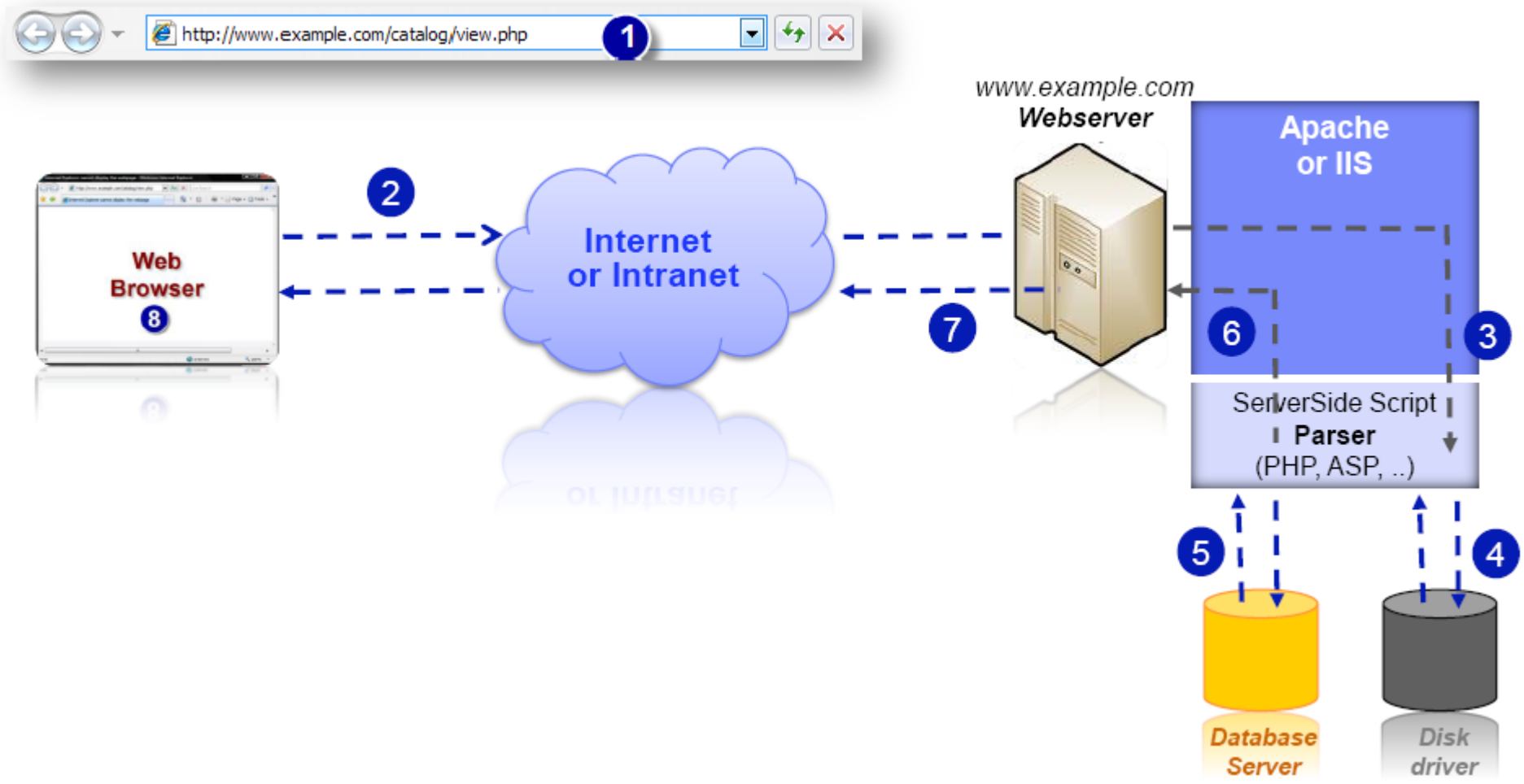
Tổng quan lập trình PHP

Hùng Nguyễn

Cơ chế tương tác người dùng đến web server

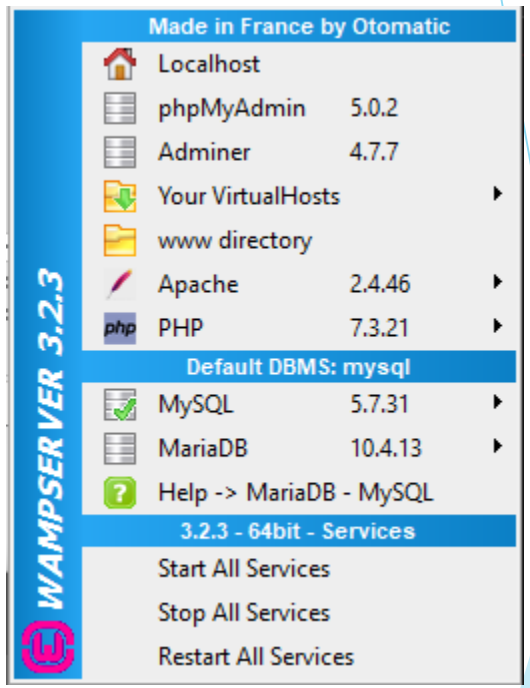


Các đặc điểm của ngôn ngữ PHP

- PHP được chạy trên hệ thống chủ (Server)
- PHP rất đơn giản
- Tốc độ xử lý nhanh, dễ sử dụng
- Luôn được cải tiến và cập nhật (mã nguồn mở)
- Có nhiều hướng dẫn sử dụng trên mạng
- Hoàn toàn miễn phí
- PHP có thể thực thi trên bất cứ hệ điều hành (Operator System) nào.
- PHP không chỉ làm việc với HTML mà còn có thể làm việc được với hình ảnh, PDF, Flash movie,…
