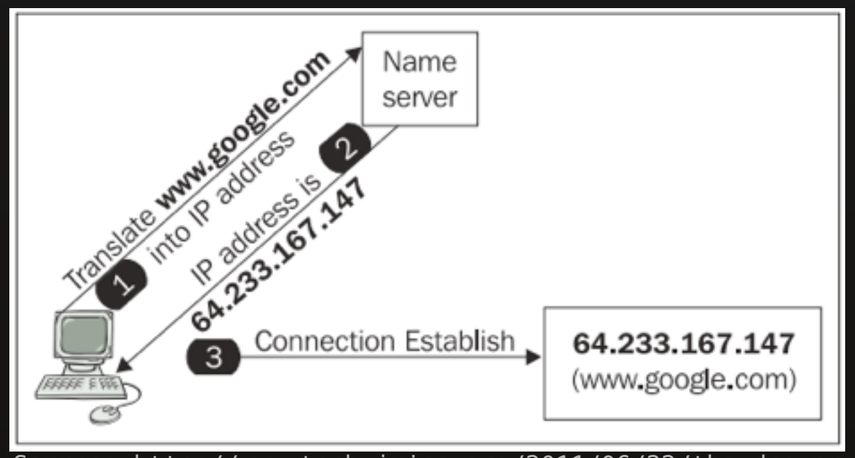
- Browser makes request.
- Could connect directly to web server, or via proxy.



Source: http://www.linuxjournal.com/magazine/paranoid-penguin-building-secure-squid-web-proxy-part-i

DNS and Host Headers

DNS Lookup



Source: http://www.tech-juice.org/2011/06/22/the-dns-protocol-explained/

Apache Config

```
<VirtualHost *:80>
ServerName www.domain.tld
ServerAlias domain.tld *.domain.tld
DocumentRoot /www/domain
</VirtualHost>

<VirtualHost *:80>
ServerName www.otherdomain.tld
DocumentRoot /www/otherdomain
</VirtualHost>
```

HTTP Headers

Request URL: https://www.google.com/search?hl=en&site=imghp&tbm=isc…
Request method: GET
Status code: 200 OK

Filter headers

Response headers (0.285 KB)

Alternate-Protocol: "443:quic"
Cache-Control: "private, max-age=0"
Content-Encoding: "gzip"
Content-Type: "text/html; charset=UTF-8"
Date: "Thu, 06 Feb 2014 21:32:45 GMT"
Expires: "-1"
Server: "gws"
X-Firefox-Spdy: "3"
X-Frame-Options: "SAMEORIGIN"
X-XSS-Protection: "1; mode=block"

Request headers (0.789 KB)

Host: "www.google.com"
User-Agent: "Mozilla/5.0 (Macintosh; Intel…) Gecko/20100101 Firefox/24.0"
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
Accept-Language: "en-US,en;q=0.5"
Accept-Encoding: "gzip, deflate"
Cookie: "PREF=ID=5bde9da75f1f156a:U…kYVoomZ5EwZdp_As69OPQKvCkw"
Connection: "keep-alive"
Cache-Control: "max-age=0"

HTTP Request/Response

http://mysite.com

Browser uses GET verb (GET /)

Server would respond back:

Header: Status 200

Data: <html><body><h1>Hello World</h1></body></html>

Or If page has moved (permanent redirect):

Header: Status 301

Location: http://mysite.com/newpage.html

Then, browser would have to request newpage.html in a **new** request.

Google

Full Sail Uni... | FastCGI · fa... | Object-Orie... | List of availa... | Backbone.js | j4mie/idiorm | Keynote styl... | Networks an... | Google

https://www.google.com

Google

FSO | FSO Kit | FSO KIT | Student FSO | GoTo Meeting | Connect | GitHub | CVue | IT Helpdesk | Staff 411 | WDDBS Dept | WDDBS Schedule | Faculty Dev | HRHQ

