

Phát Triển Phần Mềm Mã Nguồn Mở

NGUYỄN HẢI TRIỀU

Bộ môn Kỹ thuật phần mềm
Khoa Công nghệ thông tin
ĐH Nha Trang

Mục lục

- 1 Giới thiệu
- 2 Ngôn ngữ lập trình mã nguồn mở **PHP**
- 3 PHP–Ngôn Ngữ Lập Trình Hướng Đối Tượng

- 1 Giới thiệu
- 2 Ngôn ngữ lập trình mã nguồn mở PHP
- 3 PHP–Ngôn Ngữ Lập Trình Hướng Đối Tượng



Mục tiêu: cung cấp cho người học các kiến thức cơ bản về ngôn ngữ lập trình **PHP**. Cụ thể, chúng ta sẽ tập trung vào

- các khái niệm cơ bản khi thiết kế Web, đặc điểm của PHP
- biến và hằng, kiểu dữ liệu, các toán tử, các cấu trúc điều khiển
- các kiến thức liên quan đến lập trình PHP

□ Môi trường thực hành

- Do PHP là ngôn ngữ **chạy ở phía server** nên cần phải sử dụng phần mềm để tạo web server trên HĐH *Windows* như **XAMPP**/WAMP hoặc **LAMP** stack (Linux, Apache, MySQL và PHP) trên HĐH *linux*.
- IDEs** để code PHP thường được sử dụng là: PhpStorm, VS code, Eclipse, Notepad++/Sublime. Để đơn giản nên sử dụng **PhpStorm** hoặc **VS code**.



- 1 Giới thiệu
- 2 Ngôn ngữ lập trình mã nguồn mở **PHP****
- 3 PHP–Ngôn Ngữ Lập Trình Hướng Đối Tượng

Trước khi bắt đầu với định nghĩa PHP là gì, chúng ta nhắc lại các khái niệm

- HTML (Hypertext Markup Language)
- Web Programming Languages
- Web Server/ Web Client
- Database Server
- Web Browser
- URL (Uniform Resource Locator)
- HTTP (Hypertext Transfer Protocol)
- HTTPS (Hyper Text Transfer Protocol Secure)

HTML

HTML là viết tắt của cụm từ **HyperText Markup Language**. HTML được sử dụng để tạo và cấu trúc các phần, đoạn văn, tiêu đề, liên kết và chuỗi khối cho các trang web và ứng dụng.

Các phần tử HTML được biểu thị bằng các thẻ `<>`. Ví dụ:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Page Title</title>
5   </head>
6   <body>
7     <h1>This is a Heading</h1>
8     <p>This is a paragraph.</p>
9   </body>
10 </html>
```

Web Programming Languages

Web Programming Languages là các ngôn ngữ lập trình Web, các ngôn ngữ phổ biến được dùng hiện nay:

- **Front-end(Client-side)**: JavaScript, Type script
- **Back-end (Server-side)**: C++, PHP, Python, Hack, Java

Websites	Front-end	Back-end
Google.com	JavaScript, Type script	C, C++, Go, Java, Python, ...
Facebook.com	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D, XHP, Haskell
YouTube.com	JavaScript	C, C++, Python, Java, Go
Yahoo	JavaScript	PHP
Amazon.com	JavaScript	Java, C++, Perl
Wikipedia.org	JavaScript	PHP, Hack

Bảng 1: Các ngôn ngữ Web được sử dụng ở các Website lớn (nguồn: Wikipedia). Tham khảo số lượng người dùng các ngôn ngữ trên (tính đến năm 2019) ở đây <https://www.youtube.com/watch?v=Og847HVwRSI>

Web server/Web Client

Web server có thể xét về khía cạnh phần cứng hoặc phần mềm:

- **Phần cứng**

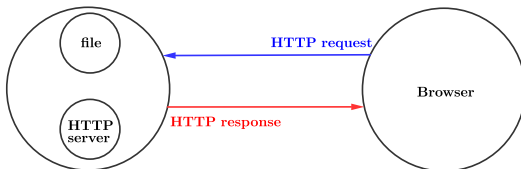
- ▶ là một máy tính lưu trữ các file thành phần của một website (ví dụ: các tài liệu HTML, các file ảnh, CSS và các file JavaScript) và có thể phân phát chúng tới thiết bị của người dùng cuối (end-user)
- ▶ nó kết nối tới mạng Internet và có thể truy cập tới thông qua một tên miền

Web server/Web Client

● Phần mềm

- ▶ gồm một số phần mềm điều khiển cách người sử dụng web truy cập tới các file được lưu trữ trên một HTTP server (máy chủ HTTP) và trả kết quả về cho **Web Client** khi nhận được yêu cầu.
- ▶ một HTTP server là một phần mềm hiểu được các URL (các địa chỉ web) và HTTP (giao thức trình duyệt của bạn sử dụng để xem các trang web)
- ▶ tất cả các Web Server đều hiểu và chạy được các file *.htm và *.html, tuy nhiên mỗi Web Server lại phục vụ một số kiểu file chuyên biệt chẳng hạn như IIS dành cho *.asp, *.aspx...; Apache dành cho *.php...; Sun Java System Web Server dành cho *.jsp...

Web server/Web Client



Hình 1: mô tả hoạt động của web server

Web Client

- máy tính dùng để truy cập các trang web
- gửi các yêu cầu tới máy có chương trình server và chờ đợi câu trả lời từ Web Server.

Database Server

Database Server

- là **máy chủ** được cài đặt các **phần mềm Hệ quản trị cơ sở dữ liệu** (HQTCSĐL) như SQL server, mySQL, Oracle ...
- một số HQTCSĐL phổ biến: **MySQL**, Oracle, PostgreSQL, SQL Server, DB2, Infomix,...
- **điểm chung** của các hệ quản trị CSDL này là chúng **đều sử dụng ngôn ngữ truy vấn theo cấu trúc** (SQL - Structured Query Language)

Web Browser

Web Browser là một phần mềm ứng dụng được gọi tắt là trình duyệt Web giúp người dùng có thể nhìn thấy và tương tác với các văn bản, hình ảnh, nhạc, đoạn video hoặc phim, các trò chơi và nhiều thông tin khác xuất hiện trên một trang web bất kì. Một số trình duyệt web thông dụng:

- Google Chrome
- Mozilla Firefox
- Safari
- Opera
- Microsoft Edge



URL

URL (“Uniform Resource Locator”) là phương tiện để người dùng sử dụng truy cập đến các tài nguyên trên mạng Internet. Một Đường dẫn URL thường có cấu trúc như sau:

```
URL_scheme://<host>[:port] [<path> [?<querystring>] ]
```

- **URL_scheme**: là các giao thức kết nối (http, https, ftp, mailto)
- **host**: là địa chỉ website bất kì, bao gồm **www**, tên miền website, và phần đuôi theo khu vực hoặc quốc tế.
- **port**: cổng dịch vụ trên máy chủ
- **path**: đường dẫn tuyệt đối của URL
- **querystring**: các truy vấn, các mục con, dùng kí tự “/” để phân chia giữa các thành phần truy vấn, mục con.

URL

Ví dụ **URL**:

- `https://www.google.com.vn/`
- `https://www.facebook.com/abcxyz/`
- `http://ntu.edu.vn/`
- `https://elearning.ntu.edu.vn/login/index.php`
- `https://drive.google.com:443`

HTTP/HTTPS

HTTP (HyperText Transfer Protocol)

- là **giao thức** truyền tải siêu văn bản
- hoạt động dựa trên **mô hình Client–Server**
- giao thức này là tập hợp các qui định dùng để **trao đổi** các tài liệu (văn bản, hình ảnh, âm thanh, video, các tập tin đa truyền thông...) giữa Web server và trình duyệt Web
- các **thông tin** được chuyển đi qua giao thức HTTP (bao gồm địa chỉ IP, các thông tin nhập liệu trên Website...) **không được mã hóa và bảo mật**

HTTP/HTTPS

HTTPS (“Hypertext Transfer Protocol Secure”)

- là giao thức truyền tải siêu văn bản bảo mật hay phiên bản an toàn của HTTP.
- giúp cho việc trao đổi thông tin một cách bảo mật trên nền Internet.

Giới thiệu ngôn ngữ PHP

PHP là viết tắt của **PHP** Hypertext **P**reprocessor do *Rasmus Lerdorf* tạo ra vào năm 1995, là ngôn ngữ lập trình **phía server** dùng để **xây dựng và phát triển các ứng dụng Website**. PHP có các đặc điểm như sau:

- chạy trên máy chủ server
- tập tin PHP có phần mở rộng là **.php**
- rất đơn giản, tốc độ xử lý nhanh.
- tính cộng đồng cao, hỗ trợ từ nhiều lập trình viên, có khá nhiều CMS, Framework được xây dựng từ PHP giúp rút gọn quá trình tạo một website

Giới thiệu ngôn ngữ PHP

- cú pháp ngôn ngữ giống ngôn ngữ C & Perl
- miễn phí, có thể chạy được trên bất cứ hệ điều hành nào.
- Có thể dễ dàng nối kết với các HQTCSĐL như: MySQL, Microsoft SQL Server 2000, Oracle, PostgreSQL, Adabas, dBase, Informix ...
- PHP không chỉ làm việc với HTML mà còn có thể làm việc được với hình ảnh, PDF, Flash movie, ...

Giới thiệu ngôn ngữ PHP

Một số website được viết bằng PHP:

- WordPress
- Facebook
- Wikipedia
- Stackoverflow
- Yahoo
- Moodle

Thiết lập môi trường để chạy PHP

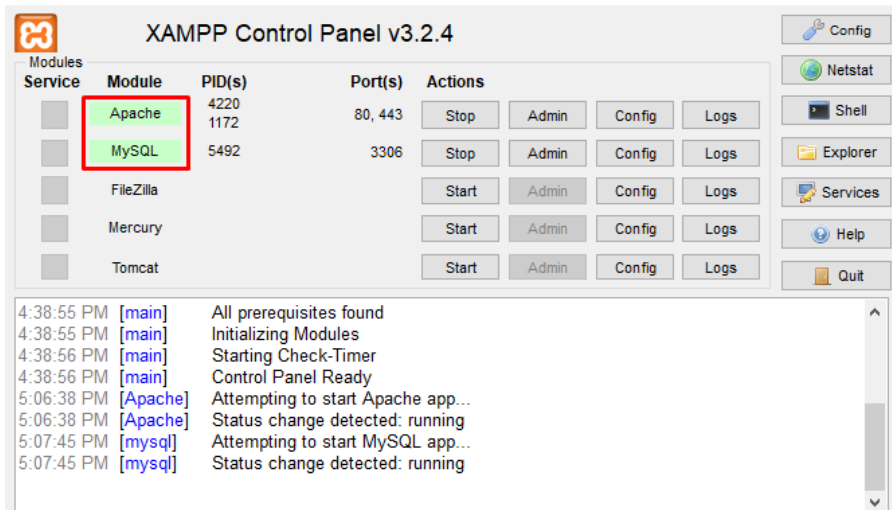
Như đã đề cập, trên HĐH Windows để chạy được PHP chúng ta cần phải cài đặt

- ❶ PHP (<https://www.php.net/downloads>)
- ❷ Apache Server (<https://httpd.apache.org/download.cgi>)
- ❸ MySQL Database (<https://www.mysql.com/downloads/>)

Hoặc có thể sử dụng các phần mềm Web server miễn phí tích hợp chung **PHP, MySQL Database, Apache Server** (All-In-One) như **WAMP/XAMPP**. Đề xuất sinh viên nên sử dụng **XAMPP** trong phần thực hành.

XAMPP bao gồm Apache web server, MySQL, PHP, Perl, FTP server và phpMyAdmin. Cài đặt XAMPP tại <https://www.apachefriends.org/index.html>

XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]



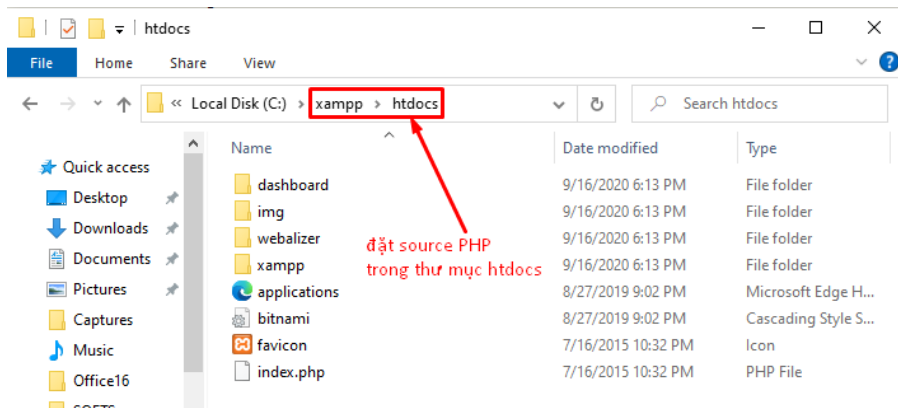
Service	Module	PID(s)	Port(s)	Actions			
<input type="checkbox"/>	Apache	4220 1172	80, 443	Stop	Admin	Config	Logs
<input type="checkbox"/>	MySQL	5492	3306	Stop	Admin	Config	Logs
<input type="checkbox"/>	FileZilla			Start	Admin	Config	Logs
<input type="checkbox"/>	Mercury			Start	Admin	Config	Logs
<input type="checkbox"/>	Tomcat			Start	Admin	Config	Logs

Log window:

- 4:38:55 PM [main] All prerequisites found
- 4:38:55 PM [main] Initializing Modules
- 4:38:56 PM [main] Starting Check-Timer
- 4:38:56 PM [main] Control Panel Ready
- 5:06:38 PM [Apache] Attempting to start Apache app...
- 5:06:38 PM [Apache] Status change detected: running
- 5:07:45 PM [mysql] Attempting to start MySQL app...
- 5:07:45 PM [mysql] Status change detected: running

Cách tổ chức và lưu trữ ứng dụng

Thư mục lưu trữ ứng dụng được đặt trong thư mục `htdocs`



Cách tổ chức và lưu trữ ứng dụng

Các loại tập tin thường gặp trong ứng dụng PHP là

Tập tin	Ý nghĩa
*.php	Tập tin mã nguồn viết theo ngôn ngữ PHP
*.js	Tập tin mã nguồn viết theo ngôn ngữ JavaScript
*.inc	Tập tin dùng chung
*.css	Dùng để định dạng style cho trang Web
*.html	Tập tin viết bằng ngôn ngữ HTML
*.jpg/png/gif	Tập tin ảnh
*.txt	Tập tin văn bản
*.sql	Tập tin dữ liệu

Thiết lập môi trường để chạy PHP

Ngoài ra, có thể sử dụng **LAMP stack** để chạy PHP trên HĐH Linux.

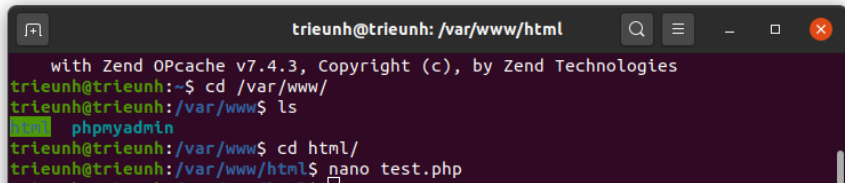
Cài LAMP stack gồm có:

- **LINUX** Operating System
- **APACHE** Web Server
- **MySQL** DataBase
- **PHP**

Tham khảo cách cài LAMP stack trên Ubuntu 18.04 tại

<https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-ubuntu-18-04>

Source code PHP được đặt ở thư mục `/var/www/html`



