Zend Framework! Programming
Shopping application

Kỹ thuật xử lý mảng

Mảng (Array) là một thành phần rất quan trọng bất kỳ ngôn ngữ lập trình nào. Thông thường khi lập trình web với PHP thuần chúng ta rất ít khi để ý đến kỹ thuật xử lý mảng điều đó đã làm hạn chế sự linh hoạt của ứng dụng.

Đối với các ứng dụng được xây dựng trên nền Zend Framework, nếu chúng ta không xử dụng tốt mảng thì thật sự sẽ khó khăn để chúng ta có thể tối ưu mã của chương trình và giúp cho ứng dụng của chúng ta chạy nhanh được.

Chính vì vậy trong bài này chúng ta sẽ học cách xử lý mảng cho mọi tình huống để nâng cao khả năng ứng dụng mảng vào Zend Framework.

Giáo trình: Zend Framework! Programming (v2.2)
Chuyên đề: Shopping application

Biên soạn: Phạm Vũ Khánh
Email: vukhanh2212@gmail.com
Điện thoại: 0908.893326
Website: www.zend.vn
Cập nhật: 07-2010

Mảng là gì?

Biến là một nơi để lưu trữ số hoặc chữ. Vấn đề là, biến chỉ có thể lưu trữ một giá trị duy nhất. Còn mảng là một biến đặc biệt, nó có thể lưu trữ nhiều giá trị trong một biến duy nhất.

Ví dụ: Chúng ta chỉ có thể lưu trữ một tên của nhân viên trong một biến. Nhưng đối với mảng chúng ta có thể lưu trữ hàng ngàn tên nhân viên khác nhau.

```
<?php
    $employee_1 = 'Nguyễn Văn A';

    $employee = array();
    $employee[] = 'Nguyễn Văn A';
    $employee[] = 'Nguyễn Văn B';
?>
```

Mảng có thể lưu trữ tất cả các giá trị biến của bạn dưới một tên duy nhất. Và bạn có thể truy cập giá trị bằng cách tham chiếu đến tên mảng. Mỗi phần tử mảng có chỉ số riêng (index) để chúng ta có thể truy cập chúng một cách dễ dàng.

Trong PHP có 3 loại mảng: Mảng số nguyên (Numeric array), Associative array, Multidimensional array.

1. Mảng số nguyên

Mảng số nguyên là mảng có chỉ số (index or key) là ở dạng số. Chúng ta thường gọi mảng này là mảng liên tục. Có 2 cách để tạo ra một mảng số nguyên.

Ví dụ 1:

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

Ví dụ 2:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

In một phần tử trong mảng:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";

echo $cars[3] . "<br>";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
```

In tất cả các phần tử trong mảng

```
$cars[0] = "Saab";
$cars[1] = "Volvo";
$cars[2] = "BMW";
$cars[3] = "Toyota";

for ($i = 0; $i <= count($cars); $i++) {
```

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

2. Mảng kết hợp

Mảng kết hợp là một mảng mà chỉ số (index or key) là một giá trị, chỉ số có thể là chuỗi hoặc số. Khi lưu trữ dữ liệu vào các phần tử mảng, các phần tử đó được đặt tên cụ thể. Mảng kết hợp là một sự bổng xung cần thiết cho thành phần mảng trong PHP vì có nhiều vấn đề mảng số nguyên không thể giải quyết được hết. Chúng ta thường gọi là mảng không liên tục.

Ví dụ 1:

```
$ages = array("Tuan"=>32, "Quang"=>30, "Long"=>34);
```

Ví dụ 2:

```
$ages["Tuan"] = 32;
$ages["Quang"] = 30;
$ages["Long"] = 34;
```

In một phần tử trong mảng

```
$ages["Tuan"] = 32;
$ages["Quang"] = 30;
$ages["Long"] = 34;

echo $ages["Tuan"];
```

In tất cả các phần tử trong mảng

```
$ages["Tuan"] = 32;
$ages["Quang"] = 30;
$ages["Long"] = 34;

foreach($ages as $key => $value){
    echo "<br>" . $key . " : " . $value;
}
```

3. Mảng đa chiều

Mảng đa chiều là mảng mà mỗi phần tử trong mảng chính có thể là một mảng và mỗi phần tử trong mảng con cũng có thể là một mảng. Chúng ta thường gọi là mảng lồng.

Ví dụ:

```
$students["SV001"] = array('id'=>1,
                        'name'=>'Tuấn',
                        'age'=> 18,
                        'points'=>array(10,5,8)
                         );
$students["SV002"] = array('id'=>1,
                        'name'=>'Tuấn',
                        'age'=> 18,
                        'points'=>array(10,5,8)
                         );
$students["SV003"] = array('id'=>1,
                        'name'=>'Tuấn',
                        'age'=> 18,
                        'points'=>array(10,5,8)
                         );
```

In phần tử trong mảng:

```
<?php

$students["SV001"] = array('id'=>1,
                        'name'=>'Tuấn',
                        'age'=> 18,
                        'points'=>array(10,5,8)
                         );
$students["SV002"] = array('id'=>1,
                        'name'=>'Tuấn',
                        'age'=> 18,
                        'points'=>array(10,5,8)
                         );
$students["SV003"] = array('id'=>1,
                        'name'=>'Tuấn',
                        'age'=> 18,
                        'points'=>array(10,5,8)
                         );

echo $students["SV003"]['name'] . '<br>';
echo $students["SV003"]['age'] . '<br>';

?>
```

4. Xem cấu trúc mảng

Để có thể thao tác tốt trên một mảng bất kỳ chúng ta phải đọc được cấu trúc mảng. Trong thực tế để đọc cấu trúc mảng, đối tượng, biến toàn cục của một hệ thống … các lập trình viên PHP thường dùng khối lệnh sau:

```
echo '<pre>';
print_r($ten_mang);
echo '</pre>';
```

Ví dụ: nếu chúng ta in ra mảng $students ở mục 3 chúng ta sẽ được như sau:

```
Array
(
    [SV001] => Array
        (
            [id] => 1
            [name] => Tuấn
            [age] => 18
            [points] => Array
                (
                    [0] => 10
                    [1] => 5
                    [2] => 8
                )

        )

    [SV002] => Array
        (
            [id] => 1
            [name] => Tuấn
            [age] => 18
            [points] => Array
                (
                    [0] => 10
                    [1] => 5
                    [2] => 8
                )

        )
```

```
    [SV003] => Array
        (
            [id] => 1
            [name] => Tuấn
            [age] => 18
            [points] => Array
                (
                    [0] => 10
                    [1] => 5
                    [2] => 8
                )

        )

)
```

5. Bài tập mảng

Để xử lý tốt mảng chúng ta cần phải thực hiện nhiều bài tập với mảng đa chiều điều này giúp chúng ta nâng cao khả năng xử lý mảng

Bài tập 1: Xây dựng hàm đưa dữ liệu trong bảng groups vào trong selectbox

```
<?php
$con = mysql_connect('localhost', 'root', '');
mysql_select_db('zf05', $con);
$sql = 'SELECT * FROM user_group';
$result = mysql_query($sql);
while ($row = mysql_fetch_assoc($result)) {

    $newArray[$row['id']] = $row['group_name'];
}

echo formSelect('group', null, $newArray);

function formSelect($name, $value = null, $option = null, $attri = null) {
    $xhtml = '<select id="' . $name . '"  name="' . $name . '">';
    foreach ($option as $key => $val) {
        $xhtml .= '<option value="' . $key . '">' . $val . '</option>';
    }

    $xhtml .= '</select>';

    return $xhtml;
}
?>
```

Bài tập 2: Nhập 2 mảng thành một mảng duy nhất theo cấu trúc được yêu cầu

```
<?php

$group[] = array('id' => 1, 'name' => 'Admin');
$group[] = array('id' => 2, 'name' => 'Manager');
$group[] = array('id' => 3, 'name' => 'Member');


$member[] = array('id' => 1, 'username' => 'Nguyen Van A', 'group_id' => 1);
$member[] = array('id' => 2, 'username' => 'Nguyen Van B', 'group_id' => 1);
$member[] = array('id' => 3, 'username' => 'Nguyen Van C', 'group_id' => 1);
$member[] = array('id' => 4, 'username' => 'Nguyen Van D', 'group_id' => 2);
$member[] = array('id' => 5, 'username' => 'Nguyen Van E', 'group_id' => 2);
$member[] = array('id' => 6, 'username' => 'Nguyen Van F', 'group_id' => 2);
$member[] = array('id' => 7, 'username' => 'Nguyen Van G', 'group_id' => 3);
$member[] = array('id' => 8, 'username' => 'Nguyen Van H', 'group_id' => 3);
$member[] = array('id' => 8, 'username' => 'Nguyen Van I', 'group_id' => 3);

$newArray = array();
```

```
foreach ($group as $key => $val) {

    $newArray[$val['name']] = array();
    foreach ($member as $key1 => $info) {
        if ($val['id'] == $info['group_id']) {
            $newArray[$val['name']][] = $info;
        }
    }
}

echo '<pre>';
print_r($newArray);
echo '</pre>';

?>
```

6. Các hàm xử lý mảng hữu ích

6.1 Hàm print_r()

print_r (array &$array): In mảng $array ra, chủ yếu dùng để debug

```
$ages["Tuan"] = 32;
$ages["Quang"] = 30;
$ages["Long"] = 34;

echo '<pre>';
print_r($ages);
echo '</pre>';
```

6.2 Hàm count()

int count (array &$array): Trả về gí trị kiểu số nguyên là số phần tử của mảng $array

```
$ages["Tuan"] = 32;
$ages["Quang"] = 30;
$ages["Long"] = 34;

count($ages);
```

6.3 Hàm array_values():

array array_values (array &$array): Trả về một mảng liên tục bao gồm các phần tử có giá trị là giá trị lấy từ các phần tử của mảng $array

Ví dụ:

```
<?php
$array = array("size" => "XL", "color" => "gold");
print_r(array_values($array));
?>
```

Hiển thị:

```
Array
(
    [0] => XL
    [1] => gold
)
```

6.4 Hàm array_keys():

array array_keys (array &$array): Trả về một mảng liên tục bao gồm các phần tử có giá trị là khóa (key) lấy từ các phần tử của mảng $array

Ví dụ:

```
$array = array(0 => 100, "color" => "red");
print_r(array_keys($array));

$array = array("blue", "red", "green", "blue", "blue");
print_r(array_keys($array, "blue"));

$array = array("color" => array("blue", "red", "green"),
               "size"  => array("small", "medium", "large"));
print_r(array_keys($array));
```

Hiển thị:

```
Array
(
    [0] => 0
    [1] => color
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
Array
(
    [0] => color
    [1] => size
)
```

6.5 Hàm array_pop():

mixed array_pop (array &$array): Loại bỏ phần tử cuối cùng của mảng $array. Hàm trả về phần tử cuối cùng đã được loại bỏ.

Ví dụ:

```
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_pop($stack);
print_r($stack);
?>
```

Hiển thị:

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
)
```

6.6 Hàm array_push():

int array_push (array &$array , mixed $var [, mixed $...]): Đưa thêm 1 hoặc nhiều phần tử vào cuối mảng $array. Hàm trả về kiểu số nguyên là số lượng phần tử của mảng $array mới

Ví dụ:

```
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

Hiển thị:

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

6.7 Hàm array_shif():

mixed array_shift (array &$array): Loại bỏ phần tử đầu tiên của mảng $array. Hàm trả về phần tử đầu tiên đã được loại bỏ.

Ví dụ:

```
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_shift($stack);
print_r($stack);
?>
```

Hiển thị:

```
Array
(
    [0] => banana
    [1] => apple
    [2] => raspberry
)
```

6.8 Hàm array_unshif():

int array_unshift (array &$array , mixed $var [, mixed $...]): Đưa thêm 1 hoặc nhiều phần tử vào vị trí đầu mảng. Hàm trả về kiểu số nguyên là số lượng phần tử của mảng $array mới

Ví dụ:

```
<?php
$queue = array("orange", "banana");
array_unshift($queue, "apple", "raspberry");
print_r($queue);
?>
```

Hiển thị:

```
Array
```

```
(
    [0] => apple
    [1] => raspberry
    [2] => orange
    [3] => banana
)
```

6.9 Hàm list():

array list (mixed $varname [, mixed $...]) = $arrValue: Đây là một cấu trúc ngôn ngữ được dùng để gán giá trị cho một danh sách các biến. Giá trị được lấy tuần tự từ tập hợp các phần tử tuần tự của mảng được gán $arrValue (tức là không lấy các phần tử có khóa (key))

Ví dụ:

```
<?php

$info = array('coffee', 'brown', 'caffeine');

// Listing all the variables
list($drink, $color, $power) = $info;
echo '<br>' .  "$drink is $color and $power makes it special.\n";

// Listing some of them
list($drink, , $power) = $info;
echo '<br>' .  "$drink has $power.\n";

// Or let's skip to only the third one
list( , , $power) = $info;
echo '<br>' .  "I need $power!\n";

// list() doesn't work with strings
list($bar) = "abcde";
var_dump($bar); // NULL
?>
```

Hiển thị

```
coffee is brown and caffeine makes it special.
coffee has caffeine.
I need caffeine!
NULL
```

Ví dụ:

```
$info = array('coffee', 'brown', 'caffeine');

list($a[0], $a[1], $a[2]) = $info;

echo '<pre>';
print_r($a);
echo '</pre>';
```

Hiển thị

```
Array
(
    [2] => caffeine
    [1] => brown
    [0] => coffee
)
```

6.10 Hàm array_flip():

array array_flip (array $array): Trả về một mảng có khóa và giá trị được hoán đổi cho nhau so với mảng $array (giá trị thành khóa và khóa thành giá trị)

Ví dụ:

```
$trans = array("a" => 1, "b" => 1, "c" => 2);
$trans = array_flip($trans);
echo '<pre>';
print_r($trans);
echo '</pre>';
```

Hiển thị:

```
Array
(
    [1] => b
    [2] => c
)
```

6.11 Hàm sort()

bool sort (array &$array): Sắp xép mảng $array theo giá trị tăng dần

Ví dụ:

```
$fruits = array("lemon", "orange", "banana", "apple");
sort($fruits);
echo '<pre>';
print_r($fruits);
echo '</pre>';
```

Hiển thị

```
Array
(
    [0] => apple
    [1] => banana
    [2] => lemon
    [3] => orange
)
```

6.12 Hàm array_reverse()

array array_reverse (array $array): Đảo ngược vị trí các phần tử của mảng, phần tử cuối cùng trở thành phần tử đầu tiên, phần tử kề cuối thành phần tử thứ nhì, ...

Ví dụ:

```
$fruits = array("lemon", "orange", "banana", "apple");
$result = array_reverse($fruits);
echo '<pre>';
print_r($result);
echo '</pre>';
```

Hiển thị

```
Array
(
    [0] => apple
    [1] => banana
    [2] => orange
```

```
    [3] => lemon
)
```

6.13 Hàm array_merge()

array array_merge (array $array1 [, array $array2 [, array $...]]): Nhập 2 hay nhiều mảng thành 1 mảng duy nhất và trả về mảng mới.

Ví dụ:

```
$fruits_1 = array("lemon", "orange");
$fruits_2 = array("banana", "apple");
$result = array_merge($fruits_1,$fruits_2);
echo '<pre>';
print_r($result);
echo '</pre>';
```

Hiển thị:

```
Array
(
    [0] => lemon
    [1] => orange
    [2] => banana
    [3] => apple
)
```

6.14 Hàm array_rand()

mixed array_rand (array $input [, int $num_req = 1]): Lấy ngẫu nhiên ra 1 hoặc hoặc nhiều phần tử mảng sau đó đưa vào một mảng mới.

Ví dụ:

```
$fruits = array("lemon", "orange", "banana", "apple");
$rand_keys = array_rand($fruits, 2);
echo '<pre>';
print_r($rand_keys);
echo '</pre>';
```

Hiển thị:

```
Array
(
    [0] => 3
    [1] => 2
)
```

6.15 array_search()

mixed array_search (mixed $needle , array $array): Tìm giá trị trong mảng $array. Hàm trả về khóa (key) của phần tử tìm được.

Ví dụ:

```
$array = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');

$key = array_search('green', $array); // $key = 2;
$key = array_search('red', $array);   // $key = 1;
```

6.16 array_slice()

array array_slice (array $array , int $offset [, int $length]): Trích lấy một đoạn phần tử của mảng $array từ vị trí bắt đầu và lấy số phần tử theo yêu cầu phần tử (vị trí đầu tiên trong mảng là 0). Trả về mảng mới.

Ví dụ

```
$input = array("a", "b", "c", "d", "e");

$output = array_slice($input, 2);      // returns "c", "d", and "e"
$output = array_slice($input, -2, 1);  // returns "d"
$output = array_slice($input, 0, 3);   // returns "a", "b", and "c"

// note the differences in the array keys
print_r(array_slice($input, 2, -1));
print_r(array_slice($input, 2, -1, true));
```

6.17 array_unique()

array array_unique (array $array): Gom những phần tử trùng nhau trong mảng $array thành 1 phần tử rồi trả về mảng mới (mảng mới sẽ không có phần tử trùng nhau về giá trị)

Ví dụ:

```
$input = array("a" => "green", "red", "b" => "green", "blue", "red");
$result = array_unique($input);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Hiển thị:

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

6.18 Hàm implode()

string implode (string $str , array $array): Chuyển các giá trị của mảng $array thành một chuỗi bao gồm các phần tử cách nhau bằng $str

Ví dụ:

```
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);

echo $comma_separated; // lastname,email,phone
```

6.19 Hàm explode()

array explode (string $delimiter , string $string [, int $limit]): Chuyển một chuỗi thành một mảng. Tách chuỗi dựa vào $delimiter, mỗi đoạn tách ra sẽ thành 1 phần tử của mảng mới

Ví dụ:

```
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo '<pre>';
print_r($pieces);
echo '</pre>';
```

Hiển thị:

```
Array
(
    [0] => piece1
    [1] => piece2
    [2] => piece3
    [3] => piece4
    [4] => piece5
    [5] => piece6
)
```

6.20 Hàm serialize()

string serialize(mixed $value): Chuyển một chuỗi/mảng/đối trương $value thành một chuỗi đặc biệt. Rất có ích để lưu vào database

Ví dụ:

```
$input = array("a" => "green", "red", "b" => "green", "blue", "red");
$result = serialize($input);

echo '<pre>';
print_r($result);
echo '</pre>';
```

6.21 Hàm unserialize()

mixed unserialize (string $str): Chuyển chuỗi đặt biệt tạo bởi serialize về trạng thái ban đầu

6.22 Hàm array_key_exists()

bool array_key_exists (mixed $key , array $array): Kiểm tra khóa $key có tồn tại trong mảng $array hay không? Nếu có trả về gia 1 trị true.

Ví dụ:

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

if (array_key_exists('b', $input)) {
    echo "Tìm thấy phần tử";
}
```

6.23 Hàm in_array()

bool in_array (mixed $value , array $array): Kiểm tra giá trị $value có tồn tại trong mảng $array hay không? Nếu tồn tại trả về giá trị true.

Ví dụ:

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

if (in_array('green', $input)) {
    echo "Tìm thấy phần tử";
}
```

6.24 Hàm array_diff()

array array_diff (array $array1 , array $array2): Trả về một mảng bao gồm các phần tử khác nhau về giá trị giữa 2 mảng $array1 và $array2.

Ví dụ:

```
$array1 = array("a" => "green", "red", "blue", "red");
$array2 = array("b" => "green", "yellow", "red");
$result = array_diff($array1, $array2);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Hiển thị:

```
Array
(
    [1] => blue
)
```

6.25 Hàm array_diff_assoc

array array_diff_assoc (array $array1 , array $array2): Trả về một mảng bao gồm các phần tử khác nhau về khóa và giá trị giữa 2 mảng $array1 và $array2.

Ví dụ:

```
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "green", "yellow", "red");
$result = array_diff_assoc($array1, $array2);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Hiển thị

```
Array
(
    [b] => brown
    [c] => blue
    [0] => red
)
```

6.26 Hàm array_intersect():

array array_intersect (array $array1 , array $array2): Trả về một mảng bao gồm các phần tử giống nhau về giá trị giữa 2 mảng $array1 và $array2.

