中山大學
SUN YAT-SEN UNIVERSITY

Lecture 6
Basic PHP for Server Side Programming

SE-805 Web 2.0 Programming

(http://my.ss.sysu.edu.cn/wiki/display/W2PSC/Home , **supported by Google;** *using some slides of & inspired by Marty Stepp's CSE 190 M courseware*)

School of Software, Sun Yat-sen University

Outline

- **Server-Side Basics**
- Introduction to PHP
- PHP Basic Syntax

URLs and Web Servers

http://my.ss.sysu.edu.cn:8080/display/W2PSC

~~~ ~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~

protocol host path

- usually when you type a URL in your browser:
  - your computer looks up the server's IP address using DNS
  - your browser connects to that IP address and requests the given file
  - the web server software (e.g. Apache) grabs that file from the server's local file system, and sends back its contents to you
- some URLs actually specify *programs* that the web server should run, and then send their output back to you as the result: http://php.net/manual/en/function.sqrt.php
  - the above URL tells the server php.net to run the program manual/en/function.sqrt.php and send back its output
~~~

Dynamic Vs. Static

- Static Page
  - Client/Consumer's Viewpoint: an url refers to an identical html file
  - Server/Producer's Viewpoint: a file stored within or sub-within the root folder of a Web Server
  - it is a html …
  - Can be display directly at a browser
- Dynamic Page
  - Client/Consumer's Viewpoint: an url refers to a dynamic html (may be vary each time requested)
  - Server/Producer's Viewpoint: a program/script produces html
  - it is **NOT a html**, but a program producing html(s)
  - Can't be display directly at a browser
- Dynamic Web Page, Dynamic HTML (DHTML), what's the difference?

Server-Side Web Programming

- server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl
- the web server contains software that allows it to run those programs and send back their output as responses to web requests
- each language/framework has its pros and cons
  - we use PHP for server-side programming in this textbook

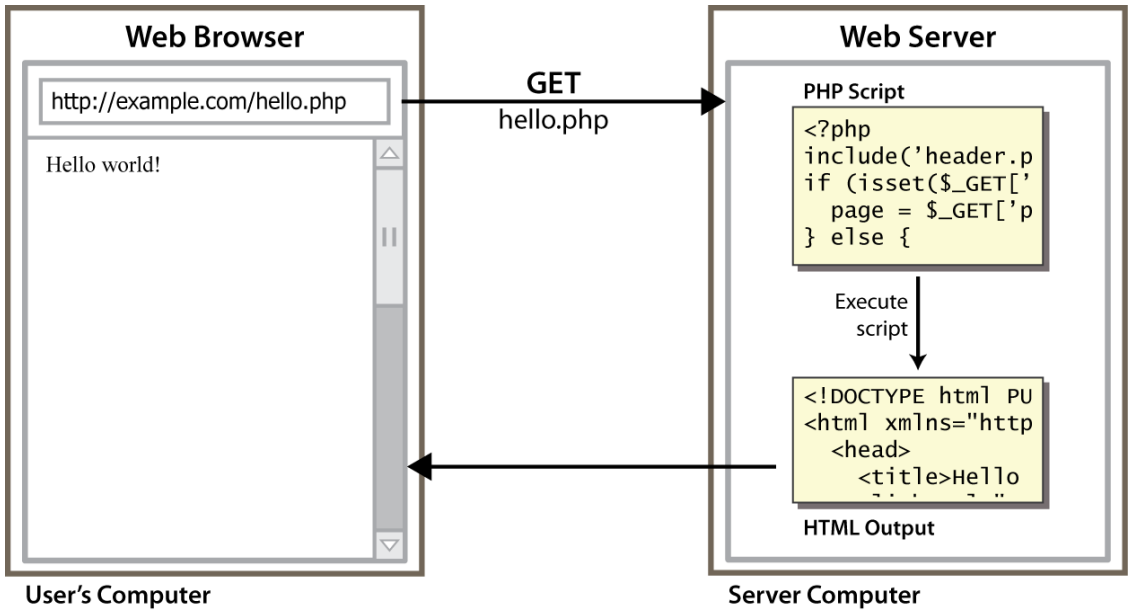
Outline

- Server-Side Basics
- **Introduction to PHP**
- PHP Basic Syntax

What is PHP?

