Array

Arrays são conjuntos de valores relacionados, como os dados enviados em um formulário, os nomes dos alunos de uma classe ou as populações de uma lista de cidades no capítulo 2 você aprendeu que uma variável é um contêiner nomeado que um valor. Um array é um contêiner que armazena vários valores.

Este capítulo mostra como trabalhar com arrays. A próxima seção, "Fundamentos dos arrays", examinará aspectos básicos como a criação de arrays e a manipulação de seus elementos. Não é raro querermos fazer algo como cada elemento de um array, como exibi-lo ou verificar se ele atende certas condições. "Percorrendo arrays" na página 87, explica como fazer isso com as estruturas foreach() e for().

Trabalhar com arrays é uma tarefa comum na programação em PHP. O capítulo 7 mostrará como processar dados de formulários, que são inseridos automaticamente em um array pelo engine PHP. Quando recuperamos informações em um banco de dados como descrito no capítulo 8, com frequência elas são empacotadas em um array. Se você conhecer bem os arrays, será mais fácil manipular esses tipos de dados

Fundamentos dos arrays

Um array é composto por elementos. Cada elemento tem uma chave e um valor. Por exemplo, um array contendo informações sobre as cores dos vegetais teria os nomes dos vegetais como chaves e as cores como valores, como mostra a figura abaixo:

Chave	Valor
corn	yellow
beet	red
carrot	orange
pepper	green
broccoli	green

Um array só pode ter um único elemento com uma determinada chave. No array de cores dos vegetais, não pode haver outro elemento com a chave corn ainda que seu valor seja blue. No entanto, o mesmo valor pode aparecer muitas vezes no mesmo array. Você pode ter pimentas verdes, brócolis verdes e aipo verde.

Qualquer valor de string ou numérico pode ser a chave de um elemento do array, como corn, 4, -36 ou Salt baked liquid. Os arrays e outros valores não escalares não podem ser chaves, mas podem ser valores dos elementos. O valor de um elemento pode ser qualquer tipo: uma string, um número, true, false ou um tipo não escalar como outro array.

Criando um array

Para criar um array, use a estrutura array() da linguagem. Especifique uma lista de pares chave/valor delimitados por vírgula, com a chave e o valor separados por =>. Isso é mostrado no exemplo abaixo:

```
<?php
//criando um array exemplo 01
$vegetables = array(
'corn' => 'yellow',
'beet' => 'red',
'carrot' => 'orange'
);
$dinner = array(
0 => 'sweet corn',
1 => 'Braised Bamboo Fungus',
2 => 'Lemon Chicken'
);
$computers = array(
'trs-80' => 'Radio Shack',
 2600 => 'Atari',
);
?>
```

As chaves e os valores dos arrays do exemplo acima são do tipo string e numérico. Eles são escritos da mesma forma que outras strings e números dos programas PHP: com aspas nas strings, mas não nos números.

Uma abreviação que é usada para a estrutura de um array() é um par de colchetes (a chama sintaxe curta dos arrays). O exemplo abaixo cria os mesmos arrays do exemplo acima, mas com a sintaxe curta.

```
$vegetables = ['corn' => 'yellow', 'beet' => 'red', 'carrot' => 'orange'];
$dinner[
0 => 'sweet corn',
1 => 'Lemon Chicken',
2 => 'Braised Bamboo Fungus'
];
$computers = ['trs-80' => 'Radio Shack', 2600 => 'Atari', 'Adam' => 'Coleco'];
```

Você também pode adicionar um elemento de cada vez a um array, atribuindo um valor a uma chave específica. O exemplo abaixo constrói os mesmos arrays dos dois exemplos anteriores, mas o faz elemento a elemento.

```
$vegetables['corn'] = 'yellow';
$vegetables['beet'] = 'red';
$vegetables['carrot'] = 'orange';
$dinner[0] = 'Sweet and aspargus';
$dinner[1] = 'Lemon Chicken';
$dinner[2] = 'Braised Bamboo Fungus';
$computers['trs-80'] = 'Radio Shack';
$computers[2600] = 'Atari';
$computers['Adam'] = 'Coleco';
```

No exemplo acima, os colchetes após o nome da variável de array referenciam uma chave específica do array. Ao atribuirmos um valor para a chave, um elemento é criado no array.