```
trieunh@trieunh: /var/www/html
with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies
trieunh@trieunh:~$ cd /var/www/
trieunh@trieunh:/var/www$ ls
html  phpmyadmin
trieunh@trieunh:/var/www$ cd html/
trieunh@trieunh:/var/www/html$ nano test.php
```

Công cụ để xây dựng ứng dụng PHP

Các Editor phổ biến để viết ứng dụng PHP gồm:

- PhpStorm
- Visual Studio Code
- Macro Media Dream Weaver
- Eclipse PHP Development Tools
- PHP Designer
- Notepad++

Thiết lập trang PHP

- là một trang động (Dynamic page)
- để hiển thị trang tiếng Việt cần phải thiết lập `charset=UTF-8`
- nhúng code PHP vào trang bằng thẻ PHP, ví dụ `<?php ?>`

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5      <title>Tiêu đề cho trang</title>
6  </head>
7  <body>
8      <?php echo "Lập trình PHP" ?>
9  </body>
10 </html>
```

Hình 2: Nhúng PHP vào HTML

Quy ước cú pháp

Code PHP được đặt trong các thẻ sau

Thẻ mở	Thẻ đóng	Chú thích
<code><?php</code>	<code>?></code>	cú pháp chính, nên sử dụng cú pháp này
<code><?</code>	<code>?></code>	cú pháp rút gọn, không nên sử dụng
<code><%</code>	<code>%></code>	cú pháp giống với ASP.
<code><script language="php"></code>	<code></script></code>	cú pháp bắt đầu bằng script

Ví dụ

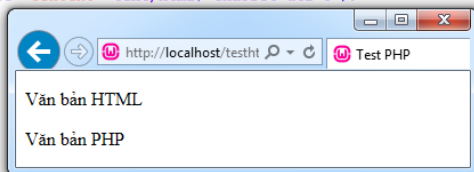
- `<?php echo " Xin Chào " ?>`
- `<? echo " Xin Chào " ?>`
- `<% echo " Xin Chào " %>`
- `<script language="php"> echo " Xin Chào "
</script>`

Ví dụ sử dụng cú pháp chính vào HTML

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5     <title>Test PHP</title>
6   </head>
7   <body>
8     <p>Văn bản HTML</p>
9     <?php echo "Văn bản PHP"; ?>
10  </body>
11 </html>

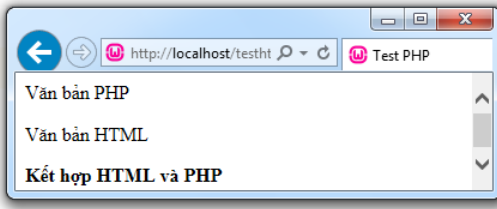
```



```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5     <title>Test PHP</title>
6   </head>
7   <body>
8     <?php
9       echo "Văn bản PHP";
10    ?>
11    <p>Văn bản HTML</p>
12    <?php echo "<b>"; ?>
13      Kết hợp HTML và PHP
14    <?php echo "</b>"; ?>
15  </body>
16 </html>

```



Lệnh và ghi chú

- Mỗi lệnh phải kết thúc bằng dấu ; (trừ lệnh cuối cùng trước khóa ?>)
- Không phân biệt khoảng trắng, tab, xuống hàng
- Khối (nhiều) lệnh được đặt trong cặp {}
- // hoặc #: chú thích có giá trị đến cuối dòng
- Để xuất dữ liệu ra trình duyệt, sử dụng: echo, print, printf, sprintf
- /* */: chú thích trên nhiều dòng

<body>

```
<?php
```

```
// chú thích trên một dòng
```

```
echo "Văn bản PHP_01";
```

```
# mỗi lệnh kết thúc bằng dấu ;
```

```
echo "<b>Văn bản PHP_02 </b>";
```

```
/*
```

```
    ghi chú trên dòng 01
```

```
    ghi chú trên dòng 02
```

```
*/
```

```
?>
```

```
<p>Văn bản HTML</p>
```

comment trong
PHP

Biến là một vùng nhớ (giống containers) dùng để lưu trữ thông tin.

- khai báo biến: `$tên_biến = value;` Trước tên biến phải có dấu **\$**
- biến **không cần khai báo kiểu dữ liệu**
- nên khởi tạo giá trị ban đầu cho biến
- tên biến:
 - ▶ có phân biệt HOA-thường
 - ▶ không trùng với tên hàm
 - ▶ không nên bắt đầu bằng số

Ví dụ:

```
1 <?php
2 $txt = "Hello_world!"; //kiểu dữ liệu String
3 $x = 1; $y = 2; // kiểu Integer
4 $N_T_U="DH_NT" //
5 $gioi_tinh=true // kiểu boolean
6 ?>
```

Cho phép thay đổi tên biến:

```
1 $varname = "Bien_moi"; $$varname = "xyz";
2 echo "$Bien_moi<br>"; // -> xyz. $$varname=$Bien_moi="xyz"
```


Xuất giá trị của biến ra màn hình:

```

1 <?php
2 $n1="Nha_Trang";
3 $n2="University";
4 echo $n1 , $n2, ":_DH-Nha_Trang<br>";
5 print ("Welcome_to_$n1_$n2<br>");
6 // xuất biến theo định dạng
7 $number = 96693983;
8 $str = "Vietnam";
9 $txt = sprintf("The_current_population_of_%s
10 is_%u_as_of_Sunday,_October_6,_2019.", $str, $number);
11 echo $txt
12 ?>

```

Một số định dạng xuất dữ liệu ra trình duyệt

Định dạng	Kiểu tham số	Đầu ra
%b	Số nguyên	Số nhị phân
%d	Số nguyên	Số nguyên
%f	Số thực	Số thực
%s	Chuỗi	Chuỗi
%u	Số nguyên	Số nguyên không dấu

Phạm vi hoạt động của biến

Loại	Vị trí khai báo	Phạm vi ảnh hưởng
Biến toàn cục	Khai báo bên ngoài hàm, nếu bên trong hàm thì phải sau từ khóa GLOBAL	Webpage
Biến cục bộ	Trong hàm	Trong hàm
Biến tĩnh	Sau từ khóa STATIC	Webpage
Tham số của hàm	Khai báo hàm	Trong hàm
Biến hệ thống	\$ _SERVER, \$ _POST, ...	Website

Biến toàn cục—Global Scope

- Có phạm vi toàn cục, có thể truy xuất bất cứ nơi nào trong trang
- Để sử dụng biến toàn cục trong hàm thì phải dùng từ khóa **global** phía trước biến hoặc dùng **\$GLOBALS["tên_biến"]**

