DESENVOLVIMENTO DE APLICAÇÕES COMPUTACIONAIS

Documento de apoio ao Guião W3: Introdução ao PHP

Edição 2023/24, versão 4.0

Objetivos

- Introdução à programação PHP;

1. Introdução

No desenvolvimento de aplicações Web, a criação de páginas dinâmicas pode realizar-se através de diversas linguagens de programação e de scripting, como o PHP, o JavaScript ou as applets Java. Estas linguagens diferenciam-se não apenas pela sua sintaxe e capacidades, mas, sobretudo, pela arquitetura subjacente ao seu modelo de execução: *client-side,* quando o código é interpretado no navegador do utilizador, ou *server-side*, quando o código é processado no servidor antes da resposta ser enviada ao cliente.

A linguagem **PHP (Hypertext Preprocessor)** é uma das soluções mais consolidadas e utilizadas para desenvolvimento Web no lado do servidor. O PHP permite a construção de páginas cujo conteúdo é gerado dinamicamente no momento da requisição HTTP, com base num funcionamento definido em scripts executados no servidor, que podem ser embebidas diretamente numa página HTML. Esta execução pode usar e processar dados enviados pelo cliente através dos formulários e/ou incorporar dados provenientes de diferentes fontes, como sistemas de bases de dados, ficheiros, valores das variáveis de ambiente das sessões de utilizador, etc.

Ao contrário de linguagens como o **JavaScript**, cuja execução ocorre no *browser* e serve tipicamente para melhorar a experiência do utilizador com a interatividade local (no *frontend*), o PHP opera no *backend*, sendo responsável por gerar o conteúdo final que será entregue ao cliente, normalmente sob a forma de HTML.

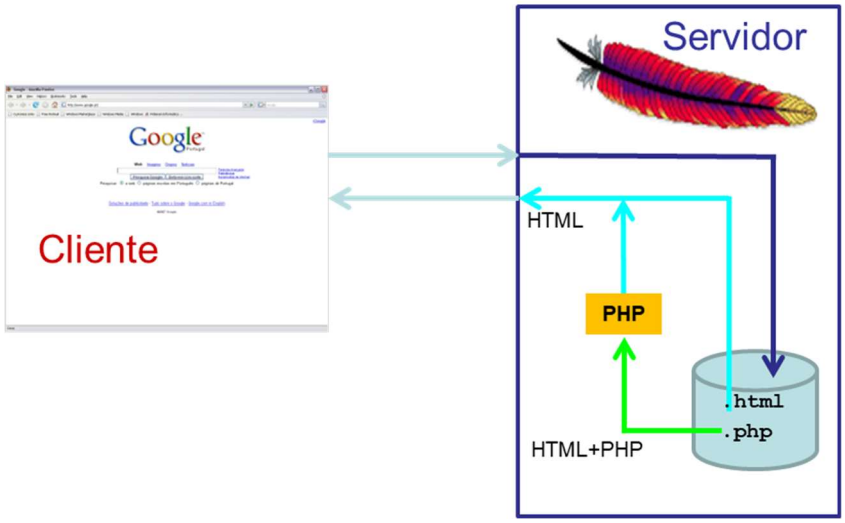
A utilização de PHP nas aplicações Web é diretamente associada aos servidores Web como o Apache, o IIS ou o Nginx, e a sua integração com sistemas de gestão de bases de dados relacionais, como o MySQL o PostgreSQL ou o SQLite, é uma das suas principais mais valias no desenvolvimento de aplicações Web robustas e escaláveis.

2. PHP - Hypertext Preprocessor

PHP[1] is a powerful server-side scripting language for creating dynamic and interactive websites.
PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. PHP is perfectly suited for Web development and can be embedded directly into the HTML code.

The PHP syntax is very similar to Perl and C. PHP is often used together with Apache (web server) on various operating systems. (…)

O PHP é uma linguagem de programação interpretada[2] conhecida, essencialmente, por ser utilizada na criação de páginas Web dinâmicas. A programação de páginas dinâmicas usando PHP faz-se embebendo diretamente o seu código no código da página HTML. Nestes casos o interpretador PHP é invocado pelo servidor Web, o que significa que ao ser pedida uma página contendo um script PHP, este vai ser interpretado, produzindo-se dinamicamente uma página (HTML) que é posteriormente enviada para o cliente (*browser*) através do protocolo HTTP. A figura seguinte esquematiza a interação referida.



