

LẬP TRÌNH WEB



Nguyễn Thị Mai Trang

1

Nội dung

- Chương 1: Tổng quan
- **Chương 2: Ngôn ngữ PHP**
- Chương 3: Controller – View – Model
- Chương 4: HTML Helpers
- Chương 5: Làm việc với Cơ sở dữ liệu

2

Chương 2

Ngôn ngữ PHP

3

Nội dung

-
1. Cấu trúc và cú pháp của PHP
 2. Cấu trúc điều khiển
 3. Hàm và đối tượng
 4. Mảng
 5. Xử lý trang web với PHP
 6. PHP và MySQL
 7. Phát triển ứng dụng web với Laravel

4

2.1 Cấu trúc và cú pháp của PHP

- Cấu trúc trang PHP
- Các kiểu dữ liệu trong PHP
- Biến
- Hằng
- Toán tử

5

2.1.1 Cấu trúc trang PHP

- Trang PHP là sự phối hợp của các thẻ HTML và PHP
- Nhúng code PHP trong trang HTML:

```
<?php
//code
?>
```

```
< ?php
    $num = 1 + 2;
    echo $num;
? >
```

- Có thể kết hợp các thẻ HTML trong code PHP

```
< body >
< ?php
    echo " < h1 > Hello < /h1 >";
? >
< /body >
```

6

Cấu trúc trang PHP (tt)

- Chú thích trong PHP

- Sử dụng ký hiệu `//` hoặc `#` cho mỗi dòng
- Sử dụng `/*` và `*/` cho nhiều dòng

- Cú pháp PHP tương tự ngôn ngữ C, C++

- Cuối câu lệnh có dấu `;`
- Mỗi phương thức đều bắt đầu `{` và đóng bằng dấu `}`

```
// This is a comment
$x += 10; // Increment $x by
10
<?php
/* This is a section
of multiline comments
which will not be
interpreted */
?>
```

2.1.2 Các kiểu dữ liệu trong PHP

- Chia thành 2 nhóm:

- Scalar (cơ bản): boolean, int, float, string,...
- Composite (đồng hợp): array, object,...

- Kiểu dữ liệu trong PHP được khởi gán và chuyển đổi kiểu một cách tự động trong quá trình khai báo hằng và biến.

- Thay đổi kiểu dữ liệu

- Ép kiểu (như ngôn ngữ C)
 - `Tenbien = (kiểu_dữ_liệu) Tenbien;`
- Sử dụng hàm `settype`
 - `settype($Tên_Biến, "Data_type");`
- Hàm `gettype`: trả về kiểu của biến

```
<?php
$don_gia = 7000;
$so_luong = 900;
$thanh_tien =
(double)($so_luong*$don_gia);
?>
```

```
<?php
$x = "12-ABC";
$check = true;
settype($x, "integer");
echo $x; //12
settype($check, "string");
echo $check; //true
?>
```

2.1.3 Biến trong PHP

- Quy tắc đặt tên cho biến:
 - Sử dụng tiền tố \$ trước tên biến, biến có phân biệt chữ hoa, thường, quy tắc định danh giống C++
- Khai báo:
 - Không cần khai báo kiểu dữ liệu
 - Nên gán giá trị khởi đầu cho biến lúc khai báo
 - Cú pháp: \$tenbien [=giatri];
 - Ví dụ:
 - \$username = "Smith";
 - \$count = 17;
 - \$cost = 17.5;

Biến trong PHP (tt)

- Kiểm tra sự tồn tại của biến:
 - Kiểm tra tồn tại: **isset (\$tenbien1, \$tenbien2,...)**
 - Trả về TRUE: tất cả các biến đều có giá trị
 - Trả về FALSE: nếu một biến bất kỳ không có giá trị
 - Ví dụ: if (isset(\$_POST["btSubmit"], \$_POST["username"])) { /*....*/ }
 - Kiểm tra giá trị rỗng: **empty (tenbien)**: kiểm tra biến có giá trị rỗng hay không
 - Trả về TRUE: biến có giá trị rỗng
 - Trả về FALSE: biến có giá trị khác rỗng
 - Các giá trị được xem là rỗng: "" (chuỗi rỗng), NULL, 0 (khi kiểu là integer), FALSE, array(), biến trong lớp được khai báo nhưng không có giá trị
 - Kiểm tra kiểu dữ liệu của biến: sử dụng các hàm có tiền tố **is_datatype**, các hàm này trả về true/false nếu kiểu dữ liệu của biến được kiểm tra là đúng
 - is_array, is_bool, is_double, is_float, is_integer, is_long, is_real, is_string, is_object,...

Biến trong PHP (tt)

• Tầm vực của biến.

- Biến cục bộ: được khai báo và truy xuất bên trong hàm, không tồn tại bên ngoài hàm
- Biến toàn cục: có thể truy xuất bất cứ nơi nào trong trang
 - Khi muốn sử dụng và cập nhật biến toàn cục trong hàm thì phải dùng từ khóa global phía trước biến hoặc dùng **`$_GLOBALS["tenbien"]`**

```
<?php
$a = 1; // toàncục
function Test()
{
    echo $a; // cụcbộ
}
Test(); → không có giá trị
echo $a; → 1
?>
```

```
<?php
$a = 1;
$b = 2;
function Sum()
{
    $_GLOBALS['b'] = $_GLOBALS['a'] + $_GLOBALS['b'];
}
Sum();
echo $b; → 3
?>
```

```
<?php
$a = 1;
$b = 2;
function Sum()
{
    global $a, $b;
    $b = $a + $b;
}
Sum();
echo $b; → 3
?>
```

Lập trình web

11

11

Biến trong PHP (tt)

• Biến static:

- Khai báo sau từ khóa static, không mất đi giá trị khi ra khỏi hàm
- Sẽ giữ nguyên giá trị trước đó khi hàm được gọi một lần nữa

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
Test(); → 0
Test(); → 1
Test(); → 2
?>
```

Lập trình web

12

12

Biến trong PHP (tt)

- Các biến toàn cục được định nghĩa trong PHP
 - **\$GLOBALS**: danh sách các biến toàn cục đã được định nghĩa, truy cập các biến này bằng cách sử dụng tên biến làm từ khóa.
 - **\$_SERVER**: mảng chứa các thông tin như tiêu đề, đường dẫn và vị trí của script. Các phần tử trong mảng này được tạo bởi máy chủ web.
 - **\$_GET**: dữ liệu được truyền thông qua phương thức HTTP GET.
 - **\$_POST**: dữ liệu được truyền thông qua phương thức HTTP POST
 - **\$_FILES**: danh sách các file được tải lên thông qua phương thức HTTP POST.
 - **\$_COOKIE**: danh sách các biến cookie.
 - **\$_SESSION**: danh sách các biến session.
 - **\$_REQUEST**: dữ liệu được truyền giữa các trang
 - **\$_ENV**: danh sách các biến môi trường

2.1.4 Hằng trong PHP

- Khai báo và sử dụng hằng
 - Khai báo: dùng **define**
 - Ví dụ: `define("MAX", 100);`
- Kiểm tra sự tồn tại của hằng: sử dụng hàm **defined**
- Các hằng được định nghĩa sẵn trong PHP:
 - __LINE__ : số dòng hiện tại của file.
 - __FILE__ : đường dẫn đầy đủ của file
 - __DIR__ : thư mục chứa file, (tương đương với `dirname(__FILE__)`)
 - __FUNCTION__ : tên phương thức.
 - __CLASS__ : tên lớp
 - __METHOD__ : tên phương thức của lớp
 - __NAMESPACE__ : Tên namespace hiện hành

```
< ?php
define ('MAX', '100');
if(defined("MAX"))
{
    echo MAX;
}
? >
```

2.1.5 Toán tử trong PHP

• Các toán tử cơ bản

Operator	Description	Example
+	Addition	<code>\$j + 1</code>
-	Subtraction	<code>\$j - 6</code>
*	Multiplication	<code>\$j * 11</code>
/	Division	<code>\$j / 4</code>
%	Modulus (the remainder after a division is performed)	<code>\$j % 9</code>
++	Increment	<code>++\$j</code>
--	Decrement	<code>--\$j</code>
**	Exponentiation (or power)	<code>\$j**2</code>

Operator	Description	Example
==	Is equal to	<code>\$j == 4</code>
!=	Is not equal to	<code>\$j != 21</code>
>	Is greater than	<code>\$j > 3</code>
<	Is less than	<code>\$j < 100</code>
>=	Is greater than or equal to	<code>\$j >= 15</code>
<=	Is less than or equal to	<code>\$j <= 8</code>
<>	Is not equal to	<code>\$j <> 23</code>
===	Is identical to	<code>\$j === "987"</code>
!==	Is not identical to	<code>\$j !== "1.2e3"</code>

Operator	Description	Example
&&	And	<code>\$j == 3 && \$k == 2</code>
and	Low-precedence and	<code>\$j == 3 and \$k == 2</code>
	Or	<code>\$j < 5 \$j > 10</code>
or	Low-precedence or	<code>\$j < 5 or \$j > 10</code>
!	Not	<code>! (\$j == \$k)</code>
xor	Exclusive or	<code>\$j xor \$k</code>

Lập trình web

15

15

2.1.6 Chuỗi

• Chuỗi trong PHP được biểu diễn bằng hai cách:

- Nằm giữa cặp nháy đơn: giữ nguyên nội dung chính xác của chuỗi
- Nằm giữa cặp nháy kép: có thể hiển thị giá trị của biến
- Ví dụ

```
$name = "Cathy";
echo 'your name is $name'; //kết quả: your name is $name
echo "your name is $name"; //kết quả: your name is Cathy
```

- Hiển thị các ký tự đặc biệt: dùng ký tự `'\'` tương tự ngôn ngữ C
- Toán tử nối chuỗi: dấu chấm (`.`)
 - Ví dụ: `"abc"."xyz" → "abcxyz"`

Lập trình web

16

16

2.2 Cấu trúc điều khiển

- Cấu trúc lựa chọn/rẽ nhánh
 - if
 - switch
- Cấu trúc lặp
 - for
 - foreach
 - while
 - do

2.2.1 Cấu trúc lựa chọn

- **Cấu trúc if**