Selecionando um nome adequado para o array

Nomes para as variáveis que contêm arrays seguem as mesmas regras dos nomes de qualquer outra variável. Os nomes de arrays e de variáveis escalares vêm do mesmo conjunto de nomes possíveis, logo, não podemos ter ao mesmo tempo uma variável de array e uma variável escalar chamadas $vegetables. Se você atribuir um valor escalar a um array (ou vice e versa), o valor anterior será removido e a variável adotará o novo valor. No exemplo 4.4, $vegetables passa ser um variável escalar e $fruits torna-se um array.

```
/*essa instrução transforma $vegetables em um array
$vegetables['corn'] = 'yellow';
essa remove qualquer traço de corn e tranforma $vegetables em uma variável escalar
$vegetables = 'delicious';
a instrução a seguir transforma $fruits em uma variável escalar
$fruits = 283;
essa não funciona -- $fruits continua sendo 283 e o engine PHP emite um aviso
$fruits['potasium'] = 'banana';
já esse sobrepões fruits, que passa a ser um array
$fruits = array ('potasium' => banana);
*/
```

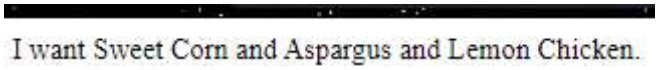
No primeiro exemplo, os arrays $vegetable e $computers armazenam cada um uma lista de relacionamentos. O array $vegetables relaciona vegetais e cores, enquanto o array $computers relaciona nomes de computador e fabricantes. No array $dinner, no entanto, só estamos interessados nos nomes das refeições, que são os valores do array. As chaves são apenas números que distinguem um elemento do outro.

Criando um array numérico

O PHP fornece alguns atalhos para o trabalho com arrays que só têm números como chaves. Se você criar um array com [ ] ou array( ) especificando apenas uma lista de valores em vez de pares chave/valor, o engine PHP atribuirá automaticamente uma chave numérica para cada valor. As chaves começam em 0 e aumentam em uma unidade para cada elemento.

```
$dinner = array('Sweet Corn and Aspargus',
                                'Lemon Chicken',
                                'Braised Bamboo Fungus');
print "I want $dinner[0] and $dinner[1].";
```

O exemplo acima exibe:



Internamente o engine PHP trata arrays com chaves numéricas a arrays com chaves de string de maneira idêntica. Devido à semelhança com recursos de outras linguagens de programação, geralmente os programadores se referem ao arrays que só tem chaves numéricas como "arrays indexados".

O PHP usará automaticamente números incrementais como chaves quando você criar um array ou adicionar elementos com a sintaxe de colchetes vazios mostrada no exemplo abaixo

```
//cria o array lunch com dois elementos
//essa instrução define $lunch[0]
$lunch[] = 'Dried Mushrooms in Brown Sauce';
//essa define $lunch[1]
$lunch[] = 'Pineapple and Yu Fungus';
print "$lunch[0]";
```

Os colchetes vazios adicionam um elemento ao array. O novo elemento tem uma chave numérica já existente no array. Se o array ainda não existir, os colchetes vazios adicionarão um elemento com a chave 0.

Descobrindo o tamanho de um array

A função count() informa o número de elementos de um array, o exemplo abaixo demostra count().

```
$dinner = array('Sweet Corn and Aspargus',
                                'Lemon Chicken',
                                'Braised Bamboo Fungus');
//print "I want $dinner[0] and $dinner[1].";
//descobrindo o tamanho de um array
$dishes = count($dinner);
print "There are $dishes things for dinner";
```

Quando recebe um array vazio, count() retorna 0. Um array vazio também é avaliado como falso na expressão de teste de if().

Percorrendo um arrays

Uma das coisas mais importantes que podemos fazer com um array é considerar cada elemento individualmente e processa-lo de alguma forma. Isso pode envolver incorpora-lo à linha de uma tabela HTML ou adicionar seu valor a um total acumulado.

