Panorâmica geral sobre PHP

- PHP é uma linguagem que permite fazer páginas dinâmicas.

- um ficheiro em PHP contem código PHP misturado com HTML.

- em vez de PHP podíamos utilizar outra linguagem qualquer desde que devidamente instalada no servidor web.

PHP (cont.)

- PHP tem uma sintaxe parecida com a do C, Java e Perl.

- PHP é interpretado.

- variáveis têm o sinal de `$`. Exemplo: `$filme`

- não é necessário declarar variáveis.

Estruturas de controlo

- PHP tem as estruturas de controlo habituais:
  - if, if...else
  - switch
  - while
  - do...while
  - for
  - foreach
- Semelhantes à linguagem C (foreach não existe em C).

Exemplo com foreach

```
$lengths = array(0,107,202,400);

// converte centimetros para inches
foreach($lengths as $cm)
{
  $inch = (100 * $cm) / 2.45;
  echo "$cm centímetros = $inch inches\n ";
}
```

Arrays

```
$numbers = array(5, 4, 8, 14);
$words = array("Pedro", "Ana", "Maria");

// escreve o terceiro elemento de $numbers: 8
echo $numbers[2]

// escreve o primeiro elemento de $words: "Pedro"
echo $words[0]
```

Alternativa:

```
$words[0] = "Pedro";
$words[1] = "Ana";
$words[2] = "Maria";
```

Arrays (cont.)

A primeira posição do array pode ser diferente de 0.

```
$words = array(1 =>"Pedro", "Ana", "Maria");
```

Podemos criar um array vazio e ir inserindo elementos:

```
$words = array();
$words[] = "Pedro";   // pos 0
$words[] = "Ana";     // pos 1
$words[] = "Maria";   // pos 2
```

Arrays associativos

Indices do array podem ser strings (neste caso, os indices costumam ser designados por chaves). Exemplo:

```
$a = array("first"=>1, "second"=>2, "third"=>3);
echo $a["second"]    // escreve 2
```

Também podem ser criados como,

```
$a["first"] = 1;
$a["second"] = 2;
$a["third"] = 3;
```

Foreach com arrays associativos

```
$a = array("first"=>1, "second"=>2, "third"=>3);

foreach($a as $s=>$v)
{
  echo "$s -- $v<br>\n";
}
```

O output seria:

```
first -- 1<br>
second -- 2<br>
third -- 3<br>
```

Arrays (cont.)

- Podemos ter arrays heterogéneos. Exemplo:

```
$mixed = array("Pedro", 22, 1.77);
```

- Podemos usar `var_dump` para debug,

```
var_dump($mixed)

array(3) { [0] => string(5) "Pedro"
           [1] => int(22)
           [2] => float(1.77) }
```

Funções para manipular arrays

- Existe uma série de funções pré-definidas para manipular arrays.
  - count()
  - in_array()
  - sort()
  - ...
- Vejam o manual antes de fazerem trabalho desnecessário.

Strings

- Pode-se usar aspas ou plicas.

- Tem de se fazer "escape" de caracteres especiais.

```
"O meu nome é Fernando"
'Olá, tudo bem'

$nome = "Fernando"
echo "Olá $nome"  // escreve Olá Fernando
echo 'Olá $nome'  // escreve Olá $nome

echo "Olá \$nome" // escreve Olá $nome
```

Strings

- Existe uma série de funções pré-definidas para manipular strings.
  - strlen, strcmp, strpad
  - strtolower, strtoupper, ucfirst, ucwords
  - trim, ltrim, rtrim
  - substr, strpos
  - implode, explode
- Vejam o manual.

Mais funções

- Com utilização de expressões regulares.
- Para manipular horas e datas.
- Funções matemáticas
- ...
- Vejam o manual.

Funções definidas pelo utilizador

```
function bold($string)
{
  echo "<b>$string</b>";
}

$myString = "Olá, tudo bem."
bold($myString)
```

Funções definidas pelo utilizador (cont.)

```
function double($var)
{
  $var = $var * 2;
  return $var;
}

$x = 5;
$temp = double($x);
echo "\$temp: $temp";
```

Escreve:

```
$temp: 10
```

Funções definidas pelo utilizador (cont.)

- Passagem de parâmetros:
  - por valor
  - por referência
- Exemplo de passagem por referência:

```
function double(&$var)
{
  $var = $var * 2;
}

$x = 5;
double($x);
echo "\$x: $x";     // escreve $x: 10
```

Argumentos podem ter valores por defeito

```
function heading($text, $headingLevel = 2)
{
  switch($headingLevel)
  case 1:
     $result = "<h1>" . ucwords($text) . "</h1>";
     break;
  case 2:
     $result = "<h2>" . ucwords($text) . "</h2>";
     break;
  case 3:
     $result = "<h3>" . ucfirst($text) . "</h3>";
     break;
  default:
     $result = "<p><b>" . ucfirst($text) . "</b>";
  return $result;
}

$test = "user defined functions";
heading($test);
```

Podemos separar a aplicação em vários ficheiros

- Depois incluímos o ficheiro.
- Pode-se usar `include` ou `require`.
- `include` faz inclusão condicional.
- `require` inclui sempre.

Exemplo

```
if($mozilla == true)
{
  include "mozilla.inc";
}
else
{
  include "other.inc";
}
```

Classes e Objectos

- PHP suporta (embora com algumas limitações) a programação orientada a objectos.

```
<?php
  //
  // Filename: Counter.php
  //
  // a class that defines a counter
  class Counter
  {
    // member variables
    var $count;
    var $startPoint;

    // methods

    function Counter($start = 0)   // constructor
    {
      $this->count = $start;
      $this->startPoint = $start;
    }

    function increment()
    {
      $this->count++;
    }
```

```
    function reset()
    {
      $this->count = $this->startPoint;
    }

    function value()
    {
      return $this->count;
    }
  }
?>
```

- Utilização da classe Counter

```
<?php
  //
  // Filename: useCounter.php
  //
  include "Counter.php";

  $aCounter = new Counter;
  echo $aCounter->value() . "<br>";
  $aCounter->increment();
  echo $aCounter->value() . "<br>";
?>
```

Erros frequentes

- Não aparece nada no browser.
  (provavelmente, esquecemo-nos de colocar um </table> ou </form>)

- Façam "View HTML source" no vosso browser para verem o HTML gerado.

- Utilizem `http://validator.w3.org/`

Erros frequentes (cont.)

- O facto da linguagem não ser compilada faz com que possam aparecer erros inesperados.

- Exemplo:

```
for($counter=0; $counter<10; $Counter++)
{
  ...
}
```

  Dá um ciclo infinito!