```
if (expr)
    statement
```

- **Cấu trúc if..else**

```
if (expr)
    statement
else
    statement
```

- **Cấu trúc if..elseif**

```
if (expr)
    statement
elseif (expr)
    statement
else
    statement
```

```
$a = $_POST["a"];
$b = $_POST["b"];
$max = ($a > $b) ? $a : $b;
```

- **Biểu thức điều kiện:** (bool_expr? value_if_true: value_if_false)

Cấu trúc lựa chọn (tt)

• Cấu trúc switch

switch(biến điều kiện)

```
{
    case giá trị 1:
        khối lệnh 1
        break;
    case giá trị 2:
        khối lệnh 2
        break;
```

...

[default: khối lệnh thực hiện khi không thỏa tất cả các case trên]

```
}
```

```
switch ($i) {
case 0:
    echo "i equals 0";
    break;
case 1:
    echo "i equals 1";
    break;
case 2:
    echo "i equals 2";
    break;
default:
    echo "i is not equal to 0, 1 or 2";
}
?>
```

2.2.2 Cấu trúc lặp

• Vòng lặp for

for (expr1; expr2; expr3)

```
{
    statement
}
```

- expr1= giá trị khởi đầu của vòng lặp for
- expr2: điều kiện lặp
- expr3: giá trị lặp

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

Cấu trúc lặp (tt)

• **foreach .. as:** thường được dùng để duyệt mảng

- Duyệt giá trị các phần tử trong mảng

```
foreach (array_expression as $value)
{
    statement
}
```

```
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
```

```
red
green
blue
yellow
```

- Duyệt cả khóa và giá trị các phần tử trong mảng

```
foreach (array_expression as $key => $value)
{
    statement
}
```

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $val) {
    echo "$x = $val<br>";
}
```

```
Peter = 35
Ben = 37
Joe = 43
```

Cấu trúc lặp (tt)

• **Vòng lặp while**

```
while (expr)
{
    statement
}
```

```
$count = 1;
while ($count <= 12)
{
    echo "$count times 12 is " . $count * 12 . "<br>";
    ++$count;
}
```

```
1 times 12 is 12
2 times 12 is 24
3 times 12 is 36
...
```

• **Vòng lặp do..while**

```
do
{
    statement
}while (expr);
```

```
$count = 1;
do
{
    echo "$count times 12 is " . $count * 12;
    echo "<br>";
} while (++$count <= 12);
```

break và continue

- **break:**

- thoát khỏi cấu trúc switch
- thoát khỏi vòng lặp

```
$so = $_POST["txtSo"];
$kq = true;
for($i=2; $i < $so; $i++) {
    if($so % $i == 0) {
        $kq = false;
        break;
    }
}
```

- **continue:** các lệnh sau continue không thực hiện mà quay về đầu vòng lặp để thực hiện bước lặp tiếp theo.

```
$tong = 0;
for($i=1; $i<=10; $i++) {
    if ($i%2==0) continue;
    $tong = $tong + $i;
}
echo $tong; →25
```

2.3 Hàm và đối tượng

- Hàm
- include, require
- Đối tượng

2.3.1 Hàm

- Khai báo hàm

- functionName: tên hàm
- parameter: danh sách tham số
- return: giá trị hàm trả về nếu có

```
function functionName ([parameter1]...[,parameterN])
{
    //statement ;
    [return ..... ;]
}
```

- Quy tắc xây dựng hàm

- Hàm có thể có hoặc không có giá trị trả về
- Tên hàm phải khác với các hàm chuẩn
- Một hàm có thể định nghĩa trong 1 hàm khác
- Hàm trong PHP có tầm vực toàn cục

Hàm (tt)

- Ví dụ:

```
<?php
function calFactorial($n)
{
    $result = 1;
    for($i=2 ; $i<=$n ; $i++)
        $result *= $i;
    return $result;
}
$n = 4;
echo $n.'!= '.calFactorial($n);
?>
```

```
<?php
function doublevalue($var=10)
{
    global $temp;
    $temp = $var * 2;
}
$temp = 5;
doublevalue();
echo "\$temp is: $temp"; //20
?>
```

Hàm (tt)

- Truyền giá trị cho hàm

```
$names = fix_names("WILLIAM", "henry", "gatES");
echo $names[0] . " " . $names[1] . " " . $names[2];
function fix_names($n1, $n2, $n3) {
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
    $n3 = ucfirst(strtolower($n3));
    return array($n1, $n2, $n3);
}
```

- Truyền tham chiếu cho hàm:

- Sử dụng toán tử &

```
$a1 = "WILLIAM";
$a2 = "henry";
$a3 = "gatES";
fix_names($a1, $a2, $a3);
echo $a1 . " " . $a2 . " " . $a3;
function fix_names (&$n1, &$n2, &$n3) {
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
    $n3 = ucfirst(strtolower($n3));
}
```

Hàm (tt)

- Một số hàm toán học trong PHP

- **abs(n)**: trả về trị tuyệt đối của n
- **max()**: trả về giá trị lớn nhất trong danh sách đối số hoặc phần tử lớn nhất trong mảng
- **min()**: trả về giá trị nhỏ nhất trong danh sách đối số hoặc phần tử nhỏ nhất trong mảng
- **mt_rand()**: trả về số nguyên ngẫu nhiên từ 0
- **mt_rand (min,max)**: trả về số nguyên ngẫu nhiên từ min (tối thiểu là 0) đến max (tối đa = mt_getrandmax())
- **pi ()**: trả về giá trị PI
- **pow (x, y)**: trả về x mũ y
- **round (number, precision, mode)**: trả về giá trị đã được làm tròn của number
 - precision: số chữ số thập phân, mặc định là 0
 - mode: tùy chọn

Hàm (tt)

• Một số hàm về chuỗi trong PHP

– **substr(string,start,length)**: trả về một chuỗi con từ string tại vị trí start, lấy length ký tự

- start: nếu là số <0: lấy từ cuối chuỗi
- length: tùy chọn, nếu không có sẽ lấy hết chuỗi

```
d
ello world
lo world
orld
```

```
d
ello world
lo world
orld
```



```
<?php
// Positive numbers:
echo substr("Hello world",10)."<br>";
echo substr("Hello world",1)."<br>";
echo substr("Hello world",3)."<br>";
echo substr("Hello world",7)."<br>";
echo "<br>";
// Negative numbers:
echo substr("Hello world",-1)."<br>";
echo substr("Hello world",-10)."<br>";
echo substr("Hello world",-8)."<br>";
echo substr("Hello world",-4)."<br>";
?>
```

Lập trình web

29

29

Hàm (tt)

• Một số hàm về chuỗi trong PHP (tt)

- **strlen(string)**: trả về số ký tự của chuỗi string
- **strpos(string,find,start)**: trả về vị trí chuỗi find trong string, từ vị trí start
 - start: tùy chọn, nếu <0: tìm từ cuối chuỗi

```
<?php
echo strpos("I love php, I love php too!","php");
?>
```

– **strrev(string)**: trả về chuỗi kết quả là đảo của chuỗi string

```
<?php
echo strrev("Hello World!"); //!dlroW olleH
?>
```

Lập trình web

30

30

Hàm (tt)

• Một số hàm về chuỗi trong PHP (tt)

– **strcmp(string1, string2)**: so sánh hai chuỗi, kết quả trả về là một số nguyên

- 0: string1 = string2
- <0: string1 < string2
- >0: string1 > string2

```
<?php
echo strcmp("Hello","Hello");//0
echo strcmp("Hello","hELLo");//-32
?>
```

– **strcasecmp(string1, string2)**: so sánh hai chuỗi không phân biệt chữ hoa, chữ thường

```
<?php
echo strcmp("Hello","Hello");//0
echo strcasecmp("Hello","hELLo");//0
?>
```

– **strncmp(string1, string2, length)**: so sánh hai chuỗi chỉ với length ký tự đầu tiên

Hàm (tt)

• Một số hàm về chuỗi trong PHP (tt)

– **join(separator, array)**: trả về một chuỗi từ mảng

```
<?php
$arr = array('Hello','World!','Beautiful','Day!');
echo join(" ", $arr)."<br>";
echo join("+", $arr)."<br>";
echo join("X", $arr);
?>
```

```
Hello World! Beautiful Day!
Hello+World!+Beautiful+Day!
HelloXWorld!XBeautifulXDay!
```

– **trim(string, charlist)**: trả về chuỗi đã được cắt bỏ từ string các ký tự trong charlist từ hai đầu, nếu không có charlist → cắt bỏ khoảng trắng

– **ltrim()**, **rtrim()**: tương tự trim() nhưng chỉ cắt bỏ từ trái hoặc bên phải

```
<?php
$str = "Hello World!";
echo trim($str,"Hed!"); //llo Worl
?>
```


Hàm (tt)

• Một số hàm về chuỗi trong PHP (tt)

– **explode(separator, string, [limit]):** trả về mảng chứa các chuỗi con tách từ chuỗi string

- separator: các ký tự phân tách chuỗi
- string: chuỗi nguồn
- limit > 0: trả về tối đa limit phần tử
- limit < 0: trả về mảng không chứa phần tử cuối
- limit = 0: trả về mảng chỉ 1 phần tử

```
<?php
$str = 'one, two, three, four';
print_r(explode(',', $str, 0));
print "<br>";
print_r(explode(',', $str, 2));
print "<br>";
print_r(explode(',', $str, -1));
?>
```

Array ([0] => one,two,three,four)
 Array ([0] => one [1] => two,three,four)
 Array ([0] => one [1] => two [2] => three)

Hàm (tt)

• Một số hàm về chuỗi trong PHP (tt)

– **strtok(string,split):** chia một chuỗi thành các chuỗi nhỏ hơn (tokens)

```
<?php
$string = "Hello world. Beautiful day today.";
$token = strtok($string, " ");
while ($token !== false)
{
    echo "$token<br>";
    $token = strtok(" ");
}
?>
```

Hello
 world.
 Beautiful
 day
 today.