```
1 <?php
2  $x = 5; // global scope
3  $y = 10; // global scope
4  function myTest() {
5      $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
6  }
7  myTest();
8  echo $y; // --> 15
9  #
10 function test_sum() {
11     global $x, $y;
12     $y = $x + $y;
13 }
14 test_sum();
15 echo $y; // --> 20
16 ?>
```

Biến cục bộ–Local Scope

- Biến được khai báo trong hàm và chỉ có thể truy cập trong hàm đó
- Khi ra khỏi hàm, biến cục bộ và giá trị của nó sẽ bị hủy bỏ

```
1 <?php
2 function myTest() {
3     $x = 5; // local scope
4     echo "<p>gia_tri_cua_bien_x_ben_trong_ham_la_: $x</p>"
5     ;
6 }
7 myTest();
8 // RA KHOI HAM myTest, bien $x da bi huy.
9 echo "<p>gia_tri_cua_bien_x_ben_ngoai_ham_la_: $x</p>";
10 // -> phat sinh loi tai day
11 ?>
```

Biến static

Khi một hàm được thi hành xong, tất cả các biến địa phương sẽ bị xóa. Tuy nhiên, nếu muốn giữ lại một vài biến để phục vụ mục đích khác, chúng ta sử dụng biến **static**. Đặc điểm của biến **static**:

- không bị mất giá trị khi đi ra khỏi hàm
- phía trước tên biến static phải có từ khóa **static**
- sẽ giữ nguyên giá trị trước đó khi hàm được gọi một lần nữa

```
1 <?php
2 function test_static() {
3     static $x = 0;
4     echo $x;
5     $x++;
6 }
7 test_static(); //->0
8 echo "<br>";
9 test_static(); //->1
10 echo "<br>";
11 test_static(); //->2
12 ?>
```

Hằng

Hằng là một biến có **giá trị không thay đổi** trong quá trình thực hiện chương trình. Ví dụ như số π , ...

- để tạo một biến hằng trong PHP, chúng ta sử dụng hàm **define(tên_hằng, giá_trị)**
- giá trị hằng chỉ được dùng các kiểu dữ liệu cơ bản như (boolean, integer, float, string)
- hằng là **biến toàn cục**, có thể **truy xuất ở mọi nơi**.

Hằng

Quy ước đặt **tên_hằng**

- giống với đặt tên biến
- không có ký tự \$

```
1 <?php
2  define("PI", 3.1412);
3  echo "Gia trị của Pi là: " .PI
4  ?>
```

Ngoài ra trong **PHP7**, chúng ta có thể tạo ra mảng hằng

```
1 <?php
2  define("cars", ["Alfa Romeo", "BMW", "Toyota"]);
3  echo cars[0];
4  ?>
```

Các kiểu dữ liệu cơ sở trong PHP

Bảng 2: Các kiểu dữ liệu chính trong PHP

Kiểu dữ liệu	Mô tả	Ví dụ
Boolean	nhận một trong hai giá trị là true hoặc false	TRUE or FALSE
Integer	giá trị là một số nguyên	1234
Float/double	số thực	1.234 hoặc 1.2e3
String	kiểu chuỗi, kí tự mỗi kí tự chiếm 1 byte	"text_string"
Array	kiểu mảng chứa các phần tử	array(1,2,3,4,5)
Object	hướng đối tượng trong PHP	<code>\$xe_hoi=new Xe();</code> # đối tượng xe

Chuyển đổi kiểu dữ liệu

Trong quá trình tính toán, chúng ta có thể cần **chuyển đổi kiểu dữ liệu**. Thực hiện chuyển đổi bằng các cách sau

- chuyển đổi tự động. Ví dụ:
 - ▶ `$var = "100" + 15;`
 - ▶ `$var = "100" + 15.5;`
 - ▶ `$var = 100 + "step";`
- **(datatype) \$var**. Ví dụ
 - ▶ `$tong_tien=(double)(500000*30000)`
 - ▶ `$var="6 feet"; (int)($var)//=6`
 - ▶ `$var="feet"; (int)($var)//=0`
- **settype(\$var, "datatype")**. Ví dụ
 - ▶ `$var="6 feet", settype($var, int)`

Phương thức kiểm tra dữ liệu của biến

Hàm	Mô tả	Cú pháp
<code>isset</code>	dùng để kiểm tra biến có giá trị hay không	<code>isset(<tên_biến 1>, <tên_biến 2>, ...)</code>
<code>empty</code>	kiểm tra biến có giá trị rỗng hay không	<code>empty(<tên_biến>)</code>
<code>is_numeric</code>	kiểm tra biến có giá trị kiểu số hay không	<code>is_numeric(<tên_biến>)</code>
<code>is_int/is_long/ is_double/is_string</code>	các hàm kiểm tra kiểu dữ liệu của biến	cú pháp chung <code>tên_hàm(<tên_biến>)</code>
<code>gettype</code>	kiểm tra biến hoặc giá trị có kiểu dữ liệu nào: integer, string, double, array, object, class	<code>gettype(<tên_biến> hoặc <giá_trị>)</code>

Hàm isset

Kết quả trả về là

- **TRUE**: nếu tất cả các biến đều có giá trị
- **FALSE**: nếu một biến bất kỳ không có giá trị

```
1 <?php
2     $var=1;
3     $kq=isset($var); //-> True
4     if($kq){
5         echo "co_gia_tri<br>"; //-> co gia tri tai day
6     }
7     else
8         echo "ko_co_gia_tri<br>";
9 ?>
```

Hàm empty

Kết quả trả về là

- **TRUE**: nếu biến là rỗng
- **FALSE**: nếu biến khác rỗng

Các biến được xem là rỗng khi giá trị của nó là:

- chuỗi rỗng "", **NULL**
- **0** khi là kiểu Integer, **false**, **array()**
- biến được khai báo nhưng không có giá trị (\$var).

Ví dụ:

```
1 <?php
2   $var1; // or $var1=0, ...
3   $kq=empty($var1); //-> TRUE
4   if($kq){
5       echo "Empty␣<br>"; //-> bien rong
6   }
7   else
8       echo "Non-empty␣<br>";
9 ?>
```

Hàm `is_numeric`

Kết quả trả về là

- **TRUE** nếu biến có giá trị kiểu số học
- **FALSE** không phải kiểu số học

Ví dụ:

```
1 <?php
2     $var1=0;
3     $kq=is_numeric($var1); //-> TRUE
4     if($kq){
5         echo "Numeric␣<br>"; //-> kieu so hoc.
6     }
7     else
8         echo "Non-numeric␣<br>";
9     $fruits = array("Apple", "Banana", "Orange");
10    $kq2=is_numeric($fruits) //-> khong phai kieu so hoc
11 ?>
```

Hàm `is_int`/`is_long`/`is_double`/`is_string`

Kết quả trả về là

- **TRUE** nếu đúng kiểu dữ liệu tương ứng với hàm
- **FALSE** nếu kiểu dữ liệu không tương ứng với hàm

Ví dụ:

```
1 <?php
2     $var1=0;
3     $kq=is_int($var1); //-> TRUE
4     if($kq){
5         echo "Integer□<br>"; //-> kiểu số nguyên
6     }
7     else
8         echo "Non-integer□<br>";
9     is_double($var1); //-> Kết quả nhận được là FALSE
10    is_string(0.1); //-> Kết quả nhận được là FALSE
11    is_string("test"); //-> Kết quả nhận được là TRUE
12 ?>
```

Hàm `gettype`

Trả về kiểu dữ liệu của biến như **integer**, **string**, **double**, **array**, **object**, **class**, ... Ví dụ:

```
1 <?php
2     $var1=0.1;
3     echo "kiểu_du_lieu_cua_var1_la:"
4         .gettype($var1)."<br>"; //->double
5     echo gettype("test")."<br>"; //->string
6     echo gettype(array("text1","text2"))."<br>"; //->array
7 ?>
```

Một số hàm xử lý số

Hàm	Mô tả	Ví dụ
<code>abs</code>	hàm lấy trị tuyệt đối	<code>abs(-1)//->1</code>
<code>round</code>	làm tròn đến số nguyên gần nhất	<code>round(4.6)//->5</code>
<code>ceil</code>	làm tròn lên	<code>ceil(4.2)//->5</code>
<code>floor</code>	làm tròn xuống	<code>floor(4.7)//->4</code>
<code>pow/sqrt</code>	lấy lũy thừa/căn bậc 2	<code>pow(2,3)//->8</code> <code>sqrt(9)//->3</code>
<code>log/log10</code>	các hàm tính logarithm	<code>log10(100)//->2</code>
<code>sin, cos, tan</code>	các hàm tính lượng giác	<code>cos(0)//->1</code>
<code>rand(min,max)</code>	giá trị ngẫu nhiên được trả về trong đoạn [min,max]	<code>rand(1,5)->?</code>
<code>min(), max()</code>	trả ra giá trị nhỏ nhất lớn nhất trong mảng	<code>max(1,2,3,4)//->4</code>
...		

Các hàm thông dụng xử lý chuỗi

Hàm	Mô tả	Ví dụ
.	toán tử nối chuỗi	<code>\$name="World"; echo "Hello " . \$name;</code>
<code>strlen</code>	trả ra độ dài của chuỗi	<code>echo strlen("Hello world!"); //-> 12</code>
<code>str_word_count</code>	số lượng từ của một chuỗi	<code>echo str_word_count("Hello world!"); //-> 2</code>
<code>str_replace</code>	tìm kiếm và thay thế chuỗi	<code>echo str_replace("world", "Dolly", "Hello world!"); //-> Hello Dolly!</code>
<code>strpos(\$str, \$chuoi_tim)</code>	tìm vị trí của \$chuoi_tim trong chuỗi \$str	<code>echo strpos("Hello world!", "world"); //-> 6</code>
<code>strrev</code>	đảo ngược lại chuỗi	<code>echo strrev("Hello world!"); //-> !dlrow olleH</code>
...		

Array

Một mảng lưu nhiều giá trị trong một biến duy nhất. Chỉ số mảng bắt đầu từ 0. ví dụ:

```
1 <?php
2 $s=array("DH","NHA_TRANG");
3 echo $s[1]; //->NHA_TRANG
4 ?>
```

Associative Arrays: mảng liên kết dùng từ khóa để kết nối giá trị với phần tử của mảng. Ví dụ:

```
1 <?php
2 $numbers=array ("one">1, "two","three");
3 echo $numbers["one"]; //->1
4 //
5 $age = array("Peter">"35", "Ben">"37", "Joe">"43");
6 echo $age["Ben"]; //->37
7 ?>
```

Một số hàm xử lý trên mảng

Hàm **count**: dùng để đếm số phần tử trong mảng

```
1 <?php
2 $cars = array("Volvo", "BMW", "Toyota");
3 echo count($cars); //->3
4 ?>
```

Hàm **min/max**: trả về giá trị nhỏ nhất/ lớn nhất của mảng. Ví dụ:

```
1 <?php
2 $n=array(-3,1,0,7,9,2);
3 echo min($n). "<br>"; //->-3
4 echo max($n); //->9
5 ?>
```

Sorting Arrays: các phần tử trong mảng có thể được sắp xếp theo alphabet hoặc số, tăng dần hoặc giảm dần. Các hàm sắp xếp cho mảng là:

- `sort()` - sắp xếp mảng theo thứ tự **tăng dần**
- `rsort()` - sắp xếp mảng theo thứ tự **giảm dần**

Một số hàm xử lý trên mảng

Các hàm sắp xếp cho **mảng liên kết** là:

- `asort()` - sắp xếp associative arrays theo thứ tự **tăng dần của giá trị**
- `ksort()` - sắp xếp associative arrays theo thứ tự **tăng dần của từ khóa**
- `arsort()` - sắp xếp associative arrays theo thứ tự **giảm dần của giá trị**
- `krsort()` - sắp xếp associative arrays theo thứ tự **giảm dần của từ khóa**

Ví dụ về các hàm Sort

asort()

```

1 <?php
2 $age = array("Ben"=>"37", "Peter"=>"35", "Joe"=>"43");
3 asort($age);
4 foreach($age as $x => $x_value) {
5     echo "Key=" . $x . ", Value=" . $x_value;
6     echo "<br>";
7 } // Key=Peter, Value=35 // Key=Ben, Value=37 // Key=Joe, Value
    =43

```

8 ?>

ksort():

```

1 <?php
2 $age = array("Joe"=>"43", "Peter"=>"35", "Ben"=>"37", "Adam"=>"50
    ");
3 ksort($age);
4 foreach($age as $x => $x_value) {
5     echo "Key=" . $x . ", Value=" . $x_value;
6     echo "<br>";
7 }
8 //Key=Adam, Value=50// Key=Ben, Value=37 //Key=Joe, Value=43//
    Key=Peter, Value=35
9 ?>

```

Một số hàm xử lý trên mảng

Ngoài ra, còn có một số hàm liên quan đến mảng như

- `reset (array)`: đặt vị trí con trỏ vào phần tử đầu tiên của mảng
- `array_push (array, elements)` thêm elements vào cuối mảng
- `array_pop (array)` lấy phần tử cuối ra khỏi mảng
- `array_shift (array)` Lấy phần tử đầu ra khỏi mảng
- `array_unshift (array elements)` Thêm elements vào đầu mảng
- `array_merge (array1,array2,...)` merge một hay nhiều mảng thành một mảng mới
- `sort (array, flag)`: flag `sort_regular`, `sort_numeric` `sort_string`, `sort_locale_string`

Một số hàm xử lý trên mảng