- PHP stands for "PHP Hypertext Preprocessor"
- a server-side scripting language
- used to make web pages dynamic:
  - provide different content depending on context
  - interface with other services: database, e-mail, etc
  - authenticate users
  - process form information
- PHP code can be embedded in XHTML code

Lifecyle of PHP Web request



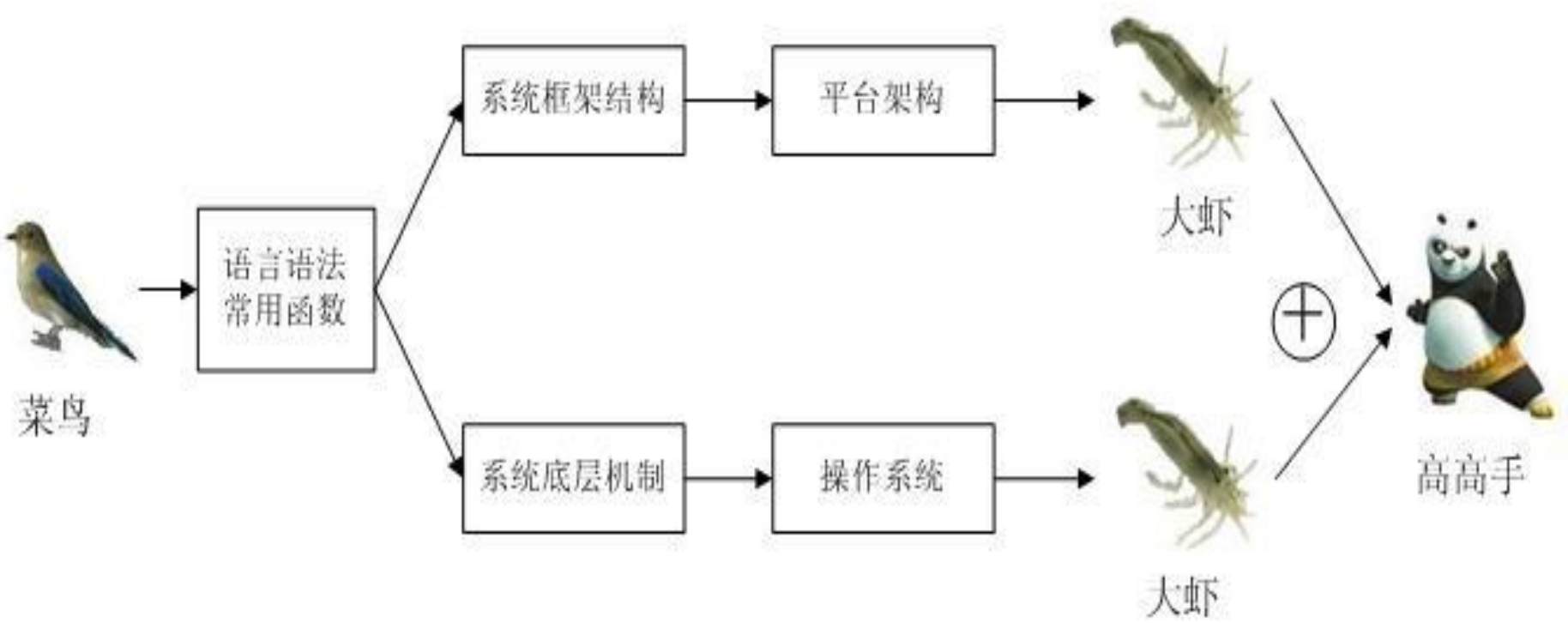
- browser requests a .html file (**static content**): server just sends that file
- browser requests a .php file (**dynamic content**): server reads it, runs any script code inside it, then sends result across the network
  - script produces output that becomes the response sent back

Why PHP?

- There are many other options for server-side languages: Ruby on Rails, JSP, ASP.NET, etc. Why choose PHP?

- **free and open source**: anyone can run a PHP-enabled server free of charge
- **compatible**: supported by most popular web servers
- **simple**: lots of built-in functionality; familiar syntax
- **available**: installed on our servers and most commercial web hosts

How to learn a Programming Language



Hello, World!