– **strtolower(string):** trả về chuỗi đã chuyển về chữ thường từ string

– **strtoupper(string):** trả về chuỗi đã chuyển về chữ hoa từ string

2.3.2 include, require

- **include:**

- Được dùng để nạp một tập tin PHP và nội dung bên trong (thường chứa thư viện các hàm cần sử dụng) → khỏi phải viết lại mã lệnh
- Cú pháp:

```
<?php
    include "library.php";
    // Your code goes here
?>
```

- **include_one:** tương tự như include nhưng không bị báo lỗi khi include một tập tin nhiều lần (nên sử dụng)
- **require:** tương tự include
- **require_one:** tương tự include_one

2.3.3 Đối tượng trong PHP

- Khai báo class

```
class ClassName //ký tự đầu luôn viết hoa
{
    //các biến, các thuộc tính của đối tượng
    //các phương thức xử lý trên đối tượng
}
```

- Tạo đối tượng: sử dụng từ khóa new
- Truy xuất thuộc tính, phương thức đối tượng
 - \$object -> property
 - \$object -> method()
- Tham chiếu **\$this**
 - Thường được dùng bên trong một phương thức của class để tham chiếu đến các thành phần của đối tượng:

```
class Fruit {
    public $name;
    public $color;
    function set_name($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
}
$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');
echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
```

Đối tượng trong PHP (tt)

- Sao chép đối tượng: **clone**

```
class User
{
    public $name, $password;
    function get_password()
    {
        return $this->password;
    }
}
```

```
$object1 = new User();
$object1->name = "Alice";
$object2 = clone $object1;
$object2->name = "Amy";
echo "object1 name = " . $object1->name . "<br>";
echo "object2 name = " . $object2->name;
```

Kết quả:

```
object1 name = Alice
object2 name = Amy
```

Đối tượng trong PHP (tt)

- Phương thức khởi tạo (Constructors)

– Cú pháp:

```
function __construct ([parameters])
{
}
```

```
class User
{
    function __construct($param1, $param2) {
        public $username = "Guest";
    }
}
```

- Phương thức hủy (Destructors)

– Cú pháp:

```
function __destruct ()
{
}
```

```
class User
{
    function __destruct() {
        //code
    }
}
```

Đối tượng trong PHP (tt)

- Sử dụng hằng trong class

- Khai báo với từ khóa **const**
- Truy xuất hằng bên trong phương thức của lớp:
self::CONST_NAME

```
Translate::lookup();
class Translate
{
    const ENGLISH = 0;
    const SPANISH = 1;
    const FRENCH = 2;
    const GERMAN = 3;
    // ...
    static function lookup()
    {
        echo self::SPANISH;
    }
}
```

Đối tượng trong PHP (tt)

- Phạm vi truy cập các thành phần của class

- **public**: có thể truy cập bất kỳ nơi đâu, mặc định
- **protected**: chỉ có thể truy cập bên trong class và các lớp kế thừa
- **private**: chỉ có thể truy cập bên trong class

```
class Example
{
    $name = "Michael"; // Same as public but deprecated
    public $age = 23; // Public property
    protected $usercount; // Protected property
    private function admin() // Private method
    {
        // Admin code goes here
    }
}
```

Đối tượng trong PHP (tt)

• Phương thức static (Static Methods)

- Khai báo sau từ khóa **static**
- Phương thức static không thể truy cập đến các thuộc tính của lớp
- Gọi phương thức static ngoài lớp:
ClassName::MethodName()
- Gọi phương thức static trong lớp:
self::MethodName()

```
User::pwd_string();
class User
{
    static function pwd_string()
    {
        echo "Please enter your password";
    }
    public function __construct() {
        self:: pwd_string();
    }
}
```

Đối tượng trong PHP (tt)

• Thuộc tính static (Static Properties)

- Các biến khai báo sau từ khóa static
- Thường được dùng để lưu dữ liệu dùng chung của lớp (số người truy cập website,...)
- Truy xuất thuộc tính static ngoài lớp: ClassName::PropertyName
- Truy xuất thuộc tính static trong lớp: self::PropertyName

```
$temp = new Test();
echo "Test A: " . Test::$static_property . "<br>";
echo "Test B: " . $temp -> get_sp() . "<br>";
echo "Test C: " . $temp -> static_property . "<br>";
class Test
{
    static $static_property = "I'm static";
    function get_sp() { return self::$static_property; }
}
```

Đối tượng trong PHP (tt)

• Kế thừa (Inheritance)

- Khai báo class kế thừa với từ khóa **extends**

```
<?php
$object = new Subscriber;
$object->name = "Fred";
$object->password = "pword";
$object->phone = "012 345 6789";
$object->email = "fred@bloggs.com";
$object->display();
class User
{
    public $name, $password;
    function save_user()
    {
        echo "Save User code goes here";
    }
}
```

```
class Subscriber extends User
{
    public $phone, $email;
    function display()
    {
        echo "Name: " . $this->name . "<br>";
        echo "Pass: " . $this->password . "<br>";
        echo "Phone: " . $this->phone . "<br>";
        echo "Email: " . $this->email;
    }
}
?>
```

```
Name: Fred
Pass: pword
Phone: 012 345 6789
Email: fred@bloggs.com
```

Lập trình web

43

43

Đối tượng trong PHP (tt)

• Kế thừa (tt)

- Từ khóa **parent**: nếu phương thức trong lớp con trùng tên với phương thức trong lớp cha → ghi đè. Muốn gọi phương thức trùng tên của lớp cha, cần dùng từ khóa **parent**

```
<?php
$object = new Son;
$object->test();
$object->test2();
class Dad
{
    function test()
    {
        echo "[Class Dad] I am your Father<br>";
    }
}
```

```
class Son extends Dad
{
    function test()
    {
        echo "[Class Son] I am Luke<br>";
    }
    function test2()
    {
        parent::test();
    }
}
?>
```

```
[Class Son] I am Luke
[Class Dad] I am your Father
```

Lập trình web

44

44

Đối tượng trong PHP (tt)

• Kế thừa (tt)

– Gọi phương thức khởi tạo của lớp cha trong phương thức khởi tạo của lớp con:

parent::__construct();

```
<?php
$object = new Tiger();
echo "Tigers have...<br>";
echo "Fur: " . $object->fur . "<br>";
echo "Stripes: " . $object->stripes;
class Wildcat
{
    public $fur; // Wildcats have fur
    function __construct()
    {
        $this->fur = "TRUE";
    }
}
```

```
class Tiger extends Wildcat
{
    public $stripes; // Tigers have stripes
    function __construct()
    {
        parent::__construct(); // Call parent constructor first
        $this->stripes = "TRUE";
    }
}
?>
```

Tigers have...
Fur: TRUE
Stripes: TRUE

Lập trình web

45

45

Đối tượng trong PHP (tt)

• Kế thừa (tt)

– Phương thức **final**: được dùng để khai báo phương thức trong lớp cha để ngăn không cho lớp con ghi đè

```
<?php
class User
{
    final function copyright()
    {
        echo "This class was written by Joe Smith";
    }
}
?>
```

Lập trình web

46

46

2.4 Mảng

- Mảng dùng chỉ số
- Mảng kết hợp
- Duyệt mảng dùng foreach
- Mảng hai chiều
- Các hàm thao tác với mảng

2.4.1 Mảng dùng chỉ số (Numerically Indexed Arrays)

- Là loại mảng mà các phần tử được xác định thông qua vị trí, bắt đầu là 0
- Tạo mảng bằng cách thêm lần lượt từng phần tử

```
$paper[] = "Copier";
$paper[] = "Inkjet";
$paper[] = "Laser";
$paper[] = "Photo";
print_r($paper);
```



```
Array
(
    [0] => Copier
    [1] => Inkjet
    [2] => Laser
    [3] => Photo
)
```

```
$paper[0] = "Copier";
$paper[1] = "Inkjet";
$paper[2] = "Laser";
$paper[3] = "Photo";
print_r($paper);
```

- Tạo mảng bằng cách thêm từng phần tử với vị trí xác định
- Truy xuất các phần tử trong mảng

```
for ($j = 0 ; $j < 4 ; ++$j)
    echo "$j: $paper[$j]<br>";
```

```
0: Copier
1: Inkjet
2: Laser
3: Photo
```


2.4.2 Mảng kết hợp (Associative Arrays)

- Loại mảng mà các phần tử được xác định thông qua một trường khóa (key)
- Được sử dụng rất phổ biến trong PHP
 - Lưu kết quả trả về từ một truy vấn trong database
 - Trích xuất thông tin từ XML và HTML, ví dụ: các search engine dùng HTML parser dạng:
 - \$html['title'] = "Trang web của tôi";
 - \$html['body'] = "... nội dung trang web ..."
- Tạo mảng bằng cách thêm lần lượt từng phần tử vào mảng

```
<?php
$paper['copier'] = "Copier & Multipurpose";
$paper['inkjet'] = "Inkjet Printer";
$paper['laser'] = "Laser Printer";
$paper['photo'] = "Photographic Paper";
echo $paper['laser'];
?>
```

Mảng kết hợp (Associative Arrays)

- Tạo mảng dùng array() với các keyword cho trước

```
<?php
$p1 = array("Copier", "Inkjet", "Laser", "Photo");
echo "p1 element: " . $p1[2] . "<br>";

$p2 = array('copier' => "Copier & Multipurpose",
            'inkjet' => "Inkjet Printer",
            'laser' => "Laser Printer",
            'photo' => "Photographic Paper");
echo "p2 element: " . $p2['inkjet'] . "<br>";
?>
```

2.4.3 Duyệt mảng

- Dùng các cấu trúc lặp for, while, do với mảng sử dụng chỉ số (như C)
- Dùng **foreach...as**
- Dùng **each** và **list**

```
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper
```

```
<?php
$paper = array('copier' => "Copier & Multipurpose",
               'inkjet' => "Inkjet Printer",
               'laser' => "Laser Printer",
               'photo' => "Photographic Paper");
foreach ($paper as $item => $description)
    echo "$item: $description<br>";
//hoặc
while (list($item, $description) = each($paper))
    echo "$item: $description<br>";
?>
```

51

2.4.4 Mảng nhiều chiều (Multidimensional Arrays)

- Mảng dùng chỉ số: là mảng mà mỗi hàng là một mảng một chiều
- Truy xuất theo chỉ số

Row number 0

- Volvo
- 22
- 18

Row number 1

- BMW
- 15
- 13



```
$cars = array (
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,15)
);
for ($row = 0; $row < 4; $row++) {
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++) {
        echo "<li>".$cars[$row][$col]."</li>";
    }
    echo "</ul>";
}
```

