PHP

- A server-side, cross-platform, HTML-embedded scripting language used to create dynamic Web pages (like Perl open source).
- We can write PHP code directly into HTML and don’t need to worry about CGI (like Java Server Pages).
- Looks like Perl and knowledge of one can help you understand the other.
- Web server needs to be configured to parse files with certain extensions (e.g. php or phtml) appropriately.

Serve-side Scripting

- SSI: Server-side includes:
  - very old scripting language
  - mostly provides #include
- PHP:
  - complete language
  - free
  - integrates nicely into Apache, IIS, other servers
- ASP: Active Server Pages
  - Microsoft version of PHP
  - language is essentially Basic
- JSP: Java Server Pages
  - Java version

Most popular third-party module for Apache:
- code and extensive documentation available from http://www.php.net/
- Pieces of code are enclosed into <?...?> tags
- Example:
  <html>
  <head><title>PHP Test</title></head>
  <body><? print("Hello World!"); ?></body>
  </html>
Parameter Decoding

- Each PHP script can be invoked as a CGI.
- Parameters are decoded automatically and appear to the script as normal variables.
- Example:

```
http://www.cse.nd.edu/~cpoellab/myphp.php?x=2&y=hello
```

```html
<html>
<head><title>PHP Test</title></head>
<body>
<? 
print("x=
");
print("y=
");
?>
</body>
</html>
```

More Preset Variables

- PHP creates many other variables automatically:
  - $HTTP_USER_AGENT, $REMOTE_ADDR, etc.
  - list: `<?phpinfo(); ?>`
- Example:

```
<? if(strstr($HTTP_USER_AGENT, 'MSIE')) {
  <center>You are using Internet Explorer</center>
} else {
  <center>You are not using Internet Explorer</center>
} ?>
```

Did you notice that you can split a PHP program into pieces separated by HTML?

Data Types

- Integer
  - Whole numbers –2,147,483,648 to 2,147,483,647
- Floating Point
  - Decimal values in the range 1.7E-308 to 1.7E308
- String
  - Sequence of characters e.g. “Hello World”

Variables

- All names begin with $ e.g. $variable
- Alphabetic character or underscore (_) must follow $
- Remaining chars are alphanumeric or underscore
- Names are case-sensitive $A is not $a
- Types determined by first assignment

String Handling

- Characters within strings can be obtained via subscripting
  - Subscripts start at 0

```
$hello = "Hello World"

$hello[1] = ?
```
Outputting a String
- The function `echo()` outputs a string to the standard output
  e.g. `echo("Hello World");`
  ```
  $astring = "Hello World"
  echo($astring);
  ```

Length of a String
- The function `strlen()` returns a string's length
  ```
  $hello = "Hello World"
  $value = strlen($hello) is 11
  ```

Expressions
- Includes normal arithmetic operations
  - `$i=$i+1` or `$i++`
  - `$i=$i-1` or `$i--`
  - Also `/` (divide) and `*` (multiply)
  - `%` is the modulo operation
- May use brackets to specify precedence
- Statements separated by semi-colons
- Expressions evaluate to true or false

Getting a Number from a String
- Use the string in a calculation
  - PHP converts as much as it can to a number
  ```
  $var = "1234"
  $num = $var[2] + 5
  $num contains the number...
  ```

PHP Control Structures
- Like all languages, PHP has control structures:
  ```
  <?
  while ($x != 0) {
    ...
  }
  if ($x == 0) {
    ...
  }
  elseif ($x == 1) {
    ...
  }
  ?>
  ```

PHP Arrays
- Very useful feature: array variable type (set of key-value associations).
  ```
  $role = array {
    "Poellabauer" => "teacher",
    "You" => "student"
  };
  $role["You"] = "student";
  $me = $role["Poellabauer"];
  ```
PHP Arrays

- Special case: array indexed by integers:

```php
$n = array("Poellabauer", "You", "Your neighbor");
$name = $names[0];
$name[3] = "Another student";
```

- What about array of arrays?

```php
$students = array (
    "You" => array ("name" => "yourname", "student nb" => 1234),
    "Another" => array ("name" => "hisname", "student nb" => 5678),
    "FooBar" => 42
);

$you = $students["You"];  //  This will print a human-readable version of $students
print_r($you);  //  This will print a human-readable version of $students
```

print_r

```php
<?
$a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x', 'y', 'z'));
print_r($a);
?>
```

PHP Functions

- You can define functions:

```php
function addition($x, $y) {
    return $x + $y;
}
$foo = addition(31, 11);
```

- Predefined functions: tons!

