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
<?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
<?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
)
```