52

2.4.4 Mảng nhiều chiều (Multidimensional Arrays)

- Mảng hai chiều kết hợp:
mỗi phần tử là một mảng kết hợp

```
paper: copier (Copier & Multipurpose)
paper: inkjet (Inkjet Printer)
paper: laser (Laser Printer)
paper: photo (Photographic Paper)
pens: ball (Ball Point)
pens: hilite (Highlighters)
pens: marker (Markers)
misc: tape (Sticky Tape)
misc: glue (Adhesives)
misc: clips (Paperclips)
```

```
$products = array(
    'paper' => array( 'copier' => "Copier & Multipurpose",
                     'inkjet' => "Inkjet Printer",
                     'laser' => "Laser Printer",
                     'photo' => "Photographic Paper"),
    'pens' => array( 'ball' => "Ball Point",
                    'hilite' => "Highlighters",
                    'marker' => "Markers"),
    'misc' => array( 'tape' => "Sticky Tape",
                    'glue' => "Adhesives",
                    'clips' => "Paperclips")
);
echo "<pre>";
foreach($products as $section => $items)
    foreach($items as $key => $value)
        echo "$section:\t$key\t($value)<br>";
echo "</pre>";
```

- Truy cập theo cú pháp:
\$products['misc']['glue'];

Lập trình web

53

53

2.4.5 Các hàm thao tác với mảng

- is_array**: kiểm tra một biến có phải là mảng hay không
 - echo (is_array(\$fred)) ? "Is an array" : "Is not an array";
- count(array, [mode])** : trả về số phần tử trong mảng
 - echo count(\$fred);
 - Đối với mảng nhiều chiều: count(\$fred, 0/1), nếu là 1: đếm tất cả các phần tử con
- Các hàm sắp xếp mảng**:
 - **sort()**: sắp xếp mảng tăng dần
 - **rsort()**: sắp xếp mảng giảm dần
 - **asort()**: sắp xếp mảng kết hợp tăng dần theo trường value
 - **ksort()**: sắp xếp mảng kết hợp tăng dần theo trường key
 - **arsort()**: sắp xếp mảng kết hợp giảm dần theo trường value
 - **rsort()**: sắp xếp mảng kết hợp giảm dần theo trường key

Lập trình web

54

54

Các hàm thao tác với mảng (tt)

- Các hàm sắp xếp mảng: ví dụ

```
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
rsort($numbers);
```

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
ksort($age);
arsort($age);
krsort($age);
```

- **explode**: tách chuỗi vào mảng theo các ký tự chỉ định

```
$arr1 = explode(' ', "This is a sentence with seven words");
print_r($temp);
$arr2 = explode('***', "A***sentence***with***asterisks");
print_r($temp);
```

Các hàm thao tác với mảng (tt)

- **min(array)**: phần tử nhỏ nhất trong mảng
 - echo min(array(4,6,8,10));
- **max(array)**: phần tử lớn nhất trong mảng
 - echo max(array(4,6,8,10));
- **reset(array)**: chuyển con trỏ về đầu mảng, trả về phần tử đầu tiên của mảng
 - reset(\$fred); // Throw away return value
 - \$item = reset(\$fred); // Keep first element of the array in \$item
- **end (array)**: chuyển con trỏ về cuối mảng, trả về phần tử cuối của mảng
 - end(\$fred);
 - \$item = end(\$fred);
- **shuffle (array)**: sắp xếp mảng ngẫu nhiên
 - shuffle(\$cards);

Các hàm thao tác với mảng (tt)

• **array_reverse**: đảo mảng

– `array_reverse($array [, $boolean]);`

- `boolean`: mặc định = `FALSE`: giữ nguyên các cặp `key=>value`

```
$array = array( "php", "js", "css", "html");
$reversed = array_reverse($array);
$preserved = array_reverse($array, true);
echo "<pre>";
print_r($array);
echo "</pre>";
echo "<pre>";
print_r($reversed);
echo "</pre>";
echo "<pre>";
print_r($preserved);
echo "</pre>";
```



```
Array
(
    [0] => php
    [1] => js
    [2] => css
    [3] => html
)
Array
(
    [0] => html
    [1] => css
    [2] => js
    [3] => php
)
Array
(
    [3] => html
    [2] => css
    [1] => js
    [0] => php
)
```

Lập trình web

57

57

Các hàm thao tác với mảng (tt)

• **array_merge** (array, array) : trộn 2 mảng

```
$a1=array("a"=>"red","b"=>"green");
$a2=array("c"=>"blue","b"=>"yellow");
print_r(array_merge($a1,$a2));
```



```
Array ( [a] => red [b] => yellow [c] => blue )
```

• **array_search**(value, array, [strict])

– `strict`: mặc định là `false`, nếu là `true`: tìm chính xác

```
$a=array("a"=>"5","b"=>5,"c"=>"5");
echo array_search(5,$a,true);
```



b

Lập trình web

58

58

Các hàm thao tác với mảng (tt)

- **array_unshift(\$array, \$element1, \$element2,):** thêm phần tử đầu mảng

```
$a=array("a"=>"red","b"=>"green");
array_unshift($a,"blue");
print_r($a);
```

Array ([0] => blue [a] => red [b] => green)

- **array_shift (\$array):** lấy phần tử ở đầu mảng

```
$a=array("a"=>"red","b"=>"green","c"=>"blue");
echo array_shift($a)."<br>";
print_r ($a);
```

red
Array ([b] => green [c] => blue)

Các hàm thao tác với mảng (tt)

- **array_push(\$array, \$element1, \$element2,):** thêm phần tử cuối mảng

```
$a=array("red","green");
array_push($a,"blue","yellow");
print_r($a);
```

Array ([0] => red [1] => green [2] => blue [3] => yellow)

- **array_pop(\$array):** lấy phần tử ở cuối mảng

```
$a=array("red","green","blue");
echo array_pop($a)."<br>";
print_r($a);
```

blue
Array ([0] => red [1] => green)

2.5 Xử lý trang PHP

- Chuyển hướng trang web
- Truyền dữ liệu giữa các trang

2.5.1 Chuyển hướng trang web

- Sử dụng hàm header:
 - **header(header, replace, http_response_code=null)**
 - header: URL của trang sẽ chuyển đến
ví dụ: header('Location: <http://cunghoclaptrinh.com/>');
 - replace: mặc định = true, nghĩa là định dạng của chuỗi header sẽ được replace chứ không phải khai báo mới (trường hợp khai báo nhiều header).
 - http_response_code: mặc định = null, là mã code trả về từ Server. (Ví dụ 404 - not found)

2.5.2 Truyền dữ liệu giữa các trang

- Sử dụng các đối tượng: \$_GET, \$_POST, \$_REQUEST, \$_COOKIE, \$_SESSION
- Trang web nhập dữ liệu:
 - Sử dụng đối tượng <form>
 - Nhập liệu thông qua các control
 - Thực hiện việc truyền dữ liệu thông qua Submit
- Trang web nhận dữ liệu (URL): Sử dụng các biến toàn cục của PHP
 - \$_POST["FieldName"]
 - \$_GET["FieldName"]
 - \$_REQUEST["FieldName"]

```
<form action = "URL" method ="get/post">
...
<input type="submit" value="Xử lý">
</form>
```

Đối tượng \$_GET

- Dữ liệu gửi từ trình duyệt lên server qua phương thức GET là phần dữ liệu nằm sau dấu ? trong địa chỉ URL
 - Ví dụ: http://abc.com/topic.php?topicId=161
→ dữ liệu gửi là: topicId=161
- Nếu có nhiều cặp **biến = giá trị** thì được phân cách bởi dấu &
 - Ví dụ: http://abc.com/index.php?method=reply&topicId=161&forumId=20.
→ dữ liệu gửi là:
 - method=reply
 - topicId=161
 - forumId=20

Đối tượng \$_GET

- Các cặp biến và giá trị đó sẽ được lưu vào mảng có tên là \$_GET.
 - Ví dụ: `http://abc.com/index.php?method=reply&topicId=161&forumId=20`
 PHP sẽ tự động sinh ra một mảng \$_GET có nội dung sau:
 - `$_GET["method"] = "reply"`
 - `$_GET["topicId"] = 161`
 - `$_GET["forumId"] = 20`
- Truy xuất dữ liệu thông qua biến toàn cục của PHP
 - `$_GET["FieldName"]`
 - `$_REQUEST["FieldName"]`

Đối tượng \$_GET

- Ví dụ: sử dụng đối tượng \$_GET

```
<body>
<h1>Tìm sách</h1>
<form action="bookresult.php" method="GET" >
  Từ khóa : <input type="text" name="txtKey"/>
  <input type="submit" value="Tìm"/>
</form>
</body>
```

```
Trang bookresult.php
<body>
<?php
  $key = $_GET["txtKey"];
?>
<h1>Tìm sách</h1>
Từ khóa tìm sách là : <?php echo $key; ?>
<br />
</body>
```

Tìm sách

Từ khóa :

Tìm sách

Từ khóa tìm sách là : C#

Đối tượng \$_POST

- Dữ liệu gửi từ trình duyệt lên server qua phương thức POST là phần dữ liệu được lưu trữ trong phần thân Request.
- Tham số truyền đi được ẩn bên trong form
- Ví dụ: URL truyền theo phương thức POST: [http:// abc.com /topic.php](http://abc.com/topic.php)
 - Khi đó, trình duyệt cũng sẽ gửi lên server một cặp: biến = giá trị (lưu trong phần thân Request), trong đó biến có tên là topicId và giá trị là 161 (topicId=161).
- Truy xuất dữ liệu thông qua biến toàn cục của PHP
 - \$_POST ["FieldName"]
 - \$_REQUEST ["FieldName"]

Đối tượng \$_POST

- Ví dụ: sử dụng đối tượng \$_POST

File register.php

```
<form method="post" action="registerprocess.php">
<label for="firstname">First name</label>
<input type="text" name="firstname" id="firstname" size="25" />
<br />
<label for="lastname">Last name</label>
<input type="text" name="lastname" id="lastname" size="25" />
<br />
<input type="submit" name="submit" value="Register" />
<br />
</form>
```

File registerprocess.php

```
<?php
if (isset($_POST["firstname"],$_POST["lastname"]))
{
    $f = $_POST["firstname"];
    $l = $_POST["lastname"];
    echo "Welcome $f $l";
}
?>
```

Đối tượng \$_REQUEST

- Được dùng để truy xuất giá trị các biến gửi bằng phương thức get hoặc post
- Ví dụ: sử dụng đối tượng \$_REQUEST

