PHP: Tipos de dados, Variáveis, Super-globais Master-Detail

Carlos Santos . NTC . DeCA . UA

Aula T12, 26 mar 2020

tipos de dados

- **em PHP**
  - os tipos de dados normalmente não são especificados pelo programador
  - os tipos de dados são decididos durante a execução, dependendo do seu contexto
- **utilizar**
  - **`var_dump(…)`** ou **`print_r(…)`** para avaliar durante o desenvolvimento (debug)
  - **`gettype(…)`** para obter o tipo de dados de uma variável
  - **`is_type(…)`** para fazer comparações durante a execução: `is_int(…)`, `is_string(…)`, `is_float(…)`, `is_numeric(…)`

como fazer debug?

- não é tão simples como em javascript…
- utilizar **`print_r(…)`** e **`var_dump(…)`** para imprimir valores temporariamente
- ver o código-fonte da página para perceber o que está mal no html, ajudando a identificar o problema no PHP
- em produção, utilizar ficheiros de log

mais informação sobre debug em PHP:
http://www.phpknowhow.com/basics/basic-debugging/

tipos de dados

```php
<?php
$a_bool = TRUE;   // a boolean
$a_str  = "foo";  // a string
$a_str2 = 'foo';  // a string
$an_int = 12;     // an integer

echo gettype($a_bool); // prints out:  boolean
echo gettype($a_str);  // prints out:  string

// If this is an integer, increment it by four
if (is_int($an_int)) {
    $an_int += 4;
}

// If $a_bool is a string, print it out
// (does not print out anything)
if (is_string($a_bool)) {
    echo "String: $a_bool";
}
?>
```


http://php.net/manual/en/language.types.intro.php

type casting

- conversão do tipo de dados armazenado numa variável (type casting)
  - `$score  = (float) 13;  -> $score = 13.0`
  - `$score  = (int) 13.22645;  -> $score = 13`
- se necessário, o PHP converte automaticamente o tipo de dados de modo a que o código possa ser corretamente processado. Exemplos:
  - `$a = 15; $b = "5";`
  - `$resultado = $a + $b; -> $resultado = 20`

  - `$cond = "1.0";`
  - `if ($cond) echo "a variável cond é verdadeira";`

type casting

- outro modo de modificar o tipo de dados
  - **gettype(var)** -> devolve o tipo de dados de var
  - **settype(var, type)** -> atribui a var o tipo de dados type

```
$foo = '1';
echo gettype($foo);  // output -> 'string'

$foo = (int) $foo;
echo gettype($foo);  // output -> 'integer'

settype($foo, float);
echo gettype($foo);  // output -> 'float'
```

identificadores

- um identificador pode identificar uma variável, uma função ou outro objecto definido pelo utilizador
  - possuem um ou mais caracteres
  - começam por uma letra ou por “_”
  - só podem conter letras, algarismos e o “_”
  - nunca devem ter caracteres especiais!

- Exemplos válidos : My_function, $Size, $_preco
  - Não esquecer: **$Carro** é diferente de **$carro**

variáveis

- armazenam valores
- identificam esses valores armazenados
- **declaração implícita** -> basta referir a variável no código
  - $a
- **atribuição direta** de valores
  - `$a = 2;`
- atribuição de valores por referência
  - `$c = &$b;`
    - se $b = 10 então $c = 10 (qualquer alteração numa reflete-se na outra)

scope (âmbito das variáveis)

- **Locais**
  - se declaradas dentro de uma função então existem apenas nesse âmbito

```
$x = 4;   // Variável global
function aMinhaFuncao() {
  $x = 0; // Variável local
}
```

  - dentro da função $x tem o valor 0
  - fora da função $x tem o valor 4
  - são variáveis distintas!

scope (âmbito das variáveis)

- **globais**
  - visíveis globalmente, mas…
  - `global $x; //Para utilizar $x dentro de uma função`
  - **Exemplo:**

```
$valor = 5;
function validar() {
   global $valor;
   if ($valor < 10) {
      $valor++;
    };
}
validar();
echo $valor; // ?
```

Alerta: As variáveis globais em PHP não funcionam do mesmo modo do Javascript!

scope (âmbito das variáveis)

- **estáticas**
  - existem no âmbito das funções e garantem que os valores não são destruídos quando se sai da função

```
function contador() {
   static $valor = 0;
   $valor = $valor + 1;
   echo $valor;
}
```

