



# Trường CNTT và TT Đại Học Cần Thơ

---

## Lập Trình PHP



Bùi Võ Quốc Bảo, Đỗ Thanh Nghị

Cần Thơ  
2010-2023

# Nội dung

---

- ❑ Giới thiệu về PHP
- ❑ Biến, kiểu dữ liệu, phép toán
- ❑ Lệnh điều khiển
- ❑ Hàm
- ❑ PHP include
- ❑ Xử lý lỗi
- ❑ Dữ liệu gửi về server
- ❑ Cookie và Session
- ❑ HTTP header, Response code, Redirect

---

## ❑ **Giới thiệu về PHP**

- ❑ Biến, kiểu dữ liệu, phép toán
- ❑ Lệnh điều khiển
- ❑ Hàm
- ❑ PHP include
- ❑ Xử lý lỗi
- ❑ Dữ liệu gửi về server
- ❑ Cookie và Session
- ❑ HTTP header, Response code, Redirect

# Giới thiệu về PHP

---

- ❑ PHP là gì ?
  - ❑ PHP là Hypertext Preprocessor
  - ❑ Ngôn ngữ script chạy trên server
  - ❑ PHP scripts chứa text, thẻ HTML, script
  - ❑ Sử dụng phần mở rộng tên file : .php, .phtml
  - ❑ PHP scripts sẽ trả về kết quả cho trình duyệt một plain HTML
  - ❑ PHP hỗ trợ để làm việc với nhiều hệ QTCSDL khác nhau: MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.
  - ❑ Phần mềm mã nguồn mở, miễn phí
  - ❑ Chạy trên nhiều platforms (Unix, Linux, Windows)

# Giới thiệu về PHP

---

- ❑ MySQL/MariaDB là gì ?
  - ❑ Hệ quản trị cơ sở dữ liệu
  - ❑ Dùng cho các ứng dụng vừa và nhỏ
  - ❑ Hỗ trợ chuẩn SQL
  - ❑ Phần mềm mã nguồn mở, miễn phí
  - ❑ Chạy trên nhiều platforms (Unix, Linux, Windows)
  - ❑ Phổ biến
  - ❑ PHP + MySQL : Web động chạy trên nhiều platforms khác nhau

# Giới thiệu về PHP

---

## ❑ Tại sao PHP ?

- ❑ Chạy trên nhiều platforms khác nhau (Unix, Linux, Windows)
- ❑ Phần mềm mã nguồn mở, miễn phí
- ❑ Tương thích với hầu hết các web server (Apache, IIS, etc)
- ❑ Dễ học và phát triển nhanh các ứng dụng trên Web

## ❑ Làm thế nào để sử dụng PHP

- ❑ Cài web server ([Apache HTTP Server](#), [NGINX](#), [Caddy](#), IIS, etc)
- ❑ Cài [PHP](#) ( $\geq 8.1$ )
- ❑ Cài MySQL/[MariaDB](#)
- ❑ Hoặc dùng các gói phần mềm tích hợp: [Xampp](#), [Laragon](#), [WampServer](#)

# Giới thiệu về PHP

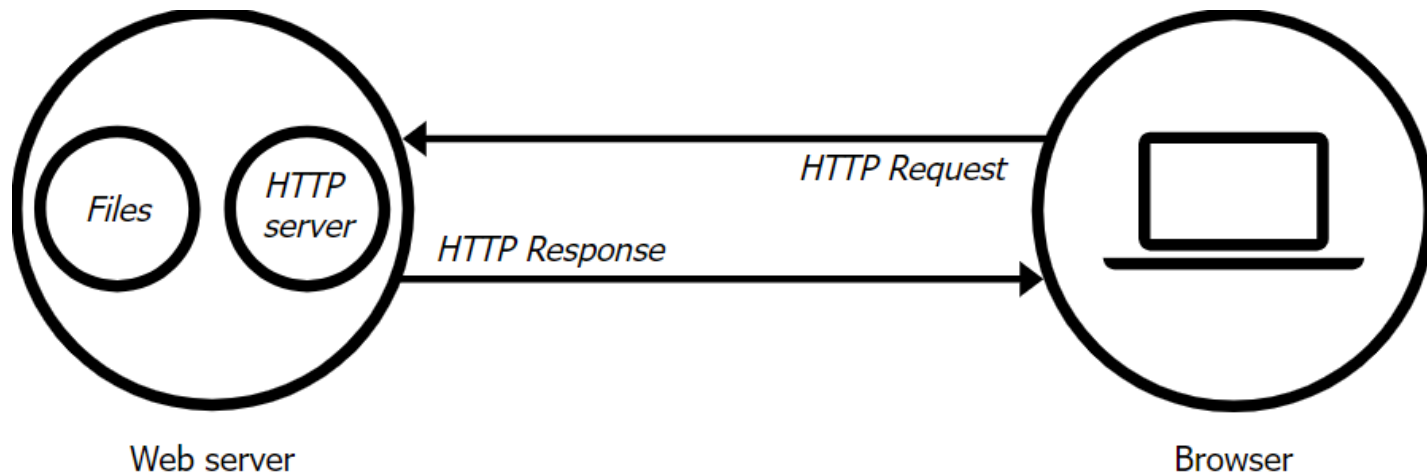
---

- Web server (máy chủ web) là gì ?
  - Thuật ngữ “web server” có thể nói đến phần cứng, phần mềm hoặc cả hai làm việc cùng nhau
  - Về phần cứng, web server là một máy tính trên đó có chạy phần mềm web server và lưu trữ các tập tin của một website (trang HTML, CSS, JavaScript, các tập tin ảnh, ...)

# Giới thiệu về PHP

---

- Web server (máy chủ web) là gì ?
  - Về phần mềm, có thể xem web server là một HTTP server (máy chủ HTTP), phần mềm có thể hiểu các URL (địa chỉ web) và giao thức HTTP, phân phát nội dung website đến thiết bị người dùng





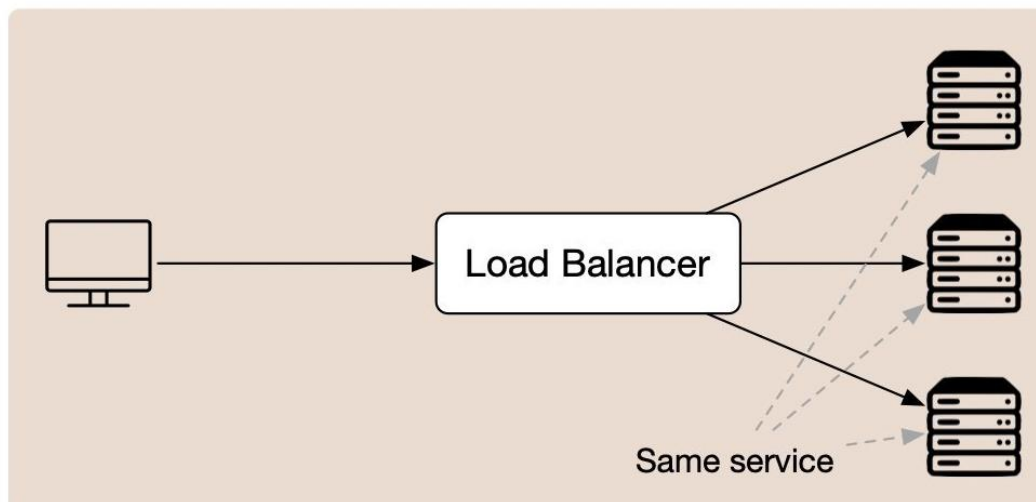
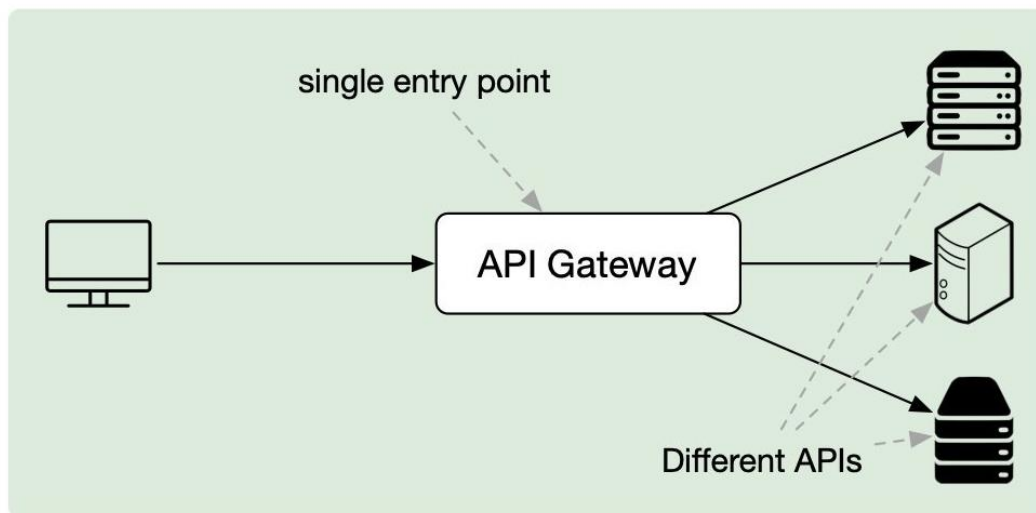
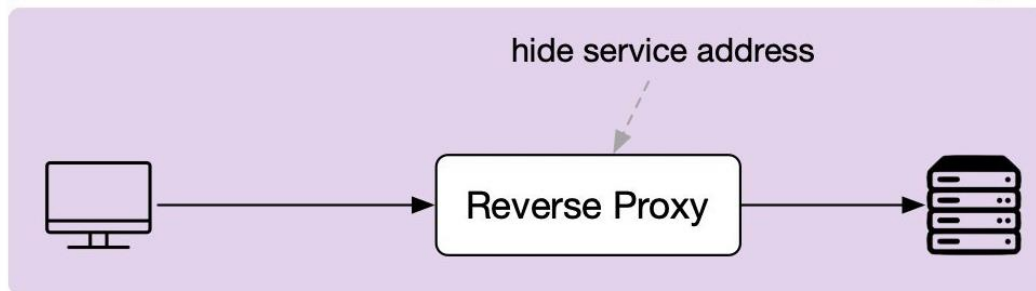
# Giới thiệu về PHP

---

- ❑ Các web server có thể được sử dụng để :
  - ❑ Phục vụ nội dung tĩnh, động\* qua giao thức HTTP
  - ❑ Trữ tạm thời nội dung website (HTTP/Web cache)
  - ❑ Cân bằng tải (load balancing)
  - ❑ Làm proxy đảo ngược (reverse proxy) hay gateway: server nằm trước các ứng dụng backend và chuyển các yêu cầu từ client đến các ứng dụng này
  - ❑ ...