```
<body>
<form action="" method="GET" >
    Từ khóa : <input type="text" name="txtKey"/>
    <input type="submit" value="Tìm"/>
</form>
<?php
    if(isset($_REQUEST["txtKey"]))
    {
        $key = $_REQUEST["txtKey"];
        print "Từ khóa tìm sách là : $key";
    }
?>
</body>
```

Đối tượng \$_COOKIE

- Là chuỗi dữ liệu được truyền đến browser từ server, dữ liệu này sẽ được browser lưu trữ (trong memory hoặc trên đĩa) và sẽ gửi ngược lên lại server mỗi khi browser tải trang web từ server.
- Những thông tin được lưu trữ trong cookie phụ thuộc vào website trên server.
- Cookie được tạo ra bởi website và gửi tới browser, do đó hai website khác nhau sẽ có 2 cookie khác nhau gửi tới browser.
- Mỗi browser quản lý và lưu trữ cookie theo cách riêng của mình, do đó hai browser cùng truy cập vào một website sẽ nhận được hai cookie khác nhau.

Đối tượng \$_COOKIE

• Sử dụng Cookie trong PHP:

- Để đặt (set) cookie: **\$_COOKIE[tên_cookie] = giá_trị;**
- Đọc (get) lại giá trị của cookie: **\$_COOKIE[tên_cookie]**

```
<?php
    $t="1111";
    setcookie("a",$t);
?>
...
<body>
Giá trị gửi lên cookies:
<?php echo $t; ?>
<a href="b.php"> qua trang b </a>
</body>
```

```
<body>
<a href="a.php"> qua trang a</a>
<?php
    if (isset($_COOKIE['a']))
    {
        echo "gia tri lay duoc ".$_COOKIE['a'];
    }
    else
        echo "khong lay duoc";
?>
</body>
```

Đối tượng \$_SESSION

- Là dữ liệu được lưu trữ trong một phiên làm việc, sẽ tự động mất đi khi người dùng đóng trình duyệt, chuyển sang website khác hoặc không thao tác trên trang web trong một khoảng thời gian (time out)
- Mỗi session sẽ có một định danh (ID).
- Các hàm liên quan đến Session:
 - session_start(): khởi tạo session.
 - session_register(tên biến): đăng ký biến session
 - \$_SESSION[tên_session] = giá_trị: đặt giá trị cho session
 - \$_SESSION[tên_session]: đọc giá trị từ session
 - session_destroy(): hủy tất cả các dữ liệu trong session
 - session_unset(): hủy tất cả các biến trong session
 - session_unregister(tên biến) hủy 1 biến trong session

Đối tượng \$_SESSION

• Ví dụ sử dụng SESSION

```
<?php
session_start();
$t=time() ; $_SESSION['username'] = 'guest';
$_SESSION['password'] = $t;
?>
<html> <head>
<title>a.php</title>
</head>
<body>
Giá trị của session đã được gán:<br> username = guest
<br> time = <?php echo $t; ?>
<br> Click
<a href="b.php">vào đây</a> de kiểm tra.
</body>
</html>
```

```
<?php
session_start();
?>
<html>
<head>
<title>Trang s_b.php</title>
</head>
<body>
Giá trị session lấy được
<a href="a.php">file a.php</a>:<br>
username =
<?php echo $_SESSION['username']; ?>
<br> time =
<?php echo $_SESSION['password']; ?>
</body>
</html>
```

Lập trình web

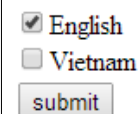
73

73

Một số ví dụ

• Truyền/nhận dữ liệu từ checkbox

```
Trang checkbox.php
<body>
<form method="get" action="checkbox.php">
<input type="checkbox" name="chk1" value="en">English <br>
<input type="checkbox" name="chk2">Vietnam<br>
<input type="submit" value="submit"><br>
</form>
<?php
echo "checkboxbox 1 : " . $_REQUEST["chk1"] . "<br>";
echo "checkboxbox 2 : " . $_REQUEST["chk2"];
?>
</body>
```



checkboxbox 1 : en
checkboxbox 2 :

Lập trình web

74

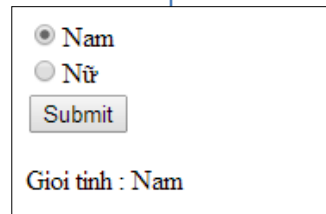
74

Một số ví dụ (tt)

• Truyền/nhận dữ liệu từ Radio button

Trang radio.php

```
<body>
<form action="radio.php" method="GET">
<input type="RADIO" NAME="rdGT" VALUE="Nam">Nam<br>
<input type="RADIO" NAME="rdGT" VALUE="Nu">Nữ<br>
<input type="SUBMIT" VALUE="Submit">
</form>
<? php
    if (isset($_GET['rdGT']))
        echo "Giới tính : " . $_GET['rdGT'];
?>
</body>
```



☒ Nam
☐ Nữ

 Giới tính : Nam

Lập trình web

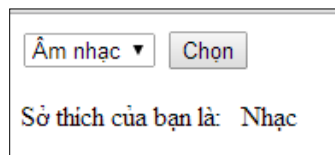
75

75

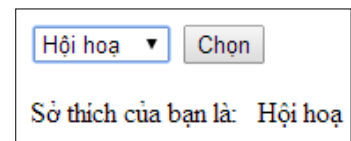
Một số ví dụ

– Truyền/nhận dữ liệu từ DropDownList

```
<body>
<form method="POST" action="dropdownlist.php">
<select name="chon">
<option value="Nhạc"><?php if($_POST["chon"]=="Nhạc") echo "selected"; ?> >Âm nhạc</option>
<option value="Thể thao"><?php if($_POST["chon"]=="Thể thao") echo "selected"; ?> >Thể thao</option>
<option value="Hội họa"><?php if($_POST["chon"]=="Hội họa") echo "selected"; ?> >Hội họa</option>
</select>
<input type="submit" name="submit" value="Chọn"/>
</form>
Sở thích của bạn là: &nbsp;    
<?php
    if (isset($_POST["chon"]))
        echo $_POST["chon"] . "<br/>";
?>
</body>
```



Âm nhạc ▼ Chọn
 Sở thích của bạn là: Nhạc



Hội họa ▼ Chọn
 Sở thích của bạn là: Hội họa

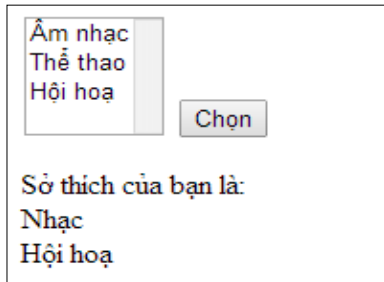
Lập trình web

76

76

Một số ví dụ

– Truyền/nhận dữ liệu từ ListBox



Sở thích của bạn là:
Nhạc
Hội hoạ

Lập trình web

```
<body>
<form method="POST" action="listbox.php">
  <select name="chon[ ]" multiple>
    <option value="Nhạc">Âm nhạc</option>
    <option value="Thể thao">Thể thao</option>
    <option>Hội hoạ</option>
  </select>
  <input type="submit" name="submit" value="Chọn"/>
</form>
Sở thích của bạn là: <br/>
<?php
if(isset($_POST["chon"])) {
    foreach ($_POST["chon"] as $item)
        echo $item. "<br/>";
}
?>
</body>
```

77

77

Một số ví dụ

– Upload file

```
<body>
<form method="post"
  enctype="multipart/form-data">
  Select image to upload:<br>
  <input type="file" name="fileUpload"
    id="fileUpload">
  <input type="submit" value="Upload Image"
    name="submit">
</form>
```

Select image to upload:
 No file chosen
 The file animals.png has been uploaded.

Lập trình web