Google

Google Search | I'm Feeling Lucky

Advertising | Business | About | Privacy & Terms | Settings

Console | Inspector | Debugger | Style Editor | Profiler | Network

Method	File	Domain	Type	Size	0 ms	160 ms	320 ms	480 ms	640 ms	800 ms	960 ms	1120 ms
GET	/	www.google.com	html	132.87 KB	→ 375 ms							
GET	chrome-48.png	www.google.com	png	2.39 KB				→ 46 ms				
GET	logo11w.png	www.google.com	png	18.25 KB				→ 66 ms				
GET	v1_b444d4f7.png	ssl.gstatic.com	png	69.58 KB				→ 42 ms				
GET	rs=AltRSTNldn9EkplKQfSkyJq_gvysXm57sA	www.google.com	js	366.95 KB				→ 45 ms				
GET	29b645ff2dff0928.js?bav=on.2,or.r_qf.	www.google.com	js	66.72 KB					→ 94 ms			
GET	rs=AltRSTNldn9EkplKQfSkyJq_gvysXm57sA	www.google.com	js	382.86 KB					→ 48 ms			
GET	tia.png	www.google.com	png	0.50 KB					→ 51 ms			
GET	rs=AltRSTN3KHDZaT8PCaVk59gxEjUWC3npFg	www.gstatic.com	js	141.92 KB					→ 60 ms			
GET	cb=gapi.loaded_0	apis.google.com	js	132.26 KB							→ 41 ms	
GET	nav_logo170.png	www.google.com	png	21.11 KB								→ 64 ms
GET	gen_204?v=3&s=webhp&action=&e=4006,1...	www.google.com	html	0 KB								→ 80 ms