- Classified into categories:

  - Apache, arrays, Aspell, BC, Bzip2, Calendar, CCVS, COM, Classes/Objects, ClibPDF, Crack, CURL, Cybercash, CyberMUT, Cyradm, ctype, dba, Date/Time, dBase, DBM, dbx, DB++, DIO, Directories, DOM XML, .NET, Errors and Logging, FrontBase, filePro, Filesystem, FDF, FniBidi, FTP, Functions, gettext, GMP, HTTP, Hypervave, ...

PHP Functions

- Look at following code:

```php
function lm($x) {
    printf("<font size=-1>[Last modified: %s]</font>", date("D, d M Y H:i:s", filemtime($x)));
}
```

```html
<html>
<body>
Download your new <a href="foo.ps.gz">assignment</a> &lt;? lm("foo.ps.gz"); &gt;
</body>
</html>
```

Getting Data From Client

- Data is often supplied to server side programs via HTML forms
- Indicated by the `<form>` tag
- Specifies HTTP method and field names
- Field names become variables in PHP scripts
  - Field name myfield becomes $myfield

A Simple Form

```html
<html>
<head>
<title>Multiply Form</title>
</head>
<body>
Enter multiplier</h1>
<form action="multiply.php" method="POST">
<input type="text" name="multiplier">
</form></body>
</html>
```

Simple Form Output

![Simple Form Output](image)

PHP Script

```html
<html>
<head><title>Times Table</title></head>
<body>
		<?for($i=1;$i<12;$i++){
				echo($i); ?
				* \t<?echo($multiplier); ?>
				=	<?echo($multiplier*$i); ?>
		?<br/>
	<?}?>
</body>
</html>
```

Output

![Output](image)
Getting it on the Server

- Treat PHP scripts like ordinary HTML pages
- Except save in files called .php
- Suitably equipped Web Server does the rest

Manipulating HTTP Headers

- By default, PHP scripts are assumed to generate HTML:
  - Content-Type field of response defaults to text/html
- You can force it otherwise:
  ```php
  <?
  $len = filesize("foo.pdf");
  header("Content-Type: application/pdf");
  header("Content-Length: $len");
  readfile("foo.pdf");  ?>
  - You can only manipulate HTTP headers BEFORE displaying anything. This won’t work:
  ```php
  <? print("foo");
  header("Content-Type: text/plain");  ?>
  ```

Access Control

- You may want to control access to certain pages:
  - administration pages, secrets, ...
- Check IP address:
  ```php
  if (!$REMOTE_ADDR=="130.207.7.245") {
    header("HTTP/1.0 403 forbidden access");
    print("You are not authorized to access my diary");
    exit;
  }
  - will only work if you know exactly who can send requests from that address
  - what if you want to access page from Cybercafe?
  ```

Access Control

- You can also use HTTP basic authentication:
  - first HTTP request: return HTTP code 401 “please authenticate yourself”
  - client types username and password
  - another HTTP request containing both: authorized to enter
- PHP has standard variables for both: $PHP_AUTH_USER and $PHP_AUTH_PW
- Documents are not encrypted:
  - even username/password are sent in clear text
  - anyone who intercepts the request can gain access

Access Control

```php
<?
//Check if the request carries an authentication
if (!isset($PHP_AUTH_USER)) {
  //No authentication
  header('HTTP/1.0 401 Unauthorized');
  header('WWW-Authenticate: Basic realm="PHP Secured"');
  print("Please authenticate yourself");
  exit;
}
elseif ($PHP_AUTH_USER=="Poellabauer" && $PHP_AUTH_PW==42) {
  //Correct authentication
  print("Hello, $PHP_AUTH_USER. You can enter.");
}
else {
  //Wrong authentication
  header('HTTP/1.0 403 Forbidden');
  print("Go away, you are not authorized here!");
  exit;
}
?>
```
Session Tracking

- By default, a web server is stateless.
- However, often you want to establish a session:
  - keep memory of past requests of user
  - example: e-commerce:
    - browse to view products
    - add items to shopping basket
    - checkout

Cookies are transmitted in HTTP response header.