```
<?php
if(isset($_POST["submit"])) {
    $dir = "Files/";
    try{
        $fileName = $dir.basename($_FILES["fileUpload"]["name"]);
        if (getimagesize($_FILES["fileUpload"]["tmp_name"]))
        {
            if (move_uploaded_file($_FILES["fileUpload"]["tmp_name"],
                $fileName))
                echo "The file ".htmlspecialchars(basename(
                    $_FILES["fileUpload"]["name"])). " has been uploaded.";
        }
    }catch(Error $e) { echo "Error uploading your file.";}
}
?>
</body>
```

78

78

2.6 PHP và MySQL

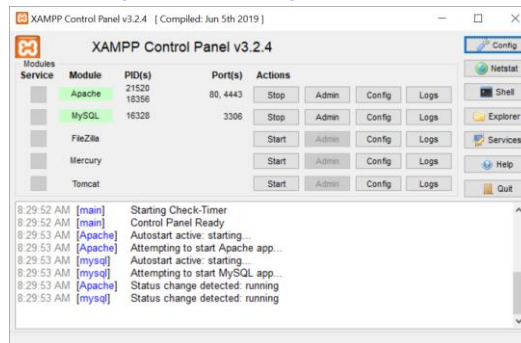
- Giới thiệu MySQL Improved Extension (MySQLi)
- Giới thiệu phpMyAdmin
- Thao tác với cơ sở dữ liệu MySQLi

2.6.1 Giới thiệu MySQL Improved Extension

- MySQLi (MySQL Improved Extension) là một thư viện mở rộng được phát triển dành cho ngôn ngữ PHP giúp lập trình viên có thể dễ dàng kết nối tới cơ sở dữ liệu MySQL.
- Thông thường khi phát triển ứng dụng web sử dụng ngôn ngữ PHP thì MySQL thường được chọn làm hệ quản trị cơ sở dữ liệu.
- MySQLi cung cấp các hàm PHP như `mysqli_connect()`, `mysqli_query()`,... giúp chúng ta có thể kết nối và thao tác với cơ sở dữ liệu MySQL
- MySQLi được cài đặt cùng với các phần mềm Web Server như XAMPP

Giới thiệu MySQL Improved Extension (tt)

- XAMPP là gói phần mềm miễn phí và được cấu hình sẵn bao gồm các thành phần như sau: Apache, MySQL, PHP, phpMyAdmin giúp loại bỏ rắc rối của việc tải xuống và cài đặt và cấu hình từng thành phần riêng lẻ trước đây.
- Download và cài XAMPP: <https://www.apachefriends.org/index.html>
- Giao diện XAMPP



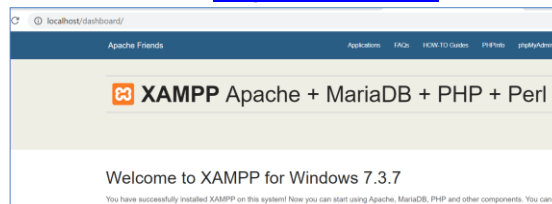
Lập trình web

81

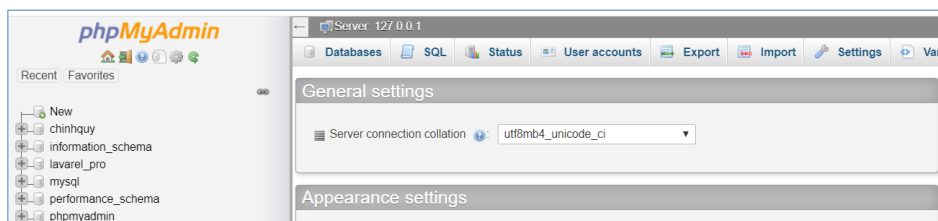
81

Giới thiệu MySQL Improved Extension (tt)

- Kiểm tra hoạt động của XAMPP: <http://localhost>



- Kiểm tra hoạt động của phpMyAdmin: <http://localhost/phpmyadmin/>



Lập trình web

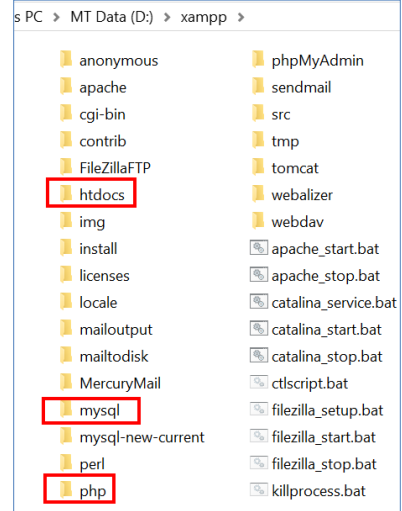
82

82

Giới thiệu MySQL Improved Extension (tt)

• Cấu trúc thư mục của XAMPP

- htdocs: thư mục chứa mã nguồn ứng dụng (url: <http://localhost> hoặc <http://127.0.0.1>)
- mysql: thư mục chứa các tập tin về cơ sở dữ liệu, trong đó có tập tin cấu hình quan trọng là my.ini
- php: thư mục chứa các tập tin về php, trong đó có tập tin cấu hình quan trọng là php.ini



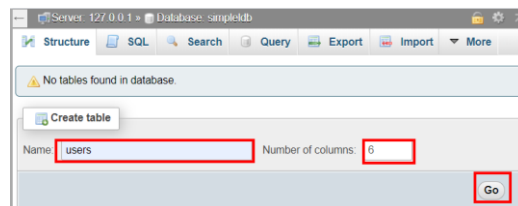
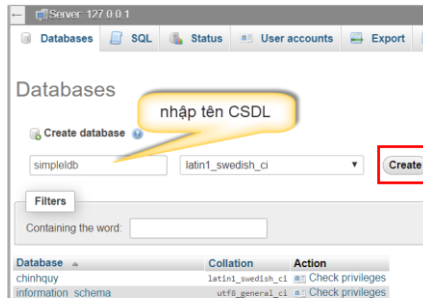
2.6.2 Giới thiệu phpMyAdmin

• Cung cấp một giao diện cho phép thao tác với cơ sở dữ liệu MySQL

- Tạo, xóa CSDL
- Tạo các bảng (table) trong CSDL
- Thao tác với các record
- Thực hiện các truy vấn
- Backup, restore CSDL

Tạo cơ sở dữ liệu

- Truy cập phpMyAdmin
- Click vào Database
- Nhập tên CSDL → Create
- Tạo Table:
 - Nhập tên table, số cột
 - Click Go



Lập trình web

85

85

Tạo cơ sở dữ liệu

- Tạo cấu trúc bảng, giả sử gồm các trường

Column name	Type	Length/Values	Default	Attributes	NULL	Index	A_I
user_id	MEDIUMINT	6	None	UNSIGNED	<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>
fname	VARCHAR	30	None		<input type="checkbox"/>		<input type="checkbox"/>
lname	VARCHAR	40	None		<input type="checkbox"/>		<input type="checkbox"/>
email	VARCHAR	50	None		<input type="checkbox"/>		<input type="checkbox"/>
psword	CHAR	40	None		<input type="checkbox"/>		<input type="checkbox"/>
registration_date	DATETIME		None		<input type="checkbox"/>		<input type="checkbox"/>

Lập trình web

86

86

Tạo cơ sở dữ liệu

- Nhập thông tin cho các trường → save

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
user_id	MEDIUMINT	6	None		UNSIGNED	<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
Pick from Central Columns									
fname	VARCHAR	30	None			<input type="checkbox"/>	---	<input type="checkbox"/>	
Pick from Central Columns									
lname	VARCHAR	40	None			<input type="checkbox"/>	---	<input type="checkbox"/>	
Pick from Central Columns									
email	VARCHAR	50	None			<input type="checkbox"/>	---	<input type="checkbox"/>	
Pick from Central Columns									
psword	CHAR	40	None			<input type="checkbox"/>	---	<input type="checkbox"/>	
Pick from Central Columns									
registration_date	DATETIME		None			<input type="checkbox"/>	---	<input type="checkbox"/>	

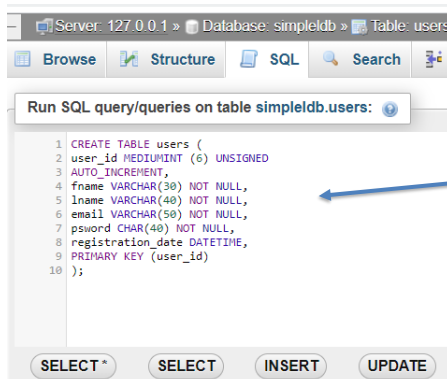
Lập trình web

87

87

Tạo cơ sở dữ liệu

- Có thể tạo bảng bằng ngôn ngữ SQL bằng cách chọn tab SQL và nhập câu truy vấn như sau:



```
CREATE TABLE users (
  user_id MEDIUMINT (6) UNSIGNED
  AUTO_INCREMENT,
  fname VARCHAR(30) NOT NULL,
  lname VARCHAR(40) NOT NULL,
  email VARCHAR(50) NOT NULL,
  psword CHAR(40) NOT NULL,
  registration_date DATETIME,
  PRIMARY KEY (user_id)
);
```

Lập trình web

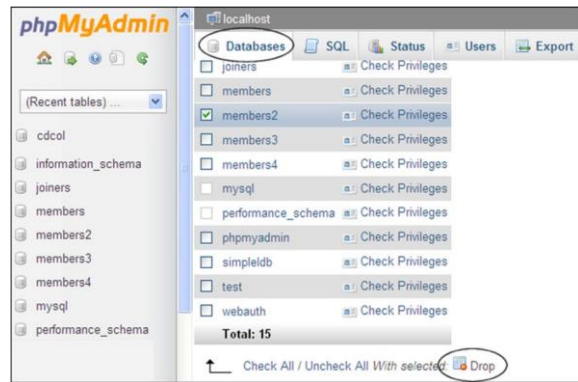
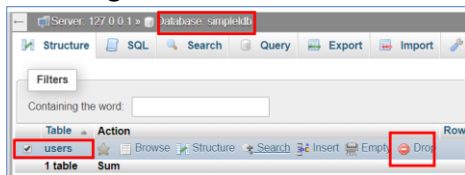
88

88

Xóa cơ sở dữ liệu, bảng

- Xóa cơ sở dữ liệu

- Xóa bảng



2.6.3 Thao tác với cơ sở dữ liệu MySQLi

- Kết nối CSDL

```
$con = mysqli_connect("localhost","my_user","my_password","my_db");
```

– Hoặc

```
$con = mysqli_connect("localhost","my_user","my_password");  
mysqli_select_db($con, "my_db");
```

– Cần kiểm tra kết nối thành công:

```
if (!$con)  
{  
    die("Connection error: " . mysqli_connect_errno());  
    //hoặc die("Connection error: " . mysqli_connect_error());  
}  
?>
```

Thao tác với cơ sở dữ liệu MySQLi (tt)

• Kết nối CSDL

– Ví dụ:

```
$servername = 'localhost';
$username = 'root';
$password = '123456';
$conn = mysqli_connect($servername, $username, $password, $db);
if(!$conn)
{
    echo "Connect Failed!". mysqli_connect_error($conn);
}
else
{
    echo "Successsful";
}
```

Lập trình web

91

91

Thao tác với cơ sở dữ liệu MySQLi (tt)

• Thực thi câu lệnh truy vấn dữ liệu

biến kết nối

```
mysqli_query (connection, query, [resultmode]);
```

Hoặc

```
$conn->query (query, [resultmode]);
```

MYSQLI_STORE_RESULT: mặc định
MYSQLI_USE_RESULT: dùng khi dữ liệu lớn

câu lệnh SQL

• Truy vấn lấy dữ liệu

- Kết nối cơ sở dữ liệu
- Thực thi câu truy vấn select...
- Xử lý kết quả trả về từ các hàm:
 - **mysqli_fetch_all()**: kết quả trả về dưới dạng một mảng kết hợp, mảng chỉ số hoặc cả hai
 - **mysqli_fetch_array()**: trả về một hàng kết quả dưới dạng mảng kết hợp, mảng chỉ số hoặc cả hai
 - **mysqli_fetch_assoc()**: trả về một hàng kết quả dưới dạng một mảng kết hợp
 - **mysqli_fetch_row()**: trả về một hàng kết quả dưới dạng một mảng chỉ số

Lập trình web

92

92

Thao tác với cơ sở dữ liệu MySQLi (tt)

- **mysqli_fetch_all()**: tập kết quả trả về dưới dạng một mảng kết hợp, một mảng chỉ số hoặc cả hai, cú pháp: **mysqli_fetch_all (\$result, \$result_type)**
 - \$result: kết quả trả về của các hàm: mysqli_query(), mysqli_store_result() hoặc mysqli_use_result()
 - \$result_type: tùy chọn, quy định kiểu mảng trả về
 - MYSQLI_ASSOC
 - MYSQLI_NUM
 - MYSQLI_BOTH (mặc định)

Thao tác với cơ sở dữ liệu MySQLi (tt)

- **mysqli_fetch_all()** ví dụ