- PHP có thể dễ dàng nối kết với các cơ sở dữ liệu như mySQL, mSQL, FrontBase, dBase, Solid, ODBC, Oracle, FilePro…

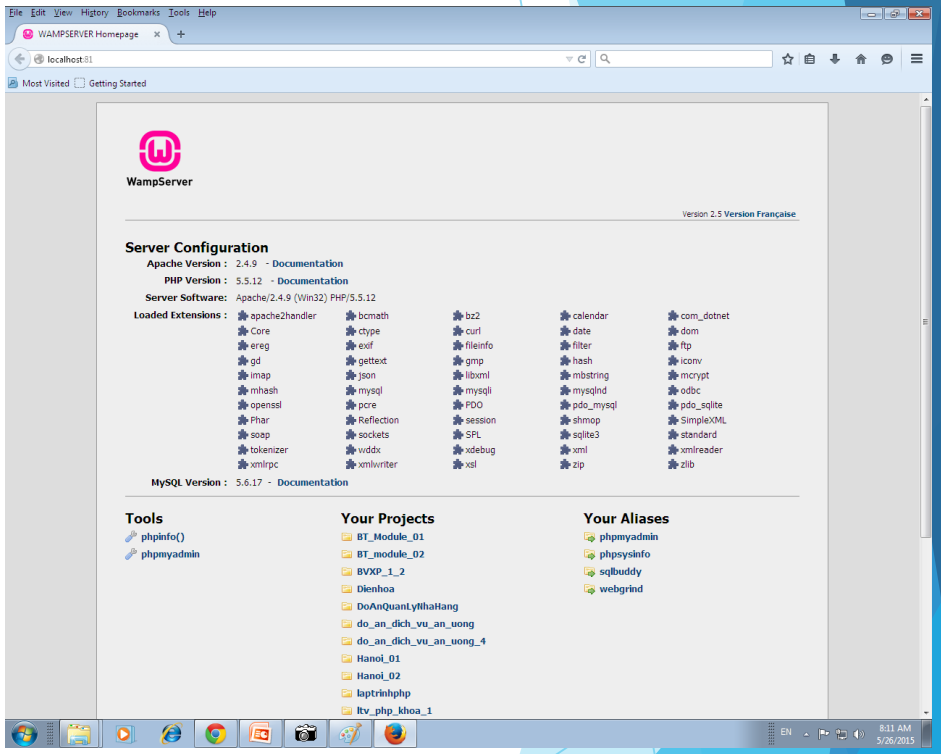
Wamp Server

- WAMP: là một gói phần mềm Web Server tất cả trong một (All-in-One), gồm có: Apache, MySQL, PHP chạy trên nền Windows.
- Có thể cài đặt dễ dàng.
- Được cập nhật đều đặn.
- Rất thuận lợi cho việc tạo máy chủ Web để chạy thử, thiết kế Website bằng PHP.
- Chạy ổn định cho Joomla các phiên bản.
- Hỗ trợ PHP7.x
- Miễn phí