\* Thường cần thêm các plugin/mô-đun hỗ trợ các ngôn ngữ script để giúp phát sinh nội dung động

# Giới thiệu



# Giới thiệu về PHP

---

- ❑ Phục vụ tài nguyên tĩnh với web server :
  - ❑ Tập tin cần gửi trả lời cho một yêu cầu web được xác định bởi :  
DocumentRoot + URL Path, ví dụ :  
DocumentRoot = C:/xampp/htdocs  
URL = http://www.example.com/**static/about.html**  
=> Đường dẫn đến tập tin là : C:/xampp/htdocs/static/about.html
  - ❑ Việc phục vụ các tập tin nằm ngoài DocumentRoot là có thể, ví dụ như thông qua chỉ thị Alias trong Apache
  - ❑ Các phần mềm web server cũng hỗ trợ chạy nhiều website trên cùng một máy tính, mỗi website có một DocumentRoot riêng (Virtual Host trong Apache, Server Block trong NGINX)

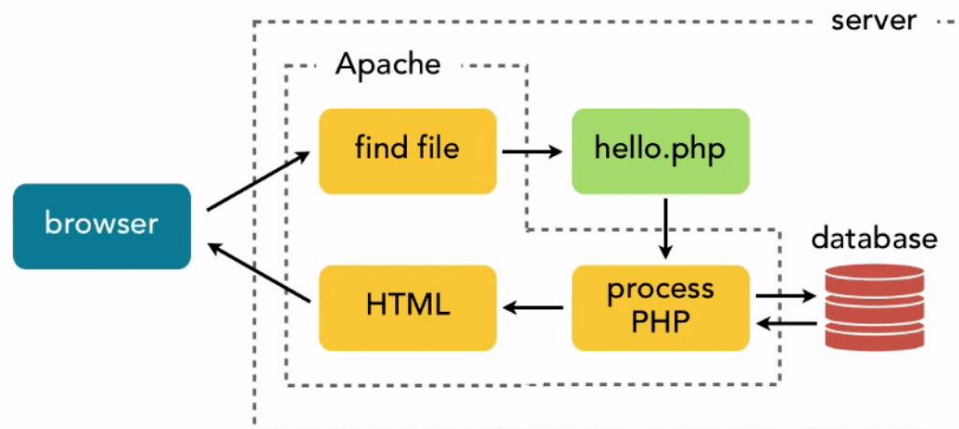
# Giới thiệu về PHP

---

- ❑ Làm thế nào web server phục vụ nội dung động phát sinh từ PHP script ?
  - ❑ PHP chạy như **một tiến trình CGI**
    - CGI (Common Gateway Interface) : giao diện cho phép các web server gọi thực thi chương trình bên ngoài để xử lý yêu cầu web
    - Web server tạo mới một tiến trình CGI PHP cho mỗi yêu cầu đến
    - Hiệu năng thấp, ngày nay ít được sử dụng

# Giới thiệu về PHP

- ❑ Làm thế nào web server phục vụ nội dung động phát sinh từ PHP script ?
  - ❑ PHP chạy như **một mô-đun trên trong web server**
    - Tùy chọn thường gặp nhất khi dùng web server Apache và cũng chỉ có web server Apache hỗ trợ cách này
    - Trình thông dịch PHP được “nhúng” vào bên trong tiến trình của Apache



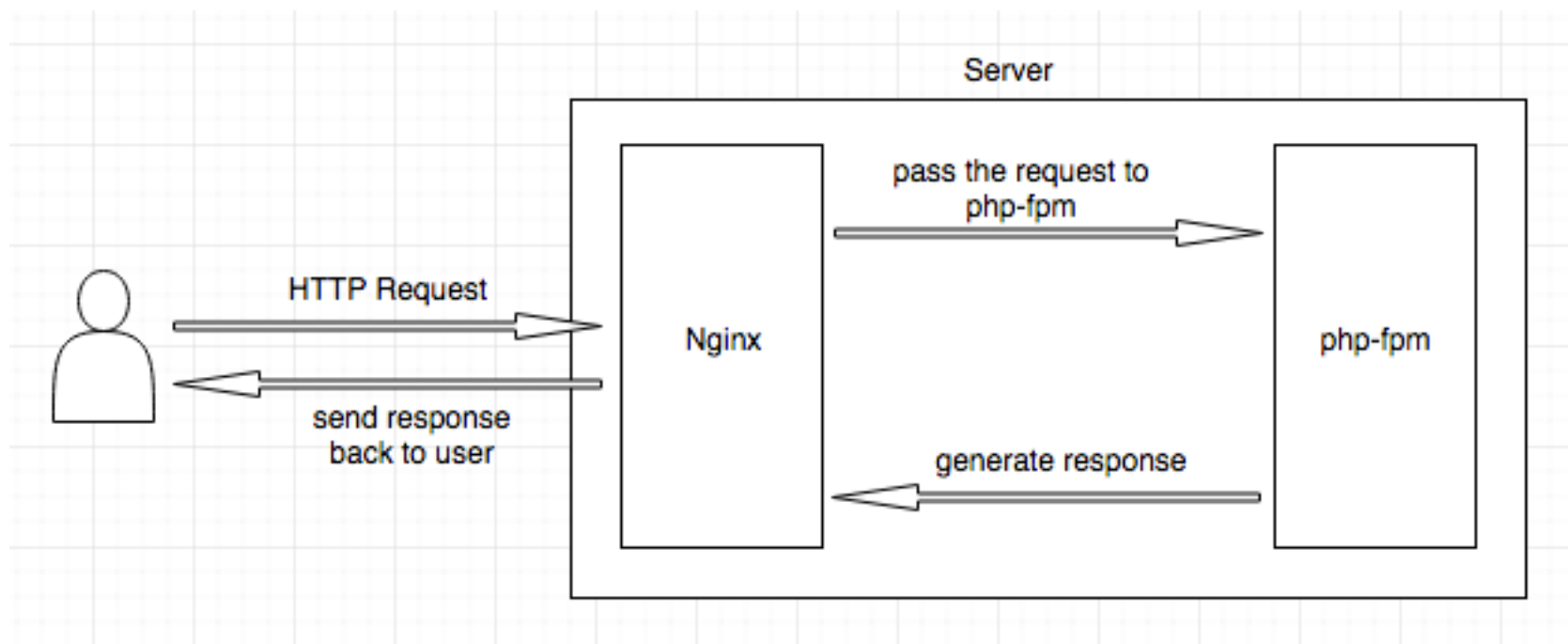
# Giới thiệu về PHP

---

- ❑ Làm thế nào web server phục vụ nội dung động phát sinh từ PHP script ?
  - ❑ PHP chạy như **một dịch vụ độc lập**
    - FastCGI : cải tiến của CGI cho phép web server chuyển yêu cầu đến tiến trình FastCGI để xử lý (qua TCP hay Unix domain socket) và nhận kết quả trả về
    - php-fpm : cài đặt FastCGI chính cho PHP, chưa hỗ trợ bản build chính thức cho Windows
    - Cách tiếp cận này mềm dẻo, có thể cho hiệu năng và hiệu suất sử dụng tài nguyên cao

# Giới thiệu về PHP

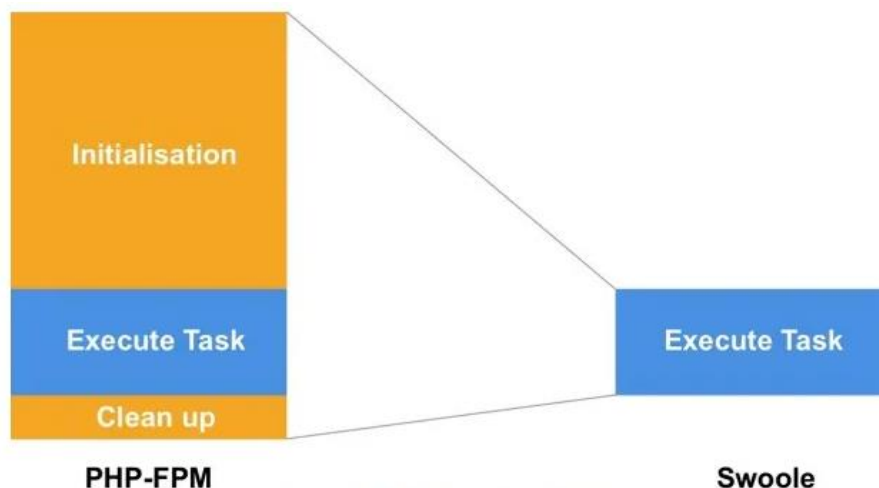
- ❑ Làm thế nào web server phục vụ nội dung động phát sinh từ PHP script ?
  - ❑ PHP chạy như **một dịch vụ độc lập**



# Giới thiệu về PHP

---

- ❑ Làm thế nào web server phục vụ nội dung động phát sinh từ PHP script ?
  - ❑ PHP chạy như **một dịch vụ độc lập**
    - RoadRunner, (Open)Swoole là các giải pháp hỗ trợ chạy tiến trình PHP lâu dài (trạng thái tiến trình PHP được duy trì qua các lần xử lý yêu cầu). Có thể tiếp nhận xử lý yêu cầu HTTP trực tiếp như một web server





# Giới thiệu về PHP

---

## ❑ Built-in web server

- ❑ PHP  $\geq 5.4$  có hỗ trợ một built-in web server đơn luồng (single threaded) dùng cho *môi trường phát triển*

- ❑ Ví dụ :

Chạy web server trong *thư mục hiện hành*:

**php -S localhost:8000**

Chạy web server với thư mục *document root* là www:

**php -S localhost:8000 -t www/**

Chạy web server với *tập tin cấu hình PHP*:

**php -S localhost:8000 -c php.ini**

# Giới thiệu về PHP

---

## ❑ Built-in web server

### ❑ Ví dụ :

Chạy web server với *tập tin định tuyến* (~ .htaccess) :

**php -S localhost:8000 router.php**

```
<?php      // router.php
```

```
if (preg_match('\.(?:png|jpg|jpeg|gif)$/',
```

```
    $_SERVER["REQUEST_URI"])) {
```

```
    return false;  // serve the requested resource as-is.
```

```
} else {
```

```
    echo "<p>Welcome to PHP</p>";
```

```
}
```

- 
- ❑ Giới thiệu về PHP
  - ❑ **Biến, kiểu dữ liệu, phép toán**
  - ❑ Lệnh điều khiển
  - ❑ Hàm
  - ❑ PHP include
  - ❑ Xử lý lỗi
  - ❑ Dữ liệu gửi về server
  - ❑ Cookie và Session
  - ❑ HTTP header, Response code, Redirect

# Cú pháp PHP

---

## □ Cú pháp

- PHP scripts chứa text, thẻ HTML, script
- Ví dụ : in ra màn hình chuỗi “Hello World”

```
<html>
```

```
<body>
```

```
<?php echo "Hello World"; ?>
```

```
</body>
```

```
</html>
```