```
$conn=mysqli_connect("localhost","my_user","my_pass","my_db");
if (mysqli_connect_errno()){
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
    exit ();
}
$sql = "SELECT Lastname, Age FROM Persons ORDER BY Lastname";
$result = mysqli_query($conn, $sql);
//hoặc $result= $conn -> query($sql);
$rows = mysqli_fetch_all($result, MYSQLI_ASSOC);
foreach ($rows as $row)
    echo $row["Lastname"]. "<br>". $row["Age"];
mysqli_free_result($result);
mysqli_close($conn);
```

Thao tác với cơ sở dữ liệu MySQLi (tt)

- **mysqli_fetch_array()**: như **mysqli_fetch_all()** nhưng chỉ trả về một hàng,
 Cú pháp: **mysqli_fetch_array(\$result, \$result_type)**

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
    exit();
}
$sql = "SELECT Lastname, Age FROM Persons Where PersonId = 1";
$result = mysqli_query($con,$sql);
$row = mysqli_fetch_array($result, MYSQLI_NUM);
printf ("%s (%s)\n", $row[0], $row[1]);
$row = mysqli_fetch_array($result, MYSQLI_ASSOC);
printf ("%s (%s)\n", $row["Lastname"], $row["Age"]);
mysqli_free_result($result);
mysqli_close($con);
```

Lập trình web

95/164

95

Thao tác với cơ sở dữ liệu MySQLi (tt)

- **mysqli_fetch_array() (tt)**

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
    exit();
}
$sql = "SELECT Lastname, Age FROM Persons ORDER BY Lastname";
$result = mysqli_query($con,$sql);
if ($result = mysqli_query($con, $sql)) {
    while ($row = mysqli_fetch_array($result)) {
        echo $row['Lastname'] . '-' . $row['Age'] . '<br/>';
        $i++;
    }
}
else echo mysqli_error($connect);
mysqli_free_result($result);
mysqli_close($con);
```

Lập trình web

96/164

96

Thao tác với cơ sở dữ liệu MySQLi (tt)

- **mysqli_fetch_assoc()**: như **mysqli_fetch_array()** nhưng chỉ trả về dạng mảng kết hợp, cú pháp: **mysqli_fetch_assoc(\$result)**

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno()){
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
    exit();
}
$result = mysqli_query($con, "Select * From Products");
if(mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result))
        echo $row['ProductId'].','.$row['ProductName']. "<br />";
}
mysqli_free_result($result);
mysqli_close($con);
```

Lập trình web

97/164

97

Thao tác với cơ sở dữ liệu MySQLi (tt)

- **mysqli_fetch_row()**: như **mysqli_fetch_array()** nhưng chỉ trả về dạng mảng chỉ số, cú pháp: **mysqli_fetch_row(\$result)**

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno()){
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$sql="SELECT Lastname, Age FROM Persons ORDER BY Lastname";
if ($result=mysqli_query($con,$sql)){
    while ($row = mysqli_fetch_row($result)){
        printf ("%s (%s)\n", $row[0], $row[1]);
    }
    mysqli_free_result($result);
}
mysqli_close($con);
```

Lập trình web

98/164

98

Thao tác với cơ sở dữ liệu MySQLi (tt)

• Truy vấn hành động:

- Thực thi các câu truy vấn Insert, Update, Delete với phương thức mysqli()
- Chèn dữ liệu:

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno()){
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
    exit();
}
$sqlInsert = "INSERT INTO Products(ProductId, ProductName) VALUES (10, 'Iphone5 Plus')";
$result = mysqli_query($con, $sqlInsert);
mysqli_close($con);
```

Thao tác với cơ sở dữ liệu MySQLi (tt)

– Xóa dữ liệu:

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error(); exit();
}
$sqlDelete = "Delete From Products Where ProductId =10";
$result = mysqli_query($con, $sqlDelete);
mysqli_close($con);
```

– Cập nhật dữ liệu:

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno()){
    echo "Failed to connect to MySQL: " . mysqli_connect_error(); exit();
}
$sqlUpdate = "Update Products Set ProductName = 'Iphone7 Plus' Where ProductId =10";
$result = mysqli_query($con, $sqlUpdate);
mysqli_close($con);
```

2.7 Giới thiệu Framework Laravel

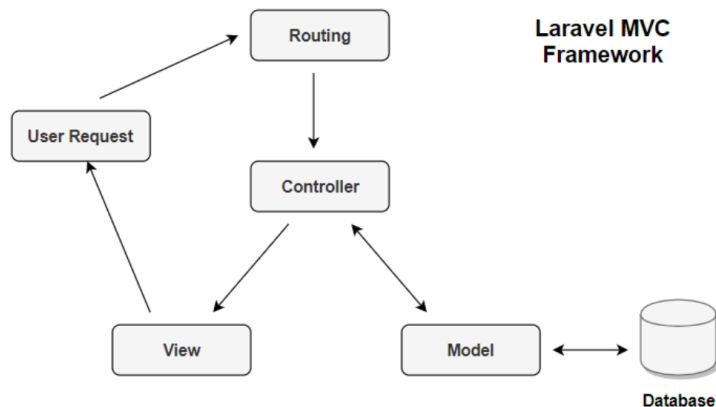
• Cài đặt

- Cài Xampp (phpMyAdmin)
- Cài composer: <https://getcomposer.org/download/>
- Cài Laravel: **composer global require "laravel/installer"**
- Cập nhật Laravel: **composer global update laravel/installer**
- Tạo project: **composer create-project laravel/laravel product_Laravel**
- Chạy Laravel service: **php artisan serve**
- Mặc định, Laravel service sử dụng port 8000, để kiểm tra, truy cập: localhost:8000

```
D:\>composer create-project laravel/laravel product_Laravel
Creating a "laravel/laravel" project at "../product_Laravel"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v9.1.2)
- Downloading laravel/laravel (v9.1.2)
- Installing laravel/laravel (v9.1.2): Extracting archive
Created project in D:\product_Laravel
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Info from https://repo.packagist.org: #StandWithUkraine
Updating dependencies
Lock file operations: 168 installs, 0 updates, 0 removals
```

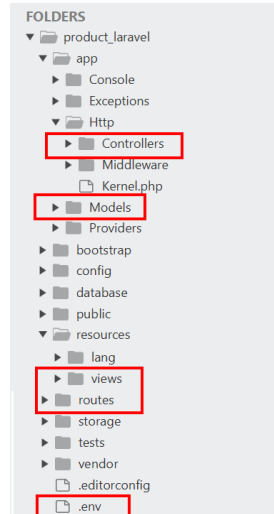
Giới thiệu Laravel framework

• Mô hình ứng dụng: MVC



Giới thiệu Laravel framework

- Mô hình ứng dụng: MVC
- Cấu trúc ứng dụng:
 - Controller: các tập tin xử lý
 - Models: các tập tin CSDL
 - views: các tập tin giao diện
 - routes: các tập tin định tuyến, mặc định là web.php
 - .env: tập tin cấu hình các biến môi trường



Route

- `Route::get($uri, $callback);`
- `Route::post($uri, $callback);`
- `Route::put($uri, $callback);`
- `Route::patch($uri, $callback);`
- `Route::delete($uri, $callback);`
- `Route::options($uri, $callback);`
- Sử dụng tham số:
 - `$uri`: 'view_name/{parameter}'
- Đặt tên:
 - `$callback`: ['as'=> 'route_name', 'uses'=> 'NameController@function_name']

Route

- Một số ví dụ:

```
Route::get('/', function () {
    return view('welcome');
});
```

```
Route::resource('index', 'ProductController');
```

```
Route::get('productdetail/{ProductId}', ['as'=>'proddetail', 'uses'=>'ProductController@getProductDetail']);
```

```
Route::post('insert', 'ProductController@insertProduct')->name('insert');
```

```
Route::get('/posts/{post}/comments/{comment}', function ($postId, $commentId) { // });
```

Route

- Chuyển hướng trang web với Route:

- {{route('route_name')}}}

- redirect()->route()

- redirect()->action('AnotherController@index');

```
<form method="POST" action="{{ route('insert') }}">
```

```
return redirect()->route('userlist', ['userId' => 5]);
```

```
return redirect()->action('HomeController@index');
```

- CSRF Protection: dùng trong kiểm chứng dữ liệu với các phương thức post, put,..

- @csrf

```
<form method="POST" action="/profile">
    @csrf
    ...
</form>
```

Controller

• Tạo Controller:

- **php artisan make:controller** NameController
- Name: tên controller phải viết hoa ký tự đầu, ví dụ: ProductController
- Các action trong Controller thường được định nghĩa trong Route

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\sanpham;
class HomeController extends Controller
{
    public function index()
    {
        $sanpham = sanpham::all();
        return view('product', compact('sanpham'));
    }
}
```

Trả về view
(product.blade.php)

Lấy dữ liệu từ Models
(Product.php)

Lập trình web

107

107

Model

• Tạo Model: **php artisan make:model** Classname

- Classname: thường là tên table trong CSDL, ví dụ: Product