Cài Đặt WAMP

- Sử dụng phiên bản Wamp 2.5 để cài đặt lần lượt theo các bước.
- Khi cài đặt xong, chúng ta sẽ thấy biểu tượng chữ W màu xanh lá ở góc phải dưới của màn hình
- Kiểm tra kết quả cài đặt



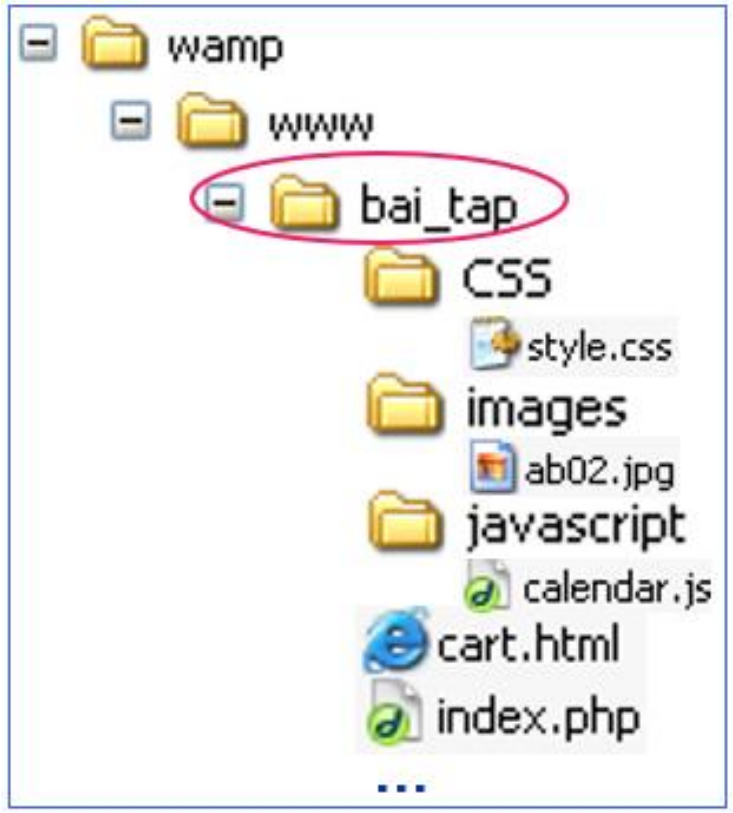
Chú ý: Khi cài đặt và sử dụng Wamp thì cần phải tắt IIS

Sử Dụng PHP

- Cách tổ chức và lưu trữ ứng dụng
- Công cụ để xây dựng ứng dụng PHP
- Các thao tác cơ bản

Cách tổ chức và lưu trữ ứng dụng

- Thư mục lưu trữ ứng dụng được đặt trong thư mục wamp/www



Cách tổ chức và lưu trữ ứng dụng

- Các loại tập tin thường gặp trong ứng dụng PHP

Tập tin	Diễn giải
.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	Tập tin html dùng để định style cho trang web
.htm, .html	Tập tin viết bằng ngôn ngữ HTML
.jpg, .gif, ...	Các tập tin hình ảnh
.txt	Tập tin dữ liệu text
.sql	Tập tin dữ liệu sql

Các ứng dụng hỗ trợ lập trình PHP

PHPStorm

Visual Studio Code

NetBean

Quy ước

Code PHP được đặt trong các cặp thẻ sau:

Thẻ mở	Thẻ đóng
<?php	?>
<?	?>
<script language="php">	</script>

Ví dụ:

```php
<?php
echo "<b>Chào các bạn.</b>"
?>
```

Quy ước

- Các lệnh cách nhau bởi dấu ;
- Ghi chú trong PHP:
  - Dùng //ghi chú
  - Hoặc /*ghi chú*/

```php
<?php

echo "<b>Chào các bạn.</b>"

// ghi chú 1 dòng

/* ----

Ghi chú nhiều dòng

---- */

?>
```

Biến

- Khai báo biến
  - Cú pháp: $tên_biến
  - Ví dụ: $tong
- Quy tắc đặt tên cho biến
  - Tên biến phải bắt đầu bằng ký tự $, theo sau là 1 ký tự hoặc dấu _, tiếp đó là ký tự, ký số hoặc dấu _
  - Nên khởi tạo giá trị ban đầu cho biến
  - Tên biến không trùng với tên hàm
  - Biến không nên bắt đầu bằng ký số