# Cú pháp PHP

---

## □ Cú pháp

- Khởi lệnh PHP script bắt đầu với

**<?php**

và kết thúc bởi

**?>**

- Khởi lệnh có thể được đặt bất cứ nơi nào trong tài liệu
- Mỗi lệnh cách nhau bởi dấu ;
- Có 2 lệnh cơ bản để xuất dữ liệu ra màn hình: **echo** và **print**
- Chú thích trong chương trình

// chú thích là 1 dòng đơn

/\* chú thích là 1 đoạn

văn bản \*/

# Cú pháp PHP

---

## □ Cú pháp

### □ Ví dụ :

```
<?php
```

```
    echo "This is a test"; // This is a one-line c++ style comment
```

```
    /* This is a multi line comment
```

```
        yet another line of comment */
```

```
    echo("This is yet another test");
```

```
    print "Hello World";
```

```
    print("Hello World");
```

```
?>
```

```
<?="This is a test"?> // <?php echo "This is a test";?>
```

# Cú pháp PHP

---

- ❑ Không phân biệt ký tự thường hoa
  - ❑ Từ khóa
  - ❑ Lớp
  - ❑ Hàm, hàm được tạo bởi người lập trình
- ❑ Chỉ phân biệt ký tự thường hoa
  - ❑ Các biến

# Biến

---

- ❑ Biến trong PHP
  - ❑ Chứa dữ liệu
  - ❑ Biến được bắt đầu bởi dấu \$
  - ❑ Tên biến bắt đầu bằng một ký tự chữ cái hoặc \_
  - ❑ Phân biệt giữa ký tự thường và hoa
  - ❑ Kiểu được tính ở thời điểm gán giá trị
  - ❑ Gán giá trị với =
  - ❑ Sử dụng & như tham chiếu



# Biến

---

## ❑ Biến trong PHP

### ❑ Ví dụ :

```
<?php
```

```
$var = 'Bob';
```

```
$Var = 'Joe';
```

```
echo "$var, $Var";    // outputs "Bob, Joe"
```

```
$4site = 'not yet';   // invalid; starts with a number
```

```
$_4site = 'not yet';  // valid; starts with an underscore
```

```
$täyte = 'mansikka';  // valid; 'ä' is (Extended) ASCII 228.
```

```
?>
```

# Biến

---

## ❑ Biến trong PHP

### ❑ Ví dụ :

```
<?php
$foo = 'Bob';           // Assign the value 'Bob' to $foo
$bar = &$foo;           // Reference $foo via $bar.
$bar = "My name is $bar"; // Alter $bar...
echo $bar;              // My name is Bob
echo $foo;              // My name is Bob
?>
```

# Biến

---

## □ Biến trong PHP

### □ Ví dụ :

```
<?php
$foo = 'Bob';
echo $foo;           // Bob
$foo = 12
echo $foo;           // 12
$foo = [1, 2, 3, 4, 5];
for($i = 0; $i < 5; $i++)
    echo $foo[$i] . "<br>";
?>
```

# Biến

---

## □ Phạm vi biến

□ Toàn cục : sử dụng từ khóa global hoặc biến \$GLOBALS

□ Ví dụ :

```
<?php
```

```
$a = 1;
```

```
$b = 2;
```

```
function sum() {
```

```
    global $a, $b;
```

```
    $b = $a + $b;
```

```
}
```

```
sum();
```

```
echo $b;
```

```
?>
```

# Biến

---

## □ Phạm vi biến

- Toàn cục : sử dụng từ khóa global hoặc biến \$GLOBALS

- Ví dụ :

```
<?php
```

```
$a = 1;
```

```
$b = 2;
```

```
function sum() {
```

```
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
```

```
}
```

```
sum();
```

```
echo $b;
```

```
?>
```

# Biến

---

- ❑ Các biến toàn cục có sẵn trong PHP, hiện diện ở tất cả các phạm vi biến (superglobals)
  - ❑ `$GLOBALS` : tất cả các biến trong phạm vi toàn cục của script
  - ❑ `$_SERVER` : tập hợp biến môi trường của Web server (các header, đường dẫn, vị trí script)
  - ❑ `$_GET`, `$_POST` : các biến được cung cấp cho script thông qua phương thức HTTP GET (query string), POST (form data)
  - ❑ `$_COOKIE` : biến cung cấp HTTP cookies cho script
  - ❑ `$_FILES` : biến cung cấp HTTP POST file uploads cho script
  - ❑ `$_ENV` : biến cung cấp môi trường cho script
  - ❑ `$_REQUEST` : cung cấp các `$_GET`, `$_POST`, `$_COOKIE`
  - ❑ `$_SESSION` : cung cấp các biến trong phiên làm việc

# Biến

---

## □ Phạm vi biến

### □ Cục bộ

### □ Ví dụ :

```
<?php
```

```
$a = 1; /* global scope */
```

```
function test() {
```

```
    $a = 10;
```

```
    echo " in test a = " . $a; /* reference to local scope variable */
```

```
}
```

```
test();
```

```
echo "<br> out test a = " . $a;
```

```
?>
```

# Biến

---

## □ Phạm vi biến

- Biến tĩnh : sử dụng từ khóa static

- Ví dụ :

```
<?php
```

```
function test() {
```

```
    static $a = 10;
```

```
    echo " in test a = " . $a;
```

```
    $a++;
```

```
}
```

```
test(); // 10
```

```
test(); // 11
```

```
?>
```



# Biến

---

## ❑ isset(), empty()

- ❑ isset() : kiểm tra biến tồn tại và có giá trị khác null
- ❑ empty() : kiểm tra biến không tồn tại hoặc có giá trị tương đương false

## ❑ Ví dụ:

```
<?php
$var = 0;
if (empty($var)) {
    echo '$var is either 0, empty, or not set at all';
}
if (isset($var)) {
    echo '$var is set even though it is empty';
}
?>
```

# Hằng số

---

- ❑ `define(constant_name, value)`
  - ❑ Định nghĩa một hằng số tại **thời điểm thực thi**
  - ❑ `defined(constant_name)` : kiểm tra nếu một hằng số đã được định nghĩa

❑ Ví dụ:

```
<?php
```

```
define('CONSTANT', 'Hello world. ');
```

```
define('ANIMALS', array('dog', 'cat', 'bird'));    // PHP >= 7.0
```

```
if (defined('ANIMALS')) {
```

```
    echo ANIMALS[1]; // outputs "cat"
```

```
}
```

```
?>
```

# Hằng số

---

## □ Từ khóa const

- Định nghĩa một hằng số tại **thời điểm biên dịch**
- Chỉ nhận các giá trị tường minh hoặc biểu thức của chúng
- Ví dụ:

```
<?php
```

```
const CONSTANT = 'Hello World';
```

```
const ANOTHER_CONST = CONSTANT.'; Goodbye World';
```

```
const ANIMALS = array('dog', 'cat', 'bird');
```

```
?>
```

# Kiểu

---

- ❑ Mỗi biểu thức trong PHP có một trong các kiểu sau :
  - ❑ int : 4 bytes, số có dấu
  - ❑ float
  - ❑ bool : true / false
  - ❑ string
  - ❑ array
  - ❑ object
  - ❑ callable
  - ❑ resource

Các hàm kiểm tra kiểu :

- *gettype()*
- *get\_debug\_type()*
- *is\_int()*, *is\_float()*, *is\_bool()*, ...

# Kiểu

---

## □ Kiểu dữ liệu

□ Ví dụ : số nguyên, số thực

<?php

\$a = 1234; // decimal number

\$a = -123; // a negative number

\$a = 0123; // octal number (equivalent to 83 decimal)

\$a = 0x1A; // hexadecimal number (equivalent to 26 decimal)

\$b = 1.234;

\$c = 1.2e3;

\$d = 7E-10;

\$e = 107\_925\_284.88; // PHP 7.4

?>

# Kiểu<sup>2</sup>

---

## □ Kiểu dữ liệu

### □ Ví dụ : luận lý

```
<?php
```

```
$foo = true; // assign the value true to $foo
```

```
if ($action == "show_version") {
```

```
    echo "The version is 1.23";
```

```
}
```

```
// this is not necessary...
```

```
if ($show_separators == true) {
```

```
    echo "<hr>\n";
```

```
}
```

```
// ...because you can simply type
```

```
if ($show_separators) {
```

```
    echo "<hr>\n";
```

```
} ?>
```

# Kiểu

## □ Kiểu dữ liệu

### □ Ví dụ : chuỗi

```
<?php
```

```
$beer = 'Heineken';
```

```
echo "$beer's taste is great"; // works, "" is an invalid character for varnames
```

```
echo "He drank some $beers"; // won't work, 's' is a valid character for varnames
```

```
echo "He drank some ${beer}s"; // works
```

```
echo "He drank some {$beer}s"; // works
```

```
$str = 'This is a test.';
```

```
$third = $str[2]; // Get the third character of a string
```

```
$str = "This is still a test.";
```

```
$last = $str{strlen($str)-1}; // Get the last character of a string.
```

```
$str = 'Look at the sea';
```

```
$str{strlen($str)-1} = 'e'; // Modify the last character of a string
```

```
?>
```

# Kiểu

---

## □ Kiểu dữ liệu

- Mảng trong PHP là có thể được xem như danh sách (list) hoặc từ điển (dictionary)
- Mảng là danh sách (list) khi các khóa là số nguyên, liên tục và bắt đầu bằng 0
- Kiểm tra mảng là một danh sách : *array\_is\_list()* (PHP 8.1)

Ví dụ :

```
<?php
$arr1 = [1, 2, 3, 4, 5];           // hoặc $arr1 = array(1, 2, 3, 4, 5);
$arr2 = ["foo" => "bar", 12 => 1]; // hoặc $arr2 = array("foo" => "bar", 12 => 1);
echo $arr1[0];                     // 1
echo $arr2["foo"];                 // bar
?>
```



# Kiểu

## □ Kiểu dữ liệu

### □ Mảng, ví dụ :

```
<?php
```

```
$arr = ["somearray" => [6 => 5, 13 => 9, "a" => 42]];
```

```
echo $arr["somearray"][6]; // 5
```

```
echo $arr["somearray"][13]; // 9
```

```
echo $arr["somearray"]["a"]; // 42
```

```
// This array is the same as ...
```

```
$a = [5 => 43, 32, 56, "b" => 12];
```

```
// ...this array
```

```
$a_n = [5 => 43, 6 => 32, 7 => 56, "b" => 12];
```

```
?>
```

# Kiểu

---

## □ Kiểu dữ liệu

- Truy xuất các phần tử mảng : `$array_name[key]`

- Ví dụ :

```
<?php
```

```
$arr = [5 => 1, 12 => 2];
```

```
$arr[] = 56; // This is the same as $arr[13] = 56;
```

```
$arr["x"] = 42; // This adds a new element to the array with key "x"
```

```
unset($arr[5]); // This removes the element from the array
```

```
unset($arr); // This deletes the whole array
```

```
?>
```

# Kiểu

---

## □ Kiểu dữ liệu

- Mảng : đếm số phần tử trong mảng

- Ví dụ :

```
<?php
```

```
$cars= ["Volvo","BMW","Toyota"];
```

```
echo count($cars);    // 3
```

```
echo sizeof($cars);    // 3, sizeof() là một tên khác (alias) của count()
```

```
?>
```

# Kiểu

---

## □ Kiểu dữ liệu

- Mảng : `isset()`

- Ví dụ :

```
<?php
```

```
$username = isset($_GET['username']) ? $_GET['username'] : 'nobody';
```

```
// PHP 7 : null coalescing operator (??)
```

```
$username = $_GET['username'] ?? 'nobody';
```

```
$username = $_GET['username'] ?? $_POST['username'] ?? 'nobody';
```

```
?>
```

# Kiểu

---

## □ Kiểu dữ liệu

□ Mảng : `extract()` – chuyển các cặp khóa-giá trị trong mảng thành các biến

□ Ví dụ :

```
<?php
```

```
$var_array = ["color" => "blue",  
              "size"  => "medium",  
              "shape" => "sphere"];
```

```
extract($var_array);
```

```
echo "$color, $size, $shape, $size\n";
```

```
?>
```

□ **Lưu ý**: không dùng `extract()` cho mảng dữ liệu từ người dùng mà chưa qua xử lý (các biến `$_GET`, `$_POST`, ...)

