

Kiểm tra/nhắc lại bài học trước

CÔNG NGHỆ WEB AN TOÀN

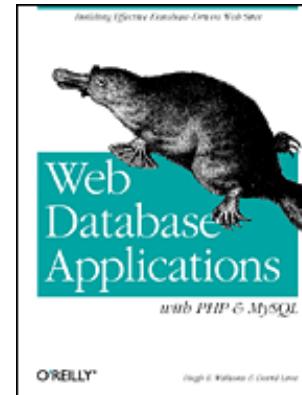
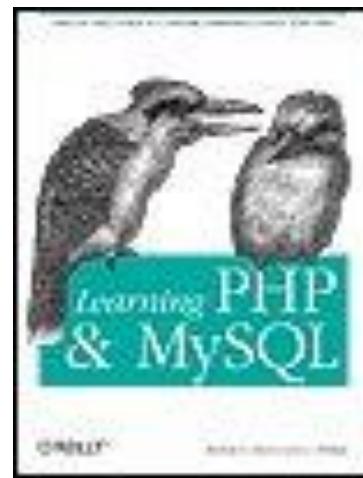
Bài 3-1. NGÔN NGỮ LẬP TRÌNH PHÍA SERVER (PHP)

Mục tiêu buổi học

1. Trình bày được cấu trúc của một trang web động
2. Trình bày được về các đặc điểm và cách thức cài đặt PHP
3. Tạo được một trang web động sử dụng HTML, CSS, JAVASCRIPT
và PHP

Tài liệu tham khảo

1. <https://www.w3schools.com/php/default.asp>
2. Michele Davis & Jon Phillips, “Learning PHP and MySQL”
3. Hugh E. Williams & David Lane, ” Web Database Applications with PHP & MySQL”
4. Lê Đình Thanh, Bài giảng “Phát triển ứng dụng Web”, Trường Đại học Công nghệ, ĐHQGHN.
5. Bộ bài tập học phần.



