



Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

PHP

- ◆ Rasmus Lerdorf desenvolveu esta linguagem em 1994 (para saber quem acedia à sua página pessoal).
- ◆ O PHP funciona do lado do servidor, sendo interpretada.
- ◆ O PHP é um híbrido de Perl, Java e C.
 - <http://www.php.net>
 - <http://php.weblogs.com/easywindows>
 - <http://www.phpmylinux.net>
 - http://www.php.net/manual/pt_BR/html/index.html

◆ Introduzir código PHP no HTML:

1. <?

```
echo ("Olá Mundo\n");
```

?>

2. <?php

```
echo("Olá Mundo\n");
```

?>

3. <script language="php">

```
echo("Olá Mundo\n");
```

</script>

4. <%

```
echo("Olá Mundo\n");
```

%>

- ◆ Separação de instruções:
 - Utilizando o ponto e vírgula (;). Na última instrução não é necessário.

- ◆ Comentários:

- `/* */` (tipo c)

```
<?php
/* isto é um
comentário tipo
várias linhas*/
echo("Olá Mundo\n");
?>
```

- `//` (tipo c++)

```
<?php
// isto é um comentário tipo uma linha
echo("Olá Mundo\n");
?>
```

- `#` (tipo shell do Unix)

```
<?php
echo("Olá Mundo\n"); # isto é um comentário tipo uma
linha
?>
```

- ◆ Tipos de dados (inteiros):

```
$a = 1234; # número decimal
```

```
$a = -123; # um número negativo
```

```
$a = 0123; # número octal (equivalente a 83 decimal)
```

```
$a = 0x12; # número hexadecimal (equivalente a 18 decimal)
```

- ◆ Tipos de dados (vírgula flutuante):

```
$a = 1.234;
```

```
$a = 1.2e3;
```

- ◆ Tipos de dados (strings):

```
<?php /* Atribuindo uma string. */
```

```
$str = "Isto é uma string";
```

```
?>
```

```

<?php
/* Atribuindo uma string. */
$str = "Isto é uma string";

/* Acrescentando a ela. */
$str = $str . " com mais um pouco de texto";

/* Outra maneira de acrescentar, inclui uma nova linha com escape.
*/
$str .= " e uma nova linha no fim.\n";

/* Esta string terminará sendo '<p>Número: 9</p>' */
$num = 9;
$str = "<p>Número: $num</p>";

```

numa string entre “ as variáveis são interpretadas

numa string entre ‘ as variáveis não são interpretadas

```

/* Esta será '<p>Número: $num</p>'
*/

```

```

$num = 9;

```

```

$str = '<p>Number: $num</p>';

```

```

/* Obtém o primeiro caracter da string
*/

```

```

$str = 'Isto é um teste.'; $first = $str
[0];

```

```

/* Obtém o último caracter de uma string. */

```

```

$str = 'Isto ainda é um teste.';

```

```

$last = $str[strlen($str)-1];

```

```

?>

```

◆ Tipos de dados (arrays):

```
$a[0] = "abc";  
$a[1] = "def";  
$b["foo"] = 13;
```

Matrizes agem atualmente como tabelas de 'hashing' (**matrizes associativas**) e matrizes indexadas (**vetores**).

```
$a[] = "olá"; // $a[2] == "olá"  
$a[] = "mundo"; // $a[3] == "mundo"
```

criar uma matriz simplesmente acrescentando valores à matriz

- as matrizes podem ser ordenadas usando as funções [asort\(\)](#), [arsort\(\)](#), [ksort\(\)](#), [rsort\(\)](#), [sort\(\)](#), [uasort\(\)](#), [usort\(\)](#), e [uksort\(\)](#)
- contar o número de itens em uma matriz usando a função [count\(\)](#)
- avançar numa matriz usando as funções [next\(\)](#) e [prev\(\)](#)

◆ Tipos de dados (arrays multi-dimensionais):

`$a[1] = $f; # exemplos de uma dimensão`

`$a["foo"] = $f;`

`$a[1][0] = $f; # duas dimensões`

`$a["foo"][2] = $f; # (você pode misturar índices numéricos e associativos)`

`$a[3]["bar"] = $f; # (você pode misturar índices numéricos e associativos)`

`$a["foo"][4]["bar"][0] = $f; # quatro dimensões!`

`$a[3]['bar'] = 'Bob';`

`echo "Isto não vai funcionar: $a[3][bar]";`

`$a[3]['bar'] = 'Bob';`

`echo "Isto vai funcionar: {$a[3][bar]}";`

- ◆ Tipos de dados (arrays multi-dimensionais):

```
$a["cor"] = "vermelho";  
$a["sabor"] = "doce";  
$a["forma"] = "redondo";  
$a["nome"] = "maçã";  
$a[3] = 4;
```

```
$a = array( "cor" => "vermelho",  
           "sabor" => "doce",  
           "forma" => "redondo",  
           "nome" => "maçã",  
           3 => 4  
);
```

```
$a = array ( "maçã" => array(  
    "cor" => "vermelho",  
    "sabor" => "doce",  
    "forma" => "redondo" ),  
    "laranja" => array(  
    "cor" => "laranja",  
    "sabor" => "azedo",  
    "forma" => "redondo" ),  
    "banana" => array(  
    "cor" => "amarelo",  
    "sabor" => "pastoso",  
    "forma" => "cilíndrico" )  
);  
echo $a["maçã"]["sabor"]; # imprimirá "doce"
```