# Kiểu

---

## □ Kiểu dữ liệu

□ Mảng : phân rã mảng (array destructuring)

□ Ví dụ :

```
<?php
```

```
$array = [1, 2, 3];
```

```
list($a, $b, $c) = $array; // Or
```

```
[$a, $b, $c] = $array;
```

```
[, , $c] = $array; // $c = 3
```

```
$array = [ 'a' => 1, 'b' => 2, 'c' => 3, ];
```

```
['c' => $c, 'a' => $a] = $array;
```

```
?>
```

# Kiểu

---

## □ Kiểu dữ liệu

□ Mảng : trải rộng mảng (array spread operator)

□ Ví dụ :

```
<?php
```

```
$arrayA = [1, 2, 3];
```

```
$arrayB = [4, 5];
```

```
$result = [0, ...$arrayA, ...$arrayB, 6, 7]; // [0, 1, 2, 3, 4, 5, 6, 7]
```

```
// PHP 8.1
```

```
$array1 = ["a" => 1];
```

```
$array2 = ["b" => 2];
```

```
$array = ["a" => 0, ...$array1, ...$array2]; // ["a" => 1, "b" => 2]
```

```
?>
```

# Kiểu

---

## □ Kiểu dữ liệu

□ Ví dụ : duyệt mảng, thường dùng foreach

```
<?php
```

```
$array = [1, 2, 3, 4, 5];
```

```
print_r($array);
```

```
foreach ($array as $i => $value)
```

```
    echo $array[$i] . "<br>";
```

```
?>
```



# Toán tử

---

- ❑ Các toán tử trong PHP :  
<https://www.php.net/manual/en/language.operators.php>
- ❑ Các toán tử số học : +, -, \*, /, %, \*\* (lũy thừa)
- ❑ Các toán tử gán : =, +=, -=, \*=, /=, %=
- ❑ Các toán tử so sánh : ==, ===, !=, <>, !==, >, <, >=, <=, <=> (spaceship)
- ❑ Các toán tử tăng giảm : ++, --
- ❑ Các toán tử luận lý : and, or, xor, !, &&, ||

# Toán tử

---

- ❑ Các toán tử chuỗi : . (nối chuỗi), .= (nối và gán chuỗi)
- ❑ Toán tử điều khiển lỗi : *@expression* (đặt cấp độ báo lỗi cho *expression* là 0)
- ❑ Toán tử kiểu : *instanceof*

- 
- ❑ Giới thiệu về PHP
  - ❑ Biến, kiểu dữ liệu, phép toán
  - ❑ **Lệnh điều khiển**
  - ❑ Hàm
  - ❑ PHP include
  - ❑ Xử lý lỗi
  - ❑ Dữ liệu gửi về server
  - ❑ Cookie và Session
  - ❑ HTTP header, Response code, Redirect

# Điều kiện

---

## □ If

### □ Cú pháp :

if (condition)

code to be executed if condition is true;

else

code to be executed if condition is false;

### □ Ví dụ :

```
<?php
```

```
$d=date("D");
```

```
if ($d=="Fri")
```

```
    echo "Have a nice weekend!";
```

```
else
```

```
    echo "Have a nice day!";
```

```
?>
```

# Điều kiện

---

## □ Switch

### □ Cú pháp :

```
switch (expression) {
```

```
case label1:
```

```
    code to be executed if expression = label1;
```

```
    break;
```

```
case label2:
```

```
    code to be executed if expression = label2;
```

```
    break;
```

```
default:
```

```
    code to be executed
```

```
    if expression is different
```

```
    from both label1 and label2;
```

```
}
```

# Điều kiện

---

## □ Switch

□ Ví dụ :

```
<?php
switch ($x) {
case 1:
    echo "Number 1"; break;
case 2:
    echo "Number 2"; break;
case 3:
    echo "Number 3"; break;
default:
    echo "No number between 1 and 3";
}
?>
```

# Điều kiện

---

## □ Match (PHP 8)

### □ Cú pháp :

```
$return_value = match (subject_expression) {  
    single_conditional_expression => return_expression,  
    conditional_expression1, conditional_expression2 => return_expression,  
};
```

# Điều kiện

---

## ❑ Match (PHP 8)

### ❑ Ví dụ :

```
<?php
$statusCode = 400;

$message = match ($statusCode) {
    200, 300 => null,
    400 => 'not found',
    500 => 'server error',
    default => 'unknown status code',
};

// $message = 'not found'
?>
```

```
<?php
// match thực hiện so sánh giá trị
// chặt chẽ hơn so với switch

$statusCode = '200';

$message = match ($statusCode) {
    200 => null,
    default => 'unknown status code',
};

// $message = 'unknown status code'
?>
```



# Lặp

---

## □ While

### □ Cú pháp :

while (condition)

code to be executed;

### □ Ví dụ :

```
<?php
```

```
$i=1;
```

```
while($i<=5) {
```

```
    echo "The number is " . $i . "<br />";
```

```
    $i++;
```

```
}
```

```
?>
```

# Lặp

---

## □ Do ... while

### □ Cú pháp :

```
do {  
    code to be executed;  
} while (condition);
```

### □ Ví dụ :

```
<?php  
$i=0;  
do {  
    $i++;  
    echo "The number is " . $i . "<br />";  
} while ($i<5);  
?>
```

# Lặp

---

## □ For

### □ Cú pháp :

```
for (initialization; condition; increment) {  
    code to be executed;  
}
```

### □ Ví dụ :

```
<?php  
for ($i=1; $i<=5; $i++) {  
    echo "Hello World!<br />";  
}  
?>
```

# Lặp

---

## □ Foreach

### □ Cú pháp :

```
foreach (array as value) {  
    code to be executed;  
}
```

### □ Ví dụ :

```
<?php  
$arr= ["one", "two", "three"];  
foreach ($arr as $value) {  
    echo "Value: " . $value . "<br />";  
}  
?>
```

- 
- ❑ Giới thiệu về PHP
  - ❑ Biến, kiểu dữ liệu, phép toán
  - ❑ Lệnh điều khiển
  - ❑ **Hàm**
  - ❑ PHP include
  - ❑ Xử lý lỗi
  - ❑ Dữ liệu gửi về server
  - ❑ Cookie và Session
  - ❑ HTTP header, Response code, Redirect

# Hàm

---

- ❑ Tham khảo các hàm có sẵn :  
<https://www.php.net/manual/en/funcref.php>
- ❑ Các hàm xử lý mảng :
  - ❑ `array_keys()` : lấy các key trong mảng
  - ❑ `array_map()` : ứng dụng một hàm callback cho từng phần tử trong mảng
  - ❑ `array_pop()` : bỏ phần tử cuối ra khỏi mảng
  - ❑ `array_push()` : thêm phần tử vào cuối mảng
  - ❑ `array_shift()` : dịch phần tử đầu tiên ra khỏi mảng và trả về phần tử đó
  - ❑ `array_slice()` : trích xuất một phần mảng
  - ❑ `array_splice()` : xóa/thay thế một phần mảng

# Hàm

---

## □ Các hàm xử lý chuỗi :

- *ltrim()* / *rtrim()* / *trim()* : bỏ khoảng trắng trong chuỗi
- *strcasecmp()* / *strcmp()* : so sánh chuỗi
- *strlen()* : lấy chiều dài chuỗi
- *substr()* : lấy chuỗi con
- *strtolower()* / *strtoupper()* : biến đổi chuỗi thường / hoa
- *htmlentities()* : chuyển đổi tất cả các ký tự có thể sang các thực thể HTML (để xuất ra trang HTML)
- *htmlspecialchars()* : chuyển đổi các ký tự đặc biệt sang các thực thể HTML (để xuất ra trang HTML)

# Hàm

---

- ❑ Các hàm xử lý ngày và thời gian :
  - ❑ *date()* : định dạng nhãn thời gian Unix địa phương
  - ❑ *microtime()* / *time()* : lấy số micro giây / giây thời gian Unix địa phương
  - ❑ *mktime()* : trả về nhãn thời gian Unix cho một ngày cụ thể
  - ❑ *date\_create()* : tạo đối tượng DateTime từ chuỗi
  - ❑ *date\_diff()* : lấy số ngày khác biệt giữa hai đối tượng DateTime



# Hàm

---

## □ Hàm do người dùng định nghĩa

### □ Cú pháp :

```
<?php
```

```
function foo($arg_1, $arg_2, /* ..., */ $arg_n) {
```

```
    echo "Example function.\n";
```

```
    return $retval;
```

```
}
```

```
?>
```

# Hàm

---

## □ Khai báo kiểu dữ liệu

- Kiểu dữ liệu sau đây có thể được khai báo cho tham số hàm và kiểu trả về: *tên lớp/giao diện, array, callable, bool, float, int, string, iterable, object, mixed, void* (chỉ cho kiểu trả về)
- Mặc định, PHP sẽ thực thi ép kiểu các tham số sang kiểu dữ liệu đã khai báo

### □ Ví dụ :

```
<?php
```

```
function square(int $num) : int {  
    return $num * $num;  
}
```

```
echo square(4); // outputs '16'.
```

```
echo square('3'); // outputs '9'.
```

```
?>
```

# Hàm

---

## ❑ Khai báo kiểu dữ liệu

❑ `declare(strict_types=1)` : bắt buộc các giá trị phải tương thích với kiểu đã khai báo (trong phạm vi một tệp mã nguồn)