O interpretador PHP também pode ser invocado diretamente na linha de comandos para processar um ficheiro, como se exemplifica no comando seguinte

```
$> php scriptXPTO.php
```

PHP é uma linguagem com licença "Open Source", de fácil aprendizagem e que suporta funcionalidades de programação avançadas. É possível definir variáveis (`$var`), vetores e matrizes de variáveis, manipular essas variáveis através de operadores (aritméticos, de atribuição, comparação e lógicos). É possível definir *strings*, concatená-las e manipulá-las com funções disponíveis na linguagem base

[1] PHP - Hypertext Preprocessor.

[2] Scripting – compilação em tempo de execução, ou seja cada vez que o script PHP é executado este é previamente compilado. Por oposição às linguagens compiladas (ex. C) nas quais é gerado um ficheiro executável.

(*strlen*, *strpos*, etc). Tem predefinido um conjunto de instruções condicionais (*if else*, *switch*), de implementação de ciclos (*while*, *do while*, *for*, *foreach*), de declaração de funções (*function*), assim como estruturas de informação para facilitar o processamento de dados inseridos em formulários Web e enviados para o servidor recorrendo aos métodos HTTP GET e POST.

Para além das anteriores funcionalidades, existem muitas outras das quais se podem referir, a título de exemplo, a possibilidade de operar e formatar informação de tempo, a data e hora na página Web (*date*), manipular ficheiros (*fopen*), fazer *upload* de ficheiros para o servidor (Upload-File Form), criar e gerir cookies (*setcookie*, $_COOKIE), estabelecer e gerir sessões (PHP session variable, $_SESSION), enviar emails diretamente a partir do *script* (*mail*). Existem ainda funções de manipulação de bases de dados (ver http://php.net/manual/en/refs.database.vendors.php), e para processamento de documentos XML (Extensible Markup Language), etc.

Programação em PHP

Os scripts PHP são delimitados por *tags* específicas. Esses scripts serão inseridos na página HTML e, quando a página é solicitada por um cliente (*browser*), o servidor inicia a interpretação do PHP nela existente. No processo de interpretação de uma página, as *tags* de script PHP serão substituídas pelo texto que resultar da interpretação desse PHP. A página resultante da interpretação já não contém quaisquer scripts em PHP, contém apenas HTML, pelo que é enviada para o cliente (*browser*) através do protocolo HTTP para ser apresentada.

Delimitação da script PHP

```
<?php  …  ?>
```

Comentários em PHP

Para inserir uma linha de comentário:

```
// Linha de comentário
```

Múltiplas linhas de comentário:

```
/*    Comentário Linha 1

                Linha N
*/
```

Variáveis

As variáveis guardam valores (números, caracteres ou sequências de números "arrays" ou caracteres "strings").

Em PHP os nomes das variáveis começam com o símbolo $.

Declaração e inicialização de uma variável:

```
$var1 = 10;
```

Em PHP não é necessário declarar o tipo da variável, pois o interpretador converte a variável automaticamente para o tipo que está a ser inicializado na variável.

Em PHP as variáveis podem guardar valores dos seguintes tipos:

- Booleanos
- Inteiros
- Vírgula flutuante
- Carácter

Operadores

Aritméticos

- '+' Adição
- '-' Subtração
- '*' Multiplicação
- '/' Divisão
- '%' Módulo (resto da divisão)
- '++' Incremento
- '--' Decremento

Atribuição

- '='
- '+=' (x+=y ⇔ x=x+y)
- '-=' (x-=y ⇔ x=x-y)
- '*=' (x*=y ⇔ x=x*y)
- '/=' (x/=y ⇔ x=x/y)
- '.=' (x.=y ⇔ x=x.y)
- '%=' (x%=y ⇔ x=x%y)

Comparação

- '==' Igual (valor)
- '===' Igual (valor e tipo)
- '!=' Diferente (valor)
- '!==' Diferente (valor e tipo)
- '>' Maior
- '<' Menor
- '>=' Maior ou igual
- '<=' Menor ou igual

Lógicos

- '&&' E
- '||' OU
- '!' Negação

Strings

Estas variáveis contêm sequências de caracteres.