Ví dụ:

```
$array1 = array("a" => "green", "red", "blue");
$array2 = array("b" => "green", "yellow", "red");
$result = array_intersect($array1, $array2);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Hiển thị:

```
Array
(
    [a] => green
    [0] => red
)
```

6.27 Hàm array_intersect_assoc()

array array_intersect_assoc (array $array1 , array $array2): Trả về một mảng bao gồm các phần tử giống nhau về khóa và giá trị giữa 2 mảng $array1 và $array2.

Ví dụ:

```
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "green", "yellow", "red");
$result_array = array_intersect_assoc($array1, $array2);

echo '<pre>';
print_r($result_array);
echo '</pre>';
```

Hiển thị

```
Array
(
    [a] => green
)
```

6.28 Hàm array_combine()

array array_combine (array $keys , array $values): Tạo một mảng mới có khóa được lấy từ mảng $keys và giá trị được lấy từ mảng $vaules theo tuần tự

Ví dụ:

```
$a = array('green', 'red', 'yellow');
$b = array('avocado', 'apple', 'banana');
$c = array_combine($a, $b);

echo '<pre>';
print_r($c);
echo '</pre>';
```

Hiển thị:

```
Array
(
```

```
    [green] => avocado
    [red] => apple
    [yellow] => banana
)
```

7. Bài tập: Trắc nghiệm trực tuyến

Câu 1: Tạo mảng Questions (cau-1.php)

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php
 $data = file('question.txt') or die("Cannot read file");

 $arrQuestion = array();
 foreach ($data as $key => $info){

        if($key != 0){

                $tmp = explode("|",$info);

                $arrTemp = array('question'=>$tmp[1]);
                $arrQuestion[$tmp[0]] = $arrTemp;

        }
 }
?>
```

Câu 2: Tạo mảng Options (cau-2.php)

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php
$data = file('options.txt') or die("Cannot read file");

$arrOption = array();
foreach ($data as $key => $info){
        if($key != 0 ){
                $tmp = explode("|",$info);
                $arrOption[$tmp[0]][$tmp[1]]['option'] = $tmp[2];
                $arrOption[$tmp[0]][$tmp[1]]['point'] = $tmp[3];
        }
}
```

Câu 3: Tạo mảng tổng hợp của 2 mảng Questions & Options (cau-3.php)

```
<?php
require_once 'cau-1.php';
require_once 'cau-2.php';
$newArray = array();
foreach($arrQuestion as $key => $info){
        $newArray[$key]["question"] = $info['question'];
        $newArray[$key]["sentences"] = $arrOption[$key];

}
```

Câu 4: Tạo các nhóm radiobox (quiz.php)

```
<?php
require_once 'cau-1.php';
require_once 'cau-2.php';
$newArray = array();
foreach($arrQuestion as $key => $info){
        $newArray[$key]["question"] = $info['question'];
        $newArray[$key]["sentences"] = $arrOption[$key];

}
?>
<form action="result.php" method="post" enctype="application/x-www-form-urlencoded">
<?php
```

```
foreach ($newArray as $key1 => $info1){
        echo "<br><b>" . $info1['question'] . "</b>";
        foreach ($info1['sentences'] as $key2 => $info2){
                echo   '<br><input   type="radio"   name="'   .   $key1   .   '"   value="'
.trim($info2['point']) . '" />' . $info2['option'];

        }
        echo "<br>-----------------------------";
}
?>
<br>
<input type="submit" value="Ket Qua"></input>
</form>
```

Câu 5: Hiển thị kết quả sau khi trắc nghiệm (result.php)

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php

$data  = file('question.txt') or die("Cannot read file");

$point = 0;

foreach($data as $key => $info){
        $tmp = explode('|',$info);
        $point =  $point + $_POST[$tmp[0]];
}

echo "<br><b> Tổng số điểm của bạn là:</b> " . $point;

$data  = file('result.txt') or die("Cannot read file");
foreach($data as $key => $info){
        $tmp = explode('|',$info);
        if($point >= $tmp[0] && $point <= $tmp[1]){
                $result = $tmp[2];
                break;
        }

}
echo "<br><b>Kết quả trắc nghiệm của bạn:</b> " . $result;
```

Zend Framework! Programming
Shopping application

Lập trình hướng đối tượng trong PHP 5.x

Lập trình hướng đối tượng là một kiểu lập trình thông dụng trong lập trình phần mềm. Nó giúp cho mã nguồn của chương trình sáng sủa, dễ dàng nâng cấp và bảo trì. Lập trình hướng đối tượng là nền tảng của các Framework. Để có thể nắm vững Zend Framework chúng ta cần nắm vững về cách phương thức cũnng như kiểu lập trình hướng đối tượng trong PHP 5.x

Giáo trình: Zend Framework! Programming (v2.2)
Chuyên đề: Shopping application

Biên soạn: Phạm Vũ Khánh
Email: vukhanh2212@gmail.com
Điện thoại: 0908.893326
Website: www.zend.vn
Cập nhật: 07-2010

A. Kiểu lập trình

Trong lập trình có 2 kiểu lập trình chúng ta thường sử dụng đó là: Lập trình hướng thủ tục và lập trình hướng đối tượng.

Lập trình hướng thủ tục: là cách lập trình để giải quyết vấn đề nào đó theo yêu cầu đưa ra và nó đi theo hướng giải quyết từng bước một đến khi đạt được kết quả. Kiểu lập trình hướng thủ tục còn được gọi là kiểu lập trình từ trên xuống hoặc lập trình theo hàm (function). Khi sử dụng kiểu lập trình này chúng ta không xây dựng sẵn các hàm xử lý mà chỉ tạo ra hàm khi gặp một vấn đề nào đó.

Lập trình hướng đối tượng: là kiểu lập trình dựa trên một nền tảng các lớp đã được xây dựng sẵn. Nghĩa là chúng ta phải xác định trước những gì sẽ phải làm, những trường hợp sẽ xảy ra để xây dựng lớp có những chức năng cần thiết cho quá trình xây dựng ứng dụng.

1. Lập trình hướng thủ tục:

Khi mới bắt đầu làm quen với lập trình thì cách lập trình hướng thủ tục là cách chúng ta thường dùng. Đây là một cách lập trình mà chúng ta không thể bỏ qua. Có thể lập trình hướng thủ tục không có nhưng ưu điểm như lập trình hướng đối tượng nhưng để nắm rõ và làm việc tốt trên lập trình hướng đối tượng bạn cần có kinh nghiệm xử lý với lập trình hướng thủ tục.

Trong lập trình hướng thủ tục gồm 2 bước sau:
- Xử lý vấn đề
- Xây dựng hàm và tối ưu mã nguồn

Ví dụ: Upload một tập tin

Bước 1: Xử lý vấn đề

Nội dung tập tin upload.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Untitled Document</title>
    </head>

    <body>
        <h1>Upload hình ảnh</h1>
        <form id="form1" name="form1" method="post" action="001_process.php"
enctype="multipart/form-data">

            <input type="file" name="upload" id="upload" />
            <br><br>
            <input type="submit" name="button" id="button" value="Submit" />
        </form>
    </body>
</html>
```

Nội dung tập tin process.php

```php
<?php

    //Lấy tên tập tin upload
    $file_name = $_FILES['upload']['name'];

    //Các kiểu tập tin mở rộng của hình ảnh
    $extent_file = "gif|jpg|png";
    $pattern = '/.+\.(gif|png|jpg)$/i';
    if (preg_match($pattern, $file_name)) {
        $file_type = true;
    } else {
        $file_type = false;
    }

    if ($file_type == true) {
        // File nguon can upload
        $source = $_FILES['upload']['tmp_name'];
        $dirUpload = '../images';
        $destination = $dirUpload . '/' . $_FILES['upload']['name'];
        if (!copy($source, $destination)) {
            $flag = false;
        } else {
            $flag = true;
        }
    } else {
        $flag = false;
    }

    if ($flag = true) {
        echo '<br> Upload thanh cong';
    } else {
        echo '<br> Upload that bai';
    }
?>
```

Bước 2: Xây dựng hàm và tối ưu mã nguồn

Nội dung tập tin functions.php

```php
<?php
    function check_extent_file($file_name, $extent_file) {
        $pattern = '/.+\.(' . $extent_file . ')$/i';
        if (preg_match($pattern, $file_name)) {
            $file_type = true;
        } else {
            $file_type = false;
        }

        return $file_type;
    }

    function upload_file($file, $dirUpload) {
        $source = $_FILES[$file]['tmp_name'];
        $destination = $dirUpload . '/' . $_FILES[$file]['name'];
        if (!copy($source, $destination)) {
            $flag = false;
        } else {
            $flag = true;
        }

        return $flag;
    }
?>
```

Nội dung tập tin process.php

```php
<?php

    require_once('functions.php');
```

```
    //Lấy tên tập tin upload
    $file_name = $_FILES['upload']['name'];

    $file_type = check_extent_file($file_name, 'gif|jpg|png');

    $flag = false;

    if ($file_type == true) {
        $flag = upload_file('upload', '../images');
    }

    if ($flag == true) {
        echo '<br> Upload thanh cong';
    } else {
        echo '<br> Upload that bai';
    }
?>
```

Kết luận: Qua ví dụ qua chúng ta đã hiểu được thế nào là lập trình hướng thủ tục và hiểu được cách tạo ra các hàm để dùng lại ở nhiều nơi.

2. Lập trình hướng đối tượng:

Lập trình hướng đối tượng: là kiểu lập trình dựa trên một nền tảng các class đã được xây dựng sẵn.

Trong ví dụ của lập trình hướng thủ tục trên chúng ta từng bước giải quyết các vấn đề sau đó tối ưu mã nguồn để đạt được một đoạn mã nguồn xử lý tốt nhất.

Nhưng trong lập trình hướng đối tượng thì ngược lại. Chúng ta phải xác định trước những chức năng cần thiết để xử lý vấnv đề. Sau đó xây dựng lớp chứa các phương thức với các chức năng đã xác định. Việc xác định các chức năng cần thiết không phải dễ dàng cho các lập trình viên vì điều này đòi hỏi lập trình viên phải nhiều kinh nghiệm thực tế trong quá trình xử lý hướng thủ tục.

Ví dụ: Xây dựng một lớp Upload với các yêu cầu sau:

- Upload một tập tin bất kỳ
- Thư mục chứa tập tin sẽ upload
- Kiểm tra phần mở rộng của tập tin
- Kiểm tra kích thước tập tin upload
- Thay đổi tên tập tin upload

Qua yêu cầu chúng ta cần xác định các phương thức cơ bản của lớp

- Nhận tập tin upload
- Kiểm tra điều kiện upload
    - Kích thước của tập tin
    - Kiểu mở rộng của tập tin
- Upload tập tin: giữ nguyên tên của tập tin gốc và thay đổi tập tin gốc

Trong phần nay chúng ta chỉ xây dựng lớp Upload đơn giản, chúng ta sẽ chưa quan tâm đến cách viết đúng chuẩn của lập trình hướng đối tượng vì chúng ta đang hiểu về sự khác nhau của 2 kiểu lập trình thông dụng

Bước 1: Xác định thuộc tính và phương thức

Thuộc tính:

- Tên của tập tin *($_fileName)*
- Kích thước tối đa được phép upload *($_fileSize)*
- Phần mở rộng của các tập tin được phép upload *($_fileExtension)*
- Tập tin tạm trước khi upload lên server *($_fileTmp)*
- Thư mục chứa tập tin upload ($_uploadDir)

Phương thức:

- Phương thức khởi tạo (*__construct()*)
- Phương thức thiết lập kích thước tập tin upload *(setFileSize())*
- Phương thức thiết lập phần mở rộng của tập tin upload *(setFileExtension())*
- Phương thức thiết lập thư mục upload tập tin *(setUploadDir())*
- Phương thức kiểm tra tất cả các điều kiện của tập tin upload *(isVail())*
- Phương thức upload tập tin *(upload())*

Bước 2: Xây dựng khung sườn cho lớp

```
<?php
class Upload{
        // Bien luu tru ten tap tin upload
        var $_fileName;

        //Bien luu tru kich thuoc cua tap tin upload
        var $_fileSize;

        //Bien luu tru phan mo rong cua tap tin upload
        var $_fileExtension;

        //Bien luu tru duong dan cua thu muc tam tap tin upload
        var $_fileTmp;

        //Bien luu tru duong tren server cua tap tin upload
        var $_uploadDir;

        //Bien luu tru error
        var $_errors;

        //Ham khoi tao doi tuong
        function __construct($file_name){

        }

        //Ham lay thanh phan mo rong cua tap tin upload
        function getFileExtension(){

        }

        //Ham thiet lap thanh phan mo rong tap tin upload
        function setFileExtension($value){

        }

        //Ham thiet lap kich thuoc tap tin upload
        function setFileSize($value){

        }
```

```
        //Ham thiet lap kich thuoc tap tin upload
        function setUploadDir($value){

        }

        //Ham kiem tra dieu kien upload
        function isVail(){

        }

        //Ham upload tap tin
        function upload($rename =  false, $prefix = 'file_'){

        }
}
```

Bước 3: Xây dựng chi tiết các phương thức xử lý

```
<?php
//upload.class.php
class Upload{
        // Bien luu tru ten tap tin upload
        var $_fileName;

        //Bien luu tru kich thuoc cua tap tin upload
        var $_fileSize;

        //Bien luu tru phan mo rong cua tap tin upload
        var $_fileExtension;

        //Bien luu tru duong dan cua thu muc tam tap tin upload
        var $_fileTmp;

        //Bien luu tru duong tren server cua tap tin upload
        var $_uploadDir;

        //Bien luu tru error
        var $_errors;

        //Ham khoi tao doi tuong
        function __construct($file_name){
                //echo __METHOD__;

                $fileInfo =  $_FILES[$file_name];
                $this->_fileName = $fileInfo['name'];
                $this->_fileSize = $fileInfo['size'];
                $this->_fileExtension = $this->getFileExtension();
                $this->_fileTmp = $fileInfo['tmp_name'];

        }

        //Ham lay thanh phan mo rong cua tap tin upload
        function getFileExtension(){
                $subject = $this->_fileName;
                $pattern = '#\.([^\.]+)$#i';
                preg_match($pattern,$subject,$matches);
                return $matches[1];
        }

        //Ham thiet lap thanh phan mo rong tap tin upload
        function setFileExtension($value){
                $subject = $this->_fileExtension;
                $pattern = '#(' . $value . ')#i';
                if(preg_match($pattern,$subject,$matches)!= 1){
                        $this->_errors[] = 'Phan mo rong khong phu hop';
                }
        }
```

```
        //Ham thiet lap kich thuoc tap tin upload
        function setFileSize($value){
                $size = $value * 1024;
                if($this->_fileSize > $size){
                        $this->_errors[] = 'Kich thuoc tap tin lon hon ' . $size . 'kb';
                }
        }

        //Ham thiet lap kich thuoc tap tin upload
        function setUploadDir($value){
                if(file_exists($value)){
                        $this->_uploadDir = $value;
                }else{
                        $this->_errors[] = 'Thu muc khong he ton tai';
                }
        }

        //Ham kiem tra dieu kien upload
        function isVail(){
                $flagErr = false;
                if(count($this->_errors)>0){
                        $flagErr =  true;
                }

                return $flagErr;
        }

        //Ham upload tap tin
        function upload($rename =  false, $prefix = 'file_'){
                if($rename ==  false){
                        $source = $this->_fileTmp;
                        $dect = $this->_uploadDir . $this->_fileName;
                }else{
                        $source = $this->_fileTmp;
                        $dect = $this->_uploadDir . $prefix . time() . '.' . $this-
>_fileExtension;
                }
                copy($source,$dect);
        }
}
```

Bước 4: Sử dụng lớp

Tạo tập tin *upload.php*

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Upload form</title>
    </head>

    <body>
        <h1>Upload hình ảnh</h1>
        <form id="myForm" name="myForm" method="post" enctype="multipart/form-data"
action="process.php">
            <input type="file" name="picture" id="picture"></input>
            <br></br>
            <input type="submit" name="button" id="button" value="Submit"></input>
        </form>
    </body>
</html>
```

Tạo tập tin *process.php*

```
<?php

 include 'upload.class.php';
```

```
$upload  = new Upload('picture');
$upload->setUploadDir('images/');
if($upload->isVail() == true){
	echo "<pre>";
	print_r($upload->_errors);
	echo "</pre>";
}else{
	$upload->upload(true,'pic_');
}
```

B. Lập trình hướng đối tượng

- Lấy đối tượng là nền tảng
- Tìm những đối tượng có sẵn hoặc xây dựng những đối tượng
- Sau đó kết hợp với nhau để giải quyết vấn đề
- Xây dựng những đối tượng mã lệnh có liên hệ khắn khít với đối tượng của thế giới thực

Ví dụ 3: Game bóng đá:

- Game bóng đá là một chương trình rất lớn chắc chắn nếu bạn muốn làm nó bạn phải xây dựng nó trên mô hình hướng đối tượng. Vậy việc đầu tiên của trước khi xây dựng game này bạn cần xác định các đối tượng chính của game

- Những đối tượng chính của game mà bạn có thể dễ dàng nhìn thấy như:
  - Câu lạc bộ
  - Sân vận động
  - Giải thi đấu
  - Cầu thủ
  - Huấn luyện viên
  - Cổ động viên…

- Trong một ứng dụng lớn như game bóng đá các bạn sẽ thấy xuất hiện rất nhiều đối tượng. Chúng ta sẽ phân tích thử một đối tượng trong game đó là đối tượng con người.

- Con người là một lớp chính trong game từ đối tượng ‘con người’ chúng ta sẽ mở rộng ra các đối khác như cầu thủ, trọng tài, huấn luyện viên, cổ động viên…

- Đơn giản hóa việc phát triển các chương trình

- Giúp tạo ra những chương trình có tính mềm dẻo và linh động cao (Khi sửa chữa bảo trì, nâng cấp dễ dàng)

3. Lớp (Class)

- Trong lập trình các bạn thường có các kiểu dữ liệu như INT (INT mô tả các số nguyên), STRING (mô tả cho chuỗi), FLOAT (mô tả cho số thực)… nhưng nếu 1 kiểu dữ liệu mới xuất hiện thì chúng phải làm sao? Lúc đó chúng ta sẽ phải tạo ra 1

định nghĩa cho kiểu dữ liệu mới đó thông qua class. Nên class là một kiểu dữ liệu được định nghĩa trong chương trình, là một sự nâng cao của structure.

- Ví dụ tôi muốn tạo ra một kiểu dữ liệu mới để mô tả một con mèo thì lúc này tôi phải tạo ra một class để định nghĩa cho kiểu dữ liệu con mèo

- Vậy một câu hỏi đặt ra muốn tạo một class để định nghĩa cho một đối tượng mới chúng ta cần phải làm như thế nào? Điều này rất đơn giản. Chúng ta chỉ cần xác định 2 vấn đề trước khi tạo một các class mô tả cho một đối tượng đó là:
    - Những thuộc tính của đối tượng.
    - Hành động của đối tượng

Ví dụ: Tạo class ConMeo

Để tạo class cho đối tượng ConMeo chúng ta cần xác định 2 phần:

- Thuộc tính (attribute)
    - Tên
    - tuổi
    - màu lông
    - …

- Hành động (phương thức - method)
    - chạy
    - kêu
    - cắn
    - cào
    - …

Trong phần này chúng ta chỉ mô tả về một đối tượng chung chung không cụ thể là một đối tượng nào cả

Tập tin *conmeo.class.php*

```
<?php

class ConMeo{

	//Khai bao cac thuoc tinh
	private $name;
	private $age;
	private $color;

	public function run(){
		return 'It is runing';
	}

	public function shout(){
		return 'It is shoutting "meo meo"';
	}

	public function climb(){
		return 'It is climbing';
	}

}
```

```
?>
```

4. Đối tượng (Object)

- Thể hiện một lớp thành một thực thể nào đó
- Có thể tạo nhiều đối tượng từ một lớp

Ví dụ:

```
$conMeoA = new ConMeo();
$conMeoB = new ConMeo();
```

Ví dụ:

```
- Nhà tôi có một con mèo
- Nhà bạn tôi có nuôi một con mèo
```

- Các đối tượng sẽ có đặc tính khác nhau

Ví dụ:

```
- Con mèo của tôi có tên là Mimi có lông màu trắng
- Con mèo của bạn tôi có tên là Doremon có lông màu vàng
```

- Tuy 2 con mèo trên có những đặc điểm khác nhau nhưng nó cùng lớp là con mèo.