- The following contents could go into a file hello.php:

```php
<?php
print "Hello, world!";
?>
```

PHP

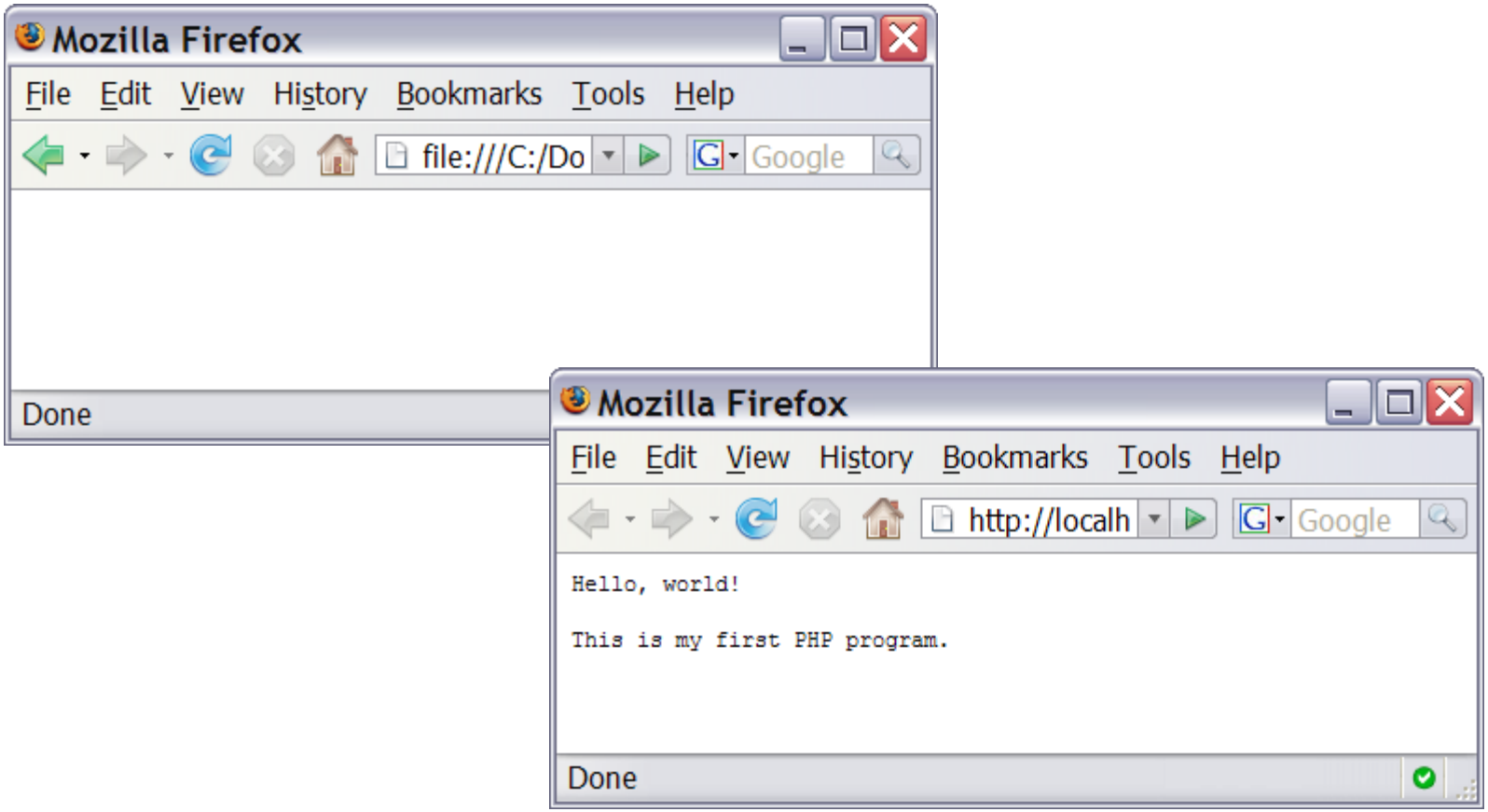
```
Hello, world!
```

output

- a block or file of PHP code begins with **<?php** and ends with **?>**
- PHP statements, function declarations, etc. appear between these endpoints

Viewing PHP output

- **Your PHP code must be run/executed first, before it reaches a browser!**



Outline

- Server-Side Basics
- Introduction to PHP
- **PHP Basic Syntax**

Comments

```php
# single-line comment

// single-line comment

/*
multi-line comment
*/
```

- like Java, but # is also allowed
  - a lot of PHP code uses # comments instead of //