1

Kiến trúc của một ứng dụng web động

2

Ngôn ngữ lập trình web phía server

3

PHP

4

Mô hình MVC

1

Kiến trúc của một ứng dụng web động

2

Ngôn ngữ lập trình web phía server

3

PHP

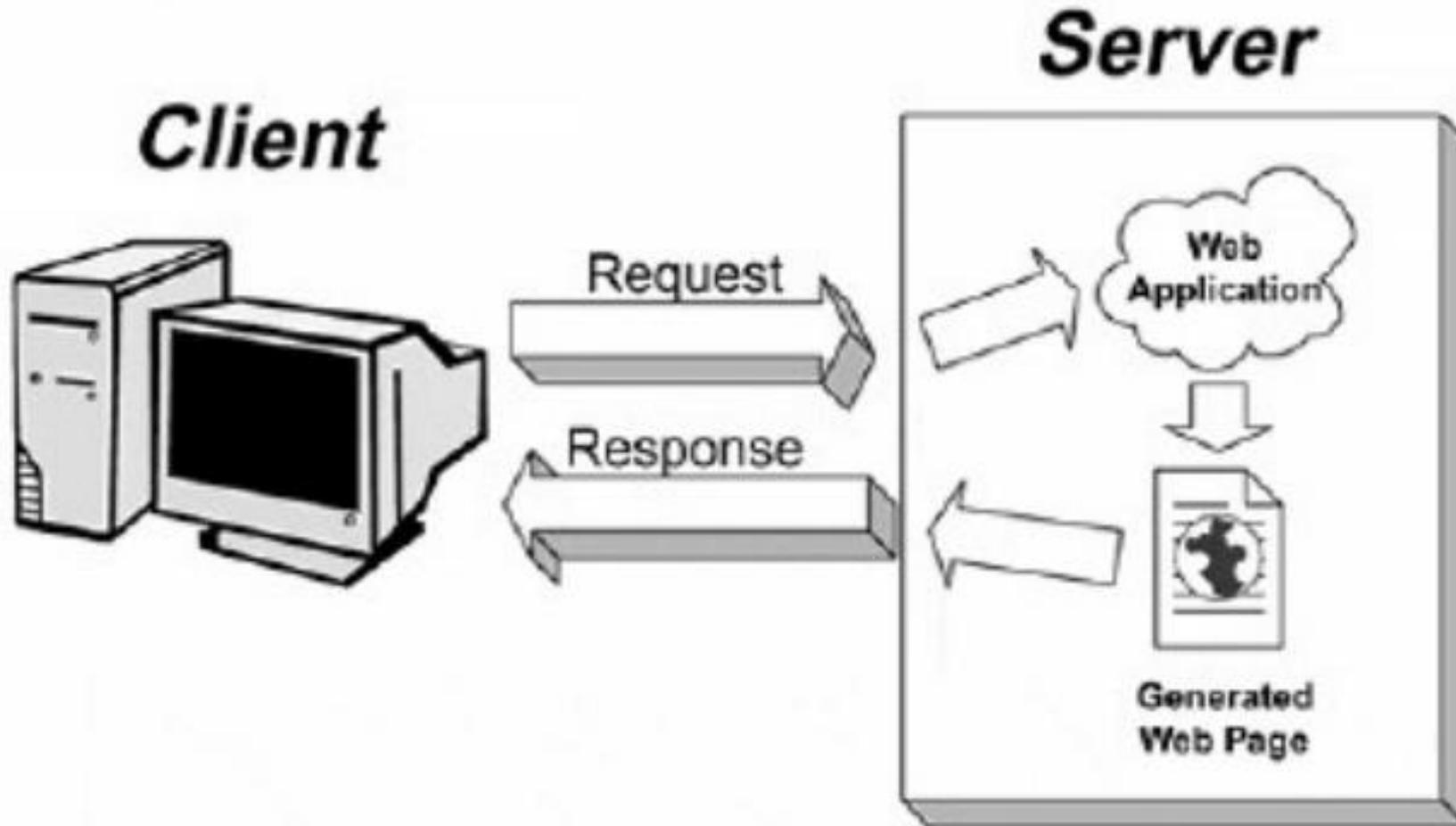
4

Mô hình MVC

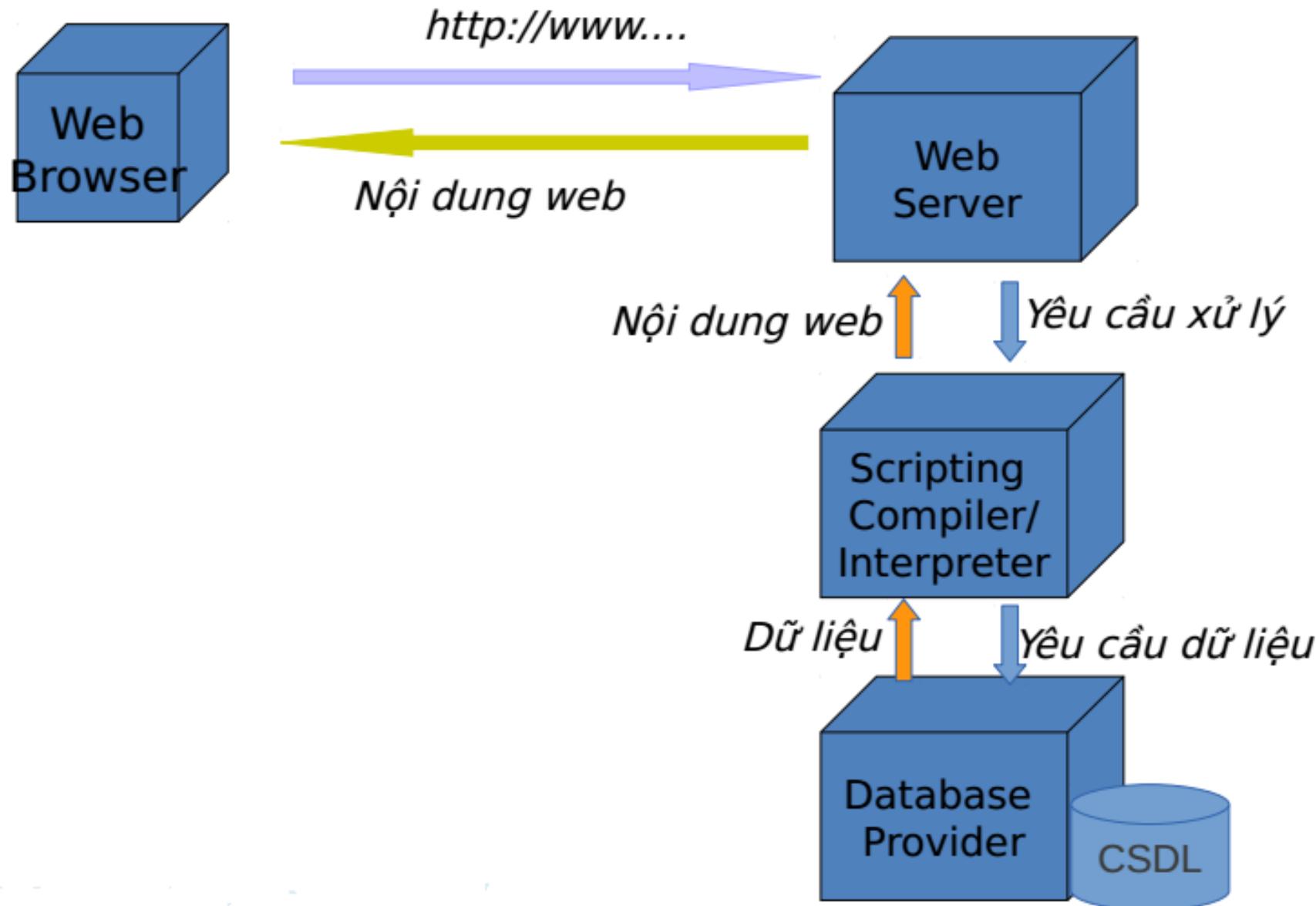
Web động

- ❑ Nội dung trang web (HTML + CSS + JavaScript) được Web Server sinh ra khi có yêu cầu từ Client.
- ❑ Rất phổ dụng: Hầu hết các trang web thương mại đều là web động.
- ❑ Sử dụng ngôn ngữ lập trình đa năng để sinh ra nội dung web.
- ❑ Sử dụng CSDL.

Kiến trúc web động

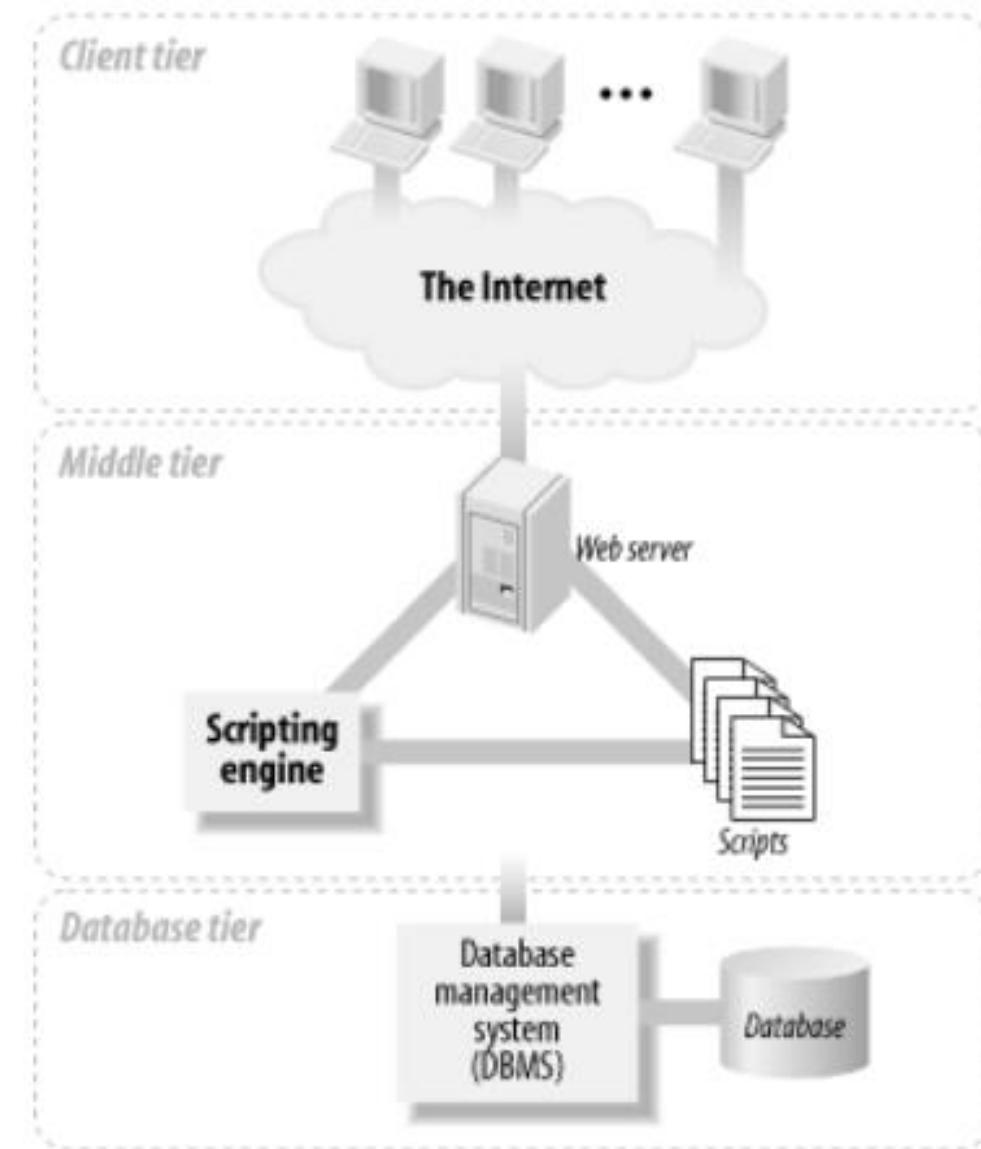


Web động với cơ sở dữ liệu



Mô hình ba tầng

- **Tầng khách**: trình diễn và tương tác với người dùng
- **Tầng giữa**: thực hiện các logic của ứng dụng
- **Tầng CSDL**: bao gồm hệ quản trị CSDL, CSDL của ứng dụng



1

Kiến trúc của một ứng dụng web động

2

Ngôn ngữ lập trình web phía server

3

PHP

4

Mô hình MVC

Ngôn ngữ lập trình web phía server

W Usage Statistics and... x +

w3techs.com/technologies/overview/programming_language/all ↗ C Google

provided by Q-Success

W3Techs

Web Technology Surveys

Home Technologies Reports Sites Quality Users Blog Forum FAQ Search

Technologies > Server-side Languages

Usage of server-side programming languages for websites

This diagram shows the percentages of websites using various server-side programming languages. See [technologies overview](#) for explanations on the methodologies used in the surveys. Our reports are updated daily.