  - a cada chamada da função contador() a variável $valor é incrementada uma unidade

scope (âmbito das variáveis)

- **parâmetros de funções**
  - existem apenas no âmbito da respetiva função

```
function soma($parcelaA, $parcelaB) {
  $resultado =  $parcelaA + $parcelaB;
  return $resultado;
}
```

super-globais

- variáveis *built-in* e sempre acessíveis em todos os scopes
- **$_SERVER**
  - fornece dados sobre o ambiente em que a página corre (servidor)
- **$_FILES**
  - dados sobre os ficheiros transferidos para o servidor pelo método POST
- **$_ENV**
  - tal como a $_SERVER fornece dados sobre o ambiente onde a página corre (servidor e cliente)
- **$_SESSION**
  - guarda os dados de todas as variáveis de sessão de um utilizador
- **$_COOKIE**
  - acesso aos dados das cookies

super-globais

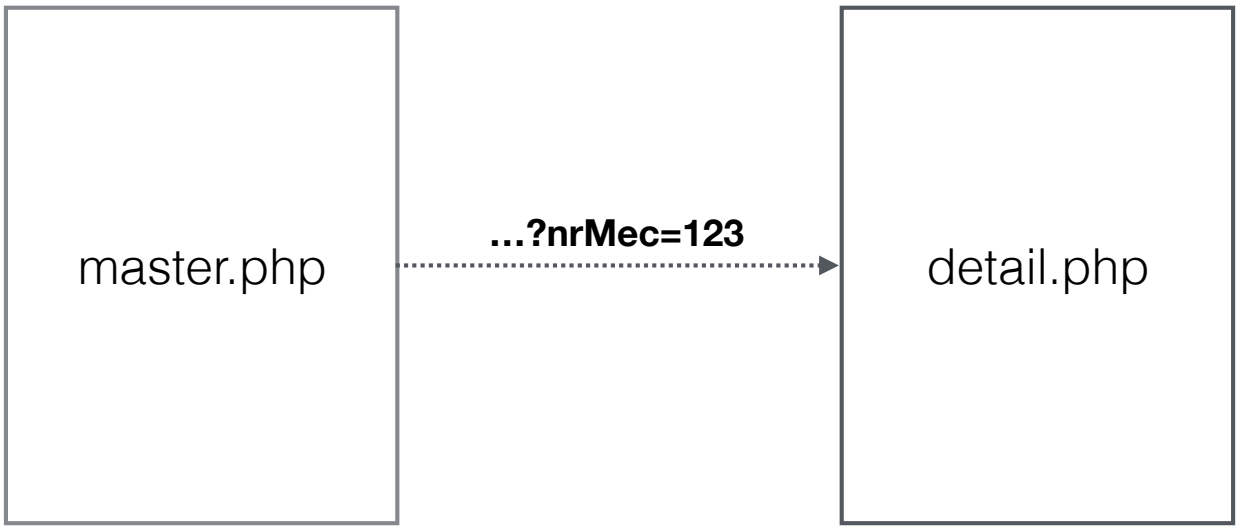
- variáveis *built-in* e sempre acessíveis em todos os scopes
- **$_GET**
  - acesso a dados enviados na query string do URL
- **$_POST**
  - acesso a dados enviados para o servidor através do método POST, normalmente um formulário

- **Estudar**: https://www.geeksforgeeks.org/php-superglobals/

super-globais

- **$_GET**
- passagem de parâmetros pelo **método GET**
  - http://www.example.com/index.php?**cat=apache&id=157**
  - `$_GET['cat'] -> "apache";`
  - `$_GET['id'] -> "157";`
- **query string**
  - tudo o que se segue ao “?” no URL (conjuntos chave/valor)
  - os dados de um formulário com método “GET” são enviados na query string

master-detail



- Uma interface do tipo **master-detail** é constituída por dois componentes:
  - A página ***master*** onde são listados vários elementos
  - A página ***details*** onde são mostrados os detalhes do elemento selecionado pelo utilizador na página master

master-detail

nrMec	Nome	Morada	Foto
123	Maria Benedita	Aveiro	aa.jpg
124	José Marcolino	Porto	bb.jpg
125	Ambrósio Esteves	Viseu	cc.jpg
127	Carlota Cardoso	Águeda	dd.jpg