❑ Nullable type : thêm ‘?’ trước tên kiểu

❑ Ví dụ :

```
<?php
```

```
function f (?string $a) : void { } // ?string = string | null
```

```
function g (string $a) : void { }
```

```
f(null); // valid
```

```
g(null); // TypeError
```

```
?>
```

# Hàm

---

## ▣ Tham số

- ▣ Truyền tham số : giá trị, tham chiếu

- ▣ Ví dụ tham số là mảng:

```
<?php
```

```
function takes_array(array $input) : void {
```

```
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
```

```
}
```

```
?>
```

# Hàm

---

## □ Tham số

□ Ví dụ số tham số không xác định :

```
<?php
```

```
function sum(int | float ...$numbers) : int | float{
```

```
    $acc = 0;
```

```
    foreach ($numbers as $n) {
```

```
        $acc += $n;
```

```
    }
```

```
    return $acc;
```

```
}
```

```
echo sum(1, 2, 3, 4);
```

```
?>
```

# Hàm

---

## ▣ Tham số

▣ Ví dụ tham số có giá trị mặc định :

```
<?php
```

```
function make_coffee(string $type = "cappuccino") : string {  
    return "Making a cup of $type.<br>";  
}
```

```
echo make_coffee();
```

```
echo make_coffee("espresso");
```

```
?>
```

# Hàm

---

## ▣ Tham số

▣ Ví dụ truyền tham chiếu :

```
<?php
```

```
function add_some_extra(&$string) {
```

```
    $string .= 'and something extra.';
```

```
}
```

```
$str = 'This is a string, ';
```

```
add_some_extra($str);
```

```
echo $str;    // outputs 'This is a string, and something extra.'
```

```
?>
```

# Hàm

---

## ▣ Tham số

▣ Ví dụ truyền tham số theo tên (PHP 8) :

```
<?php
```

```
// Using positional arguments:
```

```
array_fill(0, 100, 50);
```

```
// Using named arguments:
```

```
array_fill(start_index: 0, count: 100, value: 50);
```

```
array_fill(value: 50, count: 100, start_index: 0);
```

```
?>
```



# Hàm

---

## □ Giá trị trả về

□ Ví dụ :

```
<?php
```

```
function square(int $num) : int {
```

```
    return $num * $num;
```

```
}
```

```
echo square(4); // outputs '16'.
```

```
?>
```

# Hàm

---

## □ Giá trị trả về

□ Ví dụ :

```
<?php
```

```
function small_numbers() : array{
```

```
    return array (0, 1, 2);
```

```
}
```

```
list ($zero, $one, $two) = small_numbers();
```

```
?>
```

# Hàm

---

## □ Giá trị trả về

□ Ví dụ :

```
<?php
```

```
function &returns_reference() {
```

```
    return $someref;
```

```
}
```

```
$newref =& returns_reference();
```

```
?>
```

# Hàm

---

## □ Hàm vô danh

### □ Ví dụ :

```
<?php
$greet = function($name) {
    printf("Hello %s\r\n", $name);
};
```

```
$greet('World');
$greet('PHP');
?>
```

```
<?php
```

```
$message = 'world';
```

```
$example = function ($arg) use ($message): string {  
    return $arg . " $message";  
};
```

```
echo $example('hello');
```

```
?>
```

# Hàm

---

□ Hàm mũi tên :  $\text{fn (argument\_list)} \Rightarrow \text{expr}$

□ Ví dụ :

```
<?php
```

```
$y = 1;
```

```
$fn1 = fn($x) => $x + $y;
```

```
// equivalent to using $y by value:
```

```
$fn2 = function ($x) use ($y) {
```

```
    return $x + $y;
```

```
};
```

```
echo $fn1(3);
```

```
?>
```

# Hàm

---

- Kiểu callable : tham chiếu đến một hàm / phương thức
- Ví dụ:

```
<?php
```

```
function print_formatted(callable $format, $str) {  
    echo $format($str) . "<br>";  
}
```

```
// An anonymous function
```

```
$func = function($str) { return substr($str, 0, 5); };  
print_formatted($func , "Hello World");
```

```
// A string containing the name of a function
```

```
print_formatted("strtoupper", "Hello World");
```

# Hàm

- Kiểu callable : tham chiếu đến một hàm / phương thức
- Ví dụ:

```
<?php
```

```
function print_formatted(callable $format, $str) {
```

```
    echo $format($str) . "<br>";
```

```
}
```

```
class MyClass {
```

```
    public static function ask($str) {
```

```
        return $str . "?";
```

```
}
```

```
    public function brackets($str) {
```

```
        return "[$str]";
```

```
}
```

```
}
```

```
// An array describing a static class method
```

```
print_formatted(["MyClass", "ask"], "Hello World");
```

```
// An array describing an object method
```

```
$obj = new MyClass();
```

```
print_formatted(["$obj", "brackets"], "Hello World");
```

- 
- ❑ Giới thiệu về PHP
  - ❑ Biến, kiểu dữ liệu, phép toán
  - ❑ Lệnh điều khiển
  - ❑ Hàm
  - ❑ **PHP include**
  - ❑ Xử lý lỗi
  - ❑ Dữ liệu gửi về server
  - ❑ Cookie và Session
  - ❑ HTTP header, Response code, Redirect



# PHP include

---

- ❑ Chèn/gộp tất cả nội dung của tập tin được chỉ định vào script (và thực thi)
- ❑ Cú pháp :  
    `include 'path_to_file';`                   // hoặc  
    `include_once 'path_to_file';` // hoặc  
    `require 'path_to_file';`                   // hoặc  
    `require_once 'path_to_file';`

# PHP include

---

- Các hằng số liên quan đến đường dẫn
  - **\_\_FILE\_\_** : đường dẫn tuyệt đối đến script chứa hằng số
  - **\_\_DIR\_\_** hoặc **dirname(\_\_FILE\_\_)** : đường dẫn tuyệt đối đến thư mục của script chứa hằng số
- Ví dụ, cho cấu trúc thư mục như sau :

```
project/  
├─ public/  
│   └─ index.php  
└─ bootstrap.php
```

index.php có thể gộp bootstrap.php bởi câu lệnh sau :

**require\_once \_\_DIR\_\_ . '/../bootstrap.php';**

# PHP include

---

- ❑ Đường dẫn bắt đầu với './' hay '../' sẽ tương đối với script thực hiện include
- ❑ Ví dụ : a.php gộp b.php (a.php là script thực hiện gộp, b.php là script bị gộp). *Nếu trong b.php* có gộp c.php và d.php theo dạng :
  - ❑ require '**c.php**' : c.php tương đối với b.php
  - ❑ require './**c.php**': c.php tương đối với a.php thay vì b.php
  - ❑ require **\_\_DIR\_\_** . '/../**d.php**' : d.php tương đối với b.php
  - ❑ require '../**d.php**' : d.php tương đối với a.php thay vì b.php

# PHP include

---

## □ Ví dụ :

```
<html>
```

```
<body>
```

```
<p>Some text</p> <p>Some text</p>
```

```
<?php include 'footer.php';?>
```

```
</body>
```

```
</html>
```

```
// footer.php
```

```
<?php echo "<p>Copyright &copy; 1999-" . date("Y") .  
    " W3Schools.com</p>"; ?>
```

# PHP include

---

## □ Ví dụ :

```
// vars.php
```

```
<?php
```

```
$color='red';
```

```
$car='BMW';
```

```
?>
```

```
// home.php
```

```
<html><body>
```

```
<h1>Welcome to my home page!</h1>
```

```
<?php include 'vars.php';
```

```
    echo "I have a $color $car.";
```

```
?>
```

```
</body></html>
```

# PHP include

---

- ❑ *include vs require* : nếu tập tin được gộp không tồn tại, script sẽ dừng thực thi với require
- ❑ *include vs include\_once* : nếu tập tin đã được gộp vào rồi thì sẽ không được gộp vào lần nữa

- ❑ Ví dụ :

```
<?php
```

```
require "first.php";    // This will include the file
```

```
include_once "first.php"; // This will not as it was included using "require"
```

```
require_once "first.php"; // This will not as it was included using "require"
```

```
?>
```

- 
- ❑ Giới thiệu về PHP
  - ❑ Biến, kiểu dữ liệu, phép toán
  - ❑ Lệnh điều khiển
  - ❑ Hàm
  - ❑ PHP include
  - ❑ **Xử lý lỗi**
  - ❑ Dữ liệu gửi về server
  - ❑ Cookie và Session
  - ❑ HTTP header, Response code, Redirect

# Xử lý lỗi

---

## ❑ Cài đặt báo cáo lỗi

### ❑ Trong tập tin cấu hình **php.ini** :

- *error\_reporting* : cài đặt loại lỗi sẽ báo cáo
- *display\_errors* : lỗi sẽ được hiển thị cho client hay không
- *display\_startup\_errors* : lỗi trong quá trình khởi động (startup) của PHP sẽ được hiển thị cho client hay không

### ❑ Trong script PHP, ví dụ :

```
<?php  
error_reporting(E_ALL);  
ini_set('display_errors', '1');  
ini_set('display_startup_errors', '1');  
...  
?>
```



# Xử lý lỗi

---

## □ Tùy biến xử lý lỗi

□ Ví dụ :

```
<?php
```

```
function my_error_handler ($errno, $errstr, $errfile, $errline) {  
    echo '<br>Oppssss... An error occurred.<br>'. $errstr;  
}
```

```
set_error_handler('my_error_handler');
```

```
...
```

```
?>
```

□ Tuy nhiên một số lỗi (E\_ERROR, E\_PARSE, E\_CORE\_ERROR, E\_CORE\_WARNING, ...) bỏ qua hàm xử lý lỗi của người dùng và làm script kết thúc thực thi (lỗi fatal)

# Xử lý lỗi

---

- ❑ Tùy biến xử lý lỗi fatal (làm script kết thúc thực thi)

- ❑ Ví dụ :

```
<?php
```

```
function my_shutdown_handler() {
```

```
    $lastError = error_get_last();
```

```
    if (isset($lastError)) {
```

```
        echo '<br>Oppsss... Script terminated.<br>';
```

```
    }
```

```
}
```

```
register_shutdown_function('my_shutdown_handler');
```

```
...
```

```
?>
```

- 
- ❑ Giới thiệu về PHP
  - ❑ Biến, kiểu dữ liệu, phép toán
  - ❑ Lệnh điều khiển
  - ❑ Hàm
  - ❑ PHP include
  - ❑ Xử lý lỗi
  - ❑ **Dữ liệu gửi về server**
  - ❑ Cookie và Session
  - ❑ HTTP header, Response code, Redirect