5. Lớp và đối tượng

- Lớp là một cái chung chung
- Đối tượng là một cái cụ thể

Ví dụ:

```
- Công thức là một lớp - Cái bánh là một đối tượng
- Mô tả về con mèo là một lớp - Con mèo nhà tôi là một đối tượng
```

6. Xây dựng lớp

- Tạo class

```
Class ConMeo{

}
```

- Khởi tạo đối tượng từ class

```
$conMeoA = new ConMeo();
$conMeoB = new ConMeo();
```

7. Thuộc tính & phương thức (properties - method)

a. Thuộc tính

- Là các đặc tính, đặc điểm của một lớp. Thuộc tính bao gồm:
  - Các biến: lưu trữ các giá trị

- Biểu thức get và set: cho phép lấy và gán giá trị

Ví dụ 1:

```
<?php
//ConMeo.class.php
class ConMeo{
	private $name;
	private $age;
	private $color;

	public function getName()
	{
		return $this->name;
	}

	public function setName($value)
	{
		$this->name = $value;
	}

	public function getAge()
	{
		return $this->age;
	}

	public function setAge($value)
	{
		$this->age = $value;
	}

	public function getColor()
	{
		return $this->color;
	}

	public function setColor($value)
	{
		$this->color = $value;
	}

}
?>
```

```
<?php
	//index.php
	require_once('ConMeo.class.php');

	$conMeoA = new ConMeo();
	$conMeoA->setName('Mimi');
	$conMeoA->setAge(3);
	$conMeoA->setColor('Vàng');

	echo 'Tên: ' . $conMeoA->getName() . '<br>';
	echo 'Tuổi: ' . $conMeoA->getAge() . '<br>';
	echo 'Màu lông: ' . $conMeoA->getColor() . '<br>';
	echo '<hr>';

	$conMeoB = new ConMeo();
	$conMeoB->setName('Mimi');
	$conMeoB->setAge(3);
	$conMeoB->setColor('Vàng');

	echo 'Tên: ' . $conMeoB->getName() . '<br>';
	echo 'Tuổi: ' . $conMeoB->getAge() . '<br>';
	echo 'Màu lông: ' . $conMeoB->getColor() . '<br>';

?>
```

b. Phương thức

- Là các hành động có thể được thực hiện từ lớp
- Phương thức cũng giống như hàm nhưng là hàm riêng của lớp
- Phương thức có thể nhận vào các tham số và trả về các giá trị

Ví dụ 2:

```
//conmeo.class.php
Class ConMeo{
	.....
	public function showInfo(){
		/*echo '<br>Tên: ' . $this->getName();
		echo '<br>Tuổi: ' . $this->getAge();
		echo '<br>Color: ' . $this->getColor();
		echo '<hr>';*/

		echo '<br>Tên: ' . $this->name;
		echo '<br>Tuổi: ' . $this->age;
		echo '<br>Color: ' . $this->color;

	}

}
```

```
//index.php
<?php
	require_once('ConMeo.class.php');

	$conMeoA = new ConMeo();
	$conMeoA->setName('Mimi');
	$conMeoA->setAge(3);
	$conMeoA->setColor('Vàng');

	$conMeoA->showInfo();

?>
```

Ví dụ 3:

```
//conmeo.class.php
Class ConMeo{
	…
	public function run(){
		return 'It is runing';
	}

	public function shout(){
		return 'It is shoutting "meo meo"';
	}

	public function climb(){
		return 'It is climbing';
	}

}
```

```
//index.php
<?php
	require_once('ConMeo.class.php');

	$conMeoA = new ConMeo();
	$conMeoA->setName('Mimi');
	$conMeoA->setAge(3);
	$conMeoA->setColor('Vàng');
```

```
        $conMeoA->showInfo();

        echo 'Nó đang làm gì? - ' . $conMeoA->run();

?>
```

8. Phương thức khởi tạo (__construct())

a. Phương thức contruct()

Ví dụ 1:

```
// ConMeo.class.php
<?php

class ConMeo{

        //Khai bao cac thuoc tinh
        private $name;
        private $age;
        private $color;

        public function __construct(){
                $this->name = 'Mimi';
                $this->age = 1;
                $this->color = 'Vàng';
        }

        ...

}

?>
```

```
// index.php
<?php

class ConMeo{

        require_once('ConMeo.class.php');

        $conMeoA = new ConMeo();
        $conMeoA->showInfo();
}

?>
```

Output:

```
Tên: Mimi
Tuổi: 1
Color: Vàng
```

b. Phương thức contruct() với tham số

Ví dụ 2:

```
// ConMeo.class.php
<?php

class ConMeo{

        //Khai bao cac thuoc tinh
        private $name;
        private $age;
```

```
        private $color;

        public function __construct($name,$age,$color){
                $this->name = $name;
                $this->age = $age;
                $this->color = $color;
        }

        ...

}

?>
```

```
// index.php
<?php

class ConMeo{

        require_once('ConMeo.class.php');

        $conMeoA = new ConMeo('Doremon',1000,'Xanh');
        $conMeoA->showInfo();
}

?>
```

Output:

```
Tên: Doremon
Tuổi: 1000
Color: Xanh
```

c. Phương thức contruct() với tham số mặc định

Ví dụ 3:

```
// ConMeo.class.php
<?php

class ConMeo{

        //Khai bao cac thuoc tinh
        private $name;
        private $age;
        private $color;

        public function __construct($name = 'Mimi',$age = 1, $color = 'Vàng'){
                $this->name = $name;
                $this->age = $age;
                $this->color = $color;
        }

        ...

}

?>
```

```
// index.php
<?php

class ConMeo{

        require_once('ConMeo.class.php');

        $conMeoA = new ConMeo();
```

```
        $conMeoA->showInfo();
}

?>
```

Output:

```
Tên: Mimi
Tuổi: 1
Color: Vàng
```

d. Hàm contruct() với cách đặt tên trùng với tên class

Ví dụ 4:

```
// ConMeo.class.php
<?php

class ConMeo{

        //Khai bao cac thuoc tinh
        private $name;
        private $age;
        private $color;

        public function ConMeo($name = 'Mimi',$age = 1, $color = 'Vàng'){
                $this->name = $name;
                $this->age = $age;
                $this->color = $color;
        }

        ...

}

?>
```

```
// index.php
<?php
        require_once('ConMeo.class.php');

        $conMeoA = new ConMeo();
        $conMeoA->showInfo();
?>
```

Output:

```
Tên: Mimi
Tuổi: 1
Color: Vàng
```

e. Hàm contruct() với tham số là mảng

Ví dụ 5:

```
// ConMeo.class.php
<?php

class ConMeo{

        //Khai bao cac thuoc tinh
        private $name;
        private $age;
        private $color;

        public function __construct($arrParama){
```

```
                $this->name = $arrParama['name'];
                $this->age = $arrParama['age'];
                $this->color = $arrParama['color'];
        }

        ...

}

?>
```

```
// index.php
<?php
        require_once('ConMeo.class.php');

        $arrParama = array('name'=> 'Mimi',
                           'age' => 2,
                           'color' => 'Vàng');
        $conMeoA = new ConMeo($arrParama);

        $conMeoA->showInfo();
?>
```

Output:

```
Tên: Mimi
Tuổi: 2
Color: Vàng
```

9. Tính chất kế thừa

Tính kế thừa trong là một ưu điểm của OOP nó giúp chúng ta mở rộng và phát triển chương trình mà không làm ảnh hưởng đến những thành phần đã có sẵn.

Ví dụ: Tạo lớp ConBao kế thừa từ lớp ConMeo

Tập tin *ConBao.class.php*

```
<?php
require_once 'ConMeo.class.php';
class ConBao extends ConMeo{
        public function showInfo(){
                echo '<br>' . __METHOD__;
                echo 'Day la mot lop the hien con bao';
        }
}
```

Tập tin *index.php*

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php
require_once 'ConBao.class.php';

$arrCatInfo = array();
$arrCatInfo['name']    = 'Mabu';
$arrCatInfo['age']            = 5;
$arrCatInfo['color']   = "Xám";
$arrCatInfo['weight']  = "50 kg";
$arrCatInfo['height']  = "70 cm";
$conBaoA = new ConBao($arrCatInfo);
$conBaoA->showInfo();
```

10. Hàm __destruct()

Hàm _destruct() là một hàm thực sự rất khó hiểu trong PHP và nhiều lập trình viên không biết sử dụng nó vào mục đích gì. Theo lý thuyết hàm này sẽ được tự động gọi sau khi một đối tượng được khởi tạo và nó thực hiện hủy một số giá trị nào nào.

Hàm __destruct() là phương thức tự động chạy khi đối tượng được khởi tạo. Nó chỉ thực thi những lệnh trong thân hàm ở cuối trang mà đối tượng được khởi tạo.

Phương thức này thường dùng để
- Hủy hoặc tạo một session.
- Giải phóng bộ nhớ
- Đóng kết nối của ứng dụng đến Database
- Đóng kết nối của khi chúng ta mở một tập tin
- ...

Ví dụ 1: Vị trí hàm __destruct() thực thực thi

```
<?php
//User.class.php
class User{
        private $name;
        private $permission;
        public function __construct($name,$permission)
    {
            echo '<br> Construct function';
        $this->name = $name;
            $this->permission = $permission;

    }

        function __destruct()
    {
        echo '<br> Destruct function';

    }

}
?>
```

```
<?php
        //index.php
        require_once('User.class.php');
        $obj = new User('zendvn','Administrator');
        echo '<br>This is a test 1';
        echo '<br>This is a test 2';
        echo '<br>This is a test 3';
?>
```

Ví dụ 2: Khởi tạo một session

```
<?php
//User.class.php
class User{
        private $name;
        private $pass;
        private $lastName;
        private $firstName;
        private $website;
```

```
        public function __construct($name,$pass)
    {
           echo '<br> Construct function';
           if($name == 'KhanhPham' &&  $pass = '123456'){
               $this->name = $name;
                  $this->pass = $pass;
                  $this->getInfo();
           }

    }

    public function getInfo(){
        $this->lastName = 'Vũ Khánh';
        $this->firstName = 'Phạm';
        $this->website        = 'http://www.zend.vn';
    }

    function __destruct()
    {
        echo '<br> Destruct function';
               $_SESSION['user'] = serialize($this);
    }

}
?>
```

```
<?php
//index.php
session_start();
require_once 'User.class.php';

$user = new User('KhanhPham','123456');

<?php
session_start();
if(isset($_SESSION['user'])){
        $userInfo = unserialize($_SESSION['user']);
        echo "<pre>";
        print_r($userInfo);
        echo "</pre>";
}
unset($_SESSION['user']);
```

Ví dụ 3: Giải phóng bộ nhớ và đóng kết nối với Database

```
<?php
//Database.class.php
class Database {

        private $hostname     = 'localhost';
        private $username     = 'root';
        private $password     = '';
        private $db                   = 'test';

        private $dbCon        = null;
        private $result;

        private $connection;
        private $flagConnect;

        public function __construct() {
                $connection = mysql_connect ( $this->hostname, $this->username, $this-
>password );

                if (! $connection) {
                        die ( 'Could not connect: ' . mysql_error () );
                } else {
                        $this->connection = $connection;
```

```
                        $this->flagConnect = true;
                        mysql_select_db($this->db, $connection);
                }

        }

        public function fetchAll($table_name){
                if ($this->flagConnect == true) {
                        $sql = 'SELECT * FROM ' . $table_name;
                        $this->result = mysql_query($sql);
                        return $this->result;
                }
        }

        function __destruct() {

                if ($this->flagConnect == true) {
                        mysql_free_result($this->result);
                        mysql_close ( $this->connection );
                }

        }

}
?>
```

```
<?php
//index.php
require_once 'Database.class.php';

$db = new Database();
$result = $db->fetchAll('user_group');

while($row  = mysql_fetch_assoc($result)){
        echo '<br>' . $row['id'] . ' - ' . $row['group_name'];
}
```

11. Hàm clone

Hàm này dùng để sao chép một đối tượng từ một đối tượng khác.

Ví dụ 1: Sao chép một đối tượng không sử dụng phương thức clone

```
<?php
        require_once 'ConMeo.class.php';

        $conMeoA = new ConMeo();
        $conMeoA->setName("Tiger");
        $conMeoA->setAge(3);
        $conMeoA->setColor('Vàng');


        $conMeoB = $conMeoA;
        $conMeoB->setName("Doremon");
        $conMeoB->setAge(2);

        echo '<br>-------------------------<br>';
        echo 'In thong tin con meo A <br>';
        $conMeoA->showInfo();

        echo '<br>-------------------------<br>';
        echo 'In thong tin con meo B <br>';
        $conMeoB->showInfo();
?>
```

Ví dụ 2: Sao chép một đối tượng sử dụng phương thức clone

```
<?php
	require_once 'ConMeo.class.php';

	$conMeoA = new ConMeo();
	$conMeoA->setName("Tiger");
	$conMeoA->setAge(3);
	$conMeoA->setColor('Đen');


	$conMeoB = clone $conMeoA;
	$conMeoB->setName("Doremon");
	$conMeoB->setAge(2);

	echo '<br>-------------------------<br>';
	echo 'In thong tin con meo A <br>';
	$conMeoA->showInfo();

	echo '<br>-------------------------<br>';
	echo 'In thong tin con meo B <br>';
	$conMeoB->showInfo();
?>
```

12. self & parent

a. self

Là đại diện cho cách khởi tạo lớp hiện thời và thường được sử dụng gọi đến biến số có khóa static hay hàm nào đó trong lớp đang hiện tại.

Ví dụ 1:

```
<?php
class ConMeo{
	private $name;
	private $age;
	private $color;
	public static $maxSpeed = '30km/h';

	// Code cũ ...

	public function showInfo(){
		echo '<br> Tên con mèo là: ' . $this->getName();
		echo '<br> Tuổi: ' . $this->getAge();
		echo '<br> Màu lông: ' . $this->getColor();
		echo '<br>Nó đang làm gì? - ' . self::run();
		echo '<br>Tốc độ tối đa của nó là:  - ' . self::$maxSpeed;
	}

	// Code cũ ...
}
```

Ví dụ 2:

```
<?php
class ConMeo{
	private $name;
	private $age;
	private $color;
	public static $maxSpeed = '30km/h';

	// Code cũ ...

	public function showInfo(){
		echo '<br> Tên con mèo là: ' . $this->getName();
		echo '<br> Tuổi: ' . $this->getAge();
		echo '<br> Màu lông: ' . $this->getColor();
```

```
                echo '<br>Nó đang làm gì? - ' . ConMeo::run();
                echo '<br>Tốc độ tối đa của nó là:  - ' . ConMeo::$maxSpeed;
        }

        // Code cũ ...
}
```

b. parent

Đại diện cho class cha của lớp đang thừa kế và thường được sử dụng gọi đến biến số có khóa static hay hàm nào đó trong lớp cha của lớp hiện tại hiện tại.

```
<?php
require_once('ConMeo.class.php');

class ConBao extends ConMeo{
        public static $maxSpeed = '90km/h';


        public function showInfo(){
                parent::showInfo();
                echo '<br>Tốc độ tối đa của nó là:  - ' . self::$maxSpeed;
        }
}
?>
```

```
<?php
        require_once('ConBao.class.php');

        $conBaoA = new ConBao();
        $conBaoA->showInfo();

?>
```

13. Static

Ví dụ 1:

```
<h1> Static </h1>
<?php

$acc = new Account();
echo Account::$taiKhoan;
echo '<br>' . $acc->test();
echo '<br>' .Account::test();

class Account{

        static $taiKhoan = 100;

        static function test(){
                return 'Static method';
        }

}
```

Ví dụ 2:

```
<h1> Static </h1>
<?php

$year2010 = new Account();
$year2010->year(2010);

$year2011 = new Account();
$year2011->year(2011);
```

```
$year2012 = new Account();
$year2012->year(2012);


class Account{

	static $taiKhoan = 100;

	public function year($value){
		echo '<br> So tien ban dau la: ' . self::$taiKhoan;

		$laiSuat = self::$taiKhoan * 10/100;
		echo '<br> Tien lai xuat trong nam ' . $value . ': ' . $laiSuat;

		self::$taiKhoan = self::$taiKhoan + $laiSuat;
		echo '<br> Tong so tien trong tai khoan hien nay la: ' . self::$taiKhoan;
	}
}
```

14. PPP (public - protected - private)

Giống như mọi ngôn ngữ lập trình hướng đối tượng khác
public: Có thể truy cập từ mọi nơi
protected: chỉ sử dụng cho class đó và các class được mở rộng từ class đó
private: chỉ sử dụng ở trong chính class đó

```
<?php
	class sample {
		public $a = 1;
		private $b = 2;
		protected $c = 3;

		function __construct() {
			echo $this->a . $this->b . $this->c;
		}
	}

	class miniSample extends sample {
		function __construct() {
			echo $this->a . $this->b . $this->c;
		}
	}

	$a = new sample();
	$b = new miniSample();
	echo $a->a . $a->b . $a->c;
?>
```

15. Hằng số (Constants)

Trong PHP5 bạn có thể định nghĩa các giá trị mà các bạn muốn không bao giờ thay đổi bằng khóa const. Với khóa này thì để truy cập vào lấy giá trị của nó chúng ta chỉ có thể sử dụng toán tử (::).

PHP có thể nhận biết thế nào là một hằng số và một biến thuộc tính trong một class cho dù tên của nó giống nhau.

Ví dụ :

```
<?php
class Account{
	const money = 1000;
	public $money = 'So tien trong tai khoan';

	public function showInfo(){
		echo __METHOD__ . '<br>';
		echo $this->money . ' : ' . self::money;
	}
}

$acc  = new Account();
echo $acc->money . ' : ' . Account::money;

$acc->showInfo();
```

16. Final

- PHP5 cho phép định nghĩa class và mothod với khóa FINAL
- Đối với method (phương thức): chúng ta không thể override từ lớp con các phương thức có khóa final ở lớp cha
- Đối với lớp (class): Khi chúng ta dùng khóa FINAL thì chúng ta không thể extents từ lớp đó

Ví dụ 1: FINAL với phương thức

```
<?php
class main{

	final function bar(){
		echo 'Chay thu chuc nang doi voi khoa FINAL';
	}
}

class child extends main{

	public function bar(){
		echo  'Chay thu chuc nang doi voi khoa FINAL tai lop con';
	}
}

$test = new child();
$test->bar();
```

Ví dụ 2: FINAL với lớp

```
<?php
final class main{

	public function bar(){
		echo 'Chay thu chuc nang doi voi khoa FINAL';
	}
}

class child extends main{

	public function bar(){
		echo  'Chay thu chuc nang doi voi khoa FINAL tai lop con';
	}
}

$test = new child();
$test->bar();
```

17. Autoload

- Để làm cách khởi tạo các lớp trở nên bình thường và đơn giản PHP cung cấp cho chúng ta một hàm gọi là autoload

- Khi chạy hàm này sau khi khao báo đường dẫn đến thư viện của các class chúng ta không cần phải dùng hàm include trước khi gọi một lớp nào đó nữa

Ví dụ:

```
<?php
        function __autoload($class_name) {
                $patch = 'class/';
         require_once $patch . "{$class_name}.class.php";
    }

   $conMeoA = new ConMeo();
   $conMeoA->showInfo();
   echo '<br>-----------------</br>';

   $conBaoA = new ConBao();
   $conBaoA->showInfo();
   echo '<br>-----------------</br>';
```

18. Serialization

- Là quá trình chuyển đổi một đối tượng PHP thành một chuỗi đặc biệt và sau này có thể được sử dụng để tái tạo lại biến.
- Nó thường được sử dụng đến biến một đối tượng hay một mảng thành chuỗi để lưu trong CSDL
    - serialize (): chuyển một đối tượng nào đó thành một chuỗi đặc biệt
    - unserialize (): chuyển đặc biệt trở lại hiện tại ban đầu

Ví dụ:

```
<?php
class test {
        public $foo = 1, $bar, $baz;

        function __construct() {
                $this->bar = $this->foo * 10;
                $this->baz = ($this->bar + 3) / 2;
        }
}
$a = serialize(new test()); // encode instantiated class test
$b = unserialize($a); // restore the class into $b;
?>
```

19. _sleep() & __wakeup

Phương thức __sleep() được thực hiện khi chúng ta đưa một đối tượng vào hàm serialize. Lúc này trong phương thức __sleep() chúng ta sẽ khai báo những thành phần nào của đối tượng được chuyển thành một chuỗi đặc biệt.

Phương thức __wakeup dùng để làm sạch đối tượng như khi mới khởi tạo ban đầu với các giá trị mặc định được gán vào. Điều này giúp chúng không phải khởi tạo đối tượng một lần nữa.

Ví dụ:

```
<?php
//ConMeo.class.php
class ConMeo{
	private $name;
	private $age;
	private $color;

	public function __construct($name = 'Mimi',$age = 2,$color = 'Den'){
		echo __METHOD__;
		$this->name = $name;
		$this->age = $age;
		$this->color = $color;

	}

	public function save($value){
		if(!empty($value)){
			echo '<br> Du lieu da duoc luu vao database';
		}
	}

	function __sleep(){
		return array('name','age','color');
	}

	function __wakeup(){
		$this->name = 'Mimi';
		$this->age = 2;
		$this->color = 'Den';
	}

	//Code cũ ...
}
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php
	//index.php
	require_once 'ConMeo.class.php';

	$conMeo = new ConMeo();
	$conMeo->setName("Tiger");
	$conMeo->setColor("Xám");
	$conMeo->setAge(4);
	echo "<pre>";
	print_r($conMeo);
	echo "</pre>";

	$data = serialize($conMeo);
	echo '<br>' . $data;
	$conMeo->save($data);

	$conMeo = unserialize($data);
	echo "<pre>";
	print_r($conMeo);
	echo "</pre>";
```

20- toString()

- Phương thức này giúp chúng ta đưa các phần của đối tượng thành chuỗi. Và sử dụng lệnh echo để in ra.

Ví dụ:

```
<?php
//ConMeo.class.php
class ConMeo{
	private $name;
	private $age;
	private $color;

	public function __construct($name = 'Mimi',$age = 2,$color = 'Den'){
		echo __METHOD__;
		$this->name = $name;
		$this->age = $age;
		$this->color = $color;

	}

	function __toString(){
		$str = '';
		$str .= '<br>' . $this->name;
		$str .= '<br>' . $this->age;
		$str .= '<br>' . $this->color;

		return $str;
	}

	public function setName($value){
		$this->name = $value;
	}

	//Code cũ ...
}
```

```
<?php
//index.php
 require_once 'ConMeo.class.php';

 $conMeoA = new ConMeo();
 echo $conMeoA;
```

Zend Framework! Programming
Shopping application

Regular expressions

Regular expressions (Biểu thức chính qui) bắt nguồn từ ngôn ngữ Perl và hiện nay nó hầu như có trong tất cả các ngôn ngữ lập trình. Là một phần rất quan trong quá trình xử lý chuỗi và hỗ trợ cho các lập trình viên giảm bớt những dòng mã trong quá trình xử lý chuỗi bằng những biểu thức ngắn gọn nhưng đem lại kết quả như sự mong đợi. Nhưng để sử dụng tốt Regular expressions thì không phải dễ dàng nếu chúng ta không đi đúng hướng và hiểu hết các kí hiệu của nó. Regurlar Expression thường sử dụng trong những trường hợp sau:
- Kiểu tra giá trị các phần tử của Form
- Xử lý yêu cầu phức tạp trong chuỗi như bóc tách, thay đổi nội dung, loại bỏ các ký tự không cần thiết.

Giáo trình: Zend Framework! Programming
Chuyên đề: Shopping application

Bản quyền: ZendVN group
Biên soạn: Phạm Vũ Khánh
Email: vukhanh2212@gmail.com
Điện thoại: 0908.893326
Website: www.zend.vn
Cập nhật: 07-2010

A. Một số hàm hỗ trợ Regular expressions

1. Tham chiếu

Trong lập trình có 2 loại biến chúng ta thường sử dụng truyền giá trị vào hàm đó là biến tham trị và biến tham chiếu.

- Biến tham trị: Khi truyền giá trị vào hàm, mọi sự thay đổi của biến tham trị trong thân hàm không ảnh hưởng đến biến ban đầu.
- Biến tham chiếu: Khi truyền giá trị vào hàm, thì giá trị của biến tham trị sẽ thay đổi bởi nội dung xử lý trong thân hàm được gọi.

Ví dụ:

```
<?php
	$phepCong = phepCong(10,5,$c);
	echo "<br>" . $phepCong;
	echo "<br>" . $c;

	function phepCong($a = 0, $b = 0, &$c = 0){
		$tong = $a + $b;
		$c = $tong * 10;
		return $tong;
	}
?>
```

2. preg_match

Hàm preg_match trả về giá trị 1 nếu tìm được pattern trong chuỗi và trả về 0 nếu không tìm được. Hàm này trả về một mảng có một phần tử trong chuỗi

```
preg_match ($pattern, $subject, &$matches)
```

$pattern : chứa mẫu cần tìm kiếm, nó được xem như 1 chuỗi.
$subject : chứa chuỗi nguồn.
&$matches: Mảng tham biến này chứa phần tử được tìm thấy

Ví dụ:

```
<?php
	$subject = "abcdefasdasdasd";
	$pattern = '#a#';
	echo preg_match($pattern, $subject, $matches) . '<br>';
	print_r($matches);
?>
```

3. preg_match_all

Hàm preg_match trả về giá trị 1 nếu tìm được pattern trong chuỗi và trả về 0 nếu không tìm được. Hàm này trả về một mảng có nhiều phần tử trong chuỗi

```
preg_match_all($pattern, $subject, &$matches);
```

Ví dụ:

```
<?php
$subject = "abcdefasdasdasd";
$pattern = '#a#';
echo preg_match_all($pattern, $subject, $matches) . '<br>';
echo '<pre>';
print_r($matches);
echo '</pre>';
?>
```

Output:

```
4
Array
(
    [0] => Array
        (
            [0] => a
            [1] => a
            [2] => a
            [3] => a
        )
)
```

Hai hàm này thường dùng để kiểm tra chuỗi nhập vào có đúng với yêu cầu hay không hoặc sử dụng để bóc tách các chuỗi có cấu hình phức tạp.

B. Các ký hiệu Regular expressions

1. Tìm chuỗi

$subject	Hello, world!
$pattern	Hello
$matches	Hello, world!

2. Ký tự ^ và $

Ký tự mũ (^): Tìm giá trị ở đầu chuỗi
Ký tự dola ($): Tìm giá trị ở cuối chuỗi

Ví dụ: Tìm chữ 'who' ở đầu chuỗi:

$subject	who is who
$pattern	^who
$matches	who is who

Ví dụ: Tìm chữ 'who' ở cuối chuỗi:

$subject	who is who
$pattern	who$
$matches	who is who

3. Ký tự \

*Khi tìm kiếm chuỗi có một số giá trị đặc biệt trong cú pháp của REX chúng ta sẽ phải sử dụng dấu *

Ví dụ 1: Tìm ký hiệu '$' trong chuỗi:

$subject	12 \-\ 25
$pattern	$
$matches	12 \-\ 25 *(Not Found)*

Ví dụ 2: Tìm ký hiệu '$' trong chuỗi:

$subject	12 \-\ 25
$pattern	\$
$matches	12 \-\ 25

Ví dụ 3: Tìm ký hiệu '$' ở đầu chuỗi:

$subject	12 \-\ 25
$pattern	^\$
$matches	12 \-\ 25

Ví dụ 4: Tìm ký hiệu '$' ở cuối chuỗi:

$subject	12 \-\ 25
$pattern	\$$
$matches	12 \-\ 25

Ví dụ 5: Tìm ký hiệu '\' trong chuỗi:

$subject	12 \-\ 25
$pattern	\\
$matches	12 \-\ 25

4. Ký tự .

Ký tự là dấu chấm là đại diện cho một ký tự bất kỳ

Ví dụ 1:

$subject	Regular expressions are powerful!!!
$pattern	.
$matches	Regular expressions are powerful!!!

Ví dụ 2:

$subject	Regular expressions are powerful!!!
$pattern	…..
$matches	Regular expressions are powerful!!!

Ví dụ 3:

$subject	O.K.
$pattern	.
$matches	O.K.

Ví dụ 4: Tìm ký tự chấm (.) trong chuỗi

$subject	O.K.
$pattern	\.
$matches	O.K.

5. Ký tự [ ]

Ví dụ 1: Tìm tập hợp các ký tự oyu trong chuỗi

$subject	How do you do?
$pattern	[oyu]
$matches	How do you do?

Ví dụ 2:

$subject	How do you do?
$pattern	[dH].
$matches	How do you do?

Ví dụ 3:

$subject	How do you do?
$pattern	[owy][yow]
$matches	How do you do?

6. Ký tự ngạch ngang (-)

Ví dụ 1: Lấy các ký tự từ C-K

$subject	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
$pattern	[C-K]
$matches	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

Ví dụ 2:

$subject	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
$pattern	[CDEFGHIJK]
$matches	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

Ví dụ 3:

$subject	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
$pattern	[a-d]
$matches	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

Ví dụ 4:

$subject	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
$pattern	[2-6]
$matches	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

Ví dụ 5:

$subject	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

$pattern	[C-Ka-d2-6]
$matches	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

7. Phủ định (^)

Nếu một ký tự ^ đứng trước một ký tự hay một tập hợp có nghĩa là phủ định của ký hiệu hay tập hợp đó

Ví dụ 1:

$subject	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
$pattern	[^CDghi45]
$matches	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

Ví dụ 2:

$subject	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
$pattern	[^W-Z]
$matches	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

8. Sự lựa chọn (A|B|C)

Ví dụ 1:

$subject	Monday Tuesday Friday
$pattern	(on|ues|rida)
$matches	Monday Tuesday Friday

Ví dụ 2:

$subject	Monday Tuesday Friday
$pattern	(Mon|Tues|Fri)day
$matches	Monday Tuesday Friday

Ví dụ 3:

$subject	Monday Tuesday Friday
$pattern	..(id|esd|nd)ay
$matches	Monday Tuesday Friday

9. Ký tự * + ?

**: 0 hoặc nhiều lần xuất hiện*
+: 1 hoặc nhiều lần xuất hiện
?: 0 hoặc 1 lần xuất hiện

Ví dụ 1:

$subject	aabc abc bc
$pattern	a*b
$matches	aabc abc bc

Ví dụ 2:

$subject	aabc abc bc
$pattern	a+b
$matches	aabc abc bc

Ví dụ 3:

$subject	aabc abc bc
$pattern	a?b
$matches	aabc abc bc

10. Một số ví dụ về kí hiệu *

Ví dụ 1:

$subject	-@- *** -- "*" -- *** -@-
$pattern	.*
$matches	-@- *** -- "*" -- *** -@-

Ví dụ 2:

$subject	-@- *** -- "*" -- *** -@-
$pattern	-A*-
$matches	-@- *** -- "*" -- *** -@-

Ví dụ 3:

$subject	-@- *** -- "*" -- *** -@-
$pattern	[-@]*
$matches	-@- *** -- "*" -- *** -@-

11. Một số ví dụ về kí hiệu +

Ví dụ 1:

$subject	-@@@- * ** - - "*" -- * ** -@@@-
$pattern	*+
$matches	-@@@- * ** - - "*" -- * ** -@@@-

Ví dụ 2:

$subject	-@@@- * ** - - "*" -- * ** -@@@-
$pattern	-@+-
$matches	-@@@- * ** - - "*" -- * ** -@@@-

Ví dụ 3:

$subject	-@@@- * ** - - "*" -- * ** -@@@-
$pattern	[^]+
$matches	-@@@- * ** - - "*" -- * ** -@@@-

12. Một số ví dụ về kí hiệu ?

Ví dụ 1:

$subject	--XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-
$pattern	-X?XX?X
$matches	--XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-

Ví dụ 2:

$subject	--XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-
$pattern	-@?@?@?-
$matches	--XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-

Ví dụ 3:

$subject	--XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-
$pattern	[^@]@?@
$matches	--XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-

13. Số lần xuất hiện của ký tự {n,m}

Ví dụ 1:

$subject	One ring to bring them all and in the darkness bind them
$pattern	.{5}
$matches	One ring to bring them all and in the darkness bind them

Ví dụ 2:

$subject	One ring to bring them all and in the darkness bind them
$pattern	[els]{1,3}
$matches	One ring to bring them all and in the darkness bind them

Ví dụ 3:

$subject	One ring to bring them all and in the darkness bind them
$pattern	[a-z]{3,}
$matches	One ring to bring them all and in the darkness bind them

14. Ví dụ kết hợp * + ? {n,m}

Ví dụ 1:

$subject	AA ABA ABBA ABBBA
$pattern	AB*A
$matches	AA ABA ABBA ABBBA

Ví dụ 2:

$subject	AA ABA ABBA ABBBA
$pattern	AB{0,}A
$matches	AA ABA ABBA ABBBA

Ví dụ 3:

$subject	AA ABA ABBA ABBBA
$pattern	AB+A
$matches	AA ABA ABBA ABBBA

Ví dụ 4:

$subject	AA ABA ABBA ABBBA
$pattern	AB{1,}A
$matches	AA ABA ABBA ABBBA

Ví dụ 5:

$subject	AA ABA ABBA ABBBA
$pattern	AB?A
$matches	AA ABA ABBA ABBBA

15. Ví dụ kết hợp . * + ?

a. *: 0 hoặc nhiều lần xuất hiện
b. +: 1 hoặc nhiều lần xuất hiện
c. ?: 0 hoặc 1 lần xuất hiện

Ví dụ 1:

$subject	One ring to bring them all and in the darkness bind them
$pattern	r.*
$matches	One ring to bring them all and in the darkness bind them

Ví dụ 2:

$subject	One ring to bring them all and in the darkness bind them
$pattern	r.*?
$matches	One ring to bring them all and in the darkness bind them

Ví dụ 3:

$subject	One ring to bring them all and in the darkness bind them
$pattern	r.+
$matches	One ring to bring them all and in the darkness bind them

Ví dụ 4:

$subject	One ring to bring them all and in the darkness bind them
$pattern	r.+?
$matches	One ring to bring them all and in the darkness bind them

Ví dụ 5:

$subject	One ring to bring them all and in the darkness bind them
$pattern	r.?
$matches	One ring to bring them all and in the darkness bind them

Ví dụ 6:

$subject	One ring to bring them all and in the darkness bind them
$pattern	r.??
$matches	One ring to bring them all and in the darkness bind them

16. Tập hợp các ký tự \w tương đương [A-z0-9_]

a. *: 0 hoặc nhiều lần xuất hiện
b. +: 1 hoặc nhiều lần xuất hiện
c. ?: 0 hoặc 1 lần xuất hiện

Ví dụ 1:

$subject	A1 B2 c3 d_4 e:5 ffGG77--__--
$pattern	\w
$matches	A1 B2 c3 d_4 e:5 ffGG77--__--

Ví dụ 2:

$subject	A1 B2 c3 d_4 e:5 ffGG77--__--
$pattern	\w*
$matches	A1 B2 c3 d_4 e:5 ffGG77--__--

Ví dụ 3:

$subject	A1 B2 c3 d_4 e:5 ffGG77--__--
$pattern	[a-z]\w*
$matches	A1 B2 c3 d_4 e:5 ffGG77--__--

Ví dụ 4:

$subject	A1 B2 c3 d_4 e:5 ffGG77--__--
$pattern	\w{5}
$matches	A1 B2 c3 d_4 e:5 ffGG77--__--

Ví dụ 5:

$subject	A1 B2 c3 d_4 e:5 ffGG77--__--
$pattern	[A-z0-9_]
$matches	A1 B2 c3 d_4 e:5 ffGG77--__--

17. Tập hợp các ký tự \W tương đương [^A-z0-9_] (phủ định của \w)

*: 0 hoặc nhiều lần xuất hiện
+: 1 hoặc nhiều lần xuất hiện
?: 0 hoặc 1 lần xuất hiện

Ví dụ 1:

$subject	AS _34:AS11.23 @#$ %12^*

$pattern	\W
$matches	AS _34:AS11.23 @#$ %12^*

Ví dụ 2:

$subject	AS _34:AS11.23 @#$ %12^*
$pattern	\w
$matches	AS _34:AS11.23 @#$ %12^*

Ví dụ 3:

$subject	AS _34:AS11.23 @#$ %12^*
$pattern	[^A-z0-9_]
$matches	AS _34:AS11.23 @#$ %12^*

18. \s tập hợp những ký tự khoảng trắng - \S tập hợp những ký tự không phải khoảng trắng

*: 0 hoặc nhiều lần xuất hiện
+: 1 hoặc nhiều lần xuất hiện
?: 0 hoặc 1 lần xuất hiện

Ví dụ 1:

$subject	When young was mountain under moon;
$pattern	\s
$matches	When young was mountain under moon;

Ví dụ 2:

$subject	When young was mountain under moon;
$pattern	\S
$matches	When young was mountain under moon;

19. \d tập hợp những ký tự từ 0 đến 9 [0-9] - \D tập hợp không thuộc từ 0-9 [^0-9]

*: 0 hoặc nhiều lần xuất hiện
+: 1 hoặc nhiều lần xuất hiện
?: 0 hoặc 1 lần xuất hiện

Ví dụ 1:

$subject	Page 123; published: 1234 id=12#24@112
$pattern	\d
$matches	Page 123; published: 1234 id=12#24@112

Ví dụ 2:

$subject	Page 123; published: 1234 id=12#24@112
$pattern	\D

$matches	Page 123; published: 1234 id=12#24@112

Ví dụ 3:

$subject	Page 123; published: 1234 id=12#24@112
$pattern	[0-9]
$matches	Page 123; published: 1234 id=12#24@112

20. \A tìm từ đầu chuỗi nguồn - \A tương đương với dấu ^; \Z tìm từ cuối chuỗi nguồn -\Z với dấu $

Ví dụ 1:

$subject	Ere iron was found or tree was hewn.
$pattern	\A…
$matches	Ere iron was found or tree was hewn.

Ví dụ 2:

$subject	Ere iron was found or tree was hewn.
$pattern	…\Z
$matches	Ere iron was found or tree was hewn.

21. (?=<patter>)

Ví dụ 1:

$subject	AAAX---aaax---111
$pattern	\w+(?=X)
$matches	AAAX---aaax---111

Ví dụ 2:

$subject	AAAX---aaax---111
$pattern	\w+
$matches	AAAX---aaax---111

22. (?!<patter>)

Ví dụ 1:

$subject	AAAX---AAA
$pattern	AAA(?!X)
$matches	AAAX---AAA

Ví dụ 2:

$subject	AAAX---AAA
$pattern	AAA
$matches	AAAX---AAA

23. (?<=<patter>)

Ví dụ 1:

$subject	This is a title
$pattern	(?<=<p class\=\"Title\">).*(?=)
$matches	This is a title

24. \b vị trí ở biên của một từ

Ví dụ 1:

$subject	Ere iron was found or tree was hewn, When young was mountain under moon;
$pattern	\b.
$matches	Ere iron was found or tree was hewn, When young was mountain under moon;

Ví dụ 2:

$subject	Ere iron was found or tree was hewn, When young was mountain under moon;
$pattern	.\b
$matches	Ere iron was found or tree was hewn, When young was mountain under moon;

Ví dụ 3:

$subject	The captain wore his cap and cape proudly as he sat listening to the recap of how his crew saved the men from a capsized vessel.
$pattern	\bcap
$matches	The captain wore his cap and cape proudly as he sat listening to the recap of how his crew saved the men from a capsized vessel.

Ví dụ 4:

$subject	The captain wore his cap and cape proudly as he sat listening to the recap of how his crew saved the men from a capsized vessel.
$pattern	\bcap
$matches	The captain wore his cap and cape proudly as he sat listening to the recap of how his crew saved the men from a capsized vessel.

25. \B vị trí không ở biên của một từ

Ví dụ 1:

$subject	Ere iron was found or tree was hewn, When young was mountain under moon;
$pattern	\B.
$matches	Ere iron was found or tree was hewn, When young was mountain under moon;

Ví dụ 2:

$subject	Ere iron was found or tree was hewn, When young was mountain under moon;
$pattern	.\B
$matches	Ere iron was found or tree was hewn, When young was mountain under moon;

Special Sequences (Chuỗi đặt biệt)	
\w	- Những ký tự bình thường (a-z 0-9 _)
\W	- Những ký tự khác những ký tự bình thường.
\s	- Ký tự khoảng trắng (space, tab CRLF)
\S	- Những ký tự khác các ký tự khoảng trắng
\d	- Số (0-9)
\D	- Những ký tự không phải là số
.	- Những ký tự bất kỳ ngoại trừ ký tự xuống dòng

Meta Characters (Ký tự meta)	
^	- Bắt đầu
$	- Kết thúc
[	- Bắt đầu của một tập hợp
]	- Kết thúc của một tập hợp
|	- Dấu phân chia (a|b) matches a or b
(	- Bắt đầu mẫu phụ (subpattern)
)	- Kết thúc mẫu phụ (subpattern)
****	- Escape character

Quantifiers (Số lượng)	
n*	- 0 đến n lần xuất hiện
n+	- 1 đến n lần xuất hiện
n?	- 0 đến 1 lần xuất hiện
{n}	- số lần xuất hiện
{n,}	- Xuất hiện ít nhấn n lần (n: 0~n)
{n,m}	- Xuất hiện trong khoảng từ n đến m lần

Point based assertions (Điểm xuất hiện)	
\b	- Vị trí biên của một từ
\B	- Không phải vị trí biên của một từ
\A	- Vị trí đầu của một chủ đề
\Z	- Vị trí kết thúc hoặc xuống dòng của một chủ đề
\z	- Vị trí cuối của một chủ đề

C. Một số ứng dụng trong thực tế của Regular expressions

1. Kiểm tra giá trị đầu vào.

Để kiểm tra một giá trị nào đó có đúng với những theo yêu cầu hay không thì Regular Expression được xem là một sự lựa chọn tốt nhất. Nhưng trước khi viết một biểu thức Regular Expression chúng ta cần xác định điều kiện kiểm tra

Ví dụ 1: Kiểm tra địa chỉ email với những điều kiện cho trước

- Địa chỉ email phải bắt đầu là một ký tự
- Độ dài tối thiểu của email là 3 ký tự và độ dài tối đa là 32 ký tự
- Địa chỉ email là tập hợp các ký tự a-z, từ 0-9 và có thể có các ký tự như dấu chấm (.), dấu gạch dưới (_)
- Tên miền của email có thể là tên miền cấp 1 (vd: zend.vn) hoặc cấp 2 (vd: zend.com.vn)

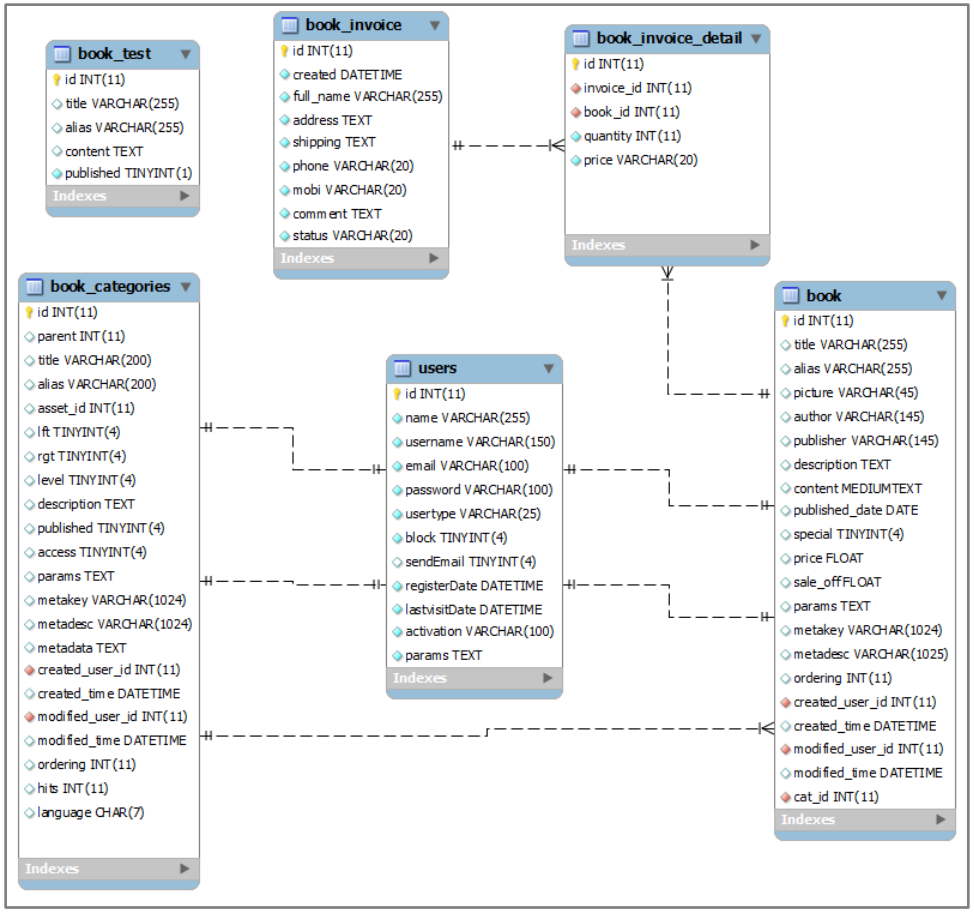
Ví dụ: Kiểm tra địa chỉ email