```
<?php

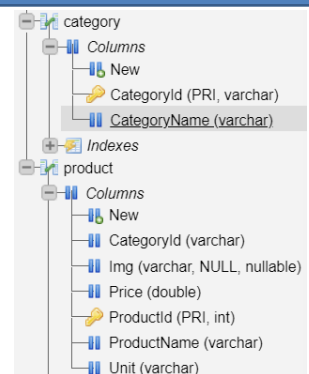
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Http\Controllers\ProductController;
class Product extends Model
{
    use HasFactory;
    protected $table = "Product";
    public $timestamps = false;
    protected $fillable = ['ProductId', 'ProductName', 'Unit', 'Price', 'CategoryId', 'Img'];
    public function Category()
    {
        return $this->belongsTo("App\Models\Category", "CategoryId", "CategoryId");
    }
}
```

Tên table trong CSDL

Không sinh 2 cột created_at và updated_at

Reference đến table Category trong quan hệ 1-n



Lập trình web

108

108

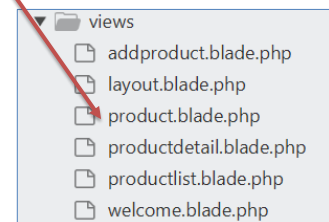
Model

- Table name: mặc định tên table là tên Model + "s", có thể thiết lập:
 - `protected $table = 'my_table';`
- Primary key:
 - Mặc định, trường id sẽ được dùng làm khóa chính, có thể thiết lập lại: `protected $primaryKey = 'my_id';`
 - Nếu khóa chính là kiểu số: mặc định là auto increment, có thể thiết lập lại: `public $incrementing = false;`
 - Nếu khóa chính khác kiểu số, có thể thiết lập lại: `protected $keyType = 'string';`
- Mặc định, Laravel bắt buộc thêm 2 trường `created_at` và `updated_at` vào table, có thể thiết lập lại:
 - `public $timestamps = false;`
- Muốn thay đổi chuỗi kết nối đã định nghĩa trong .ENV, có thể thiết lập lại:
 - `protected $connection = 'connection-name';`

View

- Tên các trang trong view có dạng: `viewname.blade.php`
- Các view thường được gọi trong các action trong Controller
- Blade là một templating engine cung cấp cú pháp định nghĩa giao diện trang web trong Laravel
(<https://laravel.com/docs/8.x/blade>)

```
public function index()
{
    $product = Product::paginate(20);
    return view('product', compact('product'));
}
```



View (tt)

- Nhúng mã lệnh PHP trong HTML: `{{code PHP}}`
- Hiển thị giá trị các biến PHP
 - `{{ $variable }}`: dạng plain text
 - `{{!! $variable !!}}`: dạng HTML
- Gọi hàm PHP: `{{ function_name() }}`, ví dụ `{{ time() }}`
- Cấu trúc lựa chọn
 - `@if ... @else ... @endif`
 - `@if ... @elseif ... @else ... @endif`
 - `@switch($i)`
 - `@case(1) ... @break`
 - `@case(2) ... @break`
 - `@default...`
 - `@endswitch`

111

View (tt)

- Cấu trúc lặp
 - `@for (...)`
 - `...`
 - `@endfor`
 - `@foreach (...)`
 - `...`
 - `@endforeach`
 - `@while ()`
 - `...`
 - `@endwhile`
- `@break, @continue`
- Tham khảo thêm trong: <https://laravel.com/docs/8.x/blade>

112

View (tt)

- Tạo layout trong blade view với kế thừa template

- Định nghĩa layout: sử dụng chỉ thị **@yield('name')**

- Các trang sử dụng layout:

- **@extends('layout.blade.php')**
- **@section('name')**
- **@endsection** hoặc **@stop**

Trang **layout.blade.php**

```
<html>
...
<body>
  <header> ... </header>
  <nav>... </nav>
  <div class="container">
    @yield('content')
  </div>
  <footer>Copyright <?php echo date('m/Y') ?></footer>
</body>
<html>
```

Trang **default.blade.php**

```
@extends('layout')
@section('content')
//Nội dung trang default.php
@endsection
```

Eloquent: Relationships

- Eloquent relationships trong Laravel: object-relational mapper (ORM)
- Cho phép định nghĩa các mối quan hệ giữa các bảng trong Model như:
 - One To One
 - One To Many
 - Many To Many
- Một table trong database sẽ có một Model tương ứng

One To One

- Ví dụ quan hệ giữa hai table user và phone được định nghĩa như sau: mỗi người dùng chỉ có duy nhất một số điện thoại và ngược lại

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class User extends Model {
    /** * Get the phone associated with the user. */
    public function phone() {
        return $this->hasOne(Phone::class);
        //return $this->hasOne(Phone::class, 'foreign_key');
        //return $this->hasOne('Phone::class, 'foreign_key',
        'owner_key');
    }
}
Muốn truy xuất Phone của có id là 1 :
$phone = User::find(1)->phone;
```

```
<?php namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Phone extends Model {
    /** * Get the user that owns the phone. */
    public function user() {
        return $this->belongsTo(User::class);
        //return $this->belongsTo(User::class, 'foreign_key');
        //return $this->belongsTo(User::class, 'foreign_key',
        'owner_key');
    }
}
```

One To Many

- Ví dụ quan hệ giữa hai table post và comment được định nghĩa như sau: mỗi bài viết (post) có nhiều comment

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Post extends Model {
    /** * Get the phone associated with the user. */
    public function comments() {
        return $this->hasMany(Comment::class);
        //return $this->hasMany(Comment::class, 'foreign_key');
        //return $this->hasMany(Comment::class, 'foreign_key',
        'owner_key');
    }
}
Truy xuất:
$comments = Post::find(1)->comments;
foreach ($comments as $comment) { ... }
```

```
<?php namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Comment extends Model {
    /** * Get the user that owns the phone. */
    public function post() {
        return $this->belongsTo(Post::class);
        //return $this->belongsTo(Post::class, 'foreign_key');
        //return $this->belongsTo(Post::class, 'foreign_key',
        'owner_key');
    }
}
```

Many To Many

- Ví dụ quan hệ giữa hai table users và roles được định nghĩa như sau: mỗi user có nhiều role và mỗi role có nhiều user
- Để biểu diễn được quan hệ này, cần sử dụng đến một bảng trung gian role_user chứa 2 cột là role_id và user_id
- Cấu trúc các bảng như sau:
 - **roles** (role_id (integer), name (string))
 - **users** (user_id (integer), name (string))
 - **role_user** (user_id (integer), role_id (integer))

Many To Many

- Ví dụ quan hệ giữa hai table users và roles được định nghĩa như sau: mỗi user có nhiều role và mỗi role có nhiều user

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class User extends Model {
    public function roles() {
        return $this->belongsToMany(Role::class);
        //return $this->belongsToMany(Role::class, 'role_user');
        // return $this->belongsToMany(Role::class, 'role_user',
        'user_id', 'role_id');
    }
}
```

```
<?php namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Role extends Model {
    public function users() {
        return $this->belongsToMany(User::class);
        //...
    }
}
```

```
Truy xuất:
$user = User::find(1);
foreach ($user->roles as $role) { ... }
$roles = User::find(1)->roles()->orderBy('name')->get();
```

Many To Many

- Lấy giá trị của bảng trung gian: sử dụng thuộc tính pivot

```
use App\Models\User;
$user = User::find(1);
foreach ($user->roles as $role) {
    echo $role->pivot->created_at;
}
```

- Mặc định, thuộc tính pivot chỉ chứa các trường khóa, created_at và updated_at, muốn truy xuất các trường khác, phải chỉ định khi khai báo quan hệ:

```
return $this->belongsToMany(Role::class)->withPivot('active', 'created_by');
```

Database Query Builder

- Để sử dụng Database Query Builder, khai báo: use Illuminate\Support\Facades\DB;
- Các truy vấn chọn
 - DB::connection('connection_name');
 - DB::table('name')->get(): lấy dữ liệu bảng
 - DB::table('name')->get('column1','column2'): lấy dữ liệu các cột trong bảng
 - DB::table('name')->distinct()->get(): lấy dữ liệu bảng bỏ qua các record trùng
 - DB::table('name')->select('column as column_alias')->get(): đổi tên cột kết quả
 - DB::table('name')->first(): lấy 1 hàng kết quả
 - DB::table('name')->skip(m)->take(n)->get();
 - DB::table('name')->offset(m) ->limit(n) ->get();
 - DB::table('name')->pluck('column');
 - DB::table('name')->lists('column');

Database Query Builder

• Truy vấn điều kiện

- DB::table('name')->where('column', 'value')->get(): truy vấn điều kiện =
- DB::table('tablename')->where('column', 'like', '%filter%')->get();
- DB::table('name')->whereBetween('column', array(1, 100))->get();
- DB::table('name')->whereIn('column', array(1, 2, 3))->get();
- DB::table('name')->whereNotIn('column', array(1, 2, 3))->get();
- DB::table('name')->whereNull('column')->get();
- DB::table('name')->whereNotNull('column')->get();
- DB::table('users')->whereDate('column', '2021-12-10')->get();
- DB::table('users')->whereMonth('column', '12')->get();
- DB::table('users')->whereDay('column', '10')->get();
- DB::table('users')->whereYear('column', '2021')->get();
- DB::table('users')->where('column1', '>', value1)->orWhere('column2', 'value2')->get();
- \$users = DB::table('users')->where(['column1', 'value1'], ['column2', '<>', 'value2'], ...)->get();

Database Query Builder

• Truy vấn gom nhóm:

- DB::table('name')->groupBy('column')->get();
- DB::table('name')->groupBy('column')->having('column2', value)->get();
- DB::table('name')->selectRaw('count(id) as alias, column1')->groupBy('column2')->havingBetween('alias', [5, 15])->get();
- DB::table('name')->orderBy('column')->get();
- DB::table('name')->orderBy('column', 'desc')->get();

• Truy vấn thống kê:

- DB::table('name')->count();
- DB::table('name')->max('column');
- DB::table('name')->min('column');
- DB::table('name')->avg('column');
- DB::table('name')->sum('column');

Database Query Builder

- Inner join
 - DB::table('name')->join('table', 'name.id', '=', 'table.id') ->select('name.id', 'table.email');
 - DB::table('users') ->join('contacts', 'users.id', '=', 'contacts.user_id') ->join('orders', 'users.id', '=', 'orders.user_id') ->select('users.*', 'contacts.phone', 'orders.price') ->get();
- Left join
 - DB::table('users') ->leftJoin('posts', 'users.id', '=', 'posts.user_id') ->get();
- Right join
 - DB::table('users') ->rightJoin('posts', 'users.id', '=', 'posts.user_id') ->get();

Database Query Builder

- Insert:
 - DB::table('users')->insert(['email' => 'kayla@example.com', 'votes' => 0]);
 - DB::table('users')->insert([['email' => 'picard@example.com', 'votes' => 0], ['email' => 'janeway@example.com', 'votes' => 0],]);
 - \$id = DB::table('users')->insertGetId(['email' => 'john@example.com', 'votes' => 0]);
- Update:
 - DB::table('users') ->where('id', 1) ->update(['votes' => 1]);
- Delete:
 - DB::table('users')->delete();
 - DB::table('users')->where('votes', '>', 100)->delete();