All | HTML | CSS | JS | XHR | Fonts | Images | Media | Flash

12 requests, 1335.46 KB, 1.27 s

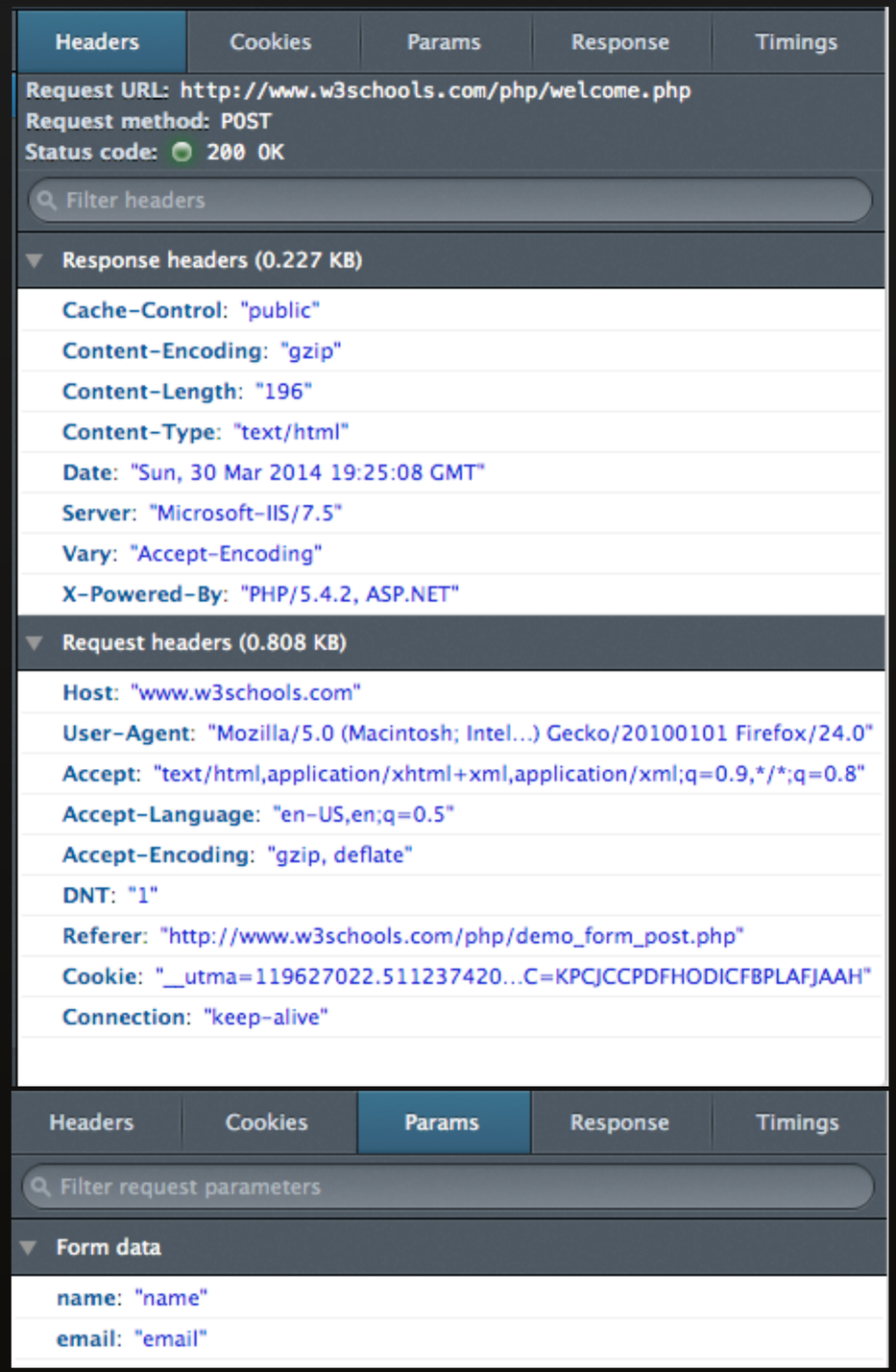
Use POST Verb

Example of HTML Form posting back to server.

```
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
```

Server responded back with:

```
<!DOCTYPE HTML>
<html><body>
Welcome name<br>
Your email address is: email
</body></html>
```



Multipart POST

Used for submitting information from the browser to web server in multiple, delimited parts.

Most often used for file uploads.

In the POST data that is sent back to server, files are Base64 encoded and sent in multiple parts.

- Base64 translates a binary file into its hexadecimal (0-9, A-F) equivalent that can be unencoded once received at the remote end.

REST (Representational State Transfer)

- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer XML, JavaScript Object Notation (JSON), or both.
- Can easily traverse forward and reverse proxy servers and firewalls
  - Because using HTTP explicitly - HTTP port 80 or HTTPS port 443

- Read More: http://www.ibm.com/developerworks/webservices/library/ws-restful/

HTTP Verbs (RESTful)

- One of the key characteristics of a RESTful Web service is the explicit use of HTTP methods in a way that follows the protocol as defined by RFC 2616.
- REST asks developers to use HTTP methods explicitly and in a way that's consistent with the protocol definition. This basic REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods.
- According to this mapping:
  - To **create** a resource on the server, use POST.
  - To **retrieve** a resource, use GET.
  - To **update** the state of a resource or to change it, use PUT.
  - To **delete** or remove a resource, use DELETE.

Example Server Request using REST

Before Implementing REST

GET /adduser?name=Robert HTTP/1.1

RESTful Implementation

POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml

```
<?xml version="1.0"?>
<user>
  <name>Robert</name>
</user>
```

MAMP Development Environment

- Download latest version of MAMP (mamp.info) 3.x and install.
- Create a class directory for SSL, and Day1 subdirectory.
- Open MAMP, under preferences set your Apache document root to the Day1 folder.
- Under Preferences, PHP, select 5.5.x version and set cache to “off”
- Under Preferences, Ports, set Apache to 8888 and MySQL to 8889
- Create a file named “phpinfo.php” under the Day1 folder, containing:
  - `<?php phpinfo(); ?>`
- Open browser to http://localhost:8888/phpinfo.php

Turn on PHP Errors (for debugging)