Biến

- Lưu ý
  - Tên biến có phân biệt chữ HOA – chữ thường
  - Ví dụ: biến $a và biến $A là hai biến khác nhau

- Gán giá trị cho biến
  - Gán giá trị trực tiếp
    - Cú pháp: $tên_biến = <giá_trị>;
    - Ví dụ:

Toán tử số học

- Gồm các toán tử + - * / %

```php
<?php
    $a=10;
    $b=5;

    $tong = $a + $b; → 15
    $tich = $a * $b; → 50

    $so_du = $a % $b; → 0
?>
```

Lưu ý:

Khi có nhiều biểu thức tính toán thì nên đưa chúng vào cặp ngoặc () để việc tính toán được tường minh.

Toán tử nối chuỗi

- Sử dụng toán tử "." để nối các chuỗi lại với nhau
  - Ví dụ:

```php
<?php
    $chuoi_1 = "vietnam";
    $chuoi_2 = "airline.com.vn";
    $chuoi =$chuoi_1.$chuoi_2;
    → "vietnamairline.com.vn"
?>
```

Toán tử gán kết hợp

- Toán tử +=
  - Sử dụng: $a+=$b; ⇔ $a=$a+$b;
  - Ví dụ:

```php
<?php
    $a = 7;
    $a+= 1; → 8
?>
```

Toán tử gán kết hợp

- Toán tử ++
  - Sử dụng: $a++; ⇔ $a=$a+1;
  - Ví dụ:

```php
<?php
    $a = 7;
    $a++; //→ 8
?>
```

Toán tử gán kết hợp

- Toán tử -=
  - Sử dụng: $a-=$b; ⇔ $a=$a-$b;
  - Ví dụ:

```php
<?php
    $a = 7;
    $a- = 1; → 6
?>
```

Toán tử gán kết hợp

- Toán tử --
  - Sử dụng: $a--; ⇔ $a=$a-1;
  - Ví dụ:

```php
<?php
    $a = 7;
    $a--; → 6
?>
```

Toán tử gán kết hợp

- Toán tử *=
  - Sử dụng: $a*=$b; ⇔ $a=$a*$b;
  - Ví dụ:

```php
<?php
    $a = 7;
    $a *= 2; → 14
?>
```

Toán tử gán kết hợp

- Toán tử /=
  - Sử dụng: $a/=$b; ⇔ $a=$a/$b;
  - Ví dụ:

```php
<?php
    $a = 8;
    $a /= 2; → 4
?>
```

Toán tử gán kết hợp

- Toán tử %=
  - Sử dụng: $a%=$b; ⇔ $a=$a%$b;
  - Ví dụ:

```php
<?php
    $a = 7;
    $a %= 2; → 1
?>
```

Toán tử gán kết hợp

- Toán tử .=
  - Sử dụng: $a.=$b; ⇔ $a=$a.$b;
  - Ví dụ:

```php
<?php
    $chuoi1 = "abc";
    $chuoi2 = "def";
    $chuoi1.= $chuoi2; → "abcdef"
?>
```

Toán tử so sánh

- Toán tử ==
  - Thực hiện phép so sánh bằng
  - Ví dụ:

```php
<?php
    $a = $_POST["a"];
    $b = $_POST["b"];
    $kq = "";
    if ($a==$b)
        $kq = "a bằng b";
?>
```

Toán tử so sánh

- Toán tử ===
  - Thực hiện phép so sánh bằng với các đối tượng có cùng kiểu dữ liệu
  - Ví dụ:

```php
<?php
    $a = $_POST["a"];
    $b = $_POST["b"];
    $kq = "";
    if ($a===$b)
        $kq = " a bằng b và có cùng kiểu dữ liệu";
?>
```

Toán tử so sánh

- Toán tử !=, <>
  - Thực hiện phép so sánh khác
  - Ví dụ:

```php
<?php
    $a = $_POST["a"];
    $b = $_POST["b"];
    $kq = "";
    if ($a<>$b)
        $kq = " a khác b";
?>
```

Toán tử so sánh

- Toán tử >, >=
  - Thực hiện phép so sánh lớn hơn, lớn hơn hoặc bằng
  - Ví dụ:

```php
<?php
    $a = $_POST["a"];
    $b = $_POST["b"];
    $kq = "";
    if ($a>$b)
        $kq = " a lớn hơn b";
?>
```