- **Exemplo**:
  - A página master lista os alunos da tabela anterior. Cada nome tem associado um link para a página de detalhes, identificando no seu URL o nrMec do aluno
  - A página de detalhes obtém o nrMec do URL, consulta a BD para obter a informação completa do aluno e mostra essa informação na página

master-detail

Alunos de LabMM4

- Maria Benedita
- José Marcolino
- Ambrósio Esteves
- Carlota Cardoso

detalhesAluno.php?nrMec=127

alunos.php

Carlota Cardoso
Águeda

detalhesAluno.php

A super-global GET facilita a implementação desta estratégia!

super-globais

- **$_POST**
- passagem de parâmetros pelo método POST

```
<form action="subscribe.php" method="post">
  <input type="text" name="email" value="">
  <input type="submit" value="Subscribe!">
</form>
```

- na página **subscribe.php** o valor introduzido no campo “email” está disponível em **$_POST['email']**

super-globais

- **$_SESSION**
- permite guardar valores durante uma **sessão**
- funciona através do **Session ID**

```
<?php
session_start(); //Antes de escrever HTML! <htlm>
if (!isset($_SESSION['count'])) {
  $_SESSION['count'] = 0;
} else {
  $_SESSION['count']++;
}
?>

unset($_SESSION['count']); // para eliminar uma
                           // variável de sessão
```

constantes e expressões

- **constantes**
  - valores que não são modificados ao longo da execução da aplicação
  - são valores globais

```
define ('PI', 3.141592);
$dobro =  2 * PI;
echo $dobro; echo PI;
```

- **expressões**
  - representam genericamente uma ação no nosso programa
  - contêm operandos e operadores

```
$a = 2;
$soma = 2 + 3;
$contador++;
```

operadores

- precedência entre operadores
  - seguem as regras matemáticas “normais”
- operadores aritméticos
  - **$a + $b** -> adição
  - **$a - $b** -> subtracção
  - **$a * $b** -> multiplicação
  - **$a / $b** -> divisão
  - **$a % $b** -> resto da divisão inteira de $a por $b

operadores

- **operadores de atribuição**
  - **$a = 5** -> atribuição -> $a igual a 5
  - **$a += 5** -> adição-atribuição -> $a = $a + 5
  - **$a *= 5** -> multiplicação-atribuição -> $a = $a * 5
  - **$a /= 5** -> divisão-atribuição -> $a = $a / 5

- **operadores de strings**
  - **$a = “abc" . "def"**; -> concatenação -> $a igual a "abcdef"
  - **$a .= "abc"** -> concatenação-atribuição > $a = $a . "abc"

operadores

- **incremento e decremento**
  - **++$a**, **$a++** -> incremento -> $a = $a + 1;
  - **--$a**, **$a--** -> decremento -> $a = $a - 1;
  - mas os resultados nem sempre são iguais!

```php
<?php
    $a=1;
    $c=++$a;
    echo $a." | ";
    echo $c;
?>
```

2 | 2

```php
<?php
    $a=1;
    $c=$a++;
    echo $a." | ";
    echo $c;
?>
```

2 | 1

operadores

- **operadores lógicos**
  - **$a && $b** -> E/AND lógico -> só é verdadeiro se ambos forem verdadeiros
  - **$a || $b** -> OU/OR lógico -> verdadeiro se pelo menos um for verdadeiro
  - **!$a** -> negação/NOT -> verdadeiro se $a for falso
  - também existem os operadores **and**, **or** e **xor**
    - mas as regras de precedência são diferentes e podem confundir
    - http://php.net/manual/en/language.operators.logical.php

operadores

- **operadores de igualdade**
  - **$a == $b** -> Será $a **igual** a $b?
  - **$a != $b** -> Será $a **diferente** a $b?
  - **$a === $b** -> Será $a **exactamente igual** a $b? (compara o valor e o tipo de dados)
- **operadores de comparação**
  - **$a < $b** -> Será $a **menor** que $b?
  - **$a > $b** -> Será $a **maior** que $b?
  - **$a <= $b** -> Será $a **menor ou igual** que $b?
  - **$a >= $b** -> Será $a **maior ou igual** que $b?

atribuição de valores por referência

- Exemplo:

```
<?php
    $a = "Gostas do SCP? ";
    $b = &$a;
    echo $b;
    $b = "Claro que sim!";
    echo $a;
?>
```

```
Gostas do SCP? Claro que sim!
```