```php
<?
if (!isset($userid)) {
    srand((double)microtime()*1000000);
    $newid = rand();
    $expire = time() + 3600; // in 1 hour
    setcookie("userid", $newid, $expire, "/myshop/", ".cse.nd.edu", 0);
    echo "From now on, you will be known as number ", $newid;
} else {
    echo "You are identified as user number ", $userid;
}
?>
```

Summary: Server Side Options

- Common Gateway Interface (CGI)
- Fast CGI
- Mod Perl
- ASP
- PHP
- Cold Fusion
**Decision Points**

- When evaluating which server side framework to use, you need to consider a number of critical factors:
  - Ease of development:
    - How easily can you build new applications?
  - Performance:
    - How fast can the framework respond to queries?
  - Scalability:
    - Can the framework scale to thousands, millions of users?
  - Security:
    - Are there any inherent security vulnerabilities?

**Option 1: CGI**

- Represents one of the earliest, practical methods for generating web content.
- Primarily written in the Perl programming language.
- Unfortunately, traditional CGI programs suffer from scalability and performance problems.
- Let’s examine these two problems…

**CGI Architecture**

- For each browser request, the web server must spawn a new operating system process.

**Option 2: Fast CGI**

- Developed by Open Market as an option for developing faster, more scalable CGI programs.
- Fast CGI works by creating a pool of processes for handling CGI requests.
- When a CGI request comes in, Fast CGI picks one of the processes from the pool and assigns it to the task.
- Without the overhead of creating new operating system processes, Fast CGI is much faster than traditional CGI.
- For more information, see [http://www.fastcgi.com](http://www.fastcgi.com)

---

**Browser initiates request**

- Web server receives the request.
- For each request, web server spawns a new operating system process to execute the CGI/Perl Program.

**Creating a New Process**

- Browser 1
  - Web Server
  - Perl 1
- Browser 2
  - Web Server
  - Perl 2
- Browser n
  - Web Server
  - Perl n

---

**Spawning a new operating system process for each request takes time and memory.**

- Hence, traditional CGI programs have inherent performance and scalability problems.
- Every other server architecture tries to address these problems.
Option 3: Mod Perl
- A module of the Apache Web Server.
- Embeds the Perl interpreter directly within the web server.
- No need to start external interpreter.
- Also, because Perl is embedded within the Server, Mod Perl does not need to create a new process for each request.
- Like FastCGI, Mod Perl is much faster than traditional CGI.
- For more information, see: http://perl.apache.org

Option 4: ASP
- Active Server Pages
- Runs on Microsoft’s Web Server: Internet Information Server (IIS)
- Programmers add ASP code directly into their HTML pages.
- When a client requests a page, the Web Server takes the HTML page, runs the ASP code within the page, and returns a complete HTML page.
- Faster than traditional CGI, but only works on Microsoft IIS.

Option 5: Cold Fusion
- Developed by Allaire Corporation (now owned by Macromedia.)
- Provides excellent database access and database tools.
- Great platform for rapid prototyping and rapid development.
- For more information: http://www.macromedia.com

Option 6: PHP
- An open source project written entirely by volunteers.
- Provides simple, but powerful database access.
- Also great for rapid development.
- For additional information: http://www.php.net

Summary: PHP
- PHP is a programming language that enables you to create interactive web sites.
- Many good features:
  - easy to learn.
  - fun to use.
  - free.
  - ideal for rapid prototyping.
  - connects to most databases, including MySQL.
  - available on most systems.
  - lots of built-in functions and built-in functionality.
- Trivia: PHP stands for?

Summary: PHP
- PHP is a programming language that enables you to create interactive web sites.
- Many good features:
  - easy to learn.
  - fun to use.
  - free.
  - ideal for rapid prototyping.
  - connects to most databases, including MySQL.
  - available on most systems.
  - lots of built-in functions and built-in functionality.
- Trivia: PHP stands for Hypertext Preprocessor