# Dữ liệu gửi về server

---

- ❑ Trong các thẻ của tài liệu HTML, có hai thẻ có thể tạo ra yêu cầu HTTP về server
  - ❑ Thẻ `<a>` : có thể phát sinh yêu cầu HTTP GET đến URL chỉ định trong thuộc tính href
  - ❑ Thẻ `<form>` : có thể phát sinh yêu cầu HTTP GET hoặc POST đến URL chỉ định trong thuộc tính action
  - ❑ Kết quả trả về từ yêu cầu HTTP sinh ra bởi thẻ `<a>` và `<form>` sẽ thay thế toàn bộ nội dung trang HTML hiện thời
  - ❑ Cần phải sử dụng đến JavaScript để có thể phát sinh các yêu cầu HTTP tùy biến hoặc chỉ muốn cập nhật lại một phần của trang HTML

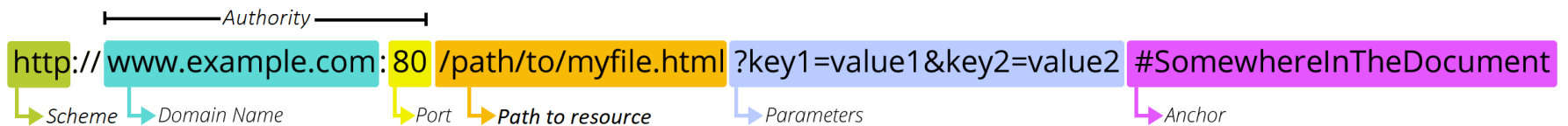
# Dữ liệu gửi về server

---

- ❑ Trình duyệt web phát sinh yêu cầu HTTP do đó có thể gửi dữ liệu về server :
  - ❑ Thông qua URL của yêu cầu HTTP
  - ❑ Thông qua các header của yêu cầu HTTP
  - ❑ Thông qua phần thân của yêu cầu HTTP

# Dữ liệu gửi về server

- Gửi qua tham số URL (chuỗi truy vấn)
  - URL (Uniform Resource Locator) : xác định một tài nguyên bằng cách cho biết địa chỉ của tài nguyên
  - Trong ngữ cảnh của Web/HTTP, URL thường được gọi là địa chỉ Web, hay liên kết (link)

The diagram shows the URL `http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument` with its parts highlighted in different colors and labeled with arrows: `http://` is green and labeled 'Scheme'; `www.example.com` is cyan and labeled 'Domain Name'; `:80` is yellow and labeled 'Port'; `/path/to/myfile.html` is orange and labeled 'Path to resource'; `?key1=value1&key2=value2` is light blue and labeled 'Parameters'; `#SomewhereInTheDocument` is purple and labeled 'Anchor'. A bracket above the first three parts is labeled 'Authority'. A red arrow points from the text 'Chuỗi truy vấn' in the following list item to the 'Parameters' part of the URL.

- Chuỗi truy vấn (query string) : một phần của URL gán giá trị cho các tham số được chỉ định

# Dữ liệu gửi về server

## □ Gửi qua tham số URL (chuỗi truy vấn)

- Chuỗi truy vấn (query string) : một phần của URL gán giá trị cho các tham số được chỉ định
- Dữ liệu trên HTML Form gửi qua phương thức GET sẽ nằm ở chuỗi truy vấn
- Sử dụng biến `$_GET` để truy xuất giá trị các tham số
- Ví dụ :

`https://localhost/example.php?name=ferret&color=purple`

`// example.php`

`<html>`

`<body>`

`Name: <?php echo $_GET["name"] ?? 'nobody'; ?>.<br>`

`Color: <?php echo $_GET["color"] ?? 'white'; ?>.`

`</body></html>`

# Dữ liệu gửi về server

## ❑ PHP kết hợp với HTML Form

- ❑ Hầu hết các thành phần của HTML Form đều có thể được truy xuất từ script PHP
- ❑ Sử dụng biến `$_POST` để truy xuất đến các thành phần của HTML Form gửi về theo phương thức POST
- ❑ Ví dụ : trang web là `welcome.html` nội dung như sau

```
<html>
```

```
<body>
```

```
<form action="welcome.php" method="POST">
```

```
Enter your name: <input type="text" name="name">
```

```
Enter your age: <input type="text" name="age">
```

```
<input type="submit" value="welcome">
```

```
</form>
```

```
</body>
```

```
</html>
```



# Dữ liệu gửi về server

---

## ❑ PHP kết hợp với HTML Form

❑ PHP script **"welcome.php"** sử dụng biến `$_POST` để truy xuất đến các thành phần của HTML Form do sử dụng **`method="POST"`**

❑ PHP script `welcome.php` nội dung như sau

```
<html>
```

```
<body>
```

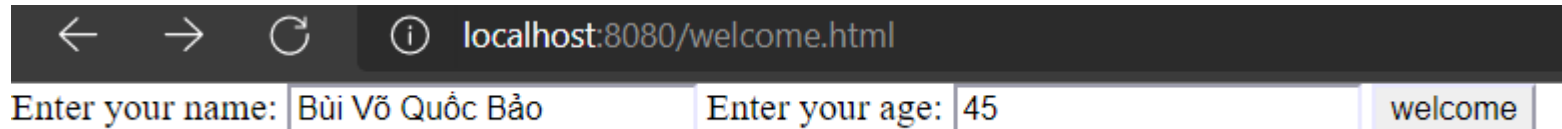
```
Welcome <?= $_POST["name"] ?? 'nobody'; ?>.<br>
```

```
You are <?= $_POST["age"] ?? 18; ?> years old!
```

```
</body>
```

```
</html>
```

# Dữ liệu gửi về server



← → ↻ ⓘ localhost:8080/welcome.html

Enter your name: Bùi Võ Quốc Bảo Enter your age: 45 welcome

Wireshark · Follow HTTP Stream (tcp.stream eq 17) · Adapter for loopback traffic capture

POST /welcome.php HTTP/1.1  
Host: localhost:8080  
Connection: keep-alive  
Content-Length: 53  
Cache-Control: max-age=0  
sec-ch-ua: " Not;A Brand";v="99", "Microsoft Edge";v="97", "Chromium";v="97"  
sec-ch-ua-mobile: ?0  
sec-ch-ua-platform: "Windows"  
Upgrade-Insecure-Requests: 1  
Origin: http://localhost:8080  
Content-Type: application/x-www-form-urlencoded  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safari/537.36 Edg/97.0.1072.62  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: navigate  
Sec-Fetch-User: ?1  
Sec-Fetch-Dest: document  
Referer: http://localhost:8080/welcome.html  
Accept-Encoding: gzip, deflate, br  
Accept-Language: en-US,en;q=0.9,vi;q=0.8  
Cookie: MOODLEID1\_=NqN%25B8%2586; PHPSESSID=kc6be9qm6336fn83bibsm7t18v; \_xsrf=2|46f8a195|e7b1de93376056b0a5578ee66785ef45|1642591848  
  
name=B%C3%B9i+V%C3%B5+Qu%E1%BB%91c+B%E1%BA%A3o&age=45

Thông điệp yêu cầu POST /welcome.php

# Dữ liệu gửi về server

Header **Content-Type** cho biết định dạng dữ liệu gửi về server

```
POST /welcome.php HTTP/1.1
```

```
Host: localhost:8080
Connection: keep-alive
Content-Length: 53
Cache-Control: max-age=0
sec-ch-ua: "Not;A Brand";v="99", "Microsoft Edge";v="97", "Chromium";v="97"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
```

```
Content-Type: application/x-www-form-urlencoded
```

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safari/537.36 Edg/97.0.1072.62
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8080/welcome.html
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,vi;q=0.8
Cookie: MOODLEID1_=NqN%25B8%2586; PHPSESSID=kc6be9qm6336fn83bibsm7t18v; _xsrf=2|46f8a195|e7b1de93376056b0a5578ee66785ef45|1642591848
```

```
name=B%C3%B9i+V%C3%B5+Qu%E1%BB%91c+B%E1%BA%A3o&age=45
```

# Dữ liệu gửi về server

Wireshark · Follow HTTP Stream (tcp.stream eq 17) · Adapter for loopback traffic capture

```
HTTP/1.1 200 OK
Host: localhost:8080
Date: Thu, 20 Jan 2022 14:10:12 GMT
Connection: close
X-Powered-By: PHP/8.0.11
Content-type: text/html; charset=UTF-8
```

```
<html>
<body>
  Welcome Bùi Võ Quốc Bảo.<br>
  You are 45 years old!
</body>
</html>
```

← → ↻ ⓘ localhost:8080/welcome.php

Welcome Bùi Võ Quốc Bảo.  
You are 45 years old!

Thông điệp trả lời

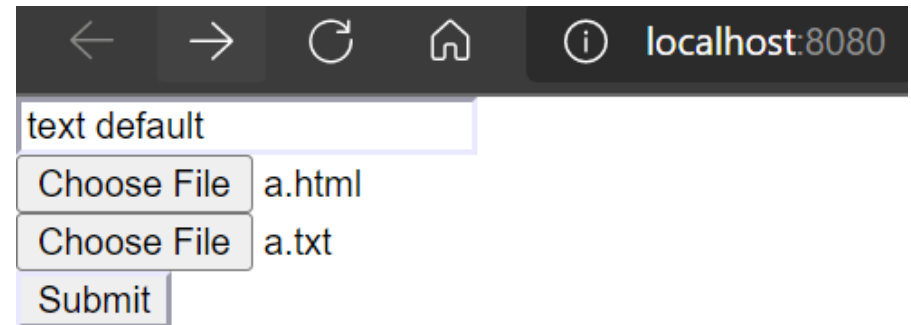
# Dữ liệu gửi về server

## □ Gửi file về server

□ Thêm thuộc tính **enctype="multipart/form-data"** vào form

□ Ví dụ :

```
<form
  action="upload.php"
  method="post"
  enctype="multipart/form-data"
>
  <p><input type="text" name="text" value="text default" /></p>
  <p><input type="file" name="file1" /></p>
  <p><input type="file" name="file2" /></p>
  <p><button type="submit">Submit</button></p>
</form>
```



# Dữ liệu gửi về server

## □ Gửi file về server

### □ Nội dung gói tin yêu cầu HTTP :

```
POST /upload.php HTTP/1.1
Host: localhost:8080
... (các header khác được loại bỏ)
Content-Length: 484
... (các header khác được loại bỏ)
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBJo5nSHgOAVzB7q2
... (các header khác được loại bỏ)

-----WebKitFormBoundaryBJo5nSHgOAVzB7q2
Content-Disposition: form-data; name="text"

text default
-----WebKitFormBoundaryBJo5nSHgOAVzB7q2
Content-Disposition: form-data; name="file1"; filename="a.html"
Content-Type: text/html

<!DOCTYPE html><title>Content of a.html.</title>
-----WebKitFormBoundaryBJo5nSHgOAVzB7q2
Content-Disposition: form-data; name="file2"; filename="a.txt"
Content-Type: text/plain

Content of a.txt.
-----WebKitFormBoundaryBJo5nSHgOAVzB7q2--
```

# Dữ liệu gửi về server

## □ Gửi file về server

- File được gửi về server sẽ nằm trong thư mục tạm, dùng hàm **move\_uploaded\_file()** để di chuyển file đến vị trí khác
- Dùng biến **\$\_FILES** để truy cập thông file