Console output: print

```php
print "text";
```

```php
print "Hello, World!\n";
print "Escape \"chars\" are the SAME as in Java!\n";

print "You can have
line breaks in a string.";

print 'A string can use "single-quotes".  It\'s cool!';
```

```
Hello, World! Escape "chars" are the SAME as in Java! You
can have line breaks in a string. A string can use "single-
quotes". It's cool!
```

output

- some PHP programmers use the equivalent **echo** instead of **print**
  - arguments of echo vs. print

Variables

```
$name = expression;                                    PHP
```

```
$user_name = "PinkHeartLuvr78";
$age = 16;
$drinking_age = $age + 5;
$this_class_rocks = TRUE;                              PHP
```

- names are case sensitive; separate multiple words with _
- names always begin with **$**, on both declaration and usage
- always implicitly declared by assignment (**type is not written**)
- a loosely typed language (like JavaScript or Python)

Types

- basic types: int, float, boolean, string, array, object, NULL
  - test what type a variable is with is_*type* functions, e.g. is_string
  - gettype function returns a variable's type as a string (not often needed)
- PHP converts between types automatically in many cases:
  - string → int auto-conversion on +
  - int → float auto-conversion on /
- type-cast with (*type*):
  - $age = *(int)* "21";

int and float types

```php
$a = 7 / 2;                    # float: 3.5
$b = (int) $a;                 # int: 3
$c = round($a);                # float: 4.0
$d = "123";                    # string: "123"
$e = (int) $d;                 # int: 123
```

- int for integers and float for reals
- division between two int values can produce a float

Arithmetic operators

- + - * / % . ++ --
  = += -= *= /= %= .=

- many operators auto-convert types: 5 + "7" is 12

bool (Boolean) type

```php
$feels_like_summer = FALSE;
$php_is_rad = TRUE;

$student_count = 217;
$nonzero = (bool) $student_count;      # TRUE
```

- the following values are considered to be **FALSE** (all others are **TRUE**):
  - 0 and 0.0 (but NOT 0.00 or 0.000)
  - "", "0", and **NULL** (includes unset variables)
  - arrays with 0 elements
- can cast to boolean using (**bool**)
- **FALSE** prints as an empty string (no output); **TRUE** prints as a 1

NULL

```php
$name = "Victoria";
$name = NULL;
if (isset($name)) {
  print "This line isn't going to be reached.\n";
}
```

- a variable is NULL if
  - it has not been set to any value (undefined variables)
  - it has been assigned the constant NULL
  - it has been deleted using the unset function
- can test if a variable is NULL using the isset function
- NULL prints as an empty string (no output)

String type

```php
$favorite_food = "Ethiopian";
print $favorite_food[2];              # h
```

- zero-based indexing using bracket notation
- string concatenation operator is **.** (**period**), not **+**
  - 5 + "2 turtle doves" === 7
  - 5 . "2 turtle doves" === "52 turtle doves"
- can be specified with " " or ' '

```php
# index  012345678901234 5
$name = "Stefanie Hatcher";
$length = strlen($name);                # 16
$cmp = strcmp($name, "Brian Le");       # > 0
$index = strpos($name, "e");            # 2
$first = substr($name, 9, 5);           # "Hatch"
$name = strtoupper($name);              # "STEFANIE HATCHER"
```

Name	Java Equivalent
strlen	length
strpos	indexOf
substr	substring
strtolower, strtoupper	toLowerCase, toUpperCase
trim	trim
explode, implode	split, join
strcmp	compareTo

Interpreted strings

- strings inside " " are interpreted variables that appear inside them will have their values inserted into the string

```php
$age = 16;
print "You are " . $age . " years old.\n";
print "You are $age years old.\n";    # You are 16 years old.
```

- strings inside ' ' are *not* interpreted:

```php
print 'You are $age years old.\n';    # You are $age years old.\n
```

Arrays

```php
$name = array();                                    # create
$name = array(value0, value1, ..., valueN);

$name[index]                                        # get element value
$name[index] = value;                               # set element value
$name[] = value;                                    # append
```

```php
$a = array();       # empty array (length 0)
$a[0] = 23;         # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";     # add string to end (at index 5)
```

- to append, use bracket notation without specifying an index
- element type is not specified; can mix types