```
<?php
	$subject = "zendvn@yahoo.com";
	$pattern = '#^[a-z][a-z0-9_\.]{2,}@[a-z0-9]{3,}(\.[a-z]{2,4}){1,2}$#';
	if(preg_match_all($pattern, $subject, $matches)==1){
		echo 'Đây là địa chỉ email';
	}
?>
```

Chương 5
Phân tích & xây dựng hệ thống Back-End

Cấu trúc database

Sơ đồ quan hệ của các bảng trong Component

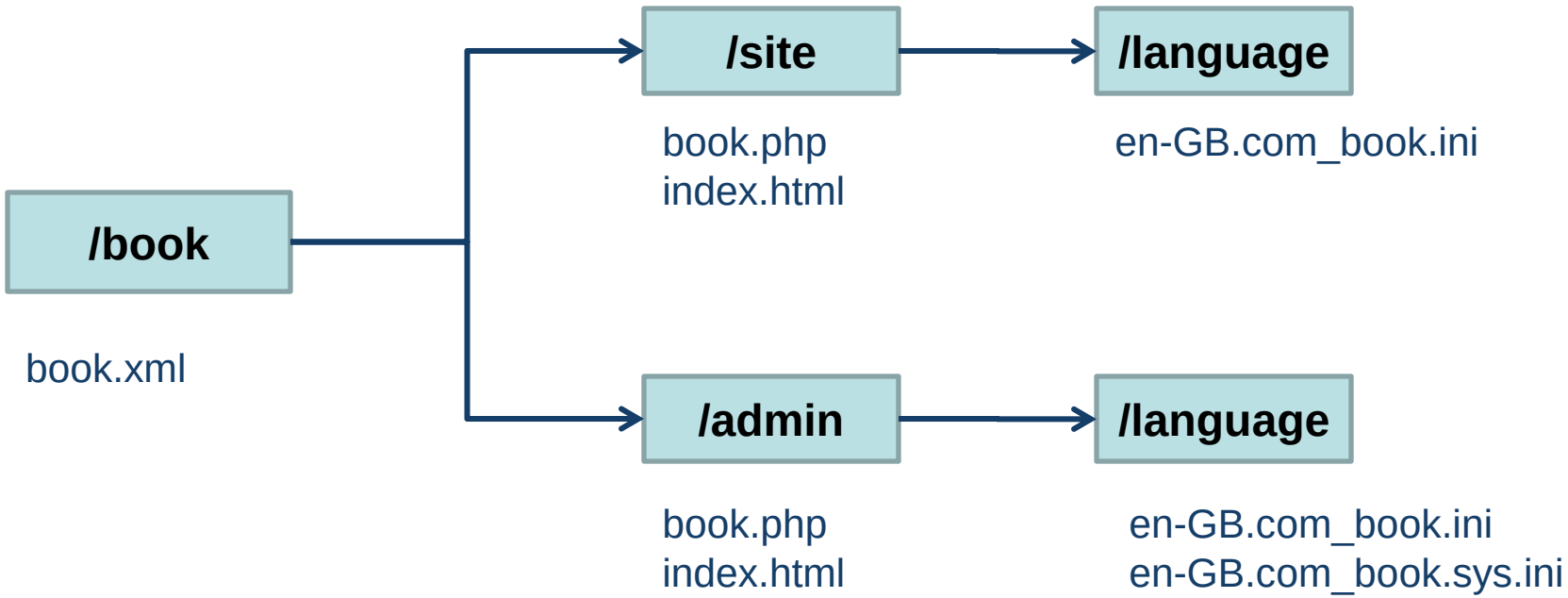


Cấu trúc database

Thông tin các bảng trong database

#__book
#__book_categories
#__book_test
#__book_invoice
#__book_invoice_detail

Tạo cấu trúc thư mục



Nội dung tập tin book.xml

Cấu trúc tổng quan của tập tin

```
<?xml version="1.0" encoding="utf-8"?>
<install type="component" method="upgrade"
version="2.5">

……….……

</install>
```

Nội dung tập tin book.xml

Khai báo thông tin của Component

```
<name>com_book</name>
<author>KhanhPham (www.zend.vn)</author>
<creationDate>02/11/2012</creationDate>
<authorEmail></authorEmail>
<authorUrl>www.zend.vn</authorUrl>
<copyright>KhanhPham</copyright>
<license>GNU/GPL</license>
<version>1.0</version>
<description>Book Shopping</description>
```

Nội dung tập tin book.xml

Khai báo thông tin của FrontEnd

- Cấu trúc thư mục
- Tập tin ngôn ngữ

Nội dung tập tin book.xml

Khai báo thông tin của BackEnd

- Cấu trúc menu
- Cấu trúc thư mục
- Tập tin ngôn ngữ

Nội dung tập tin ngôn ngữ

Cấu trúc định dạng ngôn ngữ

BackEnd
- en-GB.com_book.ini
- en-GB.com_book.sys.ini

FrontEnd
- en-GB.com_book.ini

Nội dung tập tin ngôn ngữ

Cấu trúc định dạng ngôn ngữ

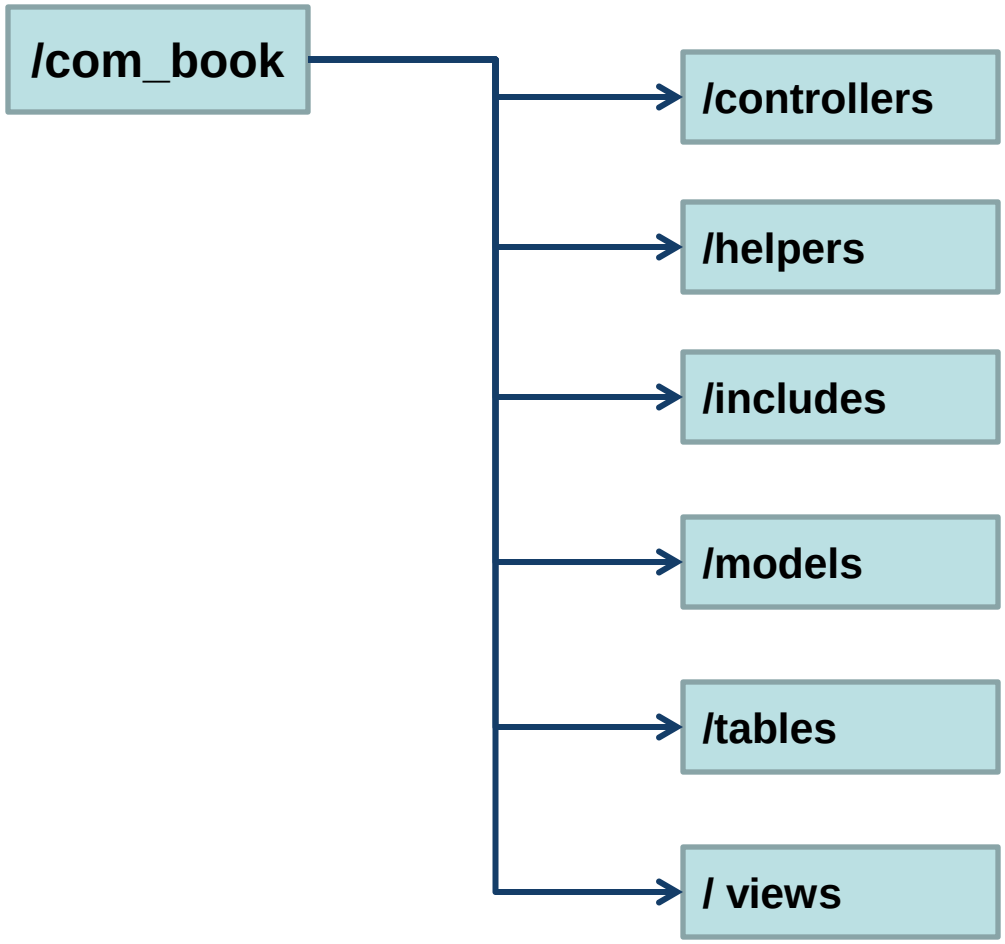
KEY = “Value”

- Luôn phải để 1 dòng trống đầu tập tin
- KEY luôn là chữ HOA
- KEY nối các từ bằng dấu gạch dưới

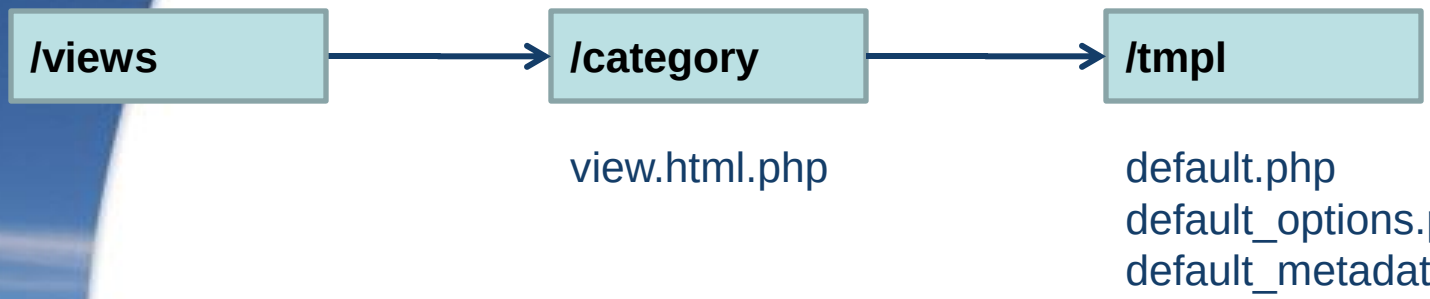
Đóng gói và cài đặt

- Nén thư mục book dưới dạng file .zip
- Sử dụng công cụ cài đạt Component

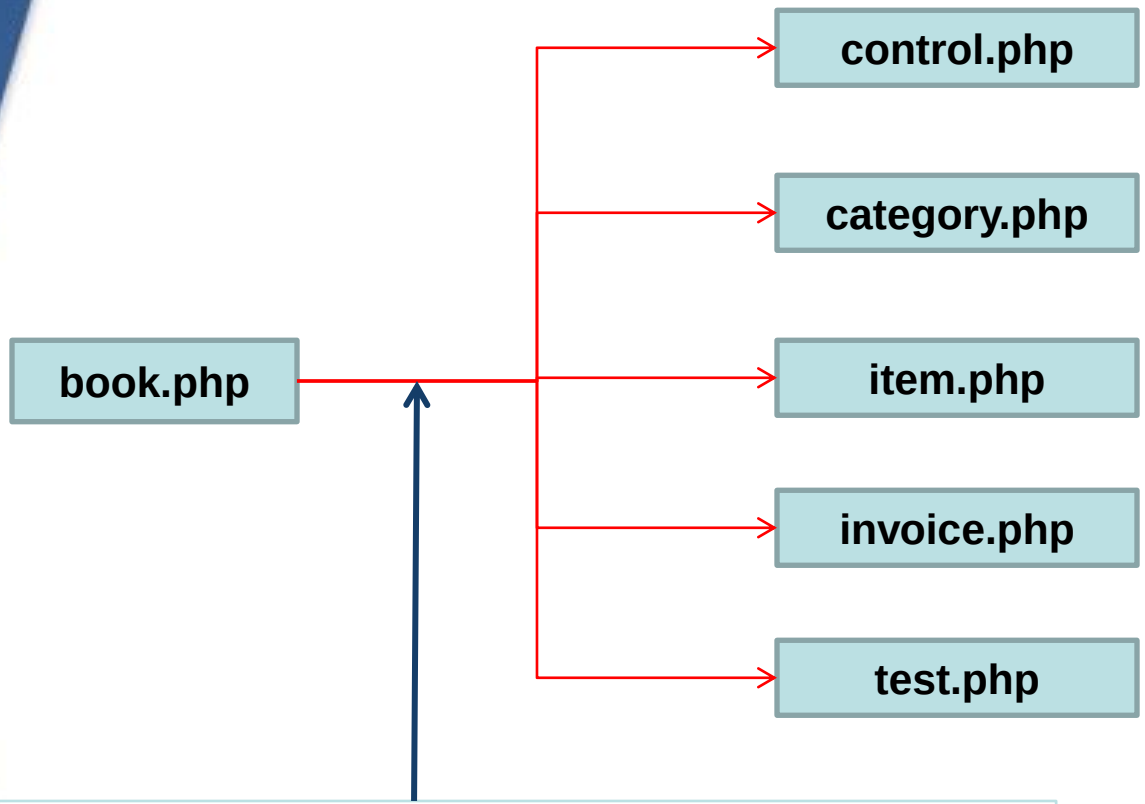
Xây dựng cấu trúc MVC



Cấu trúc thư mục VIEW

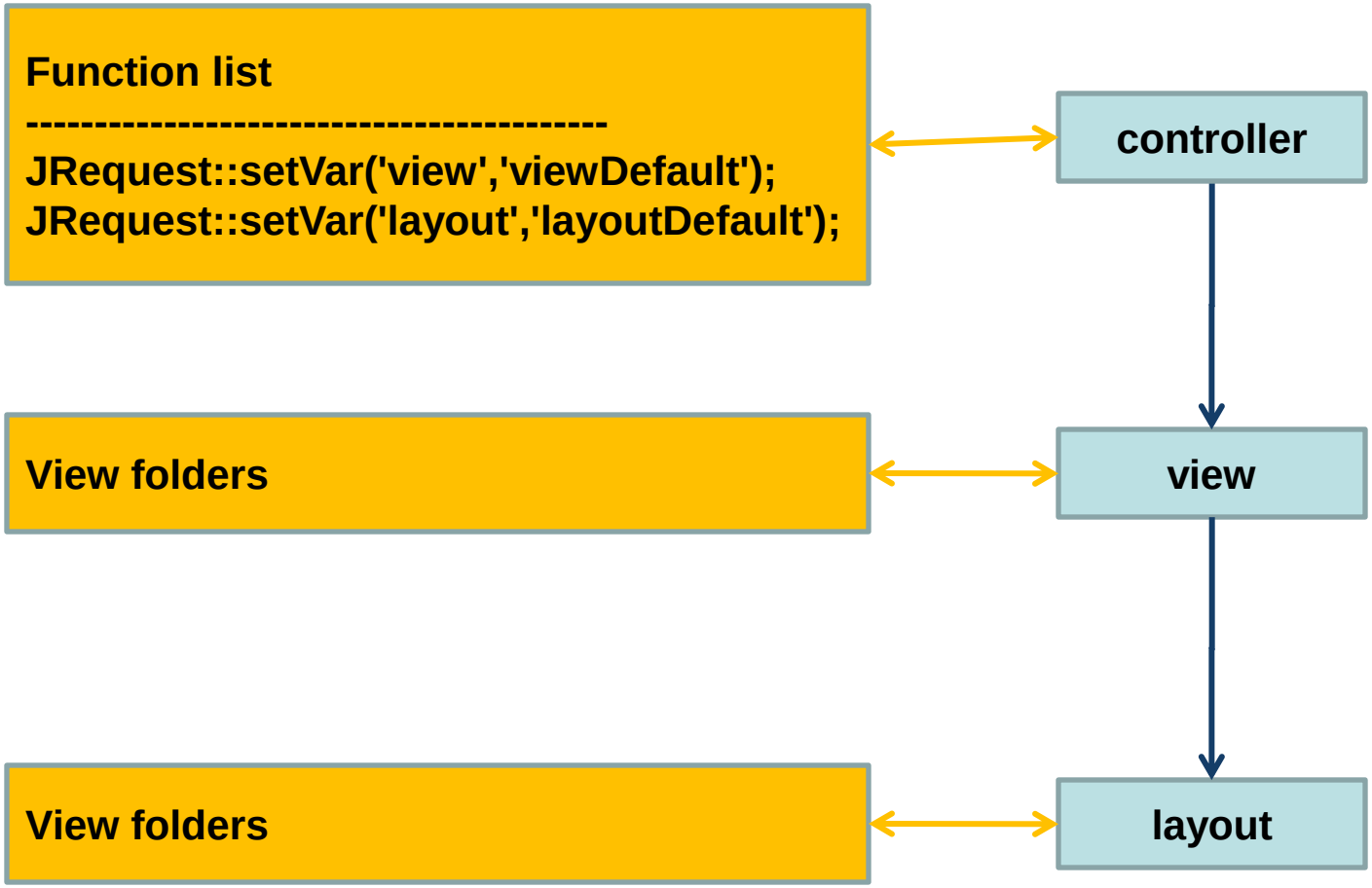


Các Controller chính



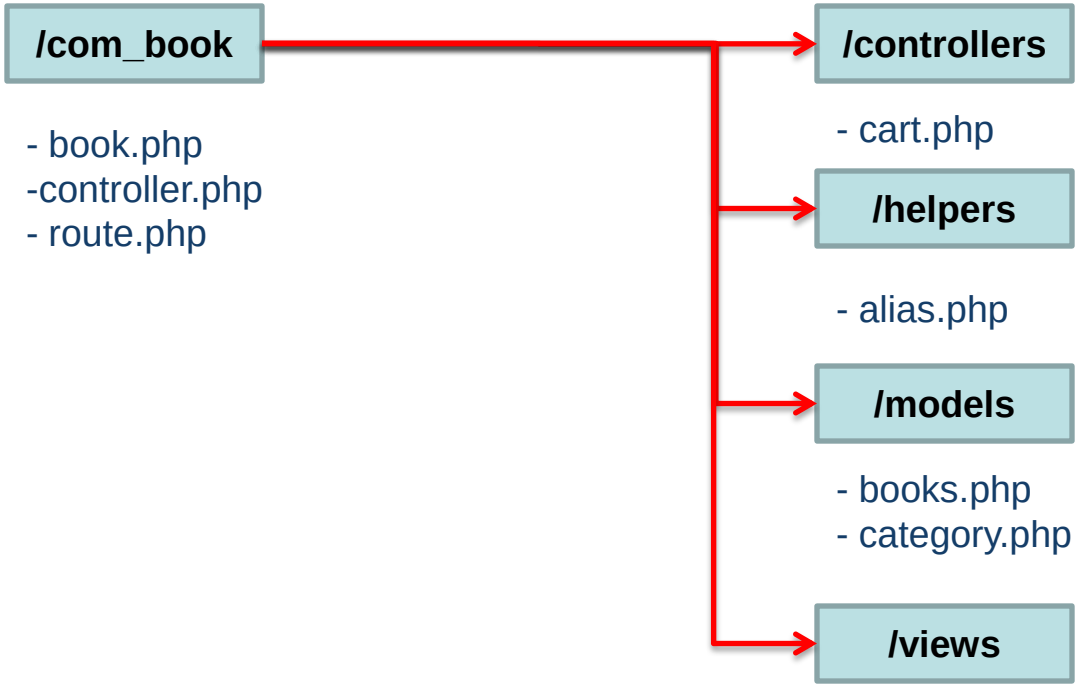