```
<?php
```

```
// CẦN KIỂM TRA KỸ LƯỜNG HƠN TRƯỚC KHI LƯU FILE TRÊN SERVER
$upload_dir = 'uploads/';
$upload_file1 = $upload_dir . basename($_FILES['file1']['name']);
$upload_file2 = $upload_dir . basename($_FILES['file2']['name']);
move_uploaded_file($_FILES['file1']['tmp_name'], $upload_file1);
move_uploaded_file($_FILES['file2']['tmp_name'], $upload_file2);
```

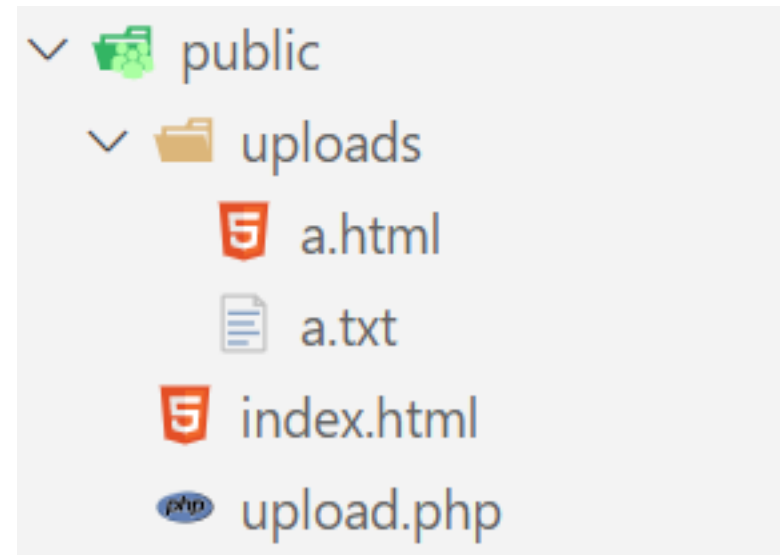
```
header('content-type: application/json');
echo json_encode([
    '$_POST' => $_POST,
    '$_FILES' => $_FILES
]);
```

**upload.php**

# Dữ liệu gửi về server

## □ Gửi file về server

```
{
  "$_POST": {
    "text": "text default"
  },
  "$_FILES": {
    "file1": {
      "name": "a.html",
      "full_path": "a.html",
      "type": "text/html",
      "tmp_name": "C:\\xampp\\tmp\\php974C.tmp",
      "error": 0,
      "size": 48
    },
    "file2": {
      "name": "a.txt",
      "full_path": "a.txt",
      "type": "text/plain",
      "tmp_name": "C:\\xampp\\tmp\\php974D.tmp",
      "error": 0,
      "size": 17
    }
  }
}
```





# Dữ liệu gửi về server

---

## □ Gửi dữ liệu JSON

- Biến `$_POST` chỉ dùng để truy xuất dữ liệu gửi về script dạng *application/x-www-form-urlencoded* hoặc *multipart/form-data*
- Để đọc dữ liệu gửi về ở định dạng khác như *application/json*, cần đọc từ stream **`php://input`** (stream dữ liệu thô phần body của yêu cầu HTTP)

□ Ví dụ : `<?php`

```
// Get the JSON contents
$json = file_get_contents('php://input');

// decode the json data
$data = json_decode($json);
?>
```

# Dữ liệu gửi về server

## □ Phân biệt yêu cầu GET và POST

- Làm thế nào script biết được yêu cầu đang được xử lý là yêu cầu GET hay POST?
- Phương thức yêu cầu được lưu trong biến : **`$_SERVER['REQUEST_METHOD']`**
- Ví dụ :

```
<?php if ($_SERVER['REQUEST_METHOD'] === 'GET') : ?>
    <form method="POST">
        Enter your name: <input type="text" name="name">
        <br>
        Enter your age: <input type="number" name="age">
        <input type="submit" value="Submit">
    </form>
<?php elseif ($_SERVER['REQUEST_METHOD'] === 'POST') : ?>
    <h1>Hello, <?php echo $_POST['name']; ?></h1>
    <h2>Your age is: <?php echo $_POST['age']; ?></h2>
<?php endif; ?>
```

# Dữ liệu gửi về server

- ❑ Biến **\$\_SERVER** :  
lưu trữ thông tin về  
môi trường thực thi  
và server

servervars.php

```
1 <?php
2 header('content-type: application/json');
3 echo json_encode($_SERVER);
```

```
localhost:8080/servervars.php

{
  "DOCUMENT_ROOT": "D:\\Projects\\php-sandbox",
  "REMOTE_ADDR": "::1",
  "REMOTE_PORT": "63545",
  "SERVER_SOFTWARE": "PHP 8.1.2 Development Server",
  "SERVER_PROTOCOL": "HTTP/1.1",
  "SERVER_NAME": "localhost",
  "SERVER_PORT": "8080",
  "REQUEST_URI": "/servervars.php",
  "REQUEST_METHOD": "GET",
  "SCRIPT_NAME": "/servervars.php",
  "SCRIPT_FILENAME": "D:\\Projects\\php-sandbox\\servervars.php",
  "PHP_SELF": "/servervars.php",
  "HTTP_HOST": "localhost:8080",
  "HTTP_CONNECTION": "keep-alive",
  "HTTP_SEC_CH_UA": "\" Not A;Brand\";v=\"99\", \"Chromium\";v=\"",
  "HTTP_SEC_CH_UA_MOBILE": "?0",
  "HTTP_SEC_CH_UA_PLATFORM": "\"Windows\"",
  "HTTP_UPGRADE_INSECURE_REQUESTS": "1",
  "HTTP_USER_AGENT": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) A",
  "HTTP_ACCEPT": "text/html,application/xhtml+xml,application/xml",
  "HTTP_SEC_FETCH_SITE": "none",
  "HTTP_SEC_FETCH_MODE": "navigate",
  "HTTP_SEC_FETCH_USER": "?1",
  "HTTP_SEC_FETCH_DEST": "document",
  "HTTP_ACCEPT_ENCODING": "gzip, deflate, br",
  "HTTP_ACCEPT_LANGUAGE": "en-US,en;q=0.9,vi;q=0.8",
  "REQUEST_TIME_FLOAT": 1646370404.569264,
  "REQUEST_TIME": 1646370404
}
```

# Dữ liệu gửi về server

## □ Kiểm tra và chuẩn hóa dữ liệu

- Không nên tin tưởng dữ liệu do người dùng gửi về : cần kiểm tra, chuẩn hóa hoặc sà n lọc dữ liệu trước khi dung/lưu trữ
- Khi ***xuất dữ liệu*** sang một môi trường khác cần “*chuẩn hóa*” (*escaping*) dữ liệu cho phù hợp với môi trường đích
  - *htmlentities()* / *htmlspecialchars()* : chuyển tất cả / chỉ các ký tự đặc biệt sang các ký tự HTML (ví dụ " thành &quot;), dùng khi xuất dữ liệu (nhận từ người dùng) ra trang HTML

<html>

<body>

Welcome <?php echo *htmlspecialchars*(\$\_POST["name"]); ?>.<br>

You are <?php echo *htmlspecialchars*(\$\_POST["age"]); ?> years old!

</body>

</html>

# Dữ liệu gửi về server

## □ Kiểm tra và chuẩn hóa dữ liệu

- *filter\_var()*, *filter\_input()*,... : dùng để *lọc bỏ (sanitize)* hoặc *kiểm tra (validate)* các ký tự không hợp lệ trong biến dữ liệu

```
<?php
$float = 0.032;
$not_float = "0.03b2";

var_dump(filter_var(
    $float,
    FILTER_SANITIZE_NUMBER_FLOAT,
    FILTER_FLAG_ALLOW_FRACTION
));
var_dump(filter_var(
    $not_float,
    FILTER_SANITIZE_NUMBER_FLOAT,
    FILTER_FLAG_ALLOW_FRACTION
));
```

Với FILTER\_SANITIZE\_\*,  
filter\_var() trả về các giá trị  
dạng chuỗi

```
string(5) "0.032"
string(5) "0.032"
```

# Dữ liệu gửi về server

## □ Kiểm tra và chuẩn hóa dữ liệu

- *filter\_var()*, *filter\_input()*,... : dùng để *lọc bỏ (sanitize)* hoặc *kiểm tra (validate)* các ký tự không hợp lệ trong biến dữ liệu

```
<?php
$float = 0.032;
$not_float = "0.03b2";

var_dump(filter_var($float, FILTER_VALIDATE_FLOAT));
var_dump(filter_var($not_float, FILTER_VALIDATE_FLOAT));
```

Với FILTER\_VALIDATE\_\*,  
filter\_var() trả về false nếu  
không đúng kiểu kiểm tra

float(0.032)  
bool(false)

- 
- ❑ Giới thiệu về PHP
  - ❑ Biến, kiểu dữ liệu, phép toán
  - ❑ Lệnh điều khiển
  - ❑ Hàm
  - ❑ PHP include
  - ❑ Xử lý lỗi
  - ❑ Dữ liệu gửi về server
  - ❑ **Cookie và Session**
  - ❑ HTTP header, Response code, Redirect

# Cookie và Session

---

- ❑ Giao thức HTTP là vô trạng thái (stateless) : các yêu cầu HTTP khác nhau là độc lập nhau
- ❑ Giải pháp cho lưu trữ dữ liệu trạng thái qua các lần tải trang :
  - ❑ Cookie : dữ liệu lưu trữ phía client
  - ❑ Session : dữ liệu lưu trữ phía server, session ID được gửi về trình duyệt dạng cookie



# Cookie

---

- ❑ Dữ liệu cookie được tạo bởi server và được đặt vào một tập tin *phía máy client* bởi trình duyệt web
- ❑ Cookie được gửi về server cho *mỗi yêu cầu* HTTP
- ❑ Một số hạn chế (tùy vào trình duyệt) :
  - ❑ Kích thước : ~4096 bytes/cookie
  - ❑ ~180 cookies/site
- ❑ PHP cho phép tạo và đọc lại những giá trị từ cookie
  - ❑ Tạo cookie : `setcookie(name, value, expire, path, domain)`
  - ❑ Đọc cookie : `$_COOKIE`

# Cookie

---

## □ Hàm tạo cookie :

setcookie(name, value, expire, path, domain);

□ Phải được gọi trước khi bất kỳ nội dung nào được gửi đi

□ Ví dụ :

```
<?php setcookie('uname', 'baobui', time()+36000); /* thời hạn 10 giờ */ ?>
```

```
<html>
```

```
<body>
```

```
<p> A cookie was set on this page!
```

```
    The cookie will be active when
```

```
    the client has sent the cookie back to the server. </p>
```

```
</body>
```

```
</html>
```

# Cookie

---

## □ Đọc dữ liệu cookie : dùng biến `$_COOKIE`

### □ Ví dụ :