Toán tử so sánh

- Toán tử <, <=
  - Thực hiện phép so sánh nhỏ hơn, nhỏ hơn hoặc bằng
  - Ví dụ:

```php
<?php
    $a = $_POST["a"];
    $b = $_POST["b"];
    $kq = "";
    if ($a<=$b)
        $kq = " a nhỏ hơn hoặc bằng b";
?>
```

Toán tử luận lý

- Toán tử !
  - Toán tử phủ định
  - Ví dụ:

```php
<?php
    $a = 2;
    $b = 1;
    if(!($a>$b))
        echo "$a không lớn hơn $b";
    else
        echo "$a lớn hơn $b";
?>
```

Toán tử luận lý

- Toán tử And, &&
  - Ý nghĩa: Đúng nếu cả hai biểu thức có giá trị đúng
  - Ví dụ:

```php
<?php
    $a = $_POST["a"];
    $b = $_POST["b"];
    if($a>0 && $b>0)
        echo "Cả $a và $b đều lớn hơn 0";
?>
```

Toán tử luận lý

- Toán tử Or, ||
  - Ý nghĩa: Đúng khi một trong hai biểu thức có giá trị TRUE, hoặc cả hai cùng có giá trị TRUE
  - Ví dụ:

```php
<?php
    $a = $_POST["a"];
    $b = $_POST["b"];
    if($a>0 || $b>0)
        echo "Có ít nhất một giá trị dương";
?>
```

Toán tử xác định kiểu

- Toán tử xác định kiểu instanceof
  - Ý nghĩa: Được dùng để kiểm tra xem biến có thuộc kiểu (đối tượng) đang dùng để so sánh hay không. Với giá trị trả về là true hoặc false
  - Ví dụ:

```
class phan_so{

}
$bien = new phan_so();
if($bien instanceof phan_so)
    echo 'Biến $bien là kiểu phân số';
else
    echo 'Biến $bien không phải là kiểu phân số';
```

Cấu trúc rẽ nhánh if else

- Cú pháp:

```
if(điều kiện 1)
{
        khối lệnh 1
}
elseif(điều kiện 2)
{
        khối lệnh 2
}
…
else
{
        khối lệnh khi không thỏa các điều kiện trên
}
```

Cấu trúc rẽ nhánh switch

- Dạng 1: mỗi trường hợp một cách xử lý khác nhau
  - Cú pháp

```
switch(biến điều kiện)
{
    case giá trị 1:
        khối lệnh 1
        break;
    case giá trị 2:
        khối lệnh 2
        break;
    ...
    [default: khối lệnh khi không thỏa tất cả các case trên]
}
```

Cấu trúc rẽ nhánh switch

- Dạng 2: mỗi nhóm các trường hợp có cùng một cách xử lý
  - Cú pháp

```
switch(biến điều kiện)
{
   case giá trị 1:
   case giá trị 2:
   ...
   khối lệnh
   ...
   [default: khối lệnh khi không thỏa tất cả các case trên]
}
```

Cấu trúc lặp

- Cấu trúc lặp cho phép thực hiện nhiều lần một khối lệnh của chương trình khi thỏa điều kiện

- Gồm có các cấu trúc: for, foreach, while, do…while

Cấu trúc lặp for

- Công dụng:
  - for được sử dụng khi chúng ta biết trước số lần cần lặp, biến đếm chạy trong khoảng giới hạn của vòng lặp, và giá trị lặp.
- Cú pháp:

```
for($biến_đếm = giá trị khởi đầu của vòng lặp for; điều kiện giới hạn của vòng lặp for; giá trị lặp của vòng lặp for)
{
        khối lệnh
}
```

Cấu trúc lặp for

- Ý nghĩa:
  - Khối lệnh chỉ được thực hiện khi biến đếm vẫn còn nằm trong khoảng giới hạn của vòng lặp.
  - Vòng lặp sẽ kết thúc khi biến đếm vượt qua khoảng giới hạn của vòng lặp
  - Cần chỉ định giá trị lặp của biến đếm. Sau mỗi lần lặp biến đếm sẽ tăng lên (hoặc giảm đi)
- Ví dụ:

```php
<?php
    $tong = 0;
    for($i=1; $i<=10;$i++)
    {
        $tong = $tong + $i;
    }
    echo $tong; → 55
?>
```