◆ Variáveis:

- As variáveis no PHP são representadas por um cifrão(\$) seguido pelo nome da variável
- Os nomes de variável no PHP fazem distinção entre maiúsculas e minúsculas
- Um nome de variável válido se inicia com uma letra ou sublinhado, seguido de qualquer número de letras, algarismos ou sublinhados

```
$var = "Bob";  
$Var = "Joe";  
echo "$var, $Var"; // imprime "Bob, Joe"  
$4site = 'not yet'; // inválido; começa com algarismo  
$_4site = 'not yet'; // válido; começa com sublinhado  
$täyte = 'mansikka'; // válido; 'ä' é ASCII 228.
```

◆ Constantes:

```
<?php  
define("CONSTANT", "Alô mundo.");  
echo CONSTANT; // imprime "Alô  
mundo."  
?>
```

constantes pré-definidas

```
__FILE__  
__LINE__  
PHP_VERSION  
PHP_OS  
TRUE  
FALSE  
E_ERROR  
E_WARNING  
E_PARSE  
E_NOTICE  
E_ALL
```

◆ Operadores aritméticos:

operador	nome	resultado
$\$a + \b	Adição	Soma de $\$a$ e $\$b$.
$\$a - \b	Subtração	Diferença entre $\$a$ e $\$b$.
$\$a * \b	Multiplicação	Produto de $\$a$ e $\$b$.
$\$a / \b	Divisão	Quociente de $\$a$ por $\$b$.
$\$a \% \b	Módulo	Resto de $\$a$ dividido por $\$b$.

◆ Operadores de atribuição (“=”):

```
$a = ($b = 4) + 5; // $a é igual a 9 agora, e $b é igual a 4.
```

```
$a = 3;
```

```
$a += 5; // atribui 8 a $a, equivalente a: $a = $a + 5;
```

```
$b = "Olá ";
```

```
$b .= "Aí!"; // atribui "Olá Aí!" a $b, como $b = $b . "Aí!";
```

◆ Operadores de comparação:

exemplo	nome	resultado
<code>\$a == \$b</code>	Igual	Verdadeiro se \$a é igual a \$b.
<code>\$a === \$b</code>	Idêntico	Verdadeiro se \$a é igual a \$b, e eles são do mesmo tipo. (somente para PHP4)
<code>\$a != \$b</code>	Diferente	Verdadeiro se \$a não é igual a \$b.
<code>\$a < \$b</code>	Menor que	Verdadeiro se \$a é estritamente menor que \$b.
<code>\$a > \$b</code>	Maior que	Verdadeiro se \$a é estritamente maior que \$b.
<code>\$a <= \$b</code>	Menor ou igual	Verdadeiro se \$a é menor ou igual a \$b.
<code>\$a >= \$b</code>	Maior ou igual	Verdadeiro se \$a é maior ou igual a \$b.

◆ Operadores de incremento/decremento:

exemplo	nome	efeito
<code>++\$a</code>	Pré-incremento	Incrementa \$a de um, e então retorna \$a.
<code>\$a++</code>	Pós-incremento	Retorna \$a, e então incrementa \$a de um.
<code>--\$a</code>	Pré-decremento	Decrementa \$a de um, e então retorna \$a.
<code>\$a--</code>	Pós-decremento	Retorna \$a, e então decrementa \$a de um.

◆ Operadores lógicos:

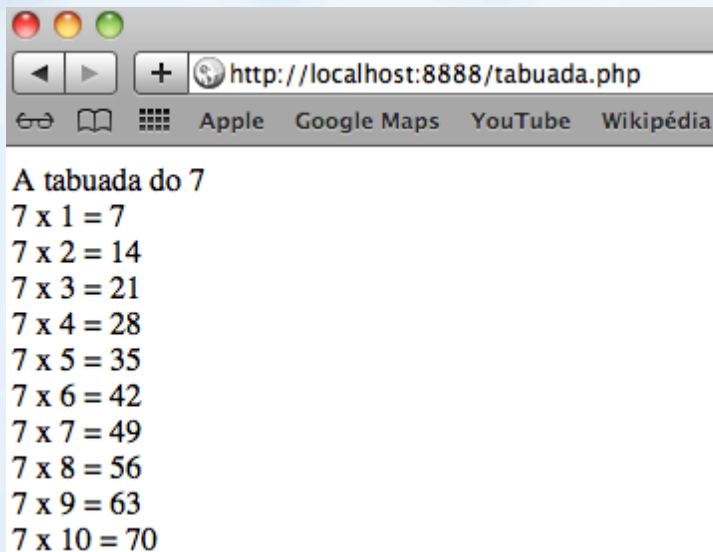
exemplo	nome	resultado
<code>\$a and \$b</code>	E	Verdadeiro se tanto \$a quanto \$b são verdadeiros.
<code>\$a or \$b</code>	OU	Verdadeiro se \$a ou \$b são verdadeiros.
<code>\$a xor \$b</code>	XOR	Verdadeiro se \$a ou \$b são verdadeiros, mas não ambos.
<code>! \$a</code>	NÃO	Verdadeiro se \$a não é verdadeiro.
<code>\$a && \$b</code>	E	Verdadeiro se tanto \$a quanto \$b são verdadeiros.
<code>\$a \$b</code>	OU	Verdadeiro se \$a ou \$b são verdadeiros.