A maneira mais fácil de iterar por cada elemento de um array é com foreach(). A estrutura foreach() permite executar um bloco de código uma vez para cada elemento do array. O exemplo abaixo usa foreach() para exibir uma tabela HTML que contém cada elemento de um array.

```
$meal = array('breakfast' => 'Walnut Bun',
```

```
                            'lunch' => 'Cashew Nuts and White Mushrooms',
                                'snack' => 'Dried Mulberies',
                                'dinner' => 'Eggplant with Chili Sauce');
print "<table  style = 'border: 1px solid black;'>";
foreach ($meal as $key => $value) {
    print "<tr><td style = 'border: 1px solid blue'>$key</td><td style = 'border: 1px solid
red'>$value</td></tr>";
}
print "</table>";
```

a saída é:

breakfast	Walnut Bun
lunch	Cashew Nuts and White Mushrooms
snack	Dried Mulberies
dinner	Eggplant with Chili Sauce

Para cada elemento de $meal, foreach() copia a chave do elemento em $key e o valor em $value. Em seguida, executa o código inserido entre chaves. No exemplo acima, o código exibe $key e $value com a notação HTML para criar uma linha da tabela. Você usar o nome de variáveis que quiser para a chave e o valor dentro do bloco de código. No entanto, se os nomes das variáveis tiverem sido usados antes de foreach(), eles serão sobrepostos por valores do array.

Dentro do bloco de código de foreach(), a alteração dos valores de variáveis do loop como $key e $value não afeta os elementos do array real. Se quiser alterar os valores dos elementos, use a variável $key como índice dentro do array. O exemplo abaixo use essa técnica para dobrar o preço de cada elemento.

```
$meals = array('Walnut Bun' => 1,
                                'Cashew' => 4.95,
                                'Dried Mullberies' => 3.00,
                                'Eggplant with chili Sauce' => 6.50);
foreach ($meals as $dish => $price) {
        $meals[$dish] = $meals[$dish] * 2;
}
foreach($meals as $dish =>$price){
        printf("The new price of %s is \$%.2f.
",$dish, $price);
}
```

O exemplo acima exibe:

The new price of Walnut Bun is $2.00.
The new price of Cashew is $9.90.
The new price of Dried Mullberies is $6.00.
The new price of Eggplant with chili Sauce is $13.00.

Há uma forma mais concisa de de foreach() para arrays numéricos mostrada no exemplo abaixo:

```
$dinner = array('Sweet Corn and Asparqus',
```

```
                                'Lemon Chicken',
                                'Braised Bamboo Fungus');
foreach($dinner as $dish){
        print "You can eat: $dish
";
}
```

Quando usar essa forma de foreach(), especifique um único nome de variável após "as" e o valor de cada elemento será copiado nessa variável. No entanto, você não poderá acessar as chaves dos elementos dentro do bloco de código.

Modificando arrays

Você pode executar operações com os elementos individuais de um array como faria com variáveis escalares, usando operadores aritméticos, lógicos e de outros tipos. O exemplo mostra algumas operações com elementos de um array

```
$dishes['Beef Chow Foon'] = 12;
$dishes['Beef Chow Foon']++;
$dishes['Roast Duck'] = 3;
$dishes['total'] = $dishes['Beef Chow Foon'] + $dishes['Roast Duck'];
if($dishes['total'] > 15){
        print "You ate a lot: ";
}
print 'You ate ' . $dishes['Beef Chow Foon'] . ' dishes of Beef Chow Foon ';
```

A saída do exemplo é:

```
You ate a lot: You ate 13 dishes of Beef Chow Foon
```

Usando arrays multidimensionais

Como mencionado em "Fundamentos dos arrays", um elemento de um array pode ter como valor outro array. Isso será útil quando quisermos armazenar dados que tenham uma estrutura mais complicada que apenas uma chave e um único valor. Um par chave/valor padrão é adequado para associar o nome de uma refeição a um único prato, mas e quando cada refeição é composta por mais de um prato? Nesse caso, os valores dos elementos devem ser arrays e não strings.

Use a estrutura array() ou a sintaxe curta dos arrays [ ] para criar arrays que tenham outros arrays como valores dos elementos como mostrado no exemplo abaixo:

```
$meals = array('breakfast' => ['Walnut', 'Coffee'],
                                'lunch' => ['Cashew Nuts', 'White Mushrooms'],
                                'snack' => ['Dried Mulberies', 'Salted Sesame Crab']);
$lunches = [ ['Chicken', 'Eggplant', 'Rice'],
                        ['Beef', 'Scallions', 'Noodles']
```

```
                        [ 'Beef', 'Scallions', 'Noodles' ],
                        ['Eggplant', 'Tofu'] ];
$flavors = array('Japanese' => array('Hot' => 'Wasabi',
                                                                    'Salty' => 'soy sauce'),
                                'Chinese' => array('hot' => 'mustard',
                'pepper' => 'prickly'
));
print $meals['lunch'][1];
```