Cấu trúc lặp for

- Ý nghĩa:
  - Khối lệnh chỉ được thực hiện khi biến đếm vẫn còn nằm trong khoảng giới hạn của vòng lặp.
  - Vòng lặp sẽ kết thúc khi biến đếm vượt qua khoảng giới hạn của vòng lặp
  - Cần chỉ định giá trị lặp của biến đếm. Sau mỗi lần lặp biến đếm sẽ tăng lên (hoặc giảm đi)
- Ví dụ:

```php
<?php
    $tong = 0;
    for($i=1; $i<=10;$i++)
    {
        $tong = $tong + $i;
    }
    echo $tong; → 55
?>
```

Cấu trúc lặp foreach

- Công dụng:
  - foreach thường được dùng để duyệt tập hợp (mảng).
- Cú pháp duyệt giá trị các phần tử trong mảng:

```
foreach ($ten_mang as $gia_tri)
{
    khối lệnh
}
```

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

```
foreach ($ten_mang as $tu_khoa => $gia_tri)
{
    khối lệnh
}
```

Cấu trúc lặp foreach

- Ví dụ 1: duyệt và in nội dung của các phần tử trong mảng

```php
<?php
    $mang_ten = array("mai", "lan", "cúc", "trúc");
    foreach($mang_ten as $ten)
    {
        echo $ten . "  ";
    }
    → mai lan cúc trúc
?>
```

Cấu trúc lặp while

- Công dụng
  - Thực hiện lặp đi lặp lại một công việc nào đó khi thỏa điều kiện.
  - while được sử dụng khi không xác định được số lần lặp (số lần lặp phụ thuộc vào điều kiện tại thời điểm thực thi)
- Cú pháp

```
while(điều kiện)
{
        khối lệnh
}
```

Cấu trúc lặp while

- Ý nghĩa
  - Điều kiện là một biểu thức logic (có kết quả TRUE hoặc FALSE)
  - Sau mỗi lần thực hiện khối lệnh trong while, điều kiện sẽ được kiểm tra lại.

    Nếu giá trị True thì thực hiện lại vòng lặp

    Nếu giá trị False thì chấm dứt vòng lặp
  - Cấu trúc này kiểm tra điều kiện trước khi thực hiện các lệnh nên sẽ không thực hiện lần nào nếu ngay lần đầu tiên điều kiện có giá trị False.

Cấu trúc lặp while

- Ví dụ 1: tính tổng các số nguyên dương <10

```php
<?php
    $i = 1;
    $tong = 0;
    while($i<10)
    {
        $tong = $tong +$i;
        $i= $i+1;
    }
    echo $tong; → 45
?>
```

Kiểm tra tồn tại: isset()

- Ý nghĩa: dùng để kiểm tra biến có giá trị hay không
- Cú pháp: isset(<tên biến 1>, <tên biến 2>, …)
- Kết quả trả về:
  - TRUE: nếu tất cả các biến đều có giá trị
  - FALSE: nếu một biến bất kỳ không có giá trị

Kiểm tra giá trị rỗng: empty()

- Ý nghĩa: dùng để kiểm tra biến có giá trị rỗng hay không
- Cú pháp: empty(<tên biến>)
  - Kết quả trả về:
    - TRUE: nếu biến có giá trị rỗng
    - FALSE: nếu một biến có giá trị khác rỗng
  - Các giá trị được xem là rỗng:
    - "" (chuỗi rỗng), NULL
    - 0 (khi kiểu là integer), FALSE, array()
    - var $var (biến trong lớp được khai báo nhưng không có giá trị)

Kiểm tra giá trị rỗng: empty()

- Ví dụ:

```php
<?php
if(empty($_POST["ten_dang_nhap"]))
    {
        echo "Vui lòng nhập tên đăng nhập";
        exit;
    }
else
    echo "Xin chào ".$_POST["ten_dang_nhap"];
?>
```

Kiểm tra trị kiểu số: is_numeric()

- Ý nghĩa: dùng để kiểm tra biến có giá trị kiểu số hay không

- Cú pháp: is_numeric(<tên biến>)
  - Kết quả trả về:
    - TRUE: nếu biến có giá trị kiểu số
    - FALSE: nếu biến có giá trị không phải kiểu số

Kiểm tra trị kiểu số: is_numeric()