How to read the diagram:
PHP is used by 82.0% of all the websites whose server-side programming language we know.

Request an extensive market report of specific server-side programming languages.
[Learn more](#)

PHP	82.0%
ASP.NET	17.2%
Java	2.8%
ColdFusion	0.7%
Perl	0.6%
Ruby	0.5%
Python	0.2%
JavaScript	0.1%
Erlang	0.1%

Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

The following server-side programming languages are used by less than 0.1% of the websites

- Lasso
- Scala

Technology Brief

Server-side Programming Languages

A programming language defines the computer instructions which are used to write programs that perform some task, e.g. composing a web page.

Featured Products and Services

WP Engine

Premium Managed WordPress Hosting

Try It for 60 Days

Secure Premium Wordpress Hosting
[WP Engine](#)

Present your product or service here

1

Kiến trúc của một ứng dụng web động

2

Ngôn ngữ lập trình web phía server

3

PHP

4

Mô hình MVC

-  1 **Giới thiệu chung về PHP**
-  2 **Cú pháp PHP**
-  3 **Biến trong PHP**
-  4 **Mảng trong PHP**
-  5 **Xâu trong PHP**
-  6 **Lớp và đối tượng trong PHP**

Tại sao sử dụng PHP – Hypertext Preprocessor

- ❑ chạy trên nhiều nền tảng khác nhau (Windows, Linux, Unix, Mac OS X, v.v.)
- ❑ Tương thích với hầu hết các máy chủ được sử dụng ngày nay (Apache, IIS, v.v.)
- ❑ Hỗ trợ nhiều loại cơ sở dữ liệu
- ❑ Miễn phí. Tải xuống từ nguồn PHP chính thức: www.php.net
- ❑ dễ học và chạy hiệu quả ở phía máy chủ

Tệp/Trang PHP

- ❑ Các tệp PHP có thể chứa văn bản, HTML, CSS, JavaScript và mã PHP
- ❑ Mã PHP được thực thi trên máy chủ và kết quả được trả về trình duyệt dưới dạng HTML thuận túy
- ❑ Tệp PHP có phần mở rộng là ".php"

Khi nào thì cần có mã HTML, CSS, javascript trong trang php?

- Những trang chỉ bao gồm mã xử lý nghiệp vụ thì không cần mã HTML, CSS, javascript.
- Những trang tạo giao diện
 - có thể chứa mã HTML, CSS, javascript
 - hoặc dùng hàm echo của php để sinh ra mã HTML, CSS, javascript.

-  1 **Giới thiệu chung về PHP**
-  2 **Cú pháp PHP**
-  3 **Biến trong PHP**
-  4 **Mảng trong PHP**
-  5 **Xâu trong PHP**
-  6 **Lớp và đối tượng trong PHP**

Cú pháp...

- Mã PHP được để trong cặp thẻ `<?php` và `?>` - được gọi là các **phân đoạn PHP**. Có thể nhúng các phân đoạn PHP vào bất kỳ vị trí nào trong trang. Bên ngoài các phân đoạn PHP có thể chứa mã HTML, CSS, javascript.
- Phần mã PHP được thực thi để sinh ra phần **động** của trang web.

Cú pháp

☐ Tựa C/C++, trừ các điểm sau:

- Định kiểu không rõ ràng: \$x = 5; \$y = 10.5;
- Tên biến bắt đầu bằng \$
(https://www.w3schools.com/php/php_variables.asp)
- Mảng là ánh xạ (https://www.w3schools.com/php/php_arrays.asp)
- Định nghĩa hàm bằng từ khóa function
(https://www.w3schools.com/php/php_functions.asp)

```
function functionName() {  
    code to be executed;  
}
```

□ Nhận yêu cầu từ client.

- Các tham số được lưu trong các mảng:
`$_REQUEST`, `$_GET`, `$_POST`

□ Tạo trả lời chứa nội dung web và gửi cho web client

- Hàm **header** thay đổi giá trị các trường tiêu đề gói HTTP Response
(https://www.w3schools.com/php/func_network_header.asp)
- Hàm **echo** ghi nội dung HTML, javascript, css vào thân gói HTTP Response

□ Lưu trạng thái phiên làm việc

- Đối tượng lưu các biến phiên: **`$_SESSION`**

Trang PHP

Mã HTML, CSS, javascript

<?php

//Mã php được thực thi bên server để sinh ra phần động của trang web

?>

Mã HTML, CSS, javascript

<?php

/*Mã php được thực thi bên server để sinh ra
phần động của trang web */

?>

Mã HTML, CSS, javascript

Ví dụ

```
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
    echo "My first PHP script!";  
?>  
  
</body>  
</html>
```

My first PHP script!

-  1 **Giới thiệu chung về PHP**
-  2 **Cú pháp PHP**
-  3 **Biến trong PHP**
-  4 **Mảng trong PHP**
-  5 **Xâu trong PHP**
-  6 **Lớp và đối tượng trong PHP**

Kiểu, biến và hằng

- ❑ Các kiểu nguyên thủy: boolean, float, integer và string
- ❑ Các kiểu phức hợp: array và object
- ❑ Tên biến được bắt đầu bằng \$
- ❑ Định kiểu không rõ ràng
- ❑ Định nghĩa hằng: define("ten_hang", gia_tri);

Phạm vi của biến...