```
1 <?php
2 $people = array("Peter", "Joe", "Glenn", "Cleveland");
3
4 echo current($people)."<br>"; // The current element is Peter
5 echo next($people)."<br>"; // The next element of Peter is Joe
6 echo current($people)."<br>"; // Now the current element is Joe
7 echo prev($people)."<br>"; // The previous element of Joe is
    Peter
8 echo end($people)."<br>"; // The last element is Cleveland
9 echo prev($people)."<br>"; // The previous element of
10 //Cleveland is Glenn
11 echo current($people)."<br>"; // Now the current element is
    Glenn
12 echo reset($people)."<br>"; // Moves the internal pointer //
13 //to the first element of the array, which is Peter
14 echo next($people)."<br>"; // The next element of Peter is Joe
15
16 print_r (each($people)); // Returns the key and value of the
17 //current element (now Joe),
18 //and moves the internal pointer forward
19 ?>
```

Các toán tử trong PHP

PHP chia các toán tử thành các nhóm sau:

- toán tử số học (Arithmetic operators)
- toán tử gán (Assignment operators)
- toán tử so sánh (Comparison operators)
- toán tử logic (Logical operators)
- toán tử chuỗi (String operators)
- toán tử mảng (Array operators)
- toán tử gán có điều kiện (Conditional assignment operators)
- toán tử tăng/giảm (Increment/Decrement operators)

Toán tử số học

Toán tử	Mô tả	Ví dụ
+	cộng	$\$x + \y
-	trừ	$\$x - \y
*	nhân	$\$x * \y
/	chia	$\$x / \y
%	modulus	$\$x \% \$y // 10 \% 6 = 4$
**	lũy thừa	$\$x ** \$y // \$x^{\$y}$

Toán tử gán

Toán tử	Mô tả	Ví dụ
$x = +y$	$x = x + y$	$\$x = 20;$ $\$x += 100; // \rightarrow 120$
$x -= y$	$x = x - y$	$\$x = 50;$ $\$x -= 30; // \rightarrow 20$
$x *= y$	$x = x * y$	$\$x = 20;$ $\$x *= 2; // \rightarrow 40$
$x /= y$	$x = x / y$	$\$x = 20;$ $\$x /= 10; // \rightarrow 2$
$x \% = y$	$x = x \% y$	$\$x = 15;$ $\$x \% = 4; // \rightarrow 3$

Toán tử so sánh

Toán tử	Mô tả
<code>==</code>	Bằng giá trị
<code>===</code>	Bằng giá trị và cùng kiểu
<code>!=</code>	Khác giá trị
<code><></code>	Khác giá trị
<code>!==</code>	Khác giá trị hoặc khác kiểu
<code>></code>	Lớn hơn
<code>>=</code>	Lớn hơn hoặc bằng
<code><</code>	Nhỏ hơn
<code><=</code>	Lớn hơn hoặc bằng
<code>\$x<=>\$y</code>	Trả ra số nguyên -1, 0 và 1 tương ứng với <code>\$x<\$y</code> , <code>\$x=\$y</code> và <code>\$x>\$y</code> (Toán tử Spaceship được sử dụng trong PHP7)

Toán tử so sánh

Ví dụ về toán tử Spaceship:

```
1 <?php
2  $x = 5;
3  $y = 10;
4
5  echo ($x <=> $y); // returns -1 because $x is less than $y
6  echo "<br>";
7
8  $x = 10;
9  $y = 10;
10
11 echo ($x <=> $y); // returns 0 because values are equal
12 echo "<br>";
13
14 $x = 15;
15 $y = 10;
16
17 echo ($x <=> $y); // returns +1 because $x is greater than
    $y
18 ?>
```

Toán tử tăng/giảm

Toán tử	Mô tả	Ví dụ
<code>++\$x</code>	tăng giá trị của <code>\$x</code> lên 1 và trả ra giá trị của <code>\$x</code>	<code>\$x = 10; echo ++\$x;//->11</code>
<code>\$x++</code>	trả ra giá trị của <code>\$x</code> và tăng <code>\$x</code> lên một	<code>\$x = 10; echo \$x++;//->10</code>
<code>--\$x</code>	giảm giá trị của <code>\$x</code> xuống 1 và trả ra giá trị của <code>\$x</code>	<code>\$x = 10; echo --\$x;//->9</code>
<code>\$x--</code>	trả ra giá trị của <code>\$x</code> và giảm <code>\$x</code> xuống một	<code>\$x = 10; echo \$x--;//->10</code>

Toán tử logic

Toán tử	Mô tả	Ví dụ
and	And	$\$x \text{ and } \$y // \rightarrow \text{True or False}$
or	Or	$\$x \text{ or } \$y // \rightarrow \text{True or False}$
xor	Xor	True if either $\$x$ or $\$y$ is true, but not both $\$x \text{ xor } \y
&&	And	
	Or	
!	Not	$!\$x$ True if $\$x$ is not true

Toán tử chuỗi

Toán tử	Mô tả	Ví dụ
.	phép nối chuỗi	<code>\$txt1 . \$txt2;</code> <code>//nối chuỗi \$txt1 và \$txt2</code>
<code>.=</code>	phép gán nối chuỗi	<code>\$txt1 .= \$txt2;</code> <code>//thêm chuỗi \$txt2 vào \$txt1</code>

Toán tử gán có điều kiện

Toán tử	Ví dụ	Mô tả
<code>?:</code>	<code>\$x=expr1 ? expr2 : expr3</code>	trả ra giá trị của \$x giá trị của \$x là: expr2 nếu expr1 = TRUE expr3 nếu expr1 = FALSE
<code>??</code>	<code>\$x = expr1 ?? expr2</code>	trả ra giá trị của \$x giá trị của \$x là: expr1 nếu expr1 tồn tại và khác NULL expr2 nếu expr1 không tồn tại hoặc NULL (sử dụng trong PHP7)

Cấu trúc rẽ nhánh

Cú pháp lệnh **if**:

```
1 if (condition) {  
2     //code to be executed if condition is true;  
3 }
```

Cú pháp lệnh **if..else**

```
1 if (condition) {  
2     //code to be executed if condition is true;  
3 } else {  
4     //code to be executed if condition is false;  
5 }
```

Ví dụ:

```
1 <?php  
2 $t = date("H");  
3 if ($t < "20") {  
4     echo "Have a good day!";  
5 } else {  
6     echo "Have a good night!";  
7 }  
8 ?>
```

Các câu lệnh if có thể lồng nhau. Cú pháp lệnh **if...elseif...else**:

```
1 if (condition) {  
2     //code to be executed if this condition is true;  
3 } elseif (condition) {  
4     //code to be executed if first condition is false  
5     //and this condition is true;  
6 } else {  
7     //code to be executed if all conditions are false;  
8 }
```

Ví dụ:

```
1 <?php  
2 $t = date("H");  
3 echo "<p>The hour (of the server) is " . $t;  
4 echo ", and will give the following message:</p>";  
5  
6 if ($t < "10") {  
7     echo "Have a good morning!";  
8 } elseif ($t < "20") {  
9     echo "Have a good day!";  
10 } else {  
11     echo "Have a good night!";  
12 }  
13 ?>
```

Cú pháp lệnh **Switch**:

```
1 switch (n) {  
2     case label1:  
3         code to be executed if n=label1;  
4         break;  
5     case label2:  
6         code to be executed if n=label2;  
7         break;  
8     ...  
9     default:  
10        code to be executed if n is different from all labels  
11 };
```

Ví dụ:

```
1 <?php $favcolor = "red";  
2 switch ($favcolor) {  
3     case "red":  
4         echo "Your favorite color is red!";  
5         break;  
6     case "blue":  
7         echo "Your favorite color is blue!";  
8         break;  
9     default:  
10        echo "Your favorite color is neither red nor blue!";  
11 }
```

while/do...while

While loop thực hiện khối lệnh trong khi điều kiện vẫn còn đúng. Cú pháp lệnh **while**:

```
1 while (condition is true) {  
2     code to be executed;  
3     // Kiểm tra điều kiện trước khi thực thi khối lệnh  
4 }
```

Cú pháp lệnh **do ... while**:

```
1 do {  
2     code to be executed;  
3     // thực thi khối lệnh trước, sau đó kiểm tra điều kiện  
4 } while (condition is true);
```

Ví dụ:

```
1 <?php  
2 $x = 1;  
3 while($x <= 5) {  
4     echo $x; //12345  
5     $x++;  
6 }  
7 ?>
```

```
1 <?php  
2 $x = 1;  
3 do {  
4     echo $x; //12345  
5     $x++;  
6 } while ($x <= 5);  
7 ?>
```

for

Vòng lặp **for** thực thi khối lệnh với số lần xác định trước. Cấu trúc vòng lặp **for**

```
1 for (init counter; test counter; increment counter) {  
2     code to be executed;  
3 }
```

Trong đó:

- init counter: thực hiện 1 lần khi bắt đầu vòng lặp
- test counter: điều kiện lặp, được xét trước mỗi lần lặp
- increment counter: thực hiện sau mỗi lần lặp

Ví dụ

```
1 <?php  
2 for ($x = 0; $x <= 3; $x++) {  
3     echo $x; //0123  
4 }  
5 ?>
```

foreach

Vòng lặp **foreach** chỉ dùng để duyệt mảng.

Cú pháp **duyệt giá trị** các phần tử trong mảng

```
1 foreach($ten_mang as $value){
2     code to be executed;
3 }
```

Cú pháp duyệt **cả khóa và giá trị** các phần tử trong mảng:

```
1 foreach($ten_mang as $key =>
    $value){
2     code to be executed;
3 }
```

Ví dụ

```
1 <?php
2 $colors = array("red", "green", "blue", "yellow");
3 foreach ($colors as $value) {
4     echo "$value_<br>"; //red//green//blue//yellow
5 }
6 ?> ;
7 <?php
8 $ar = array('a' => 1, 'b' => '2', 'c' => '3');
9 foreach ( $ar as $key => $value ){
10     echo "ar[$key]_=$value_<br>"; //ar[a] = 1// ar[b] = 2
11     //ar[c] = 3
```

Sử dụng break

break: thoát khỏi cấu trúc điều khiển dựa trên kết quả của biểu thức luận lý kèm theo (điều kiện kiểm tra)

Ví dụ

```
1 <?php
2 $number=10;
3 $isprime=true;
4 for($i=2;$i<$number;$i++){
5     if($number%$i==0){
6         $isprime=false;
7         break;
8     }
9 };
10 if($isprime ==true) echo "Prime";
11 else echo ("NOT Prime"); //-> NOT Prime
12 ?>
```

Sử dụng continue

continue: khi gặp **continue**, các lệnh bên dưới continue tạm thời không thực hiện tiếp, khi đó con trỏ sẽ nhảy về đầu vòng lặp để kiểm tra giá trị của biểu thức điều kiện còn đúng hay không.

Ví dụ

```
1 <?php
2 $total=0;
3 $n=10;
4 for ($i = 2; $i <= $n; $i++) {
5     if ($n%$i!=0) continue;
6     echo "$i_<br>"; //-> 2,5,10
7     $total=$total+$i;
8 };
9 echo $total; //->17
10 ?>
```


Xây dựng hàm

Ngoài các hàm có sẵn trong PHP, chúng ta có thể **xây dựng hàm riêng cho mình**. Mục đích xây dựng hàm:

- **tái sử dụng** lại
- tăng tính mềm dẻo, nhất quán trong ứng dụng, thời gian xây dựng và thiết kế ứng dụng.
- giảm thời gian và chi phí
- ngắn gọn, dễ hiểu, dễ quản lý

Hàm trong PHP

Tạo một hàm trong PHP bắt đầu với từ khóa **function**, cú pháp như sau:

```
1 <?php
2 function functionName(parameter1,parameter1,...) {
3     code to be executed;
4 }
5 ?>
```

Lưu ý: tên hàm (functionName) có thể bắt đầu bằng chữ cái hoặc dấu gạch dưới, **không được dùng chữ số để bắt đầu**. Tên hàm không phân biệt hoa thường.

Gọi hàm: nhập **functionName(...)** và cung cấp đầy đủ các tham số cần thiết trong cặp dấu **()**. Ví dụ

```
1 <?php
2 function writeMsg() {//define function
3     echo "Hello_world!";
4 }
5 writeMsg(); // call the function
6 //-> Hello world!
7 ?>
```

Để trả ra kết quả của hàm, ta dùng lệnh **return**.

- Nếu trong hàm không có lệnh **return** thì hàm mặc định trả ra giá trị **NULL**
- Muốn trả ra **nhiều hơn một giá trị**, ta phải dùng **mảng**.

Ví dụ

```
1 <?php
2 function addNumbers($a,$b) {
3     return $a + $b;
4 }
5 echo addNumbers(5, 10); //->15
6 ?>
7 #####
8 <?php
9 function addNumbers($a,$b) {
10     $af[0]=$a+$b;
11     $af[1]=$a*$b;
12     return $af;
13 };
14 $t=addNumbers(5, 10);
15 echo "$t[0]<br>□$t[1]"; //->15//50
16 ?>
```

Truyền tham số

Mặc định tham số được truyền vào hàm theo phương pháp **tham trị**. Trường hợp muốn **thay đổi trực tiếp trên các tham số truyền vào** thì người ta dùng phương pháp **tham chiếu**, bằng cách thêm dấu **&** trước tên tham số (**khi định nghĩa**) cũng như tên biến được truyền làm tham số (**khi gọi hàm**). Cú pháp:

```
1 <?php
2 function functionName(parameter1, &parameter2, ...){
3     function-body;
4 }
5 functionName(parameter1, parameter2, ...);
6 ?>
```

Truyền tham số

Ví dụ truyền tham trị và tham số:

```
1 <?php
2 function changeToBoldText($text){
3     $text="<b>".$text."</b>";
4 }
5 function changeToRedText(&$text){
6     $text="<font color='red'>".$text."</font>";
7 }
8 function changeToBoldRedText(&$text){
9     changeToRedText($text);
10    changeToBoldText($text);
11 }
12 $mytext="Truong_dai_hoc_Nha_Trang";
13 changeToBoldRedText($mytext);
14 echo $mytext;
15 ?>
```

Bài tập

- ❶ Viết 1 trang web nhận một giá trị ngẫu nhiên là số tự nhiên N có giá trị từ $1 \rightarrow 100$. Hãy xuất ra trình duyệt những số chẵn nằm trong khoảng $1 \rightarrow N$ đó.
- ❷ Xây dựng 1 trang web thỏa yêu cầu xuất ra bảng cửu chương từ $1 \rightarrow 10$.
- ❸ Viết 1 trang web nhận một giá trị ngẫu nhiên là số tự nhiên N có giá trị trong $[-100; 100]$. Sau đó kiểm tra N có là số dương không? Nếu thỏa thì:
 - ▶ In ra các ước số của N .
 - ▶ Viết hàm kiểm tra xem N có phải là số nguyên tố không?
 - ▶ Tính tổng các số nguyên tố $< N$.
 - ▶ Kiểm tra N có là số chính phương?

- 1 Giới thiệu
- 2 Ngôn ngữ lập trình mã nguồn mở PHP
- 3 PHP–Ngôn Ngữ Lập Trình Hướng Đối Tượng**

Object-Oriented Programming

Kể từ PHP5 trở đi, chúng ta có thể viết code PHP theo kiểu đối tượng. Một số ưu điểm của lập trình hướng đối tượng (OPP) so với lập trình thủ tục thông thường trong PHP:

- OPP nhanh và dễ dàng thực hiện hơn
- OPP cung cấp cấu trúc rõ ràng hơn cho chương trình
- OPP giúp cho code PHP tuân thủ quy tắc DRY (“Don’t Repeat Yourself”)
- OPP có sức mạnh kế thừa(tận dụng lại code)

Object-Oriented Programming

Ví dụ class & objects



Another example:



Class

Định nghĩa class Fruit cho ví dụ ở trên