```
$controller = JRequest::getCmd('controller', 'control');
```

Điều hướng Task – View - Layout

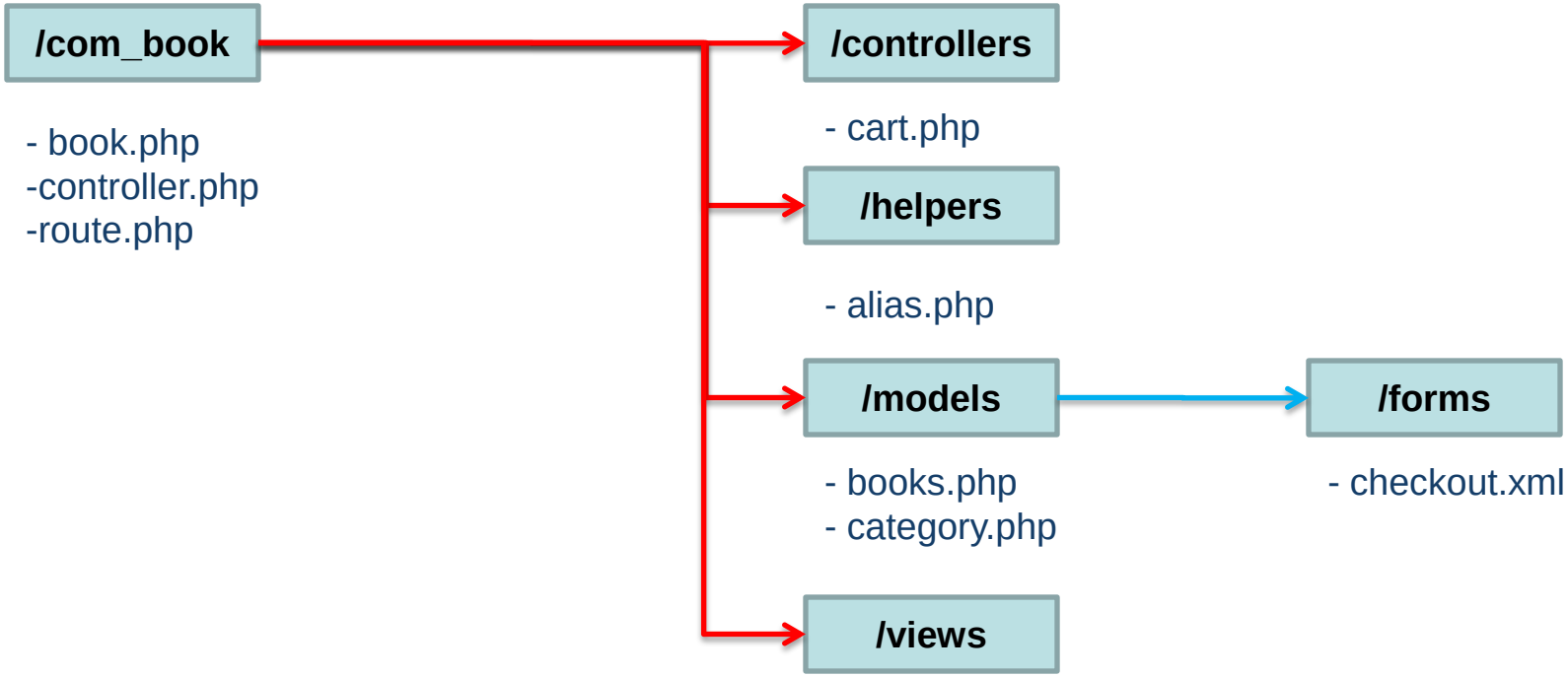


Chương 8
Xây dựng Front-End MVC

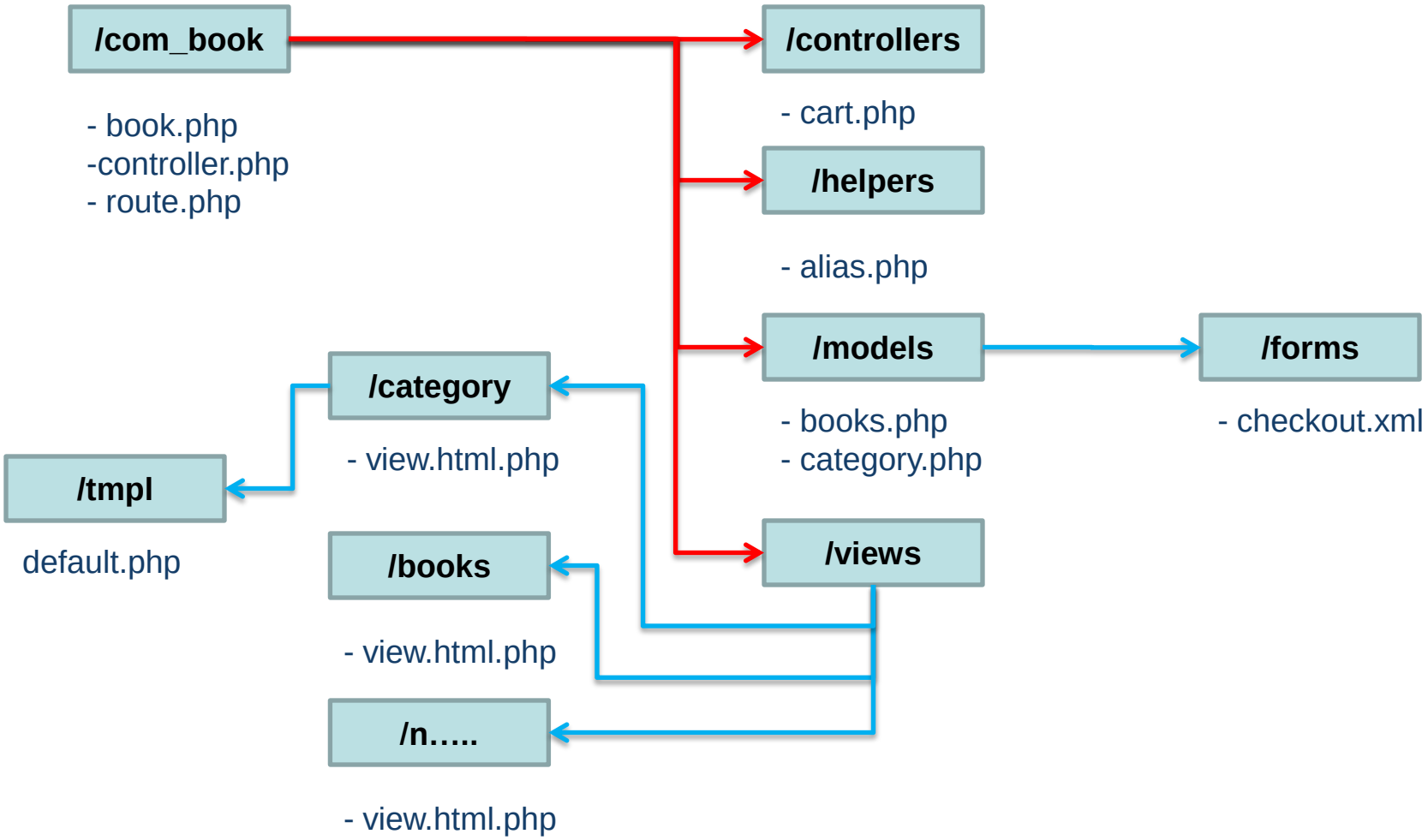
Mô hình Front-End MVC



Mô hình Front-End MVC



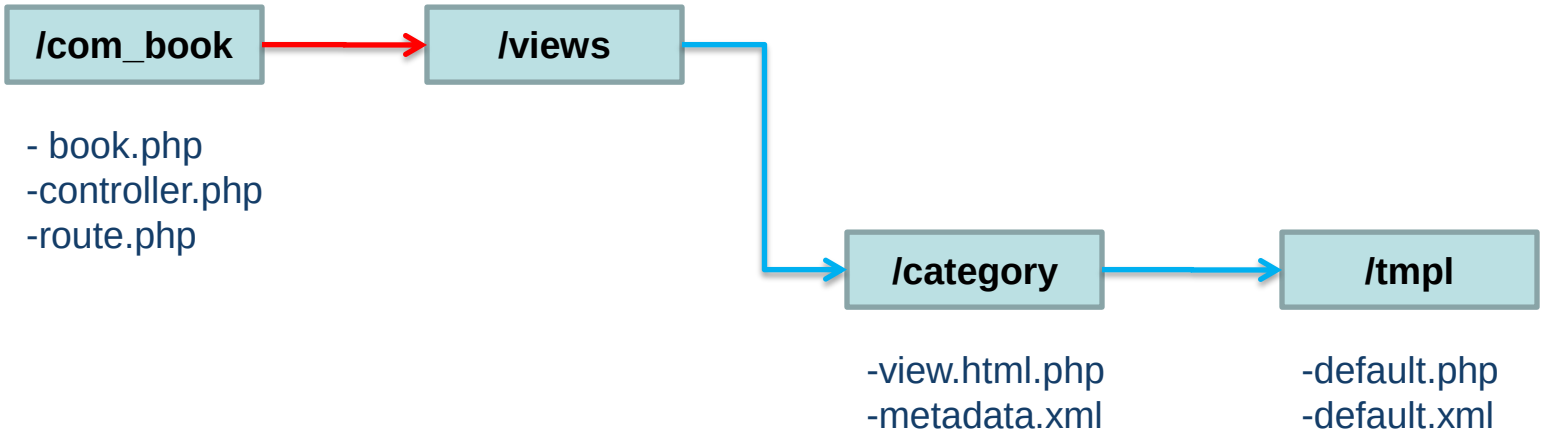
Mô hình Front-End MVC



Chức năng chính ở Front-End

- Hiển thị các category book
- Hiển thị các cuốn sách trong một category
- Hiển thị tất cả các cuốn sách
- Hiển thị chi tiết một cuốn sách
- Hiển thị những cuốn sách đặc biệt
- Hiển thị những cuốn sách đang giảm giá
- Hiển thị giỏ hàng
- Đặt hàng

Cấu trúc thư mục chứa VIEW



Nội dung tập tin book.xml

Khai báo thông tin của Component