- Phạm vi truy cập của một biến là ngũ cảnh nó được định nghĩa
 - Một biến cục bộ được định nghĩa trong một hàm chỉ có phạm vi trong hàm
 - Một biến được định nghĩa toàn cục (không trong hàm nào) có phạm vi trong tệp định nghĩa nó cùng các tệp được bao hàm

```
<?php  
    $a = 1;  
    include 'lib.inc';  
    $b = $a;  
?  
?>
```

```
<?php  
    $a = 1;  
    function test () {  
        echo $a;  
    }  
    test ();  
?  
?>
```

Phạm vi của biến...

```
<?php
$a = 1; /* global scope */
function test() {
    global $a;
    echo $a; /*reference to global scope variable*/
    echo " ";
    echo $GLOBALS['a']; /*reference to global scope variable*/
}
test();
?>
```

Biến tĩnh

- ☐ Biến tĩnh chỉ có phạm vi truy cập cục bộ trong hàm, nhưng giá trị của nó không bị mất khi thực thi của chương trình thoát khỏi hàm

```
function test() {  
    static $count = 0;  
    $count++;  
    echo $count;  
    if ($count < 10) {  
        test();  
    }  
    $count--;  
}
```

Biến biến

- ☐ Biến biến sử dụng giá trị của một biến khác làm tên

```
$a = "delta";  
$r = "array";  
$$a = 2.34; //tương đương $delta = 2.34  
$b = array('alpha','beta','delta','gama');  
${$b[2]} == 2.34;  
${$r}[1] == 'beta';  
$obj->$a //tương đương $obj->delta
```

- ☐ Tham khảo:

https://www.w3schools.com/php/php_variables.asp

Các biến dựng sẵn

`$GLOBALS` — Mảng các biến toàn cục

`$_SERVER` — Mảng các biến máy chủ

`$_GET` — Mảng các biến GET

`$_POST` — Mảng các biến POST

`$_FILES` — Mảng các tệp upload

`$_REQUEST` — Mảng các biến Request (cả GET và POST)

`$_SESSION` — Mảng các biến phiên

`$_ENV` — Mảng các biến môi trường

`$_COOKIE` — Mảng các biến Cookies

https://www.w3schools.com/php/php_superglobals.asp

-  1 **Giới thiệu chung về PHP**
-  2 **Cú pháp PHP**
-  3 **Biến trong PHP**
-  4 **Mảng trong PHP**
-  5 **Xâu trong PHP**
-  6 **Lớp và đối tượng trong PHP**

Mảng...

❑ Có ba loại mảng:

- Mảng được lập chỉ mục - Mảng có chỉ mục số
- Mảng liên kết - Mảng có các khóa được đặt tên
- Mảng nhiều chiều - Mảng chứa một hoặc nhiều mảng

❑ Mảng trong PHP là ánh xạ có thứ tự chứa các phần tử **<key, value>**

- key chỉ nhận giá trị kiểu nguyên, xâu
- value nhận giá trị kiểu bất kỳ
- Các phần tử có thể sử dụng key tăng tự động

```
$thu = array("CN"=>"Chủ nhật", 2=>"Thứ hai", "Ba", "Tư", "Năm",  
"Sáu", "Bảy");
```

```
$cars = array("Volvo", "BMW", "Toyota");
```

❑ Truy cập các phần tử mảng **array[key]**

```
echo $thu["CN"]; https://www.w3schools.com/php/php\_arrays.asp
```

Mảng...

- Sửa nếu đã tồn tại phần tử có khóa là key, thêm phần tử mới nếu ngược lại: **\$arr [key]=value;**
- Thêm phần tử mới với khóa tăng tự động: **\$arr []=value;**
- Xóa phần tử mảng: **unset (array [key]);**
- Duyệt các phần tử mảng

foreach(array as key=>value)

foreach(array as value)

Mảng

- ❑ Đếm số phần tử mảng: `count (array)` ;
- ❑ Sắp xếp các phần tử mảng: `sort (array)` ;
- ❑ Gán mảng: `array1 = array2;`
- ❑ ...
- ❑ Tham khảo: https://www.w3schools.com/php/php_arrays.asp

-  1 **Giới thiệu chung về PHP**
-  2 **Cú pháp PHP**
-  3 **Biến trong PHP**
-  4 **Mảng trong PHP**
-  5 **Xâu trong PHP**
-  6 **Lớp và đối tượng trong PHP**

Xâu...

- ❑ Xâu được đánh dấu bởi dấu nháy đơn hoặc nháy kép
- ❑ Xâu có thể chứa biến bên trong

```
$number = 45;  
$vehicle = "bus";  
$message = "This $vehicle holds $number people";
```

- ❑ Nối xâu bằng dấu chấm (.)

```
$message = "This ".$vehicle." holds ".$number."people";
```

- ❑ Các ký tự đặc biệt cần được để sau \,

- Ví dụ: \\, \', \", \\$, ...

Xâu

- ❑ `strlen($s)` – độ dài xâu \$s
- ❑ `strtolower($s)` - xâu viết thường của \$s
- ❑ `strtoupper($s)` - xâu viết hoa của \$s
- ❑ `ucfirst($s)` – xâu viết hoa ký tự đầu của \$s
- ❑ `ucwords($s)` – xâu viết hoa ký tự đầu các từ \$s
- ❑ `ltrim($s)` – xâu được bỏ dấu trắng đầu xâu của \$s
- ❑ `rtrim($s)` – xâu được bỏ dấu trắng cuối xâu của \$s
- ❑ `trim($s)` – xâu được bỏ dấu trắng đầu và cuối xâu của \$s
- ❑ Tạo dữ liệu định dạng:
 - `string sprintf (string format [, mixed args...])`
 - `printf (string format [, mixed args...])`
- ❑ Tham khảo: https://www.w3schools.com/php/php_string.asp

Xử lý ngoại lệ

❑ Ném ngoại lệ

```
throw new Exception("Mô tả ngoại lệ");
```

❑ Bắt ngoại lệ

```
try {
    //mã xử lý nghiệp vụ
} catch (Exception $e) {
    //nếu có ngoại lệ xảy ra ở khôi try thì mã xử lý ngoại lệ ở
    //khôi catch được thực hiện. Sử dụng $e->getMessage() để lấy mô tả
    //ngoại lệ
} [catch (OtherException $oe) {
    //Có thể nhiều khôi catch sau khôi try. Mỗi khôi catch bắt
    //một loại ngoại lệ
}]*
```

[finally {
 Mã được chạy bất kể ngoại lệ đã xảy ra hay không
}]

Ví dụ

```
function inverse($x) {  
    if (!$x) {  
        throw new Exception('Division by zero.');//throw  
    }  
    else return 1/$x;  
}  
  
try {  
    echo inverse(5) . "\n";  
    echo inverse(0) . "\n";  
} catch (Exception $e) {  
    echo 'Caught exception: ', $e->getMessage(), "\n";  
}
```

0.2
Caught exception: Division by
zero.

Tham khảo thêm

- ❑ Các kiểu dữ liệu:
https://www.w3schools.com/php/php_datatypes.asp
- ❑ Kiểu số: https://www.w3schools.com/php/php_numbers.asp
- ❑ Toán học: https://www.w3schools.com/php/php_math.asp
- ❑ Hằng số: https://www.w3schools.com/php/php_constants.asp
- ❑ Các toán tử: https://www.w3schools.com/php/php_operators.asp
- ❑ Rẽ nhánh: https://www.w3schools.com/php/php_if_else.asp
https://www.w3schools.com/php/php_switch.asp
- ❑ Vòng lặp: https://www.w3schools.com/php/php_looping.asp

-  1 **Giới thiệu chung về PHP**
-  2 **Cú pháp PHP**
-  3 **Biến trong PHP**
-  4 **Mảng trong PHP**
-  5 **Xâu trong PHP**
-  6 **Lớp và đối tượng trong PHP**