```
1 <?php
2 class Fruit {
3     // Properties
4     public $name;
5     public $color;
6
7     // Methods
8     function set_name($name) {
9         $this->name = $name;
10    }
11    function get_name() {
12        return $this->name;
13    }
14 }
15 ?>
```

Objects

Như đã đề cập ở trên, **class** chỉ là khung sườn

- nếu không có đối tượng nào sử dụng **class** thì class đó là vô nghĩa
- chúng ta có thể tạo được vô số đối tượng từ class **Fruit** ở trên
- các đối tượng này có cùng cấu trúc, **thừa hưởng toàn bộ đặc điểm, phương thức trong class**

Các đối tượng của một class được tạo bằng từ khóa **new**. Ví dụ:

- `$apple = new Fruit();`
- `$banana = new Fruit();`
- `$orange = new Fruit();`
- `$mangoes = new Fruit();`

Class & Objects

```
1 <?php
2 class Fruit {
3     // Properties
4     public $name;
5     public $color;
6     // Methods
7     function set_name($name) {
8         $this->name = $name;
9     }
10    function get_name() {
11        return $this->name;
12    }
13 }
14 $apple = new Fruit(); $banana = new Fruit();
15 $apple->set_name('Apple'); $banana->set_name('Banana');
16 echo $apple->get_name(); echo "<br>";
17 echo $banana->get_name();
18 ?>
```

Bài tập: Dựa vào ví dụ trên, thêm vào các phương thức `nhập_tên_màu` và `in_tên_màu` của các đối tượng trái cây cụ thể.

The `__construct` Function

Hàm này cho phép **khởi tạo thuộc tính ban đầu của đối tượng**. Khi tạo ra một đối tượng mới, **PHP** sẽ tự động gọi hàm này (**chạy đầu tiên**). Ví dụ

```
1 <?php
2 class Fruit {
3     public $name;
4     public $color;
5
6     function __construct($name) {
7         $this->name = $name;
8     }
9     function get_name() {
10         return $this->name;
11     }
12 }
13 $apple = new Fruit("Apple"); echo $apple->get_name();
14 ?>
```

Tương tự với hàm **`__destruct()`**, **PHP** sẽ tự động gọi hàm này ở cuối script.

PHP - Access Modifiers

Các thuộc tính, phương thức trong *class* được phân quyền sửa đổi, truy cập

- *public*: thuộc tính hoặc phương thức có thể được truy cập từ mọi nơi. Đây là mặc định
- *protected*: thuộc tính hoặc phương thức có thể được truy cập từ trong lớp hoặc từ trong các lớp con của lớp đó
- *private*: thuộc tính hoặc phương thức **CHỈ** có thể được truy cập trong lớp đó

PHP - Access Modifiers

Ví dụ về quyền truy cập thuộc tính: Hãy cho biết kết quả của chương trình sau

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <?php
5 class Fruit {
6     public $name;
7     public $color;
8     private $weight;
9 }
10 $mango = new Fruit(); $mango->name = 'Mango';
11 echo $mango->name; //???
12 $mango->color = 'Yellow';
13 echo $mango->color; //???
14 $mango->weight = '300';
15 echo $mango->weight; //???
16 ?>
17 </body>
18 </html>
```

PHP - Access Modifiers

Ví dụ về quyền truy cập phương thức: Hãy cho biết kết quả

```
1 <?php
2 class Fruit {
3     public $name;
4     public $color;
5     public $weight;
6     function set_name($n) { // a public function (default)
7         $this->name = $n;}
8     protected function set_color($n) { // a protected function
9         $this->color = $n;}
10    private function set_weight($n) { // a private function
11        $this->weight = $n;}
12    function __destruct() {
13        echo "all params are: " . $this->name . " " . $this->color . " " . $this->
            weight; //???
14    }
15 }
16 $mango = new Fruit();
17 $mango->set_name('Mango');
18 $mango->set_color('Yellow');
19 $mango->set_weight('300');
20 ?>
```


Inheritance

Inheritance: tính kế thừa của lớp con sinh ra từ lớp cha

- Lớp con (child class) sẽ kế thừa đầy đủ thuộc tính, phương thức có quyền truy cập public hoặc protected của lớp cha (parent class)
- một lớp con kế thừa được khai báo bằng cách sử dụng từ khóa extends

Ví dụ

```
1 <?php
2 class Fruit {
3     public $name;
4     public $color;
5     public function __construct($name, $color) {
6         $this->name = $name;
7         $this->color = $color;
8     }
9     public function intro() {
10         echo "The fruit is {$this->name} and the color is {$this->
            color}.";
11     }
12 }
13 // Strawberry is inherited from Fruit
14 class Strawberry extends Fruit {
15     public function message() {
16         echo "Am I a fruit or a berry?";
17     }
18 }
19 $strawberry = new Strawberry("Strawberry", "red");
20 $strawberry->message();
21 $strawberry->intro();
22 ?>
```

Kết quả chương trình sau?

```
1 <?php
2 class Fruit {
3     public $name;
4     public $color;
5     public function __construct($name, $color) {
6         $this->name = $name;
7         $this->color = $color;
8     }
9     protected function intro() {
10         echo "The fruit is {$this->name} and the color is {$this->
            color}.";
11     }
12 }
13
14 class Strawberry extends Fruit {
15     public function message() {
16         echo "Am I a fruit or a berry?";
17     }
18 }
19 // Try to call all three methods from outside class
20 $strawberry = new Strawberry("Strawberry", "red");
21 $strawberry->message();
22 $strawberry->intro();
23 ?>
```

Kết quả chương trình sau?

```
1 <?php
2 class Fruit {
3     public $name;
4     public $color;
5     public function __construct($name, $color) {
6         $this->name = $name;
7         $this->color = $color;
8     }
9     protected function intro() {
10         echo "The fruit is {$this->name} and the color is {$this->
            color}.";
11     }
12 }
13 class Strawberry extends Fruit {
14     public function message() {
15         echo "Am I a fruit or a berry?";
16         // Call protected method from within derived class
17         $this->intro();
18     }
19 }
20 $strawberry = new Strawberry("Strawberry", "red");
21 $strawberry->message();
22 ?>
```

Reference

- [1] Lê Thị Bích Hằng, Bài giảng *Phát Triển Phần Mềm Mã Nguồn Mở*.
- [2] [w3schools.com](https://www.w3schools.com)