Criar uma *string* (é criar uma variável genérica):

```
$string1;
```

Inicializar uma *string* (ao inicializar uma variável genérica com uma sequencia de caracteres)

```
$string1 = "A minha primeira string";
```

Imprimir uma *string* no ecrã:

```
echo $string1;
```

Operador de concatenação

Concatenação consiste em anexar o conteúdo de uma *string* a outra (colar no final). Em PHP esta operação tem um operador próprio '**.**' .

```
$string1 = "A minha primeira string";

$string2 = "PHP foi escrita em DEAPC";
```

Concatenação da *string* guardada na variável $string2 com a da $string1, guardando a *string* resultante na variável `$string1`.

```
$string1 = $string1 . $string2;
```

Concatenação colocando uma palavra no meio:

```
$string1 = $string1 . " em " . $string2;
```

Exemplos de funções para operar com *strings*

`strlen` - Determinar o tamanho de uma *string*

```
strlen ("A minha primeira string");
```

cujo resultado será 23, ou para imprimir esse valor no ecrã:

```
echo strlen ("A minha primeira string");
```

`strpos` – Procurar um carácter ou string dentro de uma string

Quando encontra o carácter ou a string retorna a posição em que a expressão de procura se encontra. Caso não encontre retorna FALSE.

Exemplo de uso:

```
strpos ("A minha primeira string","pri");
```

cujo resultado será 8 (1ª posição de string é '0').

If else

Para execução condicional de código testando condições, temos tal como noutras linguagens de programação o ciclo "if else".

Sintaxe:

```
if (condição) {
Código a ser executado caso a condição seja verdadeira }
else {
Código a ser executado caso a condição seja falsa }
```

Exemplo de uso:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")    //[3]
  echo "Bom Fim de semana!";
else
  echo "Bom estudo!";
?>

</body>
</html>
```

Ciclo While

Este ciclo permite executar uma instrução enquanto a condição for verdadeira.

Sintaxe:

```
while (condição) {
Código a ser executado }
```

Exemplo de uso:

```
<html>
<body>

  <?php
    $i=1;
    while($i<=5)
      {
      echo "Numero: " . $i . "<br />";
      $i++;
      }
  ?>
</body>
</html>
```

Ciclo For

Este ciclo permite executar determinado código *n* vezes.

Sintaxe:

```
for (inicialização; condição; incremento)
{
          Código a ser executado;
}
```

Exemplo de uso:

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
{
  echo "teste do FOR<br />";
```

[3] Função Date – permite extrair a data/tempo apresentando-os num formato à escolha do programador.

```
}
?>

</body>
</html>
```

Funções

As funções são suportadas pelo PHP. O programador pode criar funções ou utilizar funções implementadas por outros. A biblioteca de funções do PHP é extensa, é um dos seus pontos mais fortes, disponibilizando ao programador funções "prontas a utilizar" nas diferentes tarefas.

A declaração e chamada de uma função com parâmetros:

```
<html>
<body>

<?php
function adicao($x,$y)
  {
  $soma = $x + $y;
  return $soma;
  }
echo "2 + 11 = " . adicao(2,11);
?>

</body>
</html>
```

Processamento de formulários HTML em PHP

A utilização de scripts PHP facilita e simplifica a utilização de formulários de inserção de dados em páginas HTML.

Consideremos o seguinte formulário:

Name:	`<form>`
	`Name: <input type="text" name="NomeInserido" />`
	`</form>`

Método GET

O código do formulário para que os dados sejam processados pelo método http GET é o seguinte:

```
<form action="welcome.php" method="get">
Name: <input type="text" name=" NomeInseridoA" />
<input type="submit" />
</form>
```

E o conteúdo do ficheiro que processa o formulário "welcome.php" será:

```
<html><body>
Welcome <?php echo $_GET["NomeInseridoA"]; ?>.<br />
</body></html>
```

Método POST

O código do formulário para que os dados sejam processados pelo método http POST é o seguinte:

```
<form action="welcome.php" method="post">
Name: <input type="text" name="NomeInseridoB" />
<input type="submit" />
</form>
```

E o conteúdo do ficheiro que processa o formulário "welcome.php" será neste caso:

```
<html>
<body>
Welcome <?php echo $_POST["NomeInseridoB"]; ?>.<br />
</body>
</html>
```

Um a outra forma de receber informação dos formulários é através do vetor associativo $_REQUEST, que funciona de modo semelhante aos $_GET e $_POST.

Debug em PHP

O processo de programação é muita das vezes realizado de uma forma iterativa, envolvendo uma depuração sucessiva do código, e removendo os erros que vão sendo detetados. Quando a programação do código é realizada em PHP num servidor web, muitas das vezes os erros existentes no código não se tornam diretamente visíveis no browser, surgindo apenas uma página de indicação de erro no servidor.