Lớp và đối tượng

```
class SimpleClass {  
    // định nghĩa hằng  
    const constant = 'constant  
value';  
    // định nghĩa biến/thuộc tính  
    public $var = 'a default value';  
    // định nghĩa hàm/phương thức  
    public function displayVar() {  
        echo $this->var;  
    }  
}  
  
$obj = new SimpleClass();  
$obj->displayVar();  
echo $obj->var;  
echo SimpleClass::constant;
```

- `$this` được dùng để chỉ đối tượng gọi
- Tính khả kiến của các thuộc tính và phương thức có thể là `private`, `protected`, `public`

Tham khảo:

[https://www.w3schools.com/php/
php_oop_classes_objects.asp](https://www.w3schools.com/php/php_oop_classes_objects.asp)

Thuộc tính/phương thức tĩnh

```
class A {  
    public static $foo = 'I am foo';  
    public $bar = 'I am bar';  
    public static function getFoo() {  
        echo self::$foo; }  
    public static function setFoo() {  
        self::$foo = 'I am a new foo'; }  
    public function getBar() { echo $this->bar; }  
}  
  
$ob = new A();  
A::getFoo(); // output: I am foo  
$ob->getFoo(); // output: I am foo  
A::getBar(); // output:fatal error:using $this not in object context  
$ob->getBar(); // output: I am bar
```

Kế thừa

- ❑ Một lớp chỉ có thể kế thừa từ một lớp khác
- ❑ Lớp con có thể ghi đè/che phương thức lớp cha
 - Để tuy cập phương thức được kế thừa bị che, sử dụng *parent::*
 - Để định nghĩa một phương thức không thể che, thêm từ khóa *final* vào định nghĩa phương thức

```
class ExtendClass extends SimpleClass { // Redefine the parent method
function displayVar() {
    echo "Extending class\n";
    parent::displayVar();
}
$extended = new ExtendClass();
$extended->displayVar();
```

Hàm tạo

```
void __construct ([$args [, $...]] )  
class BaseClass {  
    function __construct () {  
        print "In BaseClass constructor\n";  
    }  
}  
class SubClass extends BaseClass {  
    function __construct () {  
        parent::__construct ();  
        print "In SubClass constructor\n";  
    }  
}  
$obj = new BaseClass();  
$obj = new SubClass();
```

Hàm hủy

- void **destruct** (void)

```
class MyDestructableClass {  
    function __construct() {  
        print "In constructor\n";  
        $this->name = "MyDestructableClass";  
    }  
    function __destruct() {  
        print "Destroying " . $this->name . "\n";  
    }  
}  
$obj = new MyDestructableClass();
```

Toán tử chỉ phạm vi (::)

- :: được sử dụng để
 - truy cập hằng
 - truy cập thuộc tính, phương thức tĩnh
 - truy cập phương thức của lớp cha bị ghi đè

Lớp ảo, phương thức ảo

□ Phương thức ảo

- Được định nghĩa với từ khóa **abstract**
- Chỉ có chữ ký, không có thân

□ Lớp ảo

- Được định nghĩa với từ khóa **abstract**
- Không có thể hiện
- Lớp có phương thức ảo phải được định nghĩa là lớp ảo

Lớp ảo, phương thức ảo

```
abstract class AbstractClass {  
    // Force Extending class to define this method  
    abstract protected function getValue();  
    abstract protected function prefixValue($prefix);  
    // Common method  
    public function printOut() {  
        print $this->getValue() . "\n"; }  
}  
  
class ConcreteClass extends AbstractClass {  
    protected function getValue() {  
        return "ConcreteClass"; }  
    public function prefixValue($prefix) {  
        return "{$prefix}ConcreteClass"; }  
}
```

Giao diện

Giao diện xác định các phương thức mà lớp phải cài đặt

- Tất cả các phương thức phải public

Lớp cài đặt phải cài đặt tất cả các phương thức thuộc giao diện

```
interface iTemplate {  
    public function setVariable($name, $var);  
    public function getHtml($template);  
}  
// Implement the interface  
class Template implements iTemplate {  
    private $vars = array();  
    public function setVariable($name, $var) { $this->vars[$name] = $var; }  
    public function getHtml($template) {  
        foreach($this->vars as $name => $value) {  
            $template = str_replace('{' . $name . '}', $value, $template);  
        }  
        return $template;  
    }  
}
```

Giao diện

Giao diện có thể kế thừa từ **nhiều** giao diện khác

Một lớp có thể cài đặt nhiều giao diện

```
interface a {  
    public function foo();  
}  
  
interface b {  
    public function bar();  
}  
  
interface c extends a, b {  
    public function baz();  
}  
  
class D implements a, b {  
}
```

Trait

- cung cấp một cơ chế sử dụng lại mã khác kế thừa
- được định nghĩa tương tự lớp
- không có thể hiện
- Một lớp hoặc trait có thể sử dụng nhiều trait khác
- Hữu ích cho những NNLT đơn kế thừa như PHP

```
trait Hello {  
    public function sayHello() {  
        echo 'Hello ';  
    }  
}  
  
trait World {  
    public function sayWorld() {  
        echo 'World';  
    }  
}  
  
trait HelloWorld {  
    use Hello, World;  
}
```

```
class MyHelloWorld1 {  
    use HelloWorld;  
}  
$o1= new MyHelloWorld1();  
$o1->sayHello();  
$o1->sayWorld();
```

Hello World

```
class MyHelloWorld2 {  
    use Hello, World;  
    public function sayExclamationMark() {  
        echo '!';  
    }  
}  
$o2 = new MyHelloWorld2();  
$o2->sayHello();  
$o2->sayWorld();  
$o2->sayExclamationMark();
```

Hello World!

Che

- Với các hàm trùng tên

- Độ ưu tiên: Thành viên lớp hiện tại > thành viên trait > thành viên lớp cơ sở

- Lớp không thể định nghĩa lại các thuộc tính

```
trait HelloWorld {  
    public function sayHello() {  
        echo 'Hello World!';  
    }  
}  
  
class TheWorldIsNotEnough {  
    use HelloWorld;  
    public function sayHello() {  
        echo 'Hello Universe!';  
    }  
}  
  
$o = new TheWorldIsNotEnough();  
$o->sayHello();
```

```
class Base {  
    public function sayHello() {  
        echo 'Hello ';  
    }  
}  
  
trait SayWorld {  
    public function sayHello() {  
        parent::sayHello();  
        echo 'World!';  
    }  
}  
  
class MyHelloWorld extends Base {  
    use SayWorld;  
}  
  
$o = new MyHelloWorld();  
$o->sayHello();
```

Giải quyết xung đột

- Xung đột xảy ra khi sử dụng các trait khác nhau có phương thức giống nhau trong cùng một lớp hoặc một trait khác

- Giải quyết
 - Chỉ định sử dụng phương thức của trait nào
 - Đổi tên phương thức trong một trait

```
trait A {  
    public function smallTalk() {  
        echo 'a';  
    }  
    public function bigTalk() {  
        echo 'A';  
    }  
}  
  
trait B {  
    public function smallTalk() {  
        echo 'b';  
    }  
}
```

```
echo 'b';  
}  
public function bigTalk() {  
    echo 'B';  
}  
}  
class Talker {  
use A, B {  
    B::smallTalk insteadof A;  
    A::bigTalk insteadof B;  
    B::bigTalk as talk;  
}  
}  
}
```