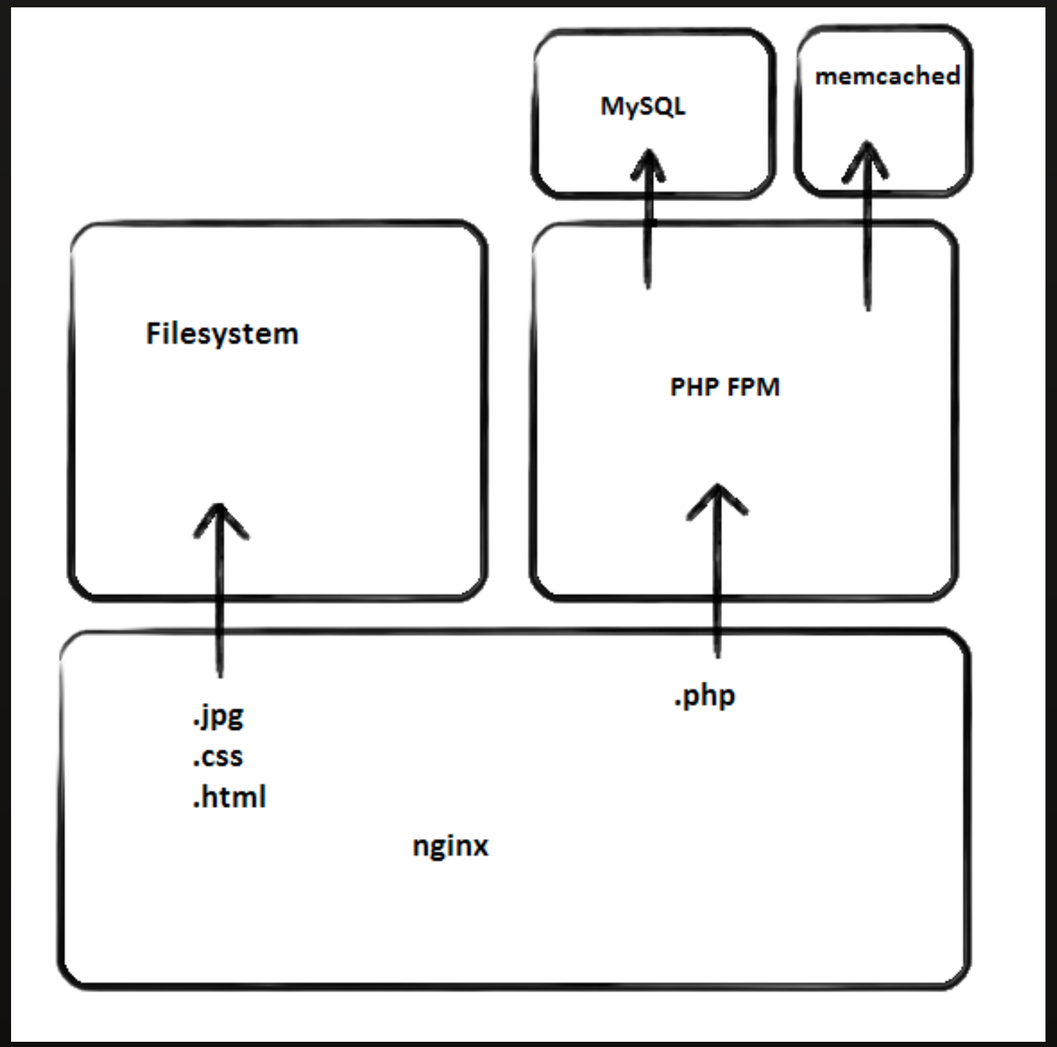
- It is useful to turn on PHP errors when debugging applications.
- Open Finder, and browse to Applications/MAMP/bin/php/php5.x.x/conf/ and open php.ini in your text editor.
  - php5.x.x will be the version number you set in MAMP preferences
- Search for display_errors and make sure it is set to On
- Restart MAMP so the changes take effect

How do PHP and Web Server work together?

- Web server (Apache, nginx, etc.) accept URL request, pass off to PHP interpreter for file extensions ending in .php
  - (see httpd.conf for Apache configuration, addtype application)

- PHP “engine” interprets and generates response
- PHP modules - Applications/MAMP/Library/Modules/*.so
- php.ini - specifies global configuration options. Some options can be overridden locally in scripts at runtime.

How PHP Integrates with Web Servers



Source: http://www.alwaysgetbetter.com/blog/2012/01/30/setting-wordpress-nginx-fastcgi/

PHP Syntax

- Files must end in .php
- Go into PHP Mode with: <?php
- Get out of PHP Mode with: ?>
- Every time you give a command, end it with a semicolon **;**
- Comment: // Single line -or- /* multiline */
- Anything not within <?php ?> tags will be output as literal, including whitespace. Will auto send HTTP Response headers for all literal.

```
$name = “Bob”;
$age = 29;
// echo sends information to the HTTP output, to browser
echo “I am $name and I am $age years old.”;
I have <?=$foo?> foo. //Shortcut Syntax
```

PHP Variables

- Types are not explicitly declared, PHP will juggle type to cast
- Letters, words, blocks of text (String)
- True/False values (Boolean)
- Numbers (Integer)
- Numbers with decimals (Floating Point)
- Other types: Array, Object, Null
- All variables begin with a **$**, assign values to with **=**
  - $name = ‘Bob’;
- Can begin with letters or the _underscore
- CANNOT begin with a number. Some $_ are reserved.
- Variables ARE case sensitive

PHP Variable Scope

- All variables outside of functions or classes are global.
  - As a rule of thumb, anything not inside { }
- Variables inside functions or classes are local within that context.
- Variables can be Persistent across page requests from same browser session. Most are non-persistent.