Para que se conseguir obter mais informação acerca dos erros, podemos ativar a geração das mensagens de erro inserindo no início das scripts as seguintes linhas de código.

```
<?php
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);

    . . .
?>
```

Acesso a base dados em PHP

A maioria dos sistemas de bases de dados (BD) existentes disponibiliza APIs para a programação em PHP. Para a interação com a maioria desses sistemas, nas scripts PHP deve ser estabelecida uma ligação com a base de dados, e através dessa devem ser trocados os comandos para executar as *queries* na base de dados.

No extrato seguinte exemplifica-se o código de uma script PHP para criação, acesso e interação com uma BD SQLite3 armazenada num ficheiro '`test.db`'.

Esta script começa por estabelecer uma nova ligação (identificada por `$db`) com a base de dados SQLite3, existente ou a criar no ficheiro '`test.db`' recorrendo à instrução

```
$db = new SQLite3('test.db');
```

Inicialmente, '`test.db`' poderá ser um ficheiro vazio.

Através da ligação estabelecida, são dadas instruções de execução de várias *queries* na BD. Neste exemplo ilustra-se (1) o envio de uma query SQL para criação de uma tabela (Viaturas), e seguidamente (2) a inserção nessa tabela de 3 entradas. Estas *queries* são ordenadas com o comando `exec( )` cujo retorno é apenas `true`, ou `false` em caso de falha.

Posteriormente, exemplifica-se (3) a utilização de uma *query* (definida em `$sqlvar`) que retornará valores os valores da base de dados que satisfazem essa *query*. Nestes casos, os resultados devolvidos são objetos estruturados com a informação dos vários campos das diversas entradas da tabela que satisfizeram a *query* processada, que neste exemplo será guardado em `$result`. Para facilitar o processamento dessa informação, pode ser usado o método **`fetchArray(`**`SQLITE3_ASSOC`**`)`** que extrai do objeto `$result` a informação linha a linha, como é exemplificado em (4), e com ela constrói um array associativo (`$row`).

```
<?php
    $db = new SQLite3('test.db');

    // criação da tabela Viaturas

    $db->exec("CREATE TABLE Viaturas(id INTEGER PRIMARY KEY, marca TEXT, preco INT)"); //(1)
    $db->exec("INSERT INTO Viaturas(marca, preco) VALUES('Fiat', 21644)");             //(2)
    $db->exec("INSERT INTO Viaturas(marca, preco) VALUES('Toyota', 35445)");           //(2)
    $db->exec("INSERT INTO Viaturas(marca, preco) VALUES('Cupra', 29090)");            //(2)

    echo "<h3>Tabela de Viaturas </h3>";

    $sqlvar = "select * from Viaturas ;";

    $result = $db->query($sqlvar);                                                     //(3)

    echo "<table>\n<th> Id </th><th> Marca </th><th> Preço </th>\n";
    while ($row = $result->fetchArray(SQLITE3_ASSOC))                                  //(4)
    {
       echo '<tr><td>' . $row['id'] . '</td><td>' . $row['marca'] . '</td><td>' . $row['preco']
. "</td></tr>\n";
    }
    echo '</table>';
    unset($db);
?>
```

3. Referências

[1] "PHP Tutorial", http://www.w3schools.com/PHP .

[2] "PHP: A simple tutorial - Manual," http://us.php.net/tut.php.

[3] "ScriptTester by vanKonGa," http://forumferney.free.fr/stester.html.

[4] "PHP: PHP Manual - Manual," http://www.php.net/manual/en/.

[5] https://www.php.net/manual/en/refs.database.vendors.php

[6] "SQLite3 PHP Extension", https://www.php.net/manual/en/book.sqlite3.php

[7] "PHP 7 Quick Scripting Reference", Mikael Olsson, Apress

4. Histórico

1) 5 de Junho de 2008, Versão 1.0, mailto: vms@isep.ipp.pt
2) 13 de Junho de 2008, Versão 2.3, vms@isep.ipp.pt; jbm@isep.ipp.pt
3) 11 de Junho de 2010, Versão 2.4, crc@isep.ipp.pt
4) 11 de Maio de 2011, Versão 2.5, jbm@isep.ipp.pt
5) 13 de Maio de 2014, Versão 2.6, jbm@isep.ipp.pt
6) 6 de Maio de 2015, Versão 2.7, jbm@isep.ipp.pt
7) 13 de Maio de 2017, Versão 2.8, jbm@isep.ipp.pt, crc@isep.ipp.pt
8) 8 de Maio de 2019, Versão 2.9, jbm@isep.ipp.pt
9) 31 de Maio de 2023, Versão 3.0, jbm@isep.ipp.pt
10) 1 de Junho de 2024, Versão 4.0, jbm@isep.ipp.pt