Có thể thay đổi tính khả kiến với toán tử **as**
B::bigTalk **as protected**;
B::bigTalk **as protected** talk;

Sao chép đối tượng

- Các biến trong PHP chỉ lưu tham chiếu đến đối tượng
- Nếu cần sao chép đối tượng (thành một thể hiện khác), sử dụng hàm

```
$copy_of_object = clone $object;
```

- * Hàm clone sao chép tất cả các thuộc tính => các thuộc tính tham chiếu vẫn giữ nguyên giá trị tham chiếu
- * Sau khi hoàn thành hàm clone, nếu hàm `void __clone(void)` được định nghĩa, hàm `_clone()` của đối tượng mới được tạo được gọi cho phép những thay đổi cần thiết giá trị các thuộc tính

Ví dụ sao chép đối tượng

```
class Class1 {  
    private $var_ = 1;  
    public function printMe() {  
        echo " var_ = $this->var_";  
    }  
    public function changeValue($v_) {  
        $this->var_ = $v_;  
    }  
}  
  
$obj1 = new Class2();  
$obj2 = clone $obj1;  
$obj1->printMe();  
$obj2->printMe();  
$obj2->changeValues(5, 6);  
$obj1->printMe();  
$obj2->printMe();
```

```
class Class2 {  
    private $var = 2;  
    private $obj;  
    public function __construct() {  
        $this->obj = new Class1();  
    }  
    public function printMe() {  
        echo "<br>var = $this->var";  
        $this->obj->printMe();  
    }  
    public function changeValues($v, $v_) {  
        $this->var = $v;  
        $this->obj->changeValue($v_);  
    }  
}
```

```
var = 2 var_ = 1  
var = 2 var_ = 1  
var = 2 var_ = 6  
var = 5 var_ = 6
```

Ví dụ sao chép đối tượng

```
class Class1 {  
    private $var_ = 1;  
    public function printMe() {  
        echo " var_ = $this->var_";  
    }  
    public function changeValue($v_) {  
        $this->var_ = $v_;  
    }  
}
```

```
$obj1 = new Class2();  
$obj2 = clone $obj1;  
$obj1->printMe();  
$obj2->printMe();  
$obj2->changeValues(5, 6);  
$obj1->printMe();  
$obj2->printMe();
```

```
class Class2 {  
    private $var = 2;  
    private $obj;  
  
    public function __construct() {  
        $this->obj = new Class1();  
    }  
  
    public function __clone() {  
        $this->obj = new Class1();  
    }  
    public function printMe() {  
        echo "<br>var = $this->var";  
        $this->obj->printMe();  
    }  
  
    public function changeValues($v, $v_) {  
        $this->var = $v;  
        $this->obj->changeValue($v_);  
    }  
}
```

```
var = 2 var_ = 1  
var = 2 var_ = 1  
var = 2 var_ = 1  
var = 5 var_ = 6
```

Không gian tên

- ❑ Không gian tên (namespace) được sử dụng để nhóm các lớp, giao diện, hàm và hằng nhằm tránh đụng độ khi sử dụng lại mã do trùng tên
 - Ở các ngôn ngữ khác:
 - .NET: namespace
 - Java: package
- ❑ Định nghĩa: Sử dụng từ khóa namespace để định nghĩa không gian tên

```
<?php  
    namespace MyNameSpace {  
        const CONNECT_OK = 1;  
        interface Conn { /*...*/ }  
        class Connection { /* ... */ }  
        function connect() { /* ... */ }  
    }  
?>
```

Không gian tên lồng nhau

- Sử dụng cú pháp biểu diễn thư mục

```
<?php  
    namespace Parent\Child\GrandChild {  
        const CONNECT_OK = 1;  
        interface Conn { /*...*/ }  
        class Connection { /* ... */ }  
        function connect() { /* ... */ }  
    }  
?>
```

Không gian tên toàn cục

- ☐ Các lớp, giao diện, hàm, hằng không được định nghĩa trong một không gian tên nào được coi nằm trong không gian tên toàn cục (\)
- ☐ Có thể sử dụng namespace không có tên để biểu thị không gian tên toàn cục

```
namespace { /*Không gian tên toàn cục*/ }
```

- Các không gian tên khác được xem như nằm trong không gian tên toàn cục.

```
\  
\namespace1  
\namespace1\subnamespace1  
\namespace2  
\namespace2\subnamespace2  
\namespace2\subnamespace2\subsubnamespace2
```

Tên đầy đủ trong không gian tên

- Tên đầy đủ của lớp, giao diện, hàm, hằng bao gồm không gian tên phía trước
 - \namespace\ClassName
 - \namespace\InterfaceName
 - \namespace\functionName
 - \namespace\CONSTANT_NAME
- Phân giải tên:
 - Khi lớp, giao diện không được viết với tên đầy đủ: Chúng được hiểu là thuộc không gian tên hiện tại
 - Khi hàm, hằng không được viết với tên đầy đủ
 - Chúng được hiểu là thuộc không gian tên hiện tại
 - Hoặc thuộc không gian tên toàn cục nếu không tìm thấy trong không gian tên hiện tại

Ví dụ phân giải tên

```
namespace ns1 {  
    class A {  
        private $var = 1;  
        public function a() {  
            echo "<br>ns1\A->var: $this->var";  
        }  
    }  
}  
  
namespace ns2\ns3 {  
    class A {  
        private $var = 3;  
        public function a() {  
            echo "<br>ns3\A->var: $this->var";  
        }  
    }  
}  
  
namespace ns2 {  
    class A {  
        private $var = 2;  
        public function a() {  
            echo "<br>ns2\A->var: $this->var";  
        }  
    }  
}  
  
$obj1 = new \ns1\A();  
$obj2 = new A();  
$obj3 = new ns3\A();  
  
$obj1->a();  
$obj2->a();  
$obj3->a();  
}
```

ns1\A->var: 1
ns2\A->var: 2
ns3\A->var: 3

Nhập và đặt bí danh

□ Để không phải viết tên đầy đủ (dài), PHP cho phép nhập và đặt bí danh cho không gian tên, lớp, và giao diện