	Non-Persistent	Persistent
Local Variables	Variables Attributes Caller Arguments This	(none)
Global Variables	Request CGI Form URL	Server Application Session Client Cookie

PHP Variable Types

```
$myinteger = 1; //integer type
$myfloat = 1.0; //floating point type
$mystring = ‘1’; //string containing 1, NOT numeric 1
$mystring = “1”; //double quotes can be used for string too
$mybool = true; //stores true or false booleans
```

Null vs. Empty vs. Set/Unset

$mystring = null; //sets the variable, but null is stored
$mystring = ‘’; //sets the variable to empty string, not same as null
unset($mystring); //variable does not exist anymore

In the first example, variable exists but stores nothing (null).
In the second, variable exists with string type and length of 0.

If we were to reference $mystring and it doesn’t exist, PHP would throw an error at runtime.
<?php echo $mystring; ?> //would throw an error for third example.

PHP String Quotes

```
$name = ‘John’;
$age = 21;
//Double quotes considered complex variables will be replaced
echo “I am $name and I am $age.”;

//Single quotes considered literal variables will not be replaced
echo ‘I am $name and I am $age.’;

//Concatenate strings and variables using dot
echo ‘I am ‘ . $name . ‘ and I am ‘ . $age;
```

Escape Sequences

```
echo “<p class=\“name\”>$name</p>”; //backslash esc double-quote
echo “\t<p>Tabs work too!</p>”; //tab in source code
echo “<p>Newlines are Pretty</p>\n”; //new line in source code
```

Exercise 1.1: Variables and Strings

Create a test file in your Day1 folder, exercise1.1.php
Set a variable for your name and age.
Output to browser: My name is *name* and age is *age*.
1. Echo out the result using double quotes.
2. Echo out the same result using single quotes.
3. Set the age variable to null and echo out age.
4. unset() your name variable, and echo out name.

Indexed Arrays

Arrays can be indexed (0, 1, 2,...)

```
$beatles=array('John', 'Paul', 'Ringo', 'George');
var_dump($beatles);
Array
(
    [0] => John
    [1] => Paul
    [2] => Ringo
    [3] => George
)
echo $beatles[0]; //outputs John
```

Associative Arrays (Key/Value Pairs)

Arrays can be referenced by keys

```
$beatles=array('keyboard' => 'John', 'vocal' => 'Paul', 'drums' =>'Ringo',
'guitar' => 'George');
var_dump($beatles);
Array
(
    [keyboard] => John
    [vocal] => Paul
    [drums] => Ringo
    [guitar] => George
)
echo $beatles['keyboard']; //outputs John
```

Multidimensional Arrays

Arrays can have multiple dimensions, and can mix types

```
$foods=array(
‘fruits’=>array(‘apple’, ‘orange’, ‘pear’),
‘vegetables’=>array(‘carrot’, ‘pea’, ‘corn’)
);
```

```
echo $foods[‘fruits’][1]; //outputs orange
```

Array Functions

Adding to Arrays

```
array_push($foods['fruits'], 'pineapple'); //add to end of array
$foods['vegetables'][3]='potato'; //access array index directly
```

Counting Arrays

```
echo count($foods['fruits']); //returns 3, previous page example
echo count($foods, COUNT_RECURSIVE); //returns 8, previous page example
```

Sorting Arrays

```
sort($foods['fruits']); //sorts dimension in alpha order, index reorder
```

See http://us3.php.net/manual/en/array.sorting.php for many array functions.

Note: When storing database results in an array, it is best to have SQL order recordset (ORDER BY) instead of using array sort.

Exercise 1.2: Using Arrays

Based on Exercise 1.1, use one variable as a key/value pair array. Set one key named *name* and another key named *age*.

1. Output "My name is *name* and age is *age*." using the array keys.
2. Output the same, but use index numbers instead.

User-Defined Functions

```
<?php
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
?>
```

Returning Values

```
<?php
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
?>
```

http://us1.php.net/manual/en/functions.arguments.php

Comparison Operators

http://us1.php.net/manual/en/language.operators.comparison.php

Use to compare variables - strings, integers, boolean, etc.

If you compare a number with a string or the comparison involves numerical strings, then each string is converted to a number and the comparison performed numerically.

Example	Name	Result
$a == $b	Equal	TRUE if *$a* is equal to *$b* after type juggling.
$a === $b	Identical	TRUE if *$a* is equal to *$b*, and they are of the same type.
$a != $b	Not equal	TRUE if *$a* is not equal to *$b* after type juggling.
$a <> $b	Not equal	TRUE if *$a* is not equal to *$b* after type juggling.
$a !== $b	Not identical	TRUE if *$a* is not equal to *$b*, or they are not of the same type.
$a < $b	Less than	TRUE if *$a* is strictly less than *$b*.
$a > $b	Greater than	TRUE if *$a* is strictly greater than *$b*.
$a <= $b	Less than or equal to	TRUE if *$a* is less than or equal to *$b*.
$a >= $b	Greater than or equal to	TRUE if *$a* is greater than or equal to *$b*.