```
<html>
<body>
<?php
if (isset($_COOKIE["uname"]))
    echo "Welcome " . $_COOKIE["uname"] . " !<br />";
else
    echo "You are not logged in!<br />";
?>
</body>
</html>
```

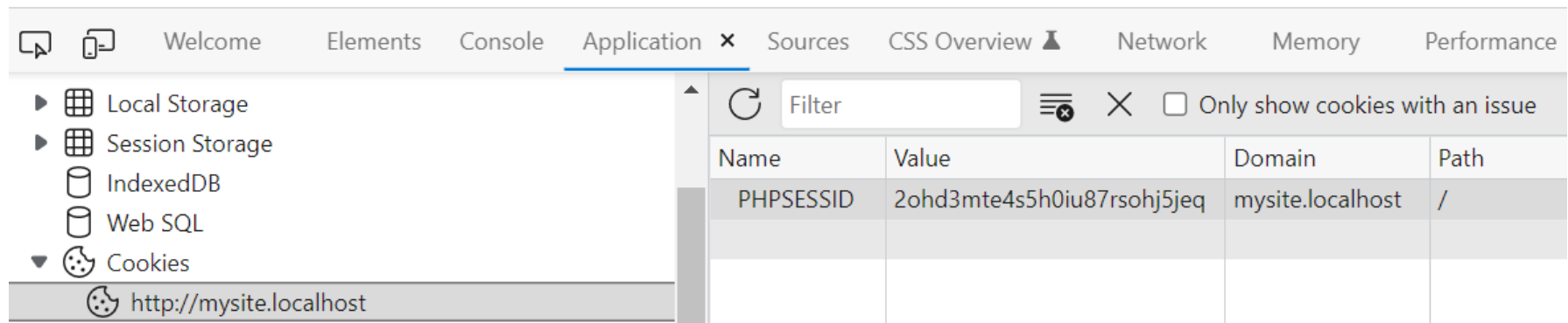
# Session

---

- ❑ Khác với cookie, dữ liệu của session không lưu trên máy tính người dùng
  - ❑ Session có thể được lưu trữ dưới dạng tập tin trên server, trong một CSDL, ... (session.save\_handler, session.save\_path)
- ❑ Mặc định, session *có thể* bị hủy sau 1440 giây/24 phút từ lần thay đổi cuối cùng (session.gc\_maxlifetime=1440)
- ❑ Các hàm làm việc với session
  - ❑ Bật session : session\_start();
  - ❑ Bỏ một biến trong session : unset(\$\_SESSION['varname']);
  - ❑ Bỏ tất cả các biến session : session\_unset();
  - ❑ Xóa session : session\_destroy();

# Session

- ❑ session\_start(): bắt đầu hay khôi phục lại phiên làm việc dựa trên một ID gửi về theo yêu cầu GET, POST hoặc cookie
  - ❑ Phải được gọi trước khi dùng biến `$_SESSION`
  - ❑ Mặc định ID được lưu phía client với dạng cookie



# Session

---

□ Ví dụ :

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

demo\_session1.php

```
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

# Session

---

## □ Ví dụ :

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

demo\_session2.php

```
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . "<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

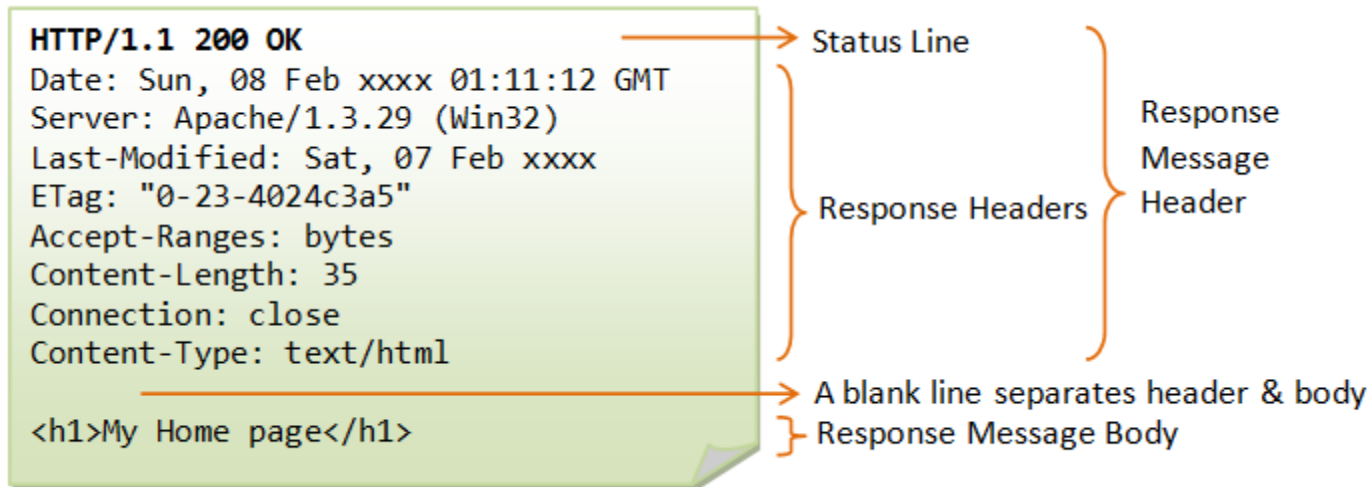
</body>
</html>
```

- 
- ❑ Giới thiệu về PHP
  - ❑ Biến, kiểu dữ liệu, phép toán
  - ❑ Lệnh điều khiển
  - ❑ Hàm
  - ❑ PHP include
  - ❑ Xử lý lỗi
  - ❑ Dữ liệu gửi về server
  - ❑ Cookie và Session
  - ❑ **HTTP header, Response code, Redirect**



# HTTP Header, Response code, Redirect

- Cấu trúc của một thông điệp trả lời HTTP gồm
  - Dòng trạng thái
  - Các header
  - Thân/nội dung thông điệp trả lời



# HTTP Header, Response code, Redirect

---

## □ HTTP header, Response code, Redirect

- Dùng *http\_response\_code(code)* để gửi mã trạng thái cùng với thông điệp trạng thái mặc định
- Dùng hàm *header()* để gửi dòng trạng thái và các HTTP header

Cú pháp :

*header(string \$header, bool \$replace = true, int \$response\_code = 0): void*

- Nếu *\$header* bắt đầu với chuỗi "HTTP/" hoặc "http/", mã trạng thái sẽ được tự động nhận biết
- Nếu *\$header* bắt đầu với chuỗi "Location:", mã trạng thái chuyển hướng trang (302) sẽ được gửi

# HTTP Header, Response code, Redirect

---

## □ HTTP header, Response code, Redirect

□ Ví dụ :

```
<?php  
header('Content-Type: application/json');  
?>
```

```
<?php  
header($_SERVER["SERVER_PROTOCOL"] . " 404 Not Found");  
?>
```

```
<?php  
header("Location: /about.php", true, 302);  
exit;  
?>
```

# HTTP Header, Response code, Redirect

---

- HTTP header, Response code, Redirect
  - *Thân thông điệp trả lời bao gồm tất cả các ký tự nằm ngoài khối lệnh <?php ?> cùng với các nội dung được echo/print*
  - *header()* phải được gọi trước khi bất kỳ nội dung trong thân thông điệp trả lời nào được gửi đi
    - Có thể dùng hàm *headers\_sent()* để kiểm tra
  - **Chú ý** : các nội dung bên ngoài khối lệnh <?php ?> được đưa vào bộ đệm xuất và gửi về client

# HTTP Header, Response code, Redirect

---

## □ HTTP header, Response code, Redirect

- **Chú ý** : các nội dung bên ngoài khối lệnh `<?php ?>` được đưa vào bộ đệm xuất và gửi về client
- Ví dụ, script sau đây sẽ bị lỗi do chuỗi `<html>` được gửi về trước `header()`:

**`<html>`**

`<?php`

`// This will result in an error due to the <html> tag above`

`header('Location: http://www.example.com/');`

`exit;`

`?>`

- Để hạn chế lỗi những lỗi tương tự như trên, các script chỉ chứa PHP code thường sẽ bỏ qua thẻ đóng `?>`

# HTTP Header, Response code, Redirect

---

## □ HTTP header, Response code, Redirect

- Có thể dùng cơ chế bộ đệm xuất (output buffering) để điều khiển khi nào dữ liệu sẽ được gửi về client

- Ví dụ :

```
<?php
```

```
ob_start();
```

```
?>
```

```
<html><body>
```

```
    <p>It's like comparing apples to oranges.</p></body></html>
```

```
<?php
```

```
$page = ob_get_contents();
```

```
ob_end_clean();
```

```
echo str_replace("apples", "oranges", $page);
```

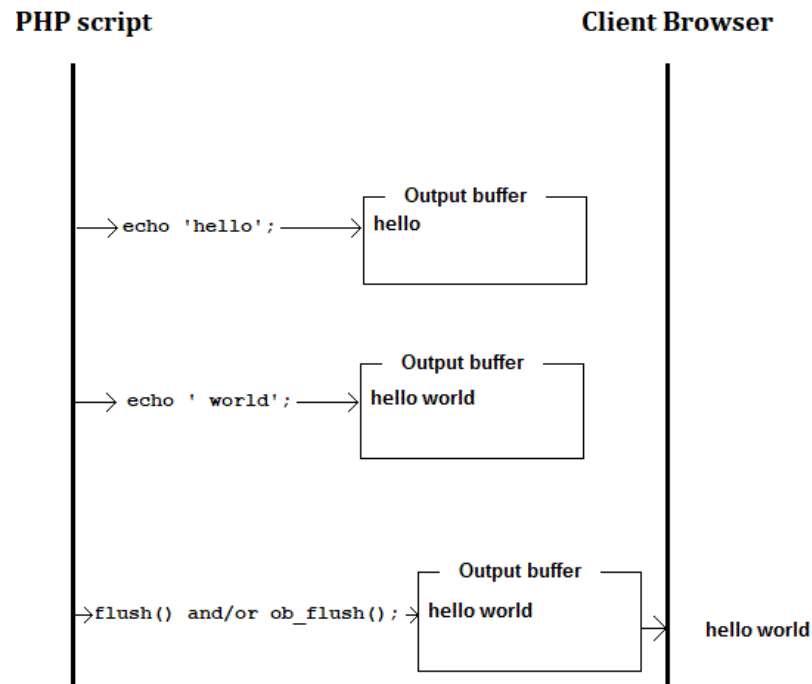
```
?>
```

# HTTP Header, Response code, Redirect

## □ HTTP header, Response code, Redirect

- Có thể dùng cơ chế bộ đệm xuất (output buffering) để điều khiển khi nào dữ liệu sẽ được gửi về client

### Output buffering





Cám ơn !