Introduction to PHP

Professor Hossein Saiedian

EECS 448: Software Engineering

November 5, 2022

THE UNIVERSITY OF KANSAS

PHP

- A widely-used open-source scripting language
  - Free to download (php.net)
- Stands for Hypertext Preprocessor
- PHP scripts are executed on the server side
  - Not on a local machine (unless PHP is installed, and you are running a local webserver)

PHP files

- Have a default .php file extension
- May contain text, HTML, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers (Apache, IIS, etc.)
- PHP has support for a wide range of databases
- PHP is relatively easy to learn and runs efficiently on the server side
  - Lots of built-in functionality; familiar syntax
- PHP is well-documented:
  - Type **php.net/functionName** in browser address bar to get docs for any function

PHP scripts

- PHP can create, open, read, write, and close files on the server
- PHP can generate dynamic page content
- PHP can collect form data
- PHP can restrict users to access some pages on your website
- PHP can add, delete, and modify data in a database

Basic syntax

- A PHP script starts with **<?php** and ends with **?>**

```
<?php
// PHP code goes here
?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code
- Each code line in PHP must end with a semicolon
  - The semicolon is a separator and is used to distinguish one set of instructions from another
- Two statements to output: **echo** and **print**

Basic syntax

```
<html>
  <head>
        <title>Example</title>
  </head>
  <body>
        <?php echo “Hello, World, I'm a PHP Script” ?>
  </body>
</html>
```

Comments in PHP (like C, C++)

```
<?php

// A single-line PHP comment
# Another single-line comment (more popular)


/*
A multi-line PHP comment
can be formed like this
*/

?>
```

PHP variables

- Variable names start with $ followed by the name
- A variable name must begin with a letter or the underscore character
- A variable name can only contain alphanumeric characters and underscores (A-z, 0-9, and _) and no spaces
- Variable names are case sensitive ($y and $Y are two different variables)

Defining variables

- PHP has no command for declaring a variable
- A variable is created the moment you first assign a value to it:

```
$text="Hello world!";
$counter=5;
```

- PHP is loosely typed and automatically converts a variable to the correct data type depending on its value
- PHP variable scopes: local, global, static, parameter

PHP operators

- + - * / %

  . ++ --

  = += -= *= /= %= .=
- Many operators auto-convert types: 7 + "7" is 14
- Expressions
  - $name = expression;
  - Implicitly declared by assignment

PHP 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
- PHP converts between types automatically in many cases:
  - string → int auto-conversion on + (e.g., **`"1" + 1 == 2`**)
  - int → float auto-conversion on / (e.g., **`3 / 2 == 1.5`**)
- Type-cast with (type):
  - **`$age = (int) "21";`**

PHP operators (similar to C)

Operator	Name	Description	Example	Result
x + y	Addition	Sum of x and y	2 + 2	4
x - y	Subtraction	Difference of x and y	5 - 2	3
x * y	Multiplication	Product of x and y	5 * 2	10
x / y	Division	Quotient of x and y	15 / 5	3
x % y	Modulus	Remainder of x divided by y	5 % 2 10 % 8 10 % 2	1 2 0
- x	Negation	Opposite of x	- 2	
a . b	Concatenation	Concatenate two strings	"Hi" . "Ha"	HiHa

PHP operators (similar to C)

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus
a .= b	a = a . b	Concatenate two strings

PHP operators (similar to C)

Operator	Name	Description
++ x	Pre-increment	Increments x by one, then returns x
x ++	Post-increment	Returns x, then increments x by one
-- x	Pre-decrement	Decrements x by one, then returns x
x --	Post-decrement	Returns x, then decrements x by one

PHP operators (similar to C)

Operator	Name	Description	Example

x and y	And	True if both x and y are true	x=6
			y=3
			(x < 10 and y > 1) returns true

x or y	Or	True if either or both x and y are true	x=6
			y=3
			(x==6 or y==5) returns true

x xor y	Xor	True if either x or y is true, but not both	x=6
			y=3
			(x==6 xor y==3) returns false

x && y	And	True if both x and y are true	x=6
			y=3
			(x < 10 && y > 1) returns true

x || y	Or	True if either or both x and y are true	x=6
			y=3
			(x==5 || y==5) returns false

! x	Not	True if x is not true	x=6
			y=3
			!(x==y) returns true

For loop (similar to Java)

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

Example

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

If statement

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

While loop