◆ Operadores de strings:

```
$a = "Olá ";
$b = $a . "mundo!"; // agora $b contém "Olá mundo!"
$a = "Olá ";
$a .= "mundo!"; // agora $a contém "Olá mundo!"
```

◆ Exemplo de um *script* em PHP:

Tabuada.php



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8888/tabuada.php`. Below the address bar, there are search engines: Apple, Google Maps, YouTube, and Wikipédia. The main content area displays the output of the PHP script, which is a multiplication table for the number 7, ranging from 7 x 1 to 7 x 10.

```
A tabuada do 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

```
<html>
<head>
  <title>LPI/LW: PHP</title>
</head>
<body>
  <?php
    $n_tabuada = 7 ;
    echo "A tabuada do $n_tabuada <br>";
    for ($n=1; $n <= 10; $n++)
      echo "$n_tabuada x $n = " . $n_tabuada*$n . "<br>";
  ?>
</body>
</html>
```

◆ Estruturas de controlo:

```
if (condição)
    instrução ;
elseif (condição) {
    instrução1;
    ...
    instrução;
}
elseif (condição)
    instrução;
else
    instrução;
```

*if**switch*

```
switch ($i) {
    case 0: print "i é igual a 0";
            break;
    case 1: print "i é igual a 1";
            break;
    case 2: print "i é igual a 2";
            break;
    default: print "i não é igual a 0, 1 ou
2";
}
```


◆ Estruturas de controlo:

```
for (expr1; expr2; expr3)
    instrução;
```

for

foreach

```
foreach(array_expression as $value)
    instrução;
foreach(array_expression as $key => $value)
    instrução;
```

```
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}
foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}
while (list($key, $value) = each ($arr)) {
    echo "Key: $key; Value: $value<br>\n";
}
foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>\n";
}
```

◆ Estruturas de controlo:

```
while (condição)
```

```
    instrução;
```

```
while (condição): instrução ... endwhile;
```

while

continue

do ... while

```
do {
```

```
    instrução;
```

```
} while (condição);
```

```
while ($i++ < 5) {  
    echo "Mais externo<br>\n";  
    while (1) {  
        echo " Meio<br>\n";  
        while (1) {  
            echo " Inner<br>\n";  
            continue 3;  
        }  
        echo "This never gets output.<br>\n";  
    }  
    echo "Neither does this.<br>\n";  
}
```

◆ Funções:

```
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Função-exemplo.\n";
    return $retval; }
```

Passagem por referência

```
function add_some_extra(&$string) {
    $string .= 'e algo mais.';
}
$str = 'Isto é uma string, ';
add_some_extra($str);
echo $str; // outputs 'Isto é uma string, e algo mais.'
```

```
function takes_array($input) {
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

```
function foo ($bar) {
    $bar .= 'e algo mais.';
}
$str = 'Isto é uma string, ';
foo ($str);
echo $str; // outputs 'Isto é uma string, '
foo (&$str);
echo $str; // outputs 'Isto é uma string, e algo mais.'
```

◆ Funções:

```
function square ($num) {  
    return $num * $num;  
}  
echo square (4); // imprime '16'.
```

Retorno de 1 valor

```
function small_numbers() {  
    return array (0, 1, 2);  
}  
list ($zero, $one, $two) = small_numbers();
```

Retorno de 3 valores

◆ Funcionalidades básicas (echo):

```
<?php
echo("Hello World");
echo "print() also works without parentheses.";
echo "This spans
multiple lines. The newlines will be
output as well";
echo "This spans\nmultiple lines. The newlines will be\noutput as well."; print
"escaping characters is done \"Like this\".";

// You can use variables inside of an print statement
$foo = "foobar";
$bar = "barbaz";
echo "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
echo 'foo is $foo'; // foo is $foo
// If you are not using any other characters, you can just print variables
echo $foo; // foobar
```

```
echo <<<END
```

This uses the "here document" syntax to output

multiple lines with \$variable interpolation. Note

that the here document terminator must appear on a

line with just a semicolon no extra whitespace!

```
END;
```

```
?>
```

◆ Funcionalidades básicas (print):

```
<?php
print("Hello World");
print "print() also works without parentheses.";
print "This spans
multiple lines. The newlines will be
output as well";
print "This spans\nmultiple lines. The newlines will be\noutput as well."; print
"escaping characters is done \"Like this\".";
// You can use variables inside of an print statement
$foo = "foobar";
$bar = "barbaz";
print "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
print 'foo is $foo'; // foo is $foo
// If you are not using any other characters, you can just print variables
print $foo; // foobar
```

```
print <<<END
```

This uses the "here document" syntax to output

multiple lines with \$variable interpolation. Note

that the here document terminator must appear on a

line with just a semicolon no extra whitespace!

```
END;
```

```
?>
```

■ Funcionalidades básicas (printf):

```
<?php
$format = "The %s contains %d monkeys";
printf($format,$num,$location);

$format = "The %2$s contains %1$d monkeys. That's a nice %2$s full of %1$d monkeys.";
printf($format, $num, $location);
?>
```

■ sprintf(), sscanf(), fscanf() e flush().