- `use ns\subns\Classname;`

//sau đó chỉ cần sử dụng Classname thay cho tên
đầy đủ

- `use ns\subns\Classname as Another;`

//sau đó sử dụng Another thay cho tên đầy
đủ

- `use ns\subns\NSname;`

//sau đó sử dụng NSname thay cho tên đầy
đủ

1

Kiến trúc của một ứng dụng web động

2

Ngôn ngữ lập trình web phía server

3

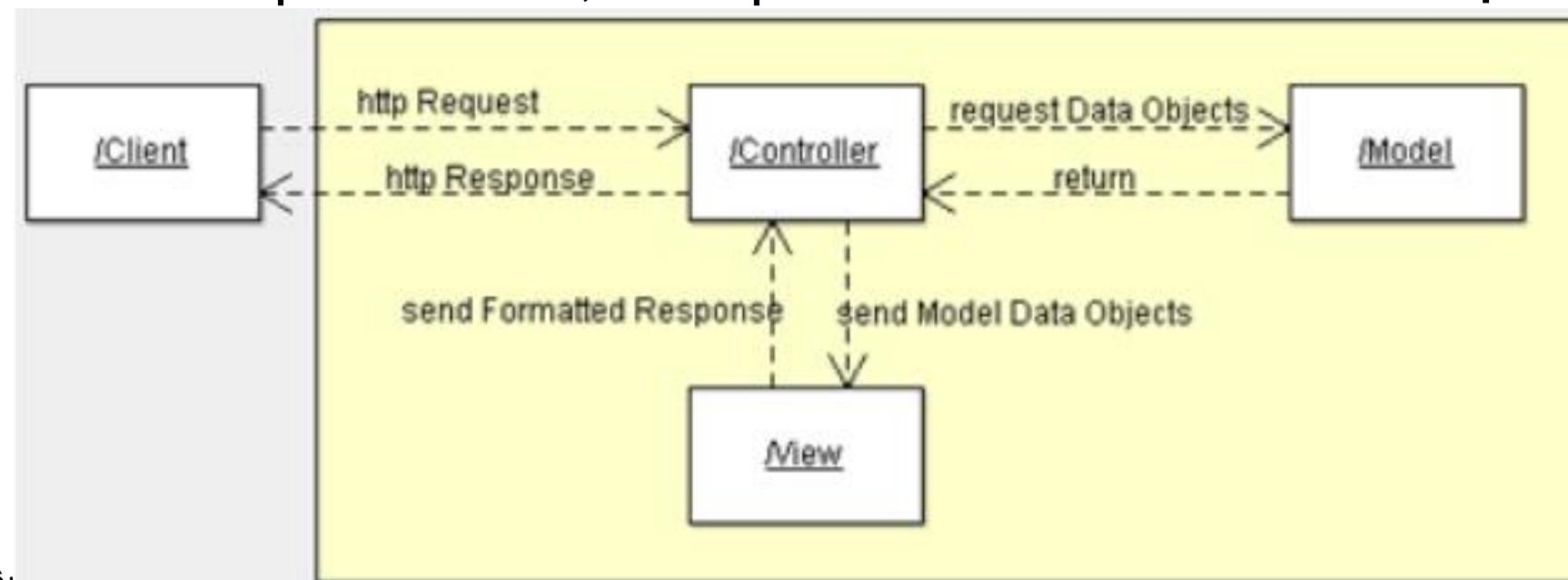
PHP

4

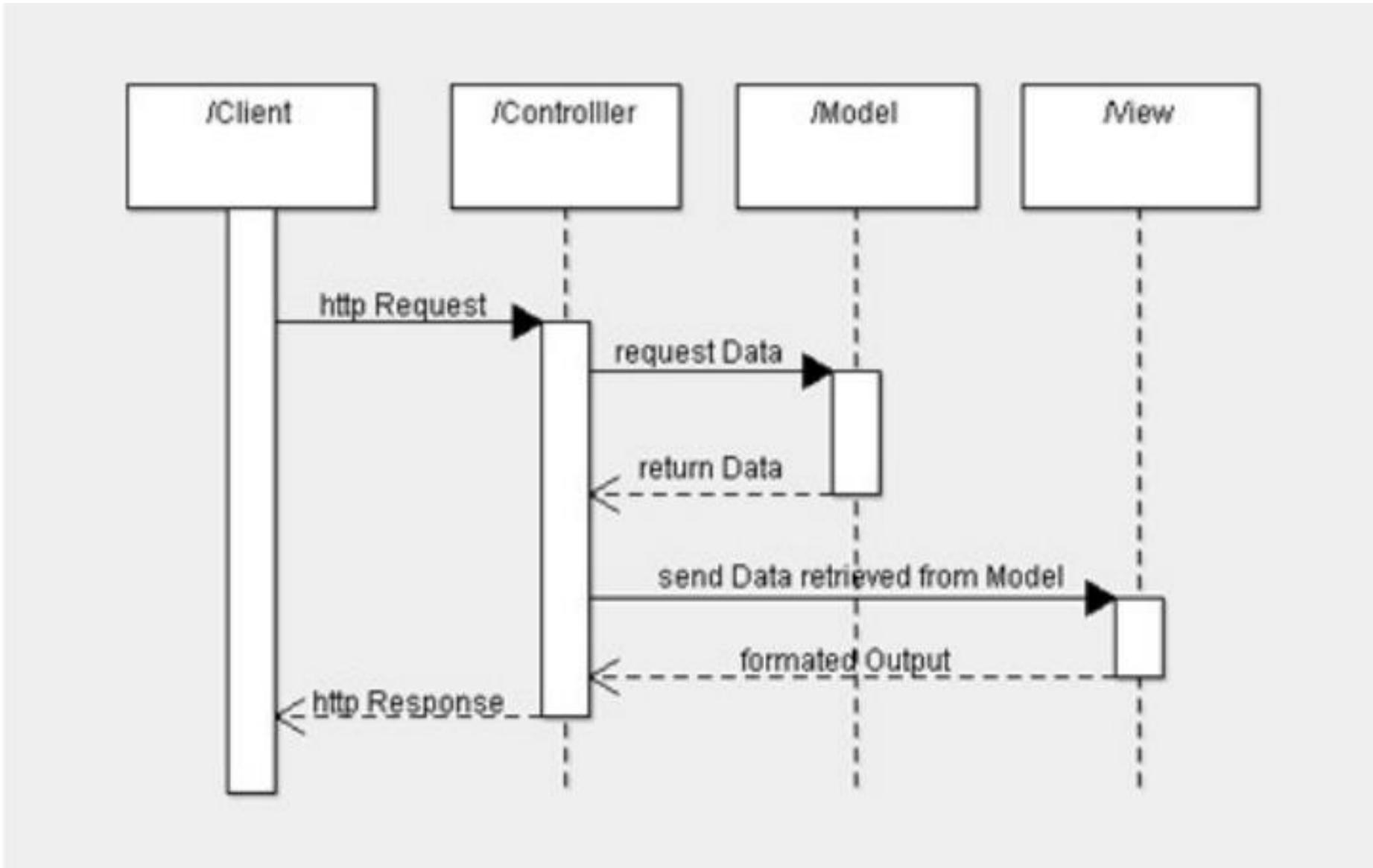
Mô hình MVC

Model – View - Control

- ☐ là mẫu thiết kế được sử dụng rộng rãi cho ứng dụng web
 - **Model:** Xử lý logic của ứng dụng, cung cấp dữ liệu, thường thao tác với CSDL
 - **View:** Hiển thị dữ liệu dưới định dạng cụ thể (với web là định dạng HTML)
 - **Control:** Giao tiếp với client, điều phối model và view làm việc



Model – View - Control



Ví dụ: Phương trình bậc 2 ...