- Ví dụ:

```php
<?php
    if(is_numeric($_POST["so_luong"]))
    {
        $so_luong = $_POST["so_luong"];
        $thanh_tien = $so_luong * $don_gia;
    }
    else
        echo "Số lượng phải là kiểu số!";
?>
```

Kiểm tra kiểu dữ liệu của biến

- is_int() / is_long()
- is_string()
- is_double()
  - Ý nghĩa: kiểm tra giá trị của biến có phải là kiểu integer - long – string – double hay không
  - Cú pháp chung: tên_hàm(<tên_biến>)

Kiểm tra kiểu dữ liệu của biến

- Ví dụ:

```php
<?php
    $a = "15";
    is_int($a); → 0 (FALSE)
    $b = 15;
    is_int($b); → 1 (TRUE)
    $a = "hello";
    is_string($a); → 1 (TRUE)
    $b = 12.5;
    is_string($b); → 0 (FALSE)
    $x = 4.2135;
    is_double($x); → 1 (TRUE)
?>
```

Xác định kiểu của biến: gettype()

- Ý nghĩa; kiểm tra biến hoặc giá trị có kiểu dữ liệu nào: integer, string, double, array, object, class, …
- Cú pháp: gettype(<tên biến> hoặc <giá trị>)
- Kết quả trả về: kiểu của giá trị hay kiểu của biến

Xác định kiểu của biến: gettype()

- Ví dụ:

```php
<?php
    $n = "Đây là chuỗi";
    $a = 123;
    $b = 123.456;
    $mang = array(1,2,3);
    echo gettype($n); → string
    echo gettype($a); → integer
    echo gettype($b); → double
    echo gettype($mang); → array
?>
```

Các hàm toán học

- Giá trị tuyệt đối
- Pi
- Lũy thừa
- Phát sinh ngẫu nhiên
- Làm tròn
- Tính căn

Các hàm toán học

- Kết quả trả về là giá trị tuyệt đối của một số
  - Cú pháp: abs(số)

- pi()
  - Kết quả trả về là hằng số PI trong toán học
  - Cú pháp: pi()
  - Ví dụ:

- pow()
  - Kết quả trả về là phép tính lũy thừa của một số theo một số mũ chỉ định
  - Cú pháp: pow(số, lũy_thừa)

Các hàm toán học

- rand()
  - Kết quả trả về là một giá trị số nguyên ngẫu nhiên bất kỳ (nằm trong khoảng từ 0 đến 32768)
  - Khi hàm rand() có hai tham số là giá trị nhỏ nhất (min) và giá trị lớn nhất (max) thì giá trị trả về sẽ là một giá trị số nằm trong khoảng từ giá trị nhỏ nhất đến giá trị lớn nhất.
  - Cú pháp: rand ([min, max])

- sqrt()
  - Tính căn bậc hai của một số dương bất kỳ
  - Kết quả trả về là căn bậc hai của số
  - Cú pháp: sqrt(số)

Các hàm toán học

- round()
  - Kết quả trả về là một số đã được làm tròn.
  - Nếu chỉ có một tham số là số được truyền vào thì kết quả trả về là một số nguyên.
  - Nếu có cả hai tham số là số và vị trí làm tròn thì kết quả trả về là một số được làm tròn dựa trên vị trí làm tròn.
- Cú pháp:

  round (số [, vị trí làm tròn])
  - Vị trí làm tròn là một số nguyên âm hoặc nguyên dương dùng để chỉ định vị trí muốn làm tròn, được tính từ vị trí dấu chấm thập phân

Các hàm chuỗi

- Bỏ khoảng trắng thừa
- Chiều dài chuỗi
- So sánh chuỗi
- Tìm kiếm và thay thế
- Lấy chuỗi con
- Kết hợp và tách chuỗi

Bỏ khoảng trắng thừa

- trim, ltrim, rtrim
  - Loại bỏ các ký tự thừa.
  - Kết quả trả về là một chuỗi không có ký tự thừa.
  - Cú pháp: ten_ham(str [, charlist])
    - Tham số bắt buộc là chuỗi str
    - Tham số tùy chọn là danh sách các ký tự muốn loại bỏ.
- Lưu ý: nếu không có tham số thứ hai thì hàm ltrim(), rtrim(), sẽ tự động loại bỏ các ký tự sau:
  - " " (ASCII 32 (0x20)), ký tự khoảng trắng.
  - "\t" (ASCII 9 (0x09)), ký tự tab.
  - "\n" (ASCII 10 (0x0A)), ký tự về đầu dòng mới.
  - "\r" (ASCII 13 (0x0D)), ký tự xuống dòng.
  - "\0" (ASCII 0 (0x00)), byte NULL.
  - "\x0B" (ASCII 11 (0x0B)), tab dọc.
- Ngoài ra, có thể liệt kê các ký tự muốn loại bỏ ở charlist