■ O sistema de ficheiros (fopen):

`int fopen (string nomedoarquivo, string mode [, int use_include_path])`

O *mode* pode ser qualquer um dos seguintes:

- 'r' - Abrir somente para leitura; coloca o ponteiro de arquivo no começo do arquivo.
- 'r+' - Abrir para leitura e gravação; colocar o ponteiro de arquivo no começo do arquivo.
- 'w' - Abrir somente para gravação; colocar o ponteiro de arquivo no começo do arquivo e truncar o arquivo para tamanho zero. Se o arquivo não existir, tentar criá-lo.
- 'w+' - Abrir para leitura e escrita; colocar o ponteiro de arquivo no início do arquivo e truncar o arquivo para tamanho zero. Se o arquivo não existir, tentar criá-lo.
- 'a' - Abrir o arquivo somente para escrita; colocar o ponteiro de arquivo no fim do arquivo. Se o arquivo não existe, tentar criá-lo.
- 'a+' - Abrir o arquivo para leitura e gravação; colocar o ponteiro no fim do arquivo. Se o arquivo não existe, tentar criá-lo.

Nota: O *mode* pode conter a letra 'b'. Isto é útil somente em sistemas que diferenciam entre arquivos binários e texto.

■ O sistema de ficheiros (fopen):

```
$fp = fopen ("/home/rasmus/arquivo.txt", "r");  
$fp = fopen ("/home/rasmus/arquivo.gif", "wb");  
$fp = fopen ("http://www.php.net/", "r");  
$fp = fopen ("ftp://usuario:senha@exemplo.com.br/", "w");  
$fp = fopen ("c:\\data\\info.txt", "r");
```

Windows

- `fopen()`, `fclose()`, `feof()`, `ftell()`, `rewind()`
- `fgets()`, `fgetc()`, `fputs()`

- O sistema de ficheiros (fileexists(), filetype(), filesize()):
 - tipo de ficheiros: fifo, char, block, link, file, dir e unknown
- copy(), delete(), dirname(), rmdir(), chdir(), mkdir(), ...
- fpassthru():

```
if (! $fp = fopen ("figura.gif", "rb")) {  
    echo ("Não foi possível abrir o  
    ficheiro\n");  
}  
else {  
    fpassthru ($fp);  
}
```

Reads to EOF on the given file pointer from the current position and writes the results to the output buffer.

■ Execução de programas externos (exec(), system()):

```
$line = exec('dir', $output, $error);  
while (list(, $line) = each($output)) {  
    echo $line, "<BR>\n";  
}  
if ($error) {  
    echo "Ocorreu um erro com o código: <$error><BR>\n";  
}
```

exec(**escapeshellcmd**(\$comando, \$output, \$error);
exec(**escapeshellargs**(\$comando, \$output, \$error);

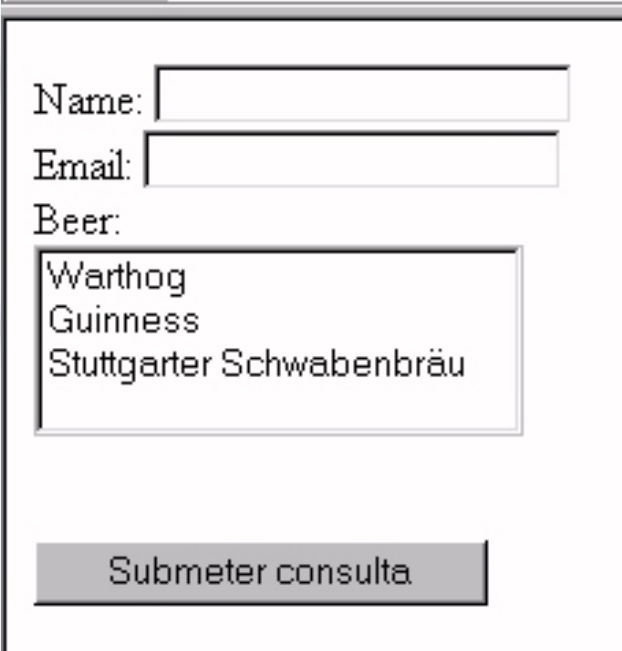
// destroy a single variable
unset(\$foo);

// destroy a single element of an array
unset(\$bar['quux']);

// destroy more than one variable
unset(\$foo1, \$foo2, \$foo3);

■ Formulários em HTML (GET e POST):

```
<form action="foo.php" method="post">  
  Name: <input type="text" name="username"><br>  
  <input type="submit">  
</form>
```



Name:

Email:

Beer:

Warthog

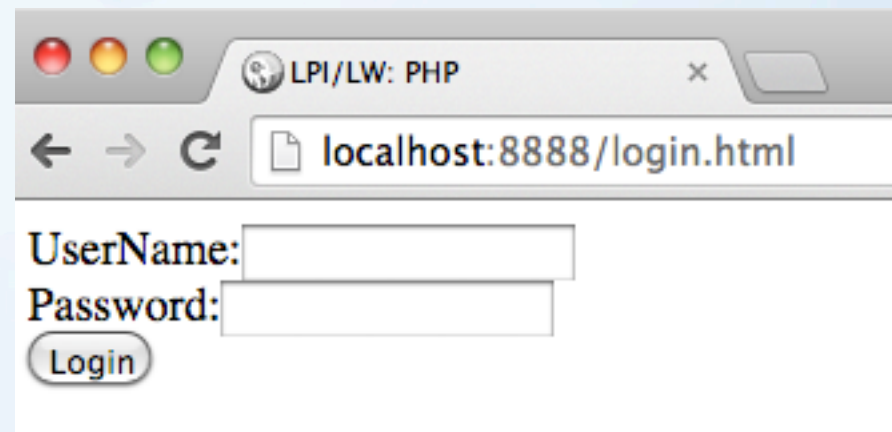
Guinness

Stuttgarter Schwabenbräu

■ Formulários em HTML:

login.html

```
<html>
<head>
  <title>LPI/LW: PHP</title>
</head>
<body>
  <form method="POST" action="verifica_login.php">
    UserName:</td><td><input type="text" name="user"><br>
    Password:</td><td><input type="password" name="password"><br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
```



A screenshot of a web browser window. The title bar shows 'LPI/LW: PHP'. The address bar shows 'localhost:8888/login.html'. The page content displays a login form with two text input fields labeled 'UserName:' and 'Password:', and a 'Login' button below them.

■ Formulários em HTML:

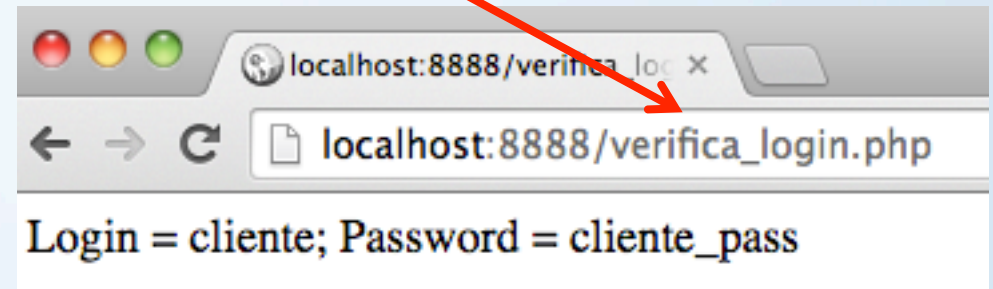
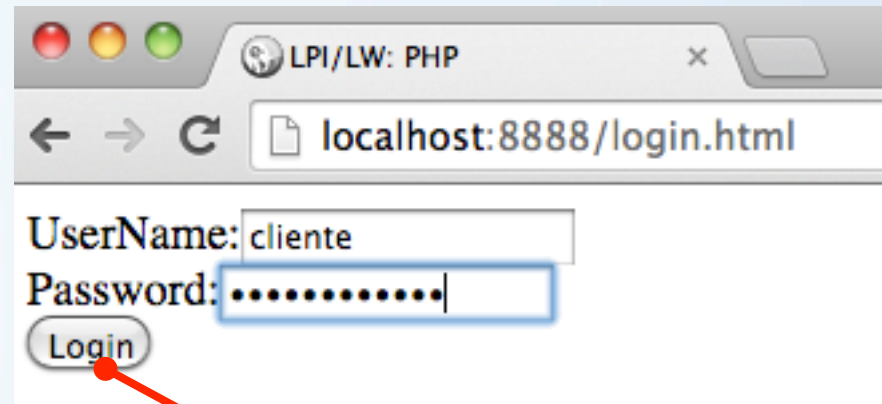
login.html

```
<html>
<head>
  <title>LPI/LW: PHP</title>
</head>
<body>
  <form method="POST" action="verifica_login.php">
    UserName:</td><td><input type="text"
    name="user"><br>
    Password:</td><td><input type="password"
    name="password"><br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
```

verifica_login.php

```
<?php
$user = $_POST["user"];
$pass = $_POST["password"];

echo "Login = $user; Password = $pass";
?>
```



■ Formulários em HTML:

```
<html>
<body>
<?php
printf ("O utilizador $nome com o email $email gosta da cerveja $beer\n");
?>
</body>
</html>
```

Se a variável de configuração *register_globals* estiver activada pode-se aceder às variáveis EGPCS (Environment, GET, POST, Cookie, Server) como globais ou não, ou seja, através das matrizes globais `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE` e `$_SERVER`

- Bases de Dados (MySQL):
 - Estas funções permitem o acesso a servidores de bases de dados MySQL
 - <http://www.mysql.com/>
 - é necessário compilar o PHP com o MySQL
 - o comportamento das funções do MySQL são afectadas pelo ficheiro global de configuração php.ini

■ Bases de Dados (MySQL):