```
while (condition) {
  statements;
}

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

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

An example of variables, operators

```
<?php
  $var = "Bob";
  $Var = "Joe";
  echo "$var, $Var";     // outputs "Bob, Joe"
  $x = 1;
  $x = 'abc';            // type can change if value changes
  $5site = 'not yet';    // invalid; starts with a number
  $_5site = 'not yet';  // valid; starts with an
  underscore
?>
```

PHP forms

- HTML forms (GET and POST)
  - Form is submitted to a PHP script
  - Information from that form is automatically made available to the script
- form.php:

```
<form action="foo.php" method="POST">
Name: <input type="text" name="username"><br>
Email: <input type="text" name="email"><br>
<input type="submit" name="submit" value="Submit me!">
</form>
```

PHP form foo.php

```
<?php // Available since PHP 4.1.0

    print $_POST['username'];
    print $_REQUEST['username'];
    import_request_variables('p', 'p_');
    print $p_username;

    // Available since PHP 3. As of PHP 5.0.0, these long
    // predefined variables can be disabled with the
    // register_long_arrays directive.

    print $HTTP_POST_VARS['username'];

    // Available if the PHP directive register_globals = on.
    // As of PHP 4.2.0 the default value of
    // register_globals = off.
    // Using/relying on this method is not preferred.

    print $username;
?>
```

Another form example

- info_form.php

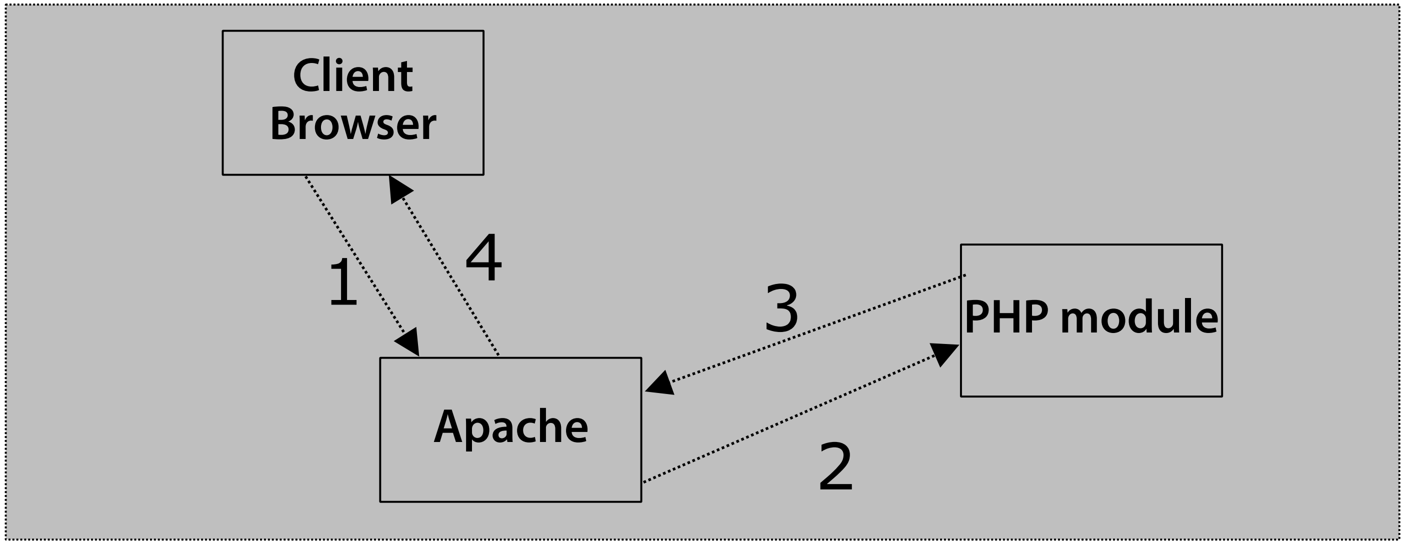
```
<form action="show_answers.php" method="POST">
  Your name: <input type="text" name="name" />
  Your age: <input type="text" name="age" />
  <input type="submit">
</form>
```