```
<name>com_book</name>
<author>KhanhPham (www.zend.vn)</author>
<creationDate>02/11/2012</creationDate>
<authorEmail>vukhanh2212@gmail.com</authorEmail>
<authorUrl>www.zend.vn</authorUrl>
<copyright>KhanhPham</copyright>
<license>GNU/GPL</license>
<version>1.0</version>
<description>Book Shopping</description>
```

Nội dung tập tin book.xml

Khai báo thông tin của FrontEnd

- Cấu trúc thư mục
- Tập tin ngôn ngữ

Nội dung tập tin book.xml

Khai báo thông tin của BackEnd

- Cấu trúc menu
- Cấu trúc thư mục
- Tập tin ngôn ngữ

Nội dung tập tin ngôn ngữ

Cấu trúc định dạng ngôn ngữ

BackEnd
- en-GB.com_book.ini
- en-GB.com_book.sys.ini

FrontEnd
- en-GB.com_book.ini

Nội dung tập tin ngôn ngữ

Cấu trúc định dạng ngôn ngữ

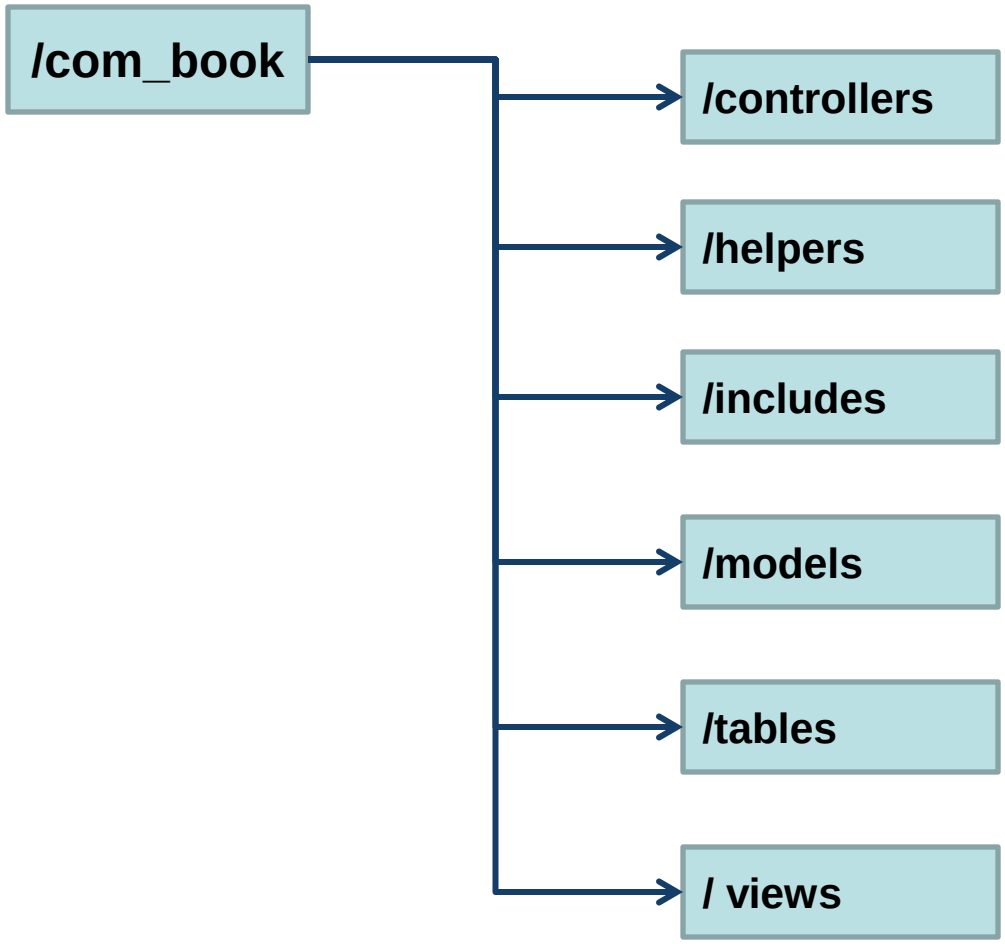
KEY = “Value”

- Luôn phải để 1 dòng trống đầu tập tin
- KEY luôn là chữ HOA
- KEY nối các từ bằng dấu gạch dưới

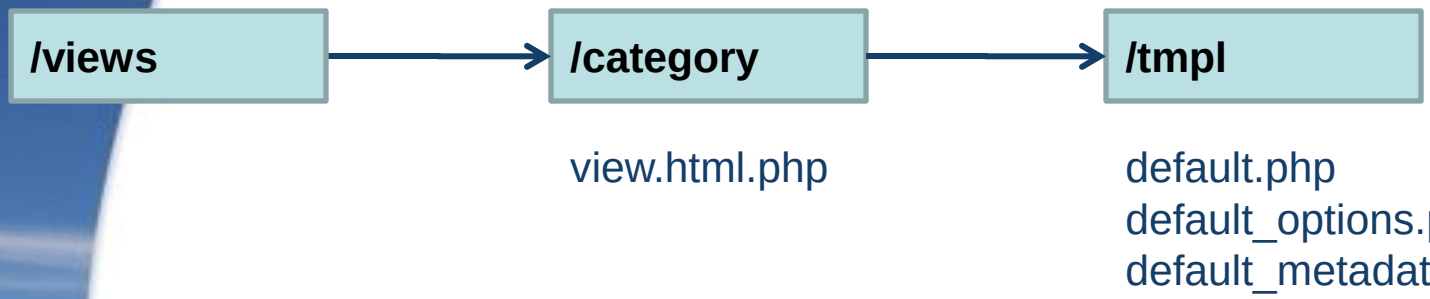
Đóng gói và cài đặt

- Nén thư mục book dưới dạng file .zip
- Sử dụng công cụ cài đạt Component

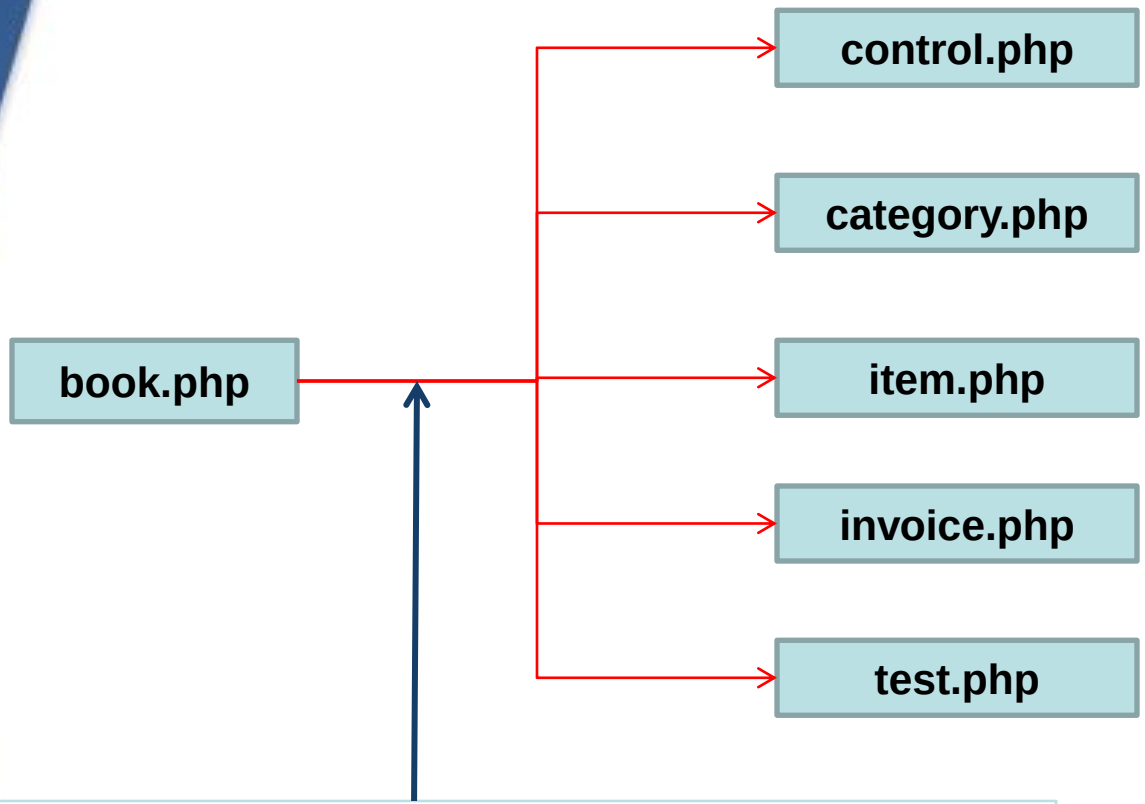
Xây dựng cấu trúc MVC



Cấu trúc thư mục VIEW

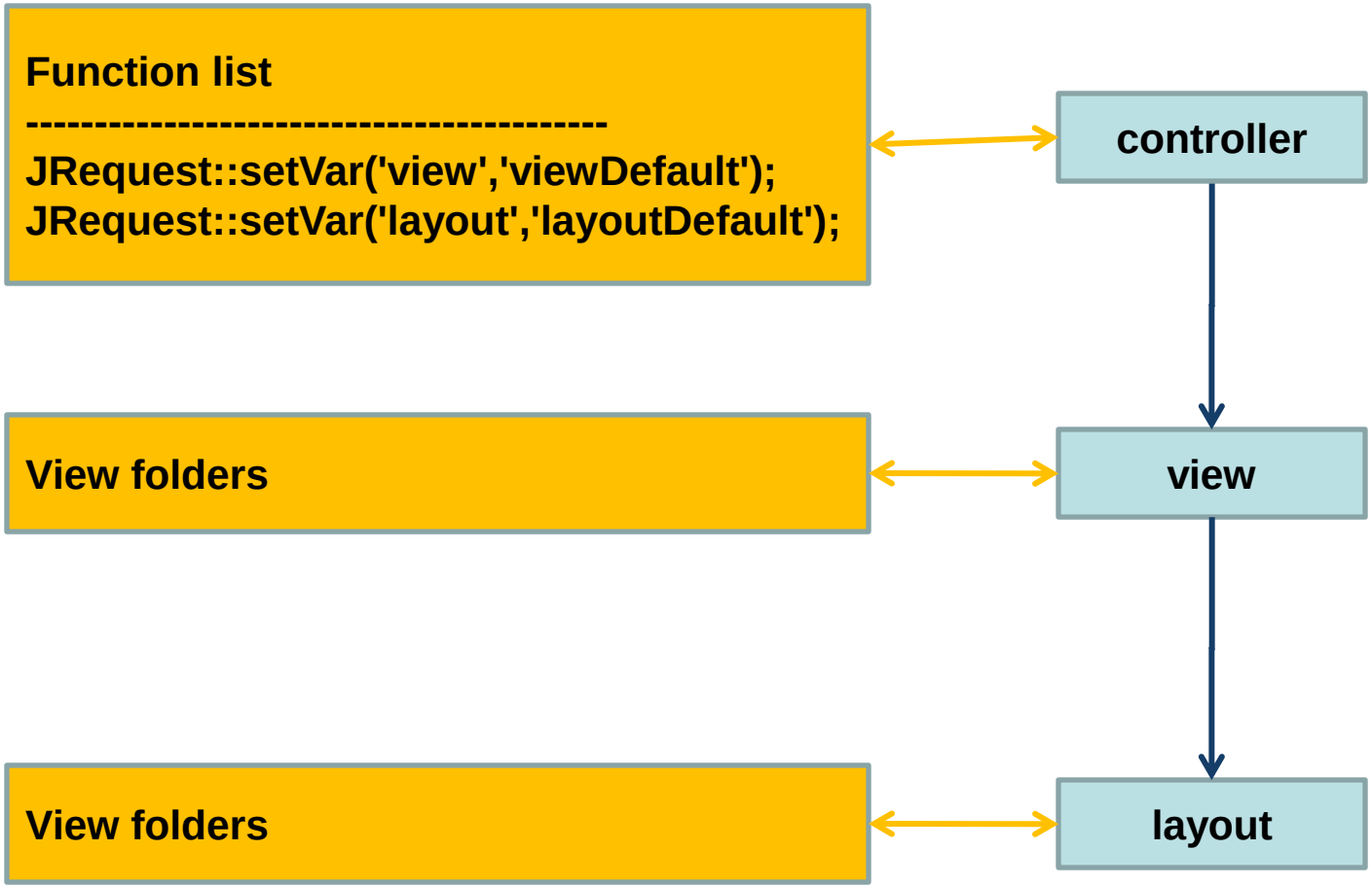


Các Controller chính

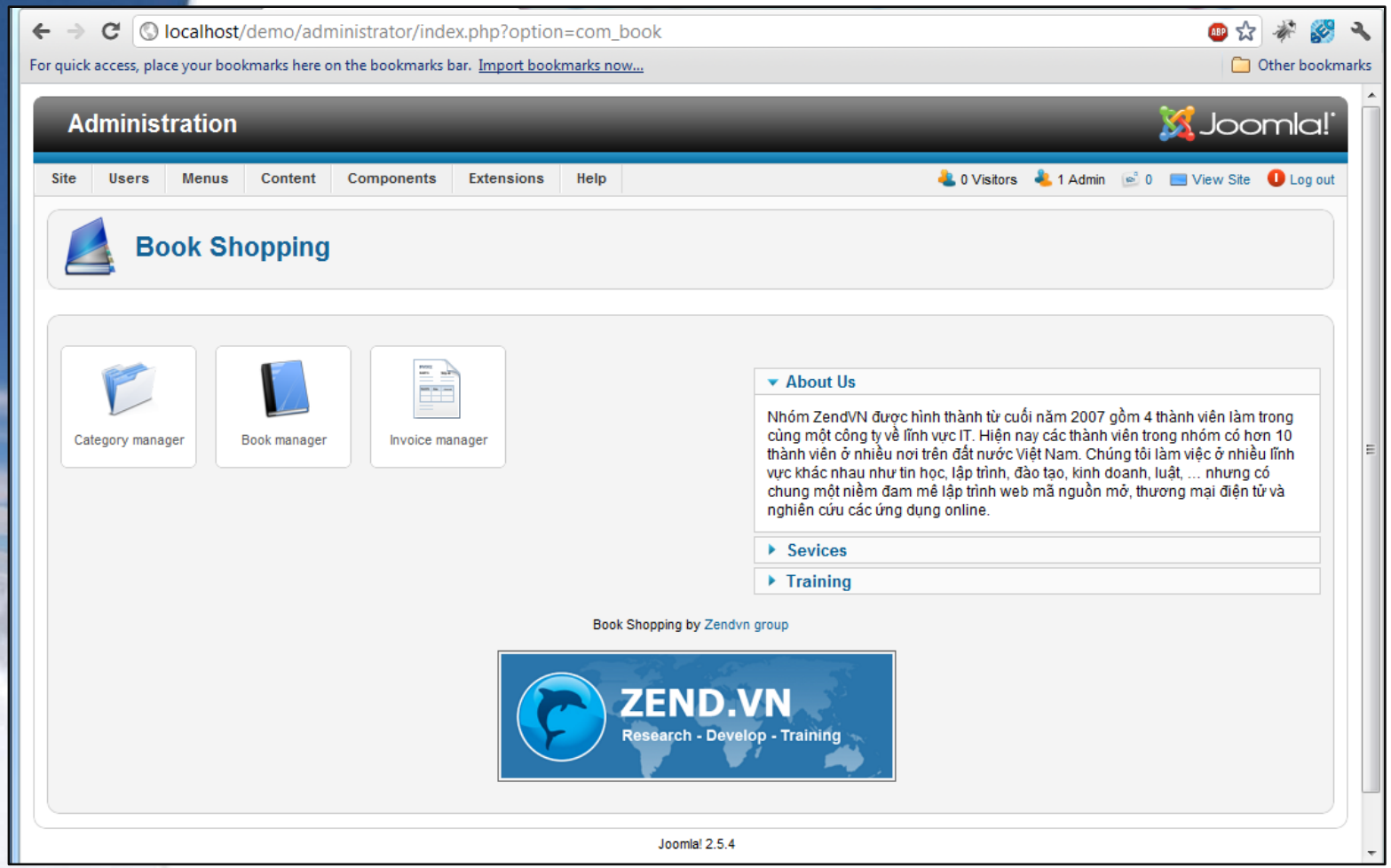



```
$controller = JRequest::getCmd('controller', 'control');
```

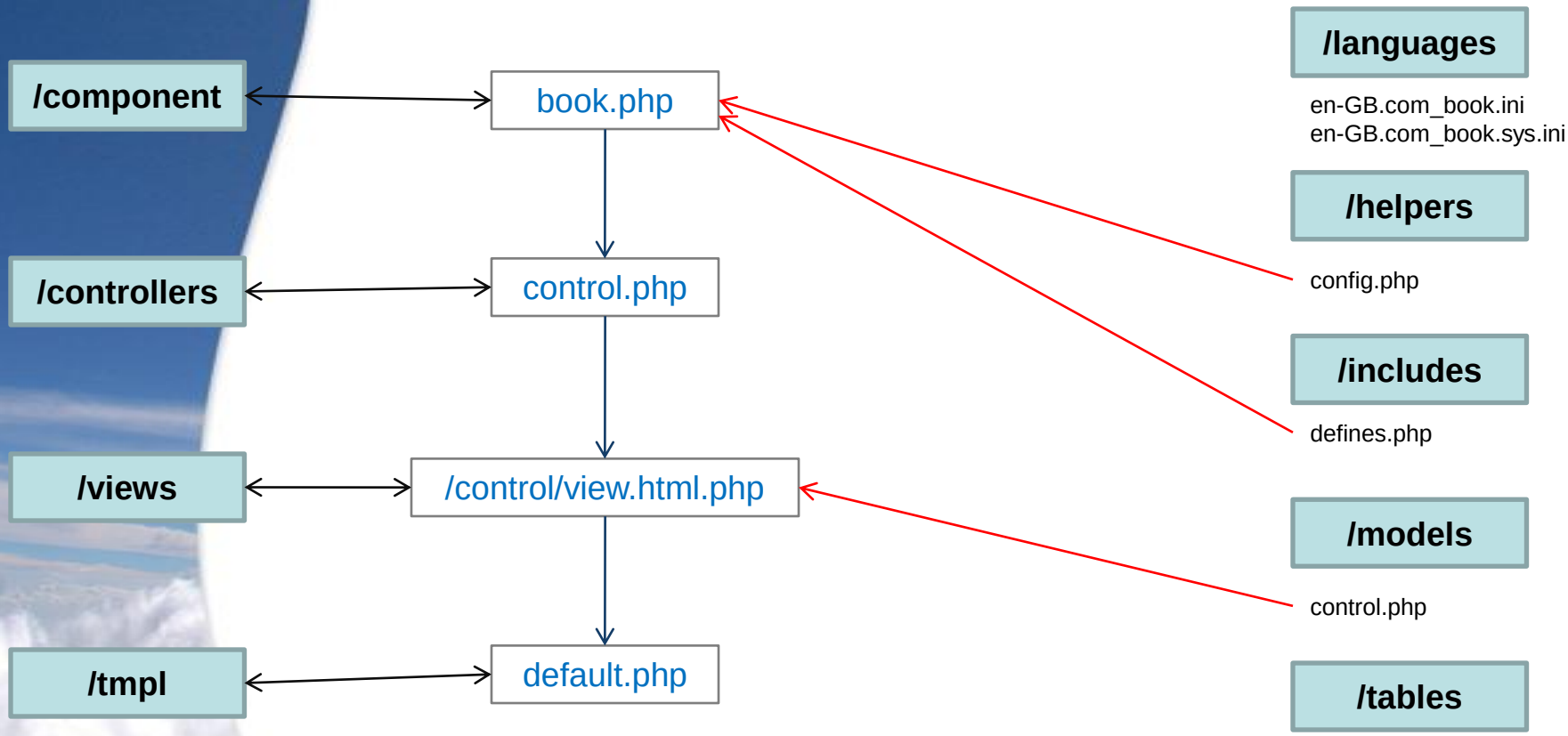
Điều hướng Task – View - Layout



Xây dựng bảng điều khiển



Xây dựng bảng điều khiển



Tương tác với database

1. Chuẩn bị

Tạo phương thức mới trong BookControllerTest là database()

```
function database(){
	echo '<br>' . __METHOD__;

	parent::display();
}
```

Tạo tập tin MODEL là /models/database.php với nội dung:

```
<?php

defined ( '_JEXEC' ) or die ( 'Direct Access to this location is not allowed.' );
jimport ( 'joomla.application.component.model' );

class BookModelDatabase extends JModel {

	var $_data;

	function __construct($config = array()) {
		parent::__construct ( $config );
	}

	function hello(){
		echo '<br>Hello Model';
	}


}
```

Tạo tập tin VIEW là /views/database/view.html.php

```
<?php
defined('_JEXEC') or die;

jimport('joomla.application.component.view');

class BookViewDatabase extends JView{

	function display($tpl = null){
		echo '<br>' . __METHOD__;

		parent::display($tpl);
	}

}
```

2. Tương tác giữa Model và Controller

Để gọi một MODEL vào CONTROLLER chúng ta sử dụng phương thức getModel().

```
$this->getModel($name = '', $prefix = '', $config = array())
```

Ví dụ: Gọi MODEL BookModelDatabase và BookModelControl

```
function database(){
	echo '<br>' . __METHOD__;

	JRequest::setVar('view','database');

	$modelDatabase = $this->getModel('Database');
	echo '<pre>';
	print_r($modelDatabase);
	echo '</pre>';

	$modelControl = $this->getModel('Control');
	echo '<pre>';
	print_r($modelDatabase);
	echo '</pre>';

	parent::display();
}
```

Để gọi một phương thức của MODEL vào CONTROLLER:
- Khởi tạo MODEL trong CONTROLLER
- Gọi đến phương thức trong MODEL

```
function database(){
	echo '<br>' . __METHOD__;
	JRequest::setVar('view','database');
	$model = $this->getModel('Database');
	$model->hello();

	parent::display();
}
```

3. Gọi Model vào VIEW

Để gọi một MODEL vào CONTROLLER chúng ta sử dụng phương thức getModel()

```
<?php
defined('_JEXEC') or die;

jimport('joomla.application.component.view');

class BookViewDatabase extends JView{

	function display($tpl = null){
		echo '<br>' . __METHOD__;

		$model = $this->getModel();

		parent::display($tpl);
	}
```

```
}
```

Gọi phương thức của MODEL vào VIEW

Cách 1: Gọi gián tiếp

```
<?php
defined('_JEXEC') or die;

jimport('joomla.application.component.view');

class BookViewDatabase extends JView{

	function display($tpl = null){
		echo '<br>' . __METHOD__;

		$model = $this->getModel();
		$model->hello();

		parent::display($tpl);
	}

}
```

Cách 2: Gọi đến các phương thức trong MODEL có từ "get" đứng đầu

Ví dụ trong MODEL BookViewDatabase() chúng ta có phương thức getAbc() với nội dung sau:

```
function getAbc(){
	echo '<br>' . __METHOD__;
}
```

Chúng ta có thể gọi phương thức getAbc() này thông qua phương thức get() của VIEW

```
<?php
defined('_JEXEC') or die;

jimport('joomla.application.component.view');

class BookViewDatabase extends JView{

	function display($tpl = null){
		echo '<br>' . __METHOD__;

		$this->get('abc');

		parent::display($tpl);
	}

}
```

4. Gọi nhiều MODEL vào sử dụng trong VIEW

Để có thể gọi được nhiều tập tin MODEL vào trong VIEW chúng ta phải thiết lập việc sử dụng các lớp MODEL này trong CONTROLLER trước, sau đó chúng ta mới có thể dùng chúng trong VIEW

Tạo một MODEL mới có tên data.php với nội dung như sau:

```
<?php

defined ( '_JEXEC' ) or die ( 'Direct Access to this location is not allowed.' );
jimport ( 'joomla.application.component.model' );

class BookModelData extends JModel {

	var $_data;

	function __construct($config = array()) {
		parent::__construct ( $config );
	}

	function hello(){
		echo '<br>Hello Model';
	}

	function getAbc(){
		echo '<br>' . __METHOD__;
	}
}
```

Ví dụ: Gọi 2 lớp MODEL vào sử dụng trong BookViewDatabase.

```
Tại tập tin
<?php
	defined('_JEXEC') or die;

	jimport('joomla.application.component.controller');

	class BookControllerTest extends JController{

		// Old code goes here
	function database(){
		echo '<br>' . __METHOD__;
		JRequest::setVar('view','database');

		$view = $this->getView('Database','html');
		$view->setModel($this->getModel('Data'));
		$view->setModel($this->getModel('Database'),true);
		$view->display();
	}

}
```

Giải thích:

```
$view = $this->getView('Database','html');
```

Lấy ra đối tượng JDocumentHTML của lớp BookViewDatabase

```
$view->setModel($this->getModel('Data'));
```

Thiết lập đối tượng BookModelData cho BookViewDatabase

```
$view->setModel($this->getModel('Database'),true);
```

Thiết lập đối tượng BookModelDatabase cho BookViewDatabase. Tham số thứ 2 bằng 'true' nghĩa là lớp MODEL đó là mặc định.