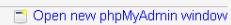
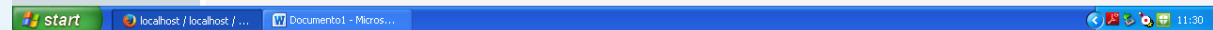
- Pode ser alterado com as funções `ini_set()`, `ini_alter()`
que é uma alias para `ini_set()`, `ini_restore()`, `ini_get()`

string **ini_set** (string varname, string newvalue)

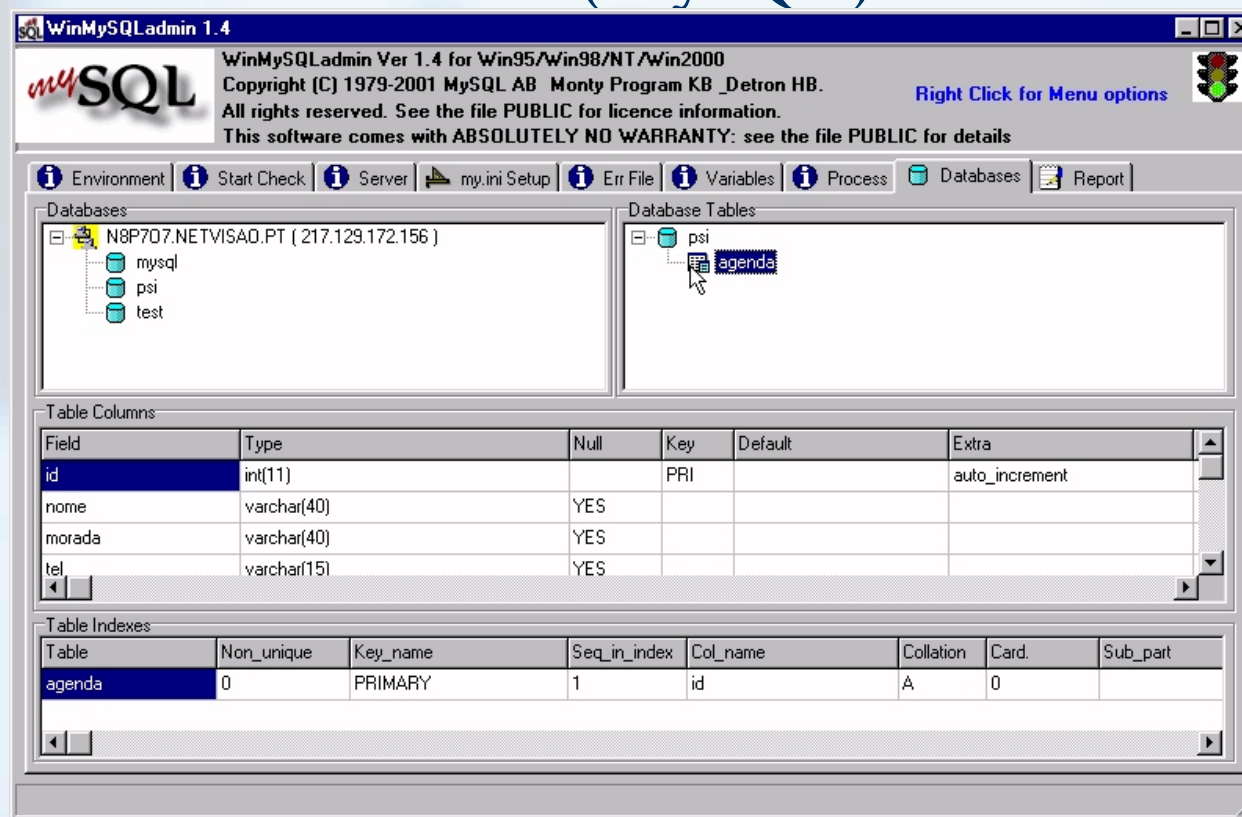
•Sets the value of the given configuration option. Returns the old value on success, **FALSE** on failure. The configuration option will keep this new value during the script's execution, and will be restored at the script's ending.
•Not all the available options can be changed using **ini_set()**.

Name	Default	Changeable
mysql.allow_persistent	"On"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL

- Bases de Dados (MySQL):
 - Utilitários de administração de MySQL:
 - ◆ PHPMyAdmin
 - ◆ NetAdmin www.it-netservice.de/support/demo
 - ◆ Artronic artronic.hr/mysql/index.htm
 - ◆ DBTools tucows.terra.com.br/business/adnload/225125_93401.html
 - ◆ Webmin www.webmin.com/webmin/ (Linux)
 - ◆ ...



■ Bases de Dados (MySQL):



■ Bases de Datos (MySQL):

resource **mysql_connect** ([string server [, string username [, string password [, bool new_link]]]])

mysql_connect() establishes a connection to a MySQL server. Returns a MySQL link identifier on success, or **FALSE** on failure.

resource **mysql_pconnect** ([string server [, string username [, string password]]])

mysql_pconnect() establishes a connection to a MySQL server. Returns a positive MySQL persistent link identifier on success, or **FALSE** on error. The link will remain open for future use (**mysql_close()** will not close links established by **mysql_pconnect()**).