□ Model: Phương trình $a*x*x + b*x + c = 0$

– Thành viên dữ liệu

- a, b, c: Các hệ số của phương trình
- retArray: Nghiệm của phương trình

– Thành viên hàm

- solve () : Giải phương trình
- getValues () : Trả về nghiệm của phương trình

Ví dụ: Phương trình bậc 2 ...

```
class QuadraticModel {  
    private $a; //he so bac hai  
    private $b; //he so bac nhat  
    private $c; //he so tu do  
    private $retArray; //ket qua:[so nghiem,nghiem 1,  
nghiem 2]  
  
    public function __construct($a, $b, $c) {  
        $this->a = $a;  
        $this->b = $b;  
        $this->c = $c;  
        $this->retArray = array();  
        $this->solve();  
    }  
    public function getValues() { return $this->retArray; }  
}
```

Ví dụ: Phương trình bậc 2 ...

```
private function solve() {  
    if ($this->a == 0) {  
        if ($this->b == 0) {  
            if ($this->c == 0) $this->retArray[] = -1;  
            else $this->retArray[] = 0;  
        } else {  
            $this->retArray[] = 1;  
            $this->retArray[] = -$this->c/$this->b;  
        }  
    } else {  
        $delta = $this->b*$this->b -  
4*$this->a*$this->c;  
        if ($delta > 0) {  
            $this->retArray[] = 2;  
            $this->retArray[] = (-$this->b+sqrt($delta))/2/$this->a;  
            $this->retArray[] = (-$this->b-sqrt($delta))/2/$this->a;  
        } else if ($delta == 0) {  
            $this->retArray[] = 1;  
            $this->retArray[] = -$this->b/2/$this->a;  
        } else {  
            $this->retArray[] = 0;  
        } } }
```

Ví dụ: Phương trình bậc 2 ...

□ View: Hiển thị kết quả

- Thành viên dữ liệu
 - arr: Nghiệm của phương trình
- Thành viên hàm
 - render(): Tạo nội dung web từ dữ liệu

Ví dụ: Phương trình bậc 2 ...

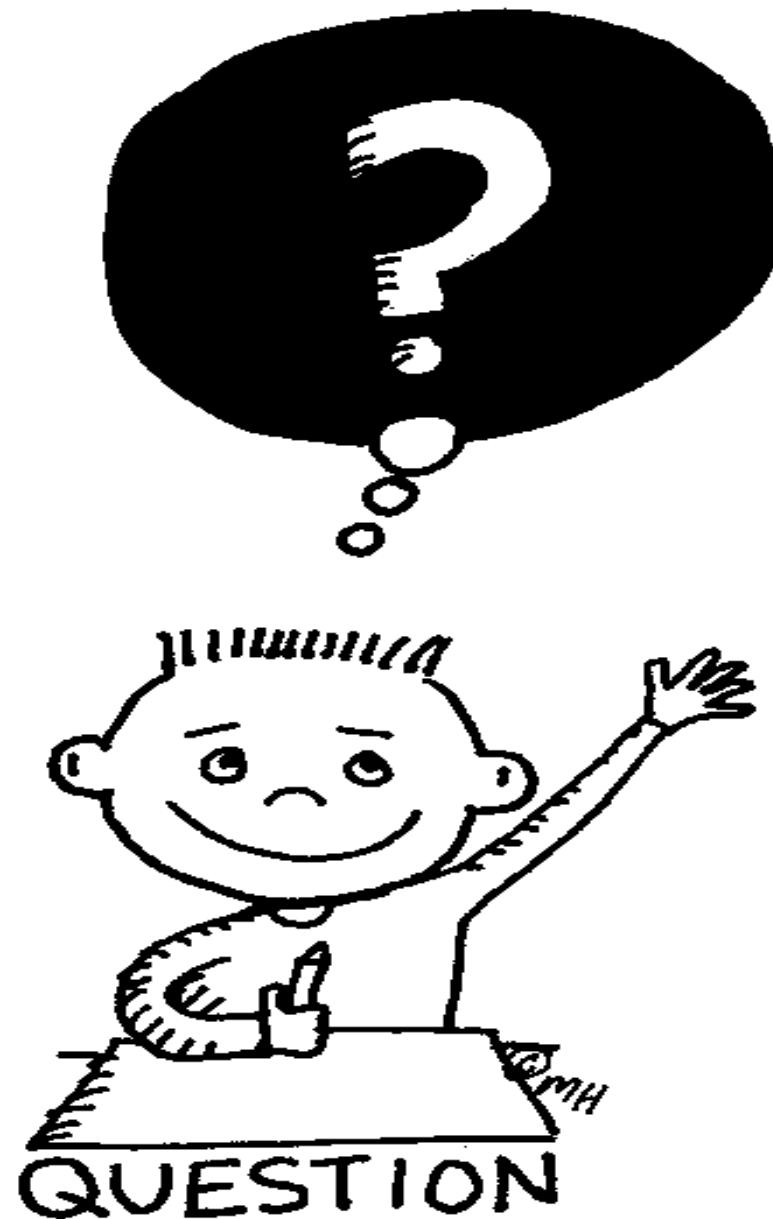
```
class QuadraticView {  
    private $arr; //la ket qua cua xu ly nghiep vu, can gui ve cho client  
    public function __construct($arr) {  
        $this->arr = $arr;  
    }  
    /**Noi dung web se gui ve cho client*/  
    public function render() {  
        $html = "";  
        if ($this->arr[0] == 0) $html.= "Phuong trinh vo nghiem";  
        else if ($this->arr[0] == 1) $html.= "Phuong trinh co MOT  
nghiem x = ".$this->arr[1];  
        else if ($this->arr[0] == 2) $html.= "Phuong trinh co HAI  
nghiem x1 = ".$this->arr[1].", x2 = ".$this->arr[2];  
        return $html;  
    }  
}
```

Ví dụ: Phương trình bậc 2 ...

- **Control:** Nhận các tham số của người dùng, điều phối model và view hoạt động, trả kết quả cho người dùng
 - Thành viên dữ liệu
 - ✓ Không
 - Thành viên hàm
 - ✓ `act()`: Thực hiện các công việc nêu trên

Ví dụ: Phương trình bậc 2 ...

```
class QuadraticControl {  
    public function act() {  
        if (isset($_GET["a"]) && isset($_GET["b"]) && isset($_GET["c"])) {  
            $a = EscapeShellCmd($_GET["a"]); //1. Nhan yeu cau, xu ly cac tham so  
            $b = EscapeShellCmd($_GET["b"]);  
            $c = EscapeShellCmd($_GET["c"]);  
            require_once("model.php"); //2. Gọi model để xử lý nghiệp vụ  
            $qmodel = new QuadraticModel($a, $b, $c);  
            $arr = $qmodel->getValues(); //ket qua xu ly nghiep vu  
            require_once("view.php"); //3. Gọi view để tạo nội dung web  
            $qview = new QuadraticView($arr);  
            $html = $qview->render();  
            echo $html; //4. Tra loi client  
        } else {  
            echo "Nhập a, b, c. Ví dụ ?a=3&b=-4&c=1";  
        } } }  
$qcontrol = new QuadraticControl();  
$qcontrol->act();
```



Tổng kết

1. Đã tìm hiểu về Web động
2. Đã tìm hiểu về PHP

Bài tập về nhà

1. Đọc thêm tài liệu về PHP
2. Làm bài tập số 3