- show_answers.php

```
Hi
<?php echo $_POST["name"]; ?>.
You are <?php echo $_POST["age"]; ?> years old.
```

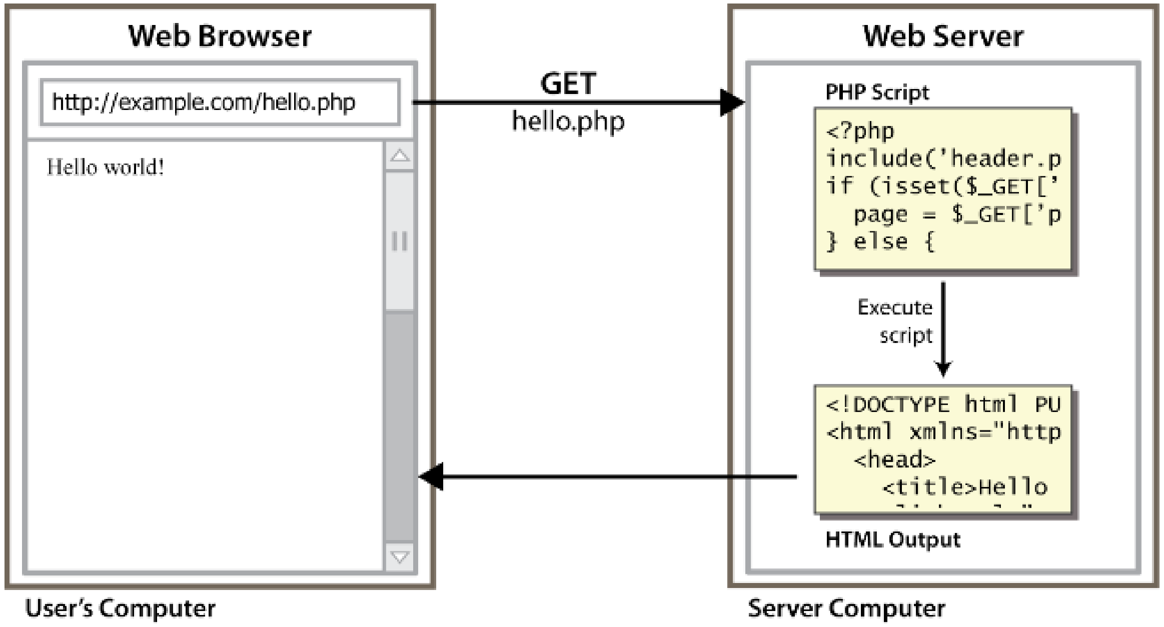
How PHP generates output

1. Client from browser send HTTP request (with POST/GET variables)
2. Apache recognizes that a PHP script is requested and sends the request to PHP module
3. PHP interpreter executes PHP script, collects script output and sends it back
4. Apache replies to client using the PHP script output as HTML output



Lifecycle of a 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



```
<?php
include('header.p
if (isset($_GET['
  page = $_GET['p
} else {
```

```
<!DOCTYPE html PU
<html xmlns="http
  <head>
    <title>Hello
```

Alternatives to PHP

- Practical extraction and Report Language (Perl)
- Active Server Pages (ASP)
- Java server pages (JSP)
- Ruby