■ Bases de Datos (MySQL):

bool **mysql_close** ([resource link_identifier])

mysql_close() closes the connection to the MySQL server that's associated with the specified link identifier. Returns **TRUE** on success, or **FALSE** on failure.

bool **mysql_select_db** (string database_name [, resource link_identifier])

mysql_select_db() sets the current active database on the server that's associated with the specified link identifier. Returns **TRUE** on success, or **FALSE** on failure.

■ Bases de Datos (MySQL):

resource **mysql_db_query** (string database, string query [, resource link_identifier])

mysql_db_query() selects a database and executes a query on it. Returns a positive MySQL result resource to the query result, or **FALSE** on error.

resource **mysql_query** (string query [, resource link_identifier [, int result_mode]])

mysql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. Only for SELECT, SHOW, EXPLAIN or DESCRIBE statements **mysql_query()** returns a resource identifier or **FALSE** if the query was not executed correctly. For other type of SQL statements, **mysql_query()** returns **TRUE** on success and **FALSE** on error. The query string should not end with a semicolon.

■ Bases de Datos (MySQL):

array **mysql_fetch_row** (resource result)

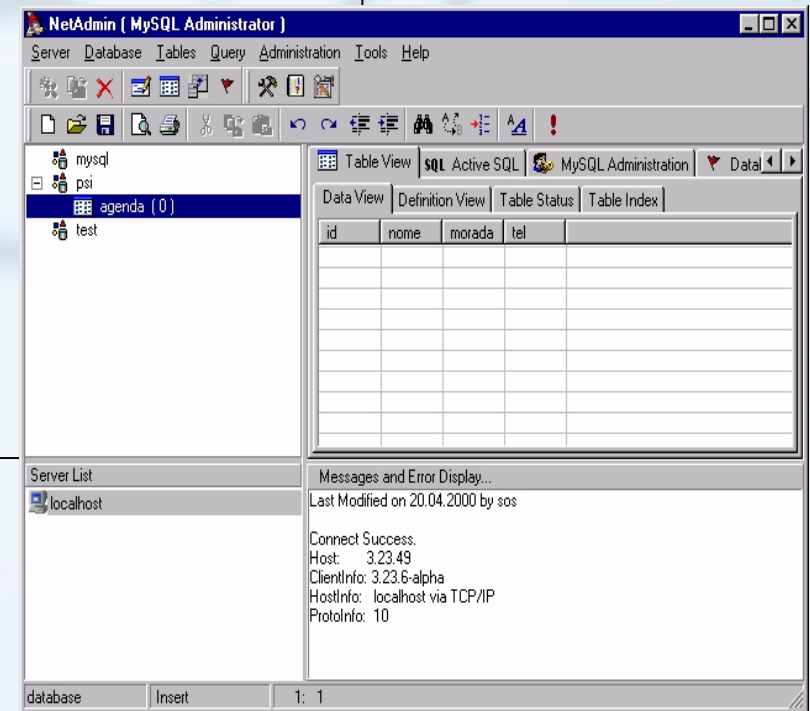
mysql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0. Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

array **mysql_fetch_array** (resource result [, int result_type])

mysql_fetch_array() is an extended version of **mysql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys. Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

■ Bases de Dados (MySQL):

```
<?php
$ligacao = mysql_connect ("localhost", "cmoa", "cmoa_pass")
    or die ("Não foi possível efectuar a ligação com a BD!!!");
mysql_select_db ("psi", $ligacao);
$sql = "select nome, tel from agenda" ;
$resultado = mysql_query ($sql, $ligacao);
while ($registro = mysql_fetch_row ($resultado)) {
    print "$registro[0] ----- $registro[1] <br>";
}
mysql_close ($ligacao);
?>
```



■ Bases de Dados (MySQL):

```
<?php
$ligacao = mysql_connect ("localhost", "cmoa", "cmoa_pass")
    or die ("Não foi possível efectuar a ligação com a BD!!!");
$sql = "select nome, tel from agenda order by nome asc" ;
$resultado = mysql_db_query ("psi", $sql, $ligacao);
if ($resultado) {
    while ($registo = mysql_fetch_array ($resultado)) {
        $nome = registo['nome'];
        $tel = registo['tel'];
        print "$nome ----- $tel <br>";
    }
}
else { print "Não há registos ..."; }
mysql_close ($ligacao);
?>
```

■ Bases de Datos (MySQL):

int **mysql_affected_rows** ([resource link_identifier])

mysql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query.

int **mysql_num_fields** (resource result)

mysql_num_fields() returns the number of fields in a result set.

int **mysql_num_rows** (resource result)

mysql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements.

■ Bases de Dados (MySQL):

int **mysql_field_len** (resource result, int field_offset)

mysql_field_len() returns the length of the specified field.

bool **mysql_free_result** (resource result)

mysql_free_result() will free all memory associated with the result identifier *result*. All associated result memory is automatically freed at the end of the script's execution. Retorna **TRUE** em caso de sucesso ou **FALSE** em falhas.

mixed **mysql_result** (resource result, int row [, mixed field])

mysql_result() returns the contents of one cell from a MySQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field name (tablename.fieldname).