Control Structures

http://us1.php.net/manual/en/control-structures.elseif.php

If/Else

```
<?php
if ($a > $b) {
  echo "a is greater than b";
} else {
  echo "a is NOT greater than b";
}
?>
```

Control Structures

If/Elseif/Else

elseif, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE.

```
<?php
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
?>
```

Exercise 1.3: Conditionals

Assign letter grades based on points earned. Using **if/else/elseif** statements, create a **function** that returns a letter grade based on the following point breakdowns:

100-90=A, 80-89=B, 79-70=C, 69-60=D, <60=F

To test your function, try it with these 5 point values and echo the result back out from the value returned by the function:

1. 94
2. 54
3. 89.9
4. 60.01
5. 102.1

Switch Statement

The switch statement is similar to a series of IF statements on the same expression. A special case is the default case that matches anything that wasn't matched by the other cases.

```
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;
    default:
       echo "i is not equal to 0, 1 or 2";
}
```

While Loops

A while statement tells PHP to execute the nested statement(s) repeatedly, as long as the while expression evaluates to TRUE. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration). Sometimes, if the while expression evaluates to FALSE from the very beginning, the nested statement(s) won't even be run once.

```
$i = 1;
while ($i <= 10) {
    echo $i
 $i++;
}
```

Do-While Loops

do-while loops are very similar to while loops, except the truth expression is checked at the end of each iteration instead of in the beginning. The main difference from regular while loops is that the first iteration of a do-while loop is guaranteed to run (the truth expression is only checked at the end of the iteration), whereas it may not necessarily run with a regular while loop (the truth expression is checked at the beginning of each iteration, if it evaluates to FALSE right from the beginning, the loop execution would end immediately).

```
$i = 0;
do {
    echo $i;
} while ($i > 0);
```

For Loops

For loops are the most complex loops in PHP.

```
for (expr1; expr2; expr3)
    statement
```

The first expression (expr1) is evaluated (executed) once unconditionally at the beginning of the loop.

In the beginning of each iteration, expr2 is evaluated. If it evaluates to TRUE, the loop continues and the nested statement(s) are executed. If it evaluates to FALSE, the execution of the loop ends.

At the end of each iteration, expr3 is evaluated (executed).

```
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
```

Exercise 1.4: Loops

Part 1: Create an array indexed by integers. Create 5 solid color values for the *even* numbers (starting at 0), then a shade of that color for the successive *odd* number.

E.g., [0] => “Red”, [1] => “Pink”, [2] => “Blue”, [3] => “Baby Blue”

Loop through the colors of the array and display the index number and color name. E.g., “Color 1 is Red”

You can do this a number of ways - for this exercise there is no right or wrong expectation to approach the problem as long as it displays the colors in order and uses a form of loop.

Part 2: Repeat the same above, but display the colors in reverse order.

Part 3: Repeat Part 1 above, but only display the solid colors

ForEach Construct

The foreach construct provides an easy way to iterate over arrays. foreach works only on arrays and objects, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable.

```
$arr = array(
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach ($arr as $key => $value) {
    echo "\$arr[$key] => $value.\n";
}
//Will output $arr[one]=>1 ...
```

Superglobal Variables

Several predefined variables in PHP are "superglobals", which means they are available in all scopes throughout a script. There is no need to do global $variable; to access them within functions or methods.

$_GET - An associative array of variables passed to the current script via the URL parameters.

```
<form method=”GET” action=”myscript.php”>
<input type=”text” name=”name” />
</form>
echo 'Hello ' . $_GET["name"] . '!';
```

Assuming the user entered http://example.com/?name=John
The above example will output something similar to:
Hello John!

Superglobal Variables

$_POST - An associative array of variables passed via the HTTP POST method.

```
//form.html
<form method=”POST” action=”myscript.php”>
<input type=”text” name=”myname” />
<input type=”submit” />
</form>
```

```
//myscript.php
<?php
  echo 'Hello ' . $_POST["myname"] . '!';
?>
```

Returns “Hello John!” to web browser.

MVC Setup

Follow along with your instructor and create your own server side framework.

- Models
- Views
- Controllers

Questions?

Lab 1: PHP Functions and Variables

Refer to FSO Lab1 and Screencast 1