Array functions

function name(s)	description
count	number of elements in the array
print_r	print array's contents
array_pop, array_push, array_shift, array_unshift	using array as a stack/queue
in_array, array_search, array_reverse, sort, rsort, shuffle	searching and reordering
array_fill, array_merge, array_intersect, array_diff, array_slice, range	creating, filling, filtering
array_sum, array_product, array_unique, array_filter, array_reduce	processing elements

Array function example

```php
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
  $tas[$i] = strtolower($tas[$i]);
}                                    # ("md", "bh", "kk", "hm", "jp")
$morgan = array_shift($tas);         # ("bh", "kk", "hm", "jp")
array_pop($tas);                     # ("bh", "kk", "hm")
array_push($tas, "ms");              # ("bh", "kk", "hm", "ms")
array_reverse($tas);                 # ("ms", "hm", "kk", "bh")
sort($tas);                          # ("bh", "hm", "kk", "ms")
$best = array_slice($tas, 1, 2);     # ("hm", "kk")
```

PHP

- the array in PHP replaces many other collections in Java
  - list, stack, queue, set, map, ...

for loop *(same as C)*

```php
for (initialization; condition; update) {
  statements;
}
```

```php
for ($i = 0; $i < 10; $i++) {
  print "$i squared is " . $i * $i . ".\n";
}
```

if/else statement

```php
if (condition) {
  statements;
} elseif (condition) {
  statements;
} else {
  statements;
}
```

PHP

- NOTE: although **elseif** keyword is much more common, **else if** is also supported

while loop *(same as c)*

```php
while (condition) {
  statements;
}
```

```php
do {
  statements;
} while (condition);
```

- break and continue keywords also behave as in Java and c

The foreach loop

```
foreach ($array as $variableName) {
  ...
}
```
PHP

```
$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($stooges); $i++) {
  print "Moe slaps {$stooges[$i]}\n";
}
foreach ($stooges as $stooge) {
  print "Moe slaps $stooge\n";  # even himself!
}
```
PHP

- a convenient way to loop over each element of an array without indexes

Math operations

```php
$a = 3;
$b = 4;
$c = sqrt(pow($a, 2) + pow($b, 2));
```

abs	ceil	cos	floor	log	log10	max
min	pow	rand	round	sin	sqrt	tan

math functions

M_PI	M_E	M_LN2

math constants

- the syntax for method calls, parameters, returns is the same as Java and C

PHP syntax template

```
HTML content

   <?php
      PHP code
   ?>

HTML content

   <?php
      PHP code
   ?>

HTML content  . . .
```
PHP

- any contents of a .php file between **<?php** and **?>** are executed as PHP code
- all other contents are output as pure HTML
- can switch back and forth between HTML and PHP "modes"

Summary

- Server-Side Basics
  - dynamic web page
  - Server-Side Programming
- Introduction to PHP
  - lifecyle of PHP Web Request
  - PHP code should be run!

Summary

- PHP Basic Syntax
  - comments, print/echo
  - variables, types, int/float, arithemetic operators
  - bool, null
  - string, string functions, interpreted strings
  - array, array functions
  - for, if/else, while, foreach
  - math functions
  - php syntax template

Exercises

- draw a sequence diagram of interactions between a web browser and a PHP web server when the browser requests a PHP page on the server
- write a PHP code snippet to calculate and output the first 20 Fibonacci numbers
- write a PHP code snippet to calculate the day of today

Further Readings

- PHP home page: http://www.php.net/
- W3Schools PHP tutorial: http://www.w3schools.com/PHP/
- Practical PHP Programming: http://hudzilla.org/phpwiki/
- PHP Cookbook: http://commons.oreilly.com/wiki/index.php/PHP_Cookbook

Thank you!


unintended uses
user contributions
effortless scalability
trust
tagging
beta
radical decentralization
customer self-service
mass service of micromarkets
software as a service
right to remix
participation
blogs
wikis
VOIP
podcasting
filtering
sharing
social search
collaboration
social bookmarking
RIAs
Ajax
SSE
RoR
BitTorrent
KatrinaList
Wikipedia
Skype
Hard to Recreate Data
Enrichment & Annotation
Copyrighted & IP Content
Unique Formulas
Software That Gets Better The More People Use it
Face
HTML, Ajax, Flash, etc.
OpenSim
Services (API)
RSS, REST, SOAP, JSON etc.
user control
small pieces
permalinks
http
two-way interaction
tcp stream
people
World-Wide Web
data
mash-up
Web 2.0