■ Bases de Dados (MySQL):

```
<?php
$ligacao = mysql_connect ("localhost", "cmoa", "cmoa_pass")
    or die ("Não foi possível efectuar a ligação com a BD!!!");
mysql_select_db ("psi", $ligacao)
    or die ("Não foi possível efectuar a ligação com a BD(2)!!!");
$sql = "select * from agenda" ;
$resultado = mysql_query ($sql, $ligacao);
$num_regs = mysql_num_rows ($resultado);
$num_campos = mysql_num_fields ($resultado);
print ("número total de registos seleccionados = $num_regs <p>");
print ("número total de campos por registo = $num_campos <p>");
if ($num_regs >0) {
    ... print ("<table>");
    for ($linha=0; $linha < $num_regs; $linhs++) {
        for ($coluna=0; $coluna < $num_campos; $num_campos++) {
            $name_field = mysql_field_name ($resultado, $coluna);
            $val_field = mysql_result ($resultado, $linha, "$name_field");
            print ("<td> $val_field </td>");
        }
        print ("<tr>");
    }
}
print </table>");
mysql_free_result ($resultado);
mysql_close ($ligacao);
?>
```

■ Bases de Dados (MySQL):

```
<?php
$ligacao = mysql_connect ("localhost", "cmoa", "cmoa_pass")
    or die ("Não foi possível efectuar a ligação com a BD!!!");
mysql_select_db ("psi", $ligacao)
    or die ("Não foi possível efectuar a ligação com a BD(2)!!!");
$sql = "insert into agenda (nome, morada, tel) values ('$NOME', '$MOR', '$TEL')";
$resultado = mysql_query ($sql, $ligacao);
$num_regs = mysql_affected_rows ($resultado);
if ($num_regs) { print ("Operação efectuada com sucesso."); }
mysql_close ($ligacao);
?>
```

■ Bases de Dados (MySQL):

```
<?php
$ligacao = mysql_connect ("localhost", "cmoa", "cmoa_pass")
    or die ("Não foi possível efectuar a ligação com a BD!!!");
mysql_select_db ("psi", $ligacao)
    or die ("Não foi possível efectuar a ligação com a BD(2)!!!");
$sql = "update agenda set morada='MORADA' where id='$ID'";
$resultado = mysql_query ($sql, $ligacao);
$num_reg = mysql_affected_rows ($resultado);
if ($num_reg) { print ("Operação efectuada com sucesso."); }
mysql_close ($ligacao);
?>
```

```
...
$sql = "delete from agenda where id='$ID'";
...
```

■ Gestão de sessões:

■ porquê gestão de sessões?!

- Permite acompanhar e analisar todas as acções dum utilizador (durante um período de tempo) enquanto estiver no *site*.

■ funcionalidades duma gestão de sessão:

- armazenamento de informação de sessão no servidor
- identificadores únicos (gerados aleatoriamente)
- O identificador único é gravado no cliente usando *cookies*, GET/POST ou *path*

O HTTP não permite manter um estado.

- **Gestão de sessões:**
 - no sistema de gestão de sessão as opções de configuração (existentes no ficheiro php.ini) podem ser alteradas:
 - **session.save_handler** defines the name of the handler which is used for storing and retrieving data associated with a session. Defaults to files
 - **session.save_path** defines the argument which is passed to the save handler. If you choose the default files handler, this is the path where the files are created. Defaults to /tmp. If session.save_path's path depth is more than 2, garbage collection will not be performed
 - **session.name** specifies the name of the session which is used as cookie name. It should only contain alphanumeric characters. Defaults to PHPSESSID
 - **session.auto_start** specifies whether the session module starts a session automatically on request startup. Defaults to 0 (disabled)
 - **session.cookie_lifetime** specifies the lifetime of the cookie in seconds which is sent to the browser. The value 0 means "until the browser is closed." Defaults to 0
 - **session.serialize_handler** defines the name of the handler which is used to serialize/deserialize data. Currently, a PHP internal format (name php) and WDDX is supported (name wddx). WDDX is only available, if PHP is compiled with WDDX support. Defaults to php
 - **session.use_cookies** specifies whether the module will use cookies to store the session id on the client side. Defaults to 1 (enabled)
 - **session.cookie_path** specifies path to set in session_cookie. Defaults to /.
 - ...

■ Gestão de sessões:

bool **session_start** (void)

session_start() creates a session (or resumes the current one based on the session id being passed via a GET variable or a cookie). This function always returns **TRUE**.

bool **session_destroy** (void)

session_destroy() destroys all of the data associated with the current session. This function returns **TRUE** on success and **FALSE** on failure to destroy the session data.

■ Gestão de sessões:

string **session_name** ([string name])

session_name() returns the name of the current session. If *name* is specified, the name of the current session is changed to its value. The session name references the session id in cookies and URLs. It should contain only alphanumeric characters.

string **session_save_path** ([string path])

session_save_path() returns the path of the current directory used to save session data. If *path* is specified, the path to which data is saved will be changed.

string **session_id** ([string id])

session_id() returns the session id for the current session. If *id* is specified, it will replace the current session id.

■ Gestão de sessões:

bool **session_register** (mixed name [, mixed ...])

session_register () só é