Chiều dài chuỗi và so sánh chuỗi

- strlen()
  - Kết quả trả về là chiều dài của một chuỗi
  - Cú pháp: strlen(str)
  - Ví dụ:

- strcmp()
  - So sánh hai chuỗi có phân biệt chữ HOA, chữ thường.
  - Kết quả trả về:
    - =0: nếu hai chuỗi bằng nhau
    - -1: nếu chuỗi str1 nhỏ hơn chuỗi str2
    - 1: nếu chuỗi str1 lớn hơn chuỗi str2
  - Cú pháp: strcmp(str1, str2)
  - Lưu ý: Việc so sánh chuỗi dựa trên bảng mã ASCII của các ký tự

Tìm vị trí chuỗi

- strpos()
  - Kết quả trả về là vị trí tìm thấy của chuỗi str2 trong chuỗi str1, nếu trong chuỗi str1 có nhiều chuỗi str2 thì kết quả trả về sẽ là vị trí đầu tiên mà chuỗi str2 xuất hiện trong chuỗi str1. Nếu không tìm thấy chuỗi str2 trong chuỗi str1 thì kết quả trả về là FALSE
  - Cú pháp: strpos(str1, str2)
  - Lưu ý: Để kiểm tra kết quả thì phải dùng toán tử === thay cho toán tử ==
- strripos()
  - Kết quả trả về tương tự như hàm strpos(). Nhưng vị trí được trả về khi tìm được sẽ là vị trí cuối cùng mà chuỗi cần tìm xuất hiện.
  - Cú pháp: strripos(str1, str2)
  - Lưu ý: Để kiểm tra kết quả thì phải dùng toán tử === thay cho toán tử ==

Thay thế chuỗi và lấy chuỗi con

- str_replace: thay thế chuỗi.
  - Cú pháp:

  mixed str_replace (mixed search, mixed replace, mixed subject [, int &count])

- substr: lấy chuỗi con của một chuỗi
  - Cú pháp: string substr(string string, int start [, int length])

  Tham số:

    - string: chuỗi gốc
    - start: vị trí bắt đầu
    - length: chiều dài chuỗi con

  Kết quả trả về: chuỗi con

Kết hợp và tách chuỗi – Tách chuỗi

- explode()
  - Tách chuỗi str thành nhiều chuỗi con bằng cách chỉ định chuỗi tách separator và gán từng chuỗi con này vào các phần tử của mảng
  - Cú pháp: explode(separator, str)
- implode()
  - Kết hợp các phần tử của mảng thành một chuỗi và các phần tử khi ráp thành chuỗi sẽ cách nhau bằng chỉ định cách separator
  - Cú pháp: implode(separator, array)

Các hàm thời gian

- Kiểm tra ngày hợp lệ
- Định dạng ngày
- Đổi thời gian sang đơn vị giây
- Lấy thời gian hiện tại

Các hàm thời gian

- checkdate()
  - Kiểm tra ngày với tháng, ngày, năm được truyền vào, nếu ngày hợp lệ thì kết quả trả về là TRUE, ngược lại trả về FALSE
  - Cú pháp: checkdate(tháng, ngày, năm)

- date()
  - Kết quả trả về là ngày hiện tại sau khi đã được định dạng
  - Cú pháp: date(chuỗi định dạng [, số giây])

Các hàm thời gian

- mktime()
  - Kết quả trả về là tổng số giây của thời gian nếu các tham số thời gian được truyền vào phù hợp, ngược lại sẽ trả về FALSE
  - Tổng số giây = thời gian được truyền vào trong hàm mktime() - January 1 1970 00:00:00 (vì thời gian bắt đầu là: January 1 1970 00:00:00)
  - Cú pháp: mktime ([hour [, minute [, second [, month [, day [, year [, is_dst]]]]]]])
- time()
  - Kết quả trả về là thời gian hiện tại được đo lường theo giá trị giây (tính từ 1/1/1970 00:00:00 GMT)
  - Cú pháp: time()

Thảo luận