```
$view->display();
```

Thiết lập lại giá trị VIEW mới, lớp VIEW này sẽ quản lý 2 lớp MODEL

Sử dụng 2 đối tượng MODEL trong BookViewDatabase

```
<?php
defined('_JEXEC') or die;

jimport('joomla.application.component.view');

class BookViewDatabase extends JView{

	function display($tpl = null){
		echo '<br>' . __METHOD__;

		$model1 = $this->getModel();
		echo '<pre>';
		print_r($model1);
		echo '</pre>';

		$model2 = $this->getModel('Data');
		echo '<pre>';
		print_r($model2);
		echo '</pre>';

		parent::display($tpl);
	}

}
```

Cách khác gọi trực tiếp MODEL vào VIEW
Trong CONTROLLER (BookControllerTest)

```
function database(){
	echo '<br>' . __METHOD__;
	JRequest::setVar('view','database');

	parent::display();
}
```

Trong VIEW (BookViewDatabase)

```
$dataModel = JModel::getInstance('Data','BookModel');
echo '<pre>';
print_r($dataModel);
echo '</pre>';
```

5. Đối tượng JModel

JModel là một đối tượng quản lý chung về database trong Joomla. Ngoài JModel còn nhiều lớp Model khác kế thừa từ JModel như

- JModelForm: Thực hiện để load dữ liệu đưa vào Form
- JModelList: thực hiện trong các thao tác hiển thị danh sách dữ liệu
- JModelItem: thực hiện thao tác lấy ra thông tin của 1 Item khi biết ID của nó

- **Khởi tạo đối tượng**

```
$model = JModel::getInstance('<FileName>','<ComponentName>Model');
```

Với phương thức này chúng ta có thể khởi tạo MODEL của Component bất kỳ nơi đâu.

Ví dụ: Khởi tạo MODEL trong VIEW

```
$model = JModel::getInstance('Data','BookModel');
echo '<pre>';
print_r($model);
echo '</pre>';
```

- **Đếm số record trong câu truy vấn**

```
$this->_getListCount($query);
```

Ví dụ: Đếm số record trong bảng book_test của model BookModelDatabase

Trong MODEL

```
function _buildQuery(){
	$query = 'SELECT title FROM '. _JOOM_TABLE_BOOK_TEST . ' WHERE published = 0';
	return $query;
}

function totalItem(){
	$query = $this->_buildQuery();
	$total = $this->_getListCount($query);
	return $total;
}
```

Trong VIEW

```
function display($tpl = null){
	echo '<br>' . __METHOD__;

	$model = $this->getModel();
	$model->totalItem();

	parent::display($tpl);
}
```

- Lấy kết quả từ database trong một câu truy vấn

```
$this->_getList($query, $limitstart, $limit)
```

Ví dụ: Hiển thị kết quả 3 cuốn sách đầu tiên trong database

Trong MODEL

```
function _buildQuery(){
	$query = 'SELECT title FROM '. _JOOM_TABLE_BOOK_TEST . ' WHERE published = 0';
	return $query;
}

function getData(){
```

```
        $query = $this->_buildQuery();
        $result = $this->_getList($query,0,3);
        return $result;
}
```

Trong VIEW

```
function display($tpl = null){
        echo '<br>' . __METHOD__;

        $model = $this->getModel();
        $items = $model->getData();
        //OR $items = $this->get('Data');

        echo '<pre>';
        print_r($items);
        echo '</pre>';

        parent::display($tpl);
}
```

- **Thiết lập giá trị biến cho lớp**

```
$this->setState($property, $value = null)
```

Ví dụ: Thiết lập các giá trị limit cho phân trang

Trong MODEL

```
function __construct($config = array()) {
        parent::__construct ( $config );
        $this->setState('limitstart',2);
        $this->setState('limit',0);
}
```

- **Lấy giá trị các biến đã được thiết lập trong lớp**

```
$this->getState($property = null, $default = null)
```

Ví dụ: lấy giá trị đưa vào phương thức _getList()

```
function getData(){
        $query = $this->_buildQuery();
        //$result = $this->_getList($query,0,3);
        $result = $this->_getList($query,$this->getState('limit'),$this->getState('limitstart'));
        return $result;
}
```

6. Đối tượng JDatabaseQueryMySQL

Đối tượng này là đối tượng để xây dựng những câu truy vấn cho ứng dụng. Khi sử dụng tốt các phương thức của lớp này sẽ tránh được những lỗi trong quá trình viết truy vấn

- Khởi tạo đối tượng JDatabaseQueryMySQL

Cách 1: Trong MODEL

```
$db =  $this->getDbo();
```

```
$select = $db->getQuery(true); //JDatabaseQueryMySQL
```

Cách 2: Khởi tạo trực tiếp

```
$test = new JDatabaseQueryMySQL();
```

Ví dụ 1: Truy xuất dữ liệu trong bảng #__book_test theo cách 1

Trong MODEL

```
function _buildQuery2(){
	$db =  $this->getDbo();
	$select = $db->getQuery(true)
			->select(array('id','title'))
			->from(_JOOM_TABLE_BOOK_TEST);

	$db->setQuery($testQuery);
	$result = $db->loadAssocList();

	echo '<pre>';
	print_r($result);
	echo '</pre>';
}
```

Trong VIEW

```
function display($tpl = null){
	echo '<br>' . __METHOD__;

	$model = $this->getModel();
	$model->_buildQuery2();

	parent::display($tpl);
}
```

Ví dụ 2: Truy xuất dữ liệu trong bảng #__book_test theo cách 2

Trong VIEW

```
function display($tpl = null){
	echo '<br>' . __METHOD__;

	$model = $this->getModel();

	$db =  $model->getDbo();

	$queryObj = new JDatabaseQueryMySQL();
	$query = $queryObj->select(array('id','title'))
			->from(_JOOM_TABLE_BOOK_TEST);

	$db->setQuery($query);
	$result = $db->loadAssocList();

	echo '<pre>';
	print_r($result);
	echo '</pre>';

	parent::display($tpl);
}
```

Chúng ta cũng có thể đưa cấu truy vấn bình thường vào trong phương thức setQuery() để thực hiện truy vấn kết quả

```
function display($tpl = null){
	echo '<br>' . __METHOD__;

	$model = $this->getModel();

	$db =  $model->getDbo();

	$query = 'SELECT id,title FROM ' . _JOOM_TABLE_BOOK_TEST;
	echo '<br>' . $query;
	$db->setQuery($query);
	$result = $db->loadAssocList();

	echo '<pre>';
	print_r($result);
	echo '</pre>';

	parent::display($tpl);
}
```

- Các phương thức xây dựng câu SQL trong JDatabaseQueryMySQL

Ví dụ 1:

Trong MODEL

```
function getData3(){
	$db = $this->getDbo();

	$query = $db->getQuery(true)
		->select('bt.id, bt.title, bt.content')
		->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
		->where('bt.published = 0','AND')
		->where('bt.id <5')
		->order('bt.id DESC');

	echo '<br>' . $query;

	$db->setQuery($query);
	$result = $db->loadAssocList();
	return $result;
}
```

Trong VIEW

```
function display($tpl = null){
	echo '<br>' . __METHOD__;

	$model = $this->getModel();
	$items =  $model->getData3();
	echo '<pre>';
	print_r($items);
	echo '</pre>';

	parent::display($tpl);
}
```

Ví dụ 2:

Trong MODEL

```
function getData4(){
        $db = $this->getDbo();

        $query = $db->getQuery(true)
                ->select('bt.id, bt.title, bt.content')
                ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                ->where('bt.published = 1','OR')
                ->where('bt.id >10')
                ->order('bt.id ASC');

        echo '<br>' . $query;

        $db->setQuery($query);
        $result = $db->loadAssocList();
        return $result;
}
```

Trong VIEW

```
function display($tpl = null){
        echo '<br>' . __METHOD__;

        $model = $this->getModel();
        $items =  $model->getData4();
        echo '<pre>';
        print_r($items);
        echo '</pre>';

        parent::display($tpl);
}
```

Ví dụ 3: Thực thi với khóa LIKE

```
function getData5(){
        $db = $this->getDbo();

        $titleParamUrl = "PHP";

        $query = $db->getQuery(true);
        $sql = $query->select('bt.id, bt.title, bt.content')
                  ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                  ->where('bt.published = 1','OR')
                  ->where("bt.title LIKE ". $query->quote('%' .$titleParamUrl . '%') )
                  ->order('bt.id ASC');

        echo '<br>' . $sql;

        $db->setQuery($sql);
        $result = $db->loadAssocList();
        return $result;
}
```

Ví dụ 4: phương thức quote($text, $escape = true)

```
function getData5(){
        $db = $this->getDbo();

        $titleParamUrl = "1 OR '1=1'";

        $query = $db->getQuery(true);
```

```
        $sql = $query->select('bt.id, bt.title, bt.content')
                  ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                  //->where("bt.id = ". $titleParamUrl )
                  ->where("bt.id = ". $db->quote($titleParamUrl,true) )
                  ->order('bt.id ASC');

        echo '<br>' . $sql;

        $db->setQuery($sql);
        $result = $db->loadAssocList();
        return $result;
}
```

Ví dụ 5: phương thức clear($clause)

```
function getData6(){
        $db = $this->getDbo();

        $query = $db->getQuery(true);
        $sql = $query->select('bt.id, bt.title, bt.content')
                  ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                  ->where('bt.published = 1')
                  ->order('bt.id ASC');
        $query->clear('where');
        $query->clear('order');
        echo '<br>' . $sql;

        $db->setQuery($sql);
        $result = $db->loadAssocList();
        return $result;
}
```

Nếu phương thức này tham số $clause để rỗng nó sẽ xóa toàn bộ câu SQL

Một số phương thức khác chúng ta sẽ học trong bài tập sau:

```
->having($conditions)
->group($columns)
->join($type, $conditions)
->leftJoin($condition)
```

Ví dụ 6: Thêm một dòng vào bảng

```
function getData7(){
        $db = $this->getDbo();

        $query = $db->getQuery(true);
        $sql = $query->insert(_JOOM_TABLE_BOOK_TEST,true)
                    ->columns(array('title','alias','content'))
                    ->values("'PHP pro','Happy code','This is a content'");

        echo '<br>' . $sql;
        $db->setQuery($sql);
        $db->query();
}
```

Ví dụ 7: Update dữ liệu

```
function getData8(){
        $db = $this->getDbo();
```

```
        $query = $db->getQuery(true);
        $sql = $query->update(_JOOM_TABLE_BOOK_TEST)
                ->set("title ='PHP pro 2012'")
                ->where("title ='PHP pro 123'");

        echo '<br>' . $sql;
        $db->setQuery($sql);
        $db->query();

}
```

Ví dụ 8: Xóa dữ liệu trong bảng

```
function getData4(){
        $db = $this->getDbo();

        $query = $db->getQuery(true);
        $sql = $query->delete(_JOOM_TABLE_BOOK_TEST)
                    ->where("id = 14");

        echo '<br>' . $sql;
        $db->setQuery($sql);
        $db->query();
}
```

7. Đối tượng JDatabaseMySQL

- **Khởi tạo đối tượng JDatabaseMySQL**

Để khởi tạo đối tượng JDatabaseMySQL chúng ta có nhiều cách khởi tạo khác nhau tùy vào lớp chung ta dùng.

Khởi tạo ở bất kỳ nơi đâu

```
$db = JFactory::getDbo()
```

Khởi tạo trong MODEL (JModel)

```
$db = $this->getDbo();
//OR
$db = $this->_db;
```

Khởi tạo trong JTable

```
$db = $this->getDbo();
```

- **Lấy font chữ của Database**

```
$db->getCollation();
```

- Trả kết quả truy vấn về một mảng dữ liệu

```
$db->loadAssocList();
```

Ví dụ:

```
function getData(){
        $db = $this->getDbo();
```

```
        $query = $db->getQuery(true)
                ->select('bt.id, bt.title, bt.content')
                ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                ->where('bt.published = 1','OR')
                ->where('bt.id >10')
                ->order('bt.id ASC');

        echo '<br>' . $query;

        $db->setQuery($query);
        $result = $db->loadAssocList();
        return $result;
}
```

- Trả về kết quả là một danh sách đối tượng

```
$db->loadObjectList();
```

Ví dụ:

```
function getData(){
        $db = $this->getDbo();
        $query = $db->getQuery(true)
                ->select('bt.id, bt.title, bt.content')
                ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                ->where('bt.published = 1','OR')
                ->where('bt.id >10')
                ->order('bt.id ASC');

        echo '<br>' . $query;

        $db->setQuery($query);
        $result = $db->loadObjectList();
        return $result;
}
```

- **Trả về kết quả của một dòng trong bảng**

```
$db->loadAssoc();
$db->loadRow();
```

Ví dụ:

```
function getData4(){
        $db = $this->getDbo();
        $query = $db->getQuery(true)
                ->select('bt.id, bt.title, bt.content')
                ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                ->where('bt.id = 10');

        echo '<br>' . $query;

        $db->setQuery($query);
        $result = $db->loadAssoc();
        return $result;
}
```

- Trả kết quả về là mảng một chiều chứa các thông tin của cột đầu tiên trong câu truy vấn

```
$db->loadColumn();
```

Ví dụ:

```
function getData4(){
        $db = $this->getDbo();
        $query = $db->getQuery(true)
                ->select('bt.id, bt.title, bt.content')
                ->from(_JOOM_TABLE_BOOK_TEST . ' AS bt')
                ->where('bt.published = 1','OR')
                ->where('bt.id >10')
                ->order('bt.id ASC');


        echo '<br>' . $query;

        $db->setQuery($query);
        $result = $db->loadColumn();
        return $result;
}
```

- Lấy tiền tố của bảng trong database

```
$db->getPrefix();
```

- Lấy danh sách các bảng trong database

```
$db->getTableList();
```

- Thiết lập font UTF-8 cho database

```
$db->setUTF();
```

8. Đối tượng JTable

JTable là một lớp dùng để quản lý tổng thể các bảng trong Joomla. Một số phương thức thông dụng trong lớp JTable

- Khởi tạo JTable

Trong MODEL

```
$this->getTable($name,$prefix,$options);
```

- Lấy thông tin chi tiết của các cột trong bảng

```
$table->getFields();
```

- Lấy giá trị của một record trong bảng

```
$table->load($key);
```

Ví dụ: Lấy ra trong bảng #__book_test một record có ID = 10

```
$table->load(10);
echo '<pre>';
print_r($table);
echo '</pre>';
```

- Trích xuất dữ liệu của record vào một mảng riêng

```
$table->getProperties();
```

Ví dụ:

```
$table->load(10);
$properties = $table->getProperties();
echo '<pre>';
print_r($properties);
echo '</pre>';
```

- Chuyển một mảng thành một đối tượng hay ngược lại

```
JArrayHelper::toObject($properties);
```

Ví dụ:

```
$table->load(10);
$properties = $table->getProperties();

$item = JArrayHelper::toObject($properties);

echo '<pre>';
print_r($item);
echo '</pre>';
```

- Đưa dữ liệu vào đối tượng JTable

```
$table->bind($src,$ignore);
```

Ví dụ:

```
//Mang du lieu se dua vao doi tuong JTable
$src    = array(
        'id' => 10,
        'title' => 'ASP master',
        'alias' => 'asp-master',
        'content' => 'This is a test',
        'published' => 1,
        );

//Loai bo cac phan tu trong mang
$ignore = array('id');

//Dua mang vao doi tuong JTable
$table->bind($src,$ignore);
```

- Lưu dữ liệu vào database

```
$table->store();
```

Ví dụ:

```
function getData2(){
        $table = $this->getTable('Test','Table');
        $src    = array(
                        'id' => 10,
                        'title' => 'ASP master',
                        'alias' => 'asp-master',
                        'content' => 'This is a test',
                        'published' => 1,
                        );
        $ignore = array('id');
```

```
        $table->bind($src,$ignore);
        $table->store();

        echo '<pre>';
        print_r($table);
        echo '</pre>';
}
```

Ví dụ: Chỉnh sửa record

```
function getData2(){
        $table = $this->getTable('Test','Table');

        $table->load(15);
        $table->title = 'ASP master 123';
        $table->store();
        echo '<pre>';
        print_r($table);
        echo '</pre>';
}
```

- Xóa một dòng trong bảng

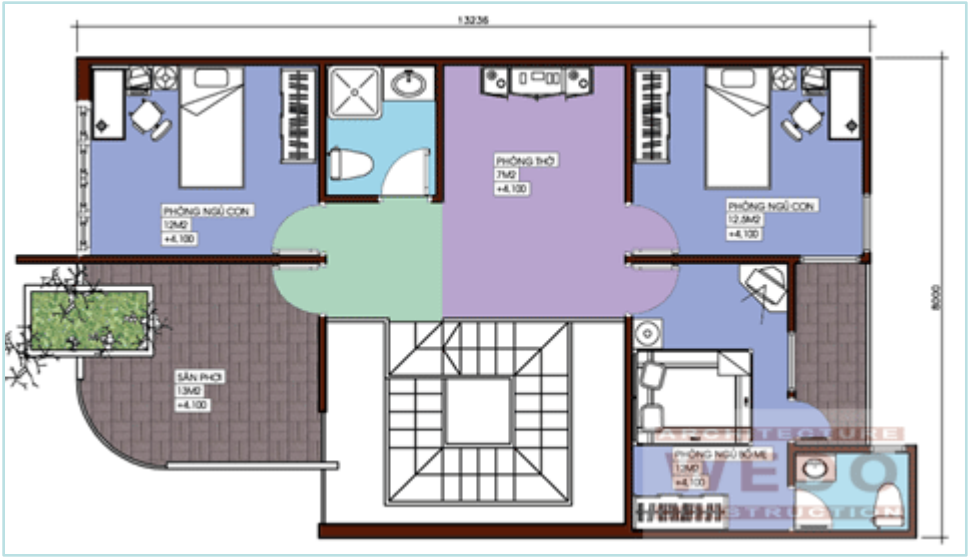
```
$table->delete($pk);
```

Ví dụ:

```
function getData2(){
        $table = $this->getTable('Test','Table');
        $table->delete(15);
}
```

Chương 11
Xây dựng Plugin

Plugin là gì?


13236
PHÒNG NGỦ CON
12M2
+4.100
PHÒNG THỜ
7M2
+4.100
PHÒNG NGỦ CON
12.5M2
+4.100
SÂN PHƠI
13M2
+4.100
8000


SINO

Nội dung tập tin ngôn ngữ

- Plugin đơn giản là một mã nguồn PHP
- Đượcthực hiện tại một hoặc nhiều thời điểm trong chu kỳ Joomla hoạt động
- Những thời điểm này được gọi là sự kiện
- Plugin được gắn với sự kiện

Để hiểu được Plugin trong Joomla

- Plugin thuộc vào nhóm nào
- Plugin sẽ hoạt động khi nào Plugin
- Phụ thuộc vào sự kiện nào?
- Plugin sẽ có bao nhiêu tham số
- Khi hoạt động Plugin sẽ tương tác đến thành phần nào trong lõi của Joomla?
- Chúng ta sẽ làm gì để biết chắc chắn Plugin sẽ hoạt động?

Kiến thức cần có để hiểu về Plugin

- Nắm rõ các sự kiện trong Joomla
- Sử dụng thành thạo hệ thống Joomla
- Tìm được vị trí gắn các plugin của Joomla
- Đọc hiểu mã nguồn của Joomla
- Nghĩ ra nhiều ý tưởng và thực hiện để tương tác hệ thống core của Joomla
- Luôn cập nhật kiến thức trong các phiên bản mới của Joomla

Nhóm sự kiện

- System events
- Content events
- User events
- Editor events
- Contact events

Các sự kiện

System

- onAfterInitialise
- onAfterRoute
- onAfterDispatch
- onAfterRender

Các sự kiện

Content

- onAfterDisplay
- onAfterContentSave
- onAfterDisplayTitle
- onAfterDisplayContent
- onContentPrepare
- onBeforeDisplay
- onBeforeContentSave
- onBeforeDisplayContent
- onContentPrepareForm
- onContentPrepareData

Các sự kiện

User

- onUserBeforeDelete
- onUserAfterDelete
- onUserBeforeSave
- onUserAfterSave
- onUserLogin
- onUserLogout
- onUserLoginFailure
- onUserLogoutFailure

Các sự kiện

Editors

- onInit
- onSave
- onSetContent
- onCustomEditorButton (editors-xtd)
- onDisplay
- onGetContent
- onGetInsertMethod

Các sự kiện

Authentication
- onUserAuthorisation
- onUserAuthorisationFailure

Search
- onSearch
- onSearchAreas

Các sự kiện

Contact
- onSubmitContact
- onValidateContact

XML-RPC
- onGetWebServices

Custom events

Các sự kiện

Contact
- onSubmitContact
- onValidateContact

XML-RPC
- onGetWebServices

Custom events

Cấu trúc Plugin

+ <Tên thư mục plugin>

- index.html
- <Tên plugin>.php
- <Tên tập tin cấu hình>.xml

\myplugin

- index.html
- myplugin.php
- myplugin.xml

Cấu trúc Plugin

+ <Tên thư mục plugin>

- index.html
- <Tên plugin>.php
- <Tên tập tin cấu hình>.xml

\myplugin

- index.html
- myplugin.php
- myplugin.xml

Cấu trúc Plugin

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="2.5" type="plugin" group="<nhóm plugin>">
        // Thong tin ve Plugin

        // He thong tap tin va thu muc trong plugin

        // Danh sach cac tap tin ngon ngu

        //He thong  tham so cau hinh cho plugin
</extension>
```

Thông tin Plugin

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="2.5" type="plugin" group="<nhóm plugin>">
        <name>plg_<nhom plugin>_<ten plugin></name>
        <author></author>
        <creationDate></creationDate>
        <copyright></copyright>
        <license></license>
        <authorEmail></authorEmail>
        <authorUrl>www.zend.vn</authorUrl>
        <version></version>
        <description></description>

        // He thong tap tin va thu muc trong plugin

</extension>
```

Hệ thống thư mục và tập tin Plugin

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="2.5" type="plugin" group="<nhóm plugin>">
        // Thong tin ve Plugin
<files>
        <filename plugin="<ten plugin>"><ten plugin>.php</filename>
        <filename>index.html</filename>
        <folder>something</folder>
</files>


</extension>
```

Các tập tin language

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="2.5" type="plugin" group="<nhóm plugin>">
        // He thong tap tin va thu muc trong plugin
        // Thong tin ve Plugin
<languages>
        <language tag="en-GB">
            en-GB.plg_<nhóm plugin>_<ten plugin>.ini
        </language>
        <language tag="en-GB">
            en-GB.plg_<nhóm plugin>_<ten plugin>.sys.ini
        </language>
    </languages>
</extension>
```

Hệ thống tham số cấu hình cho Plugin

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="2.5" type="plugin" group="<nhóm plugin>">
        // Thong tin ve Plugin
        // He thong tap tin va thu muc trong plugin
        // Danh sach cac tap tin ngon ngu
        <config>
                //Su dung JForm
        </config>
</extension>
```

Nội dung tập tin Plugin

```
<?php

defined( '_JEXEC' ) or die;

jimport('joomla.plugin.plugin');

class plg<nhóm plugin><ten plugin> extends JPlugin {


}
```

Nội dung tập tin Plugin

```
class plg<nhóm plugin><ten plugin> extends JPlugin {

        public function __construct( &$subject, $config ) {
                parent::__construct( $subject, $config );
        }

        function <ten su kien 1>() { }

        function <ten su kien 2>() { }

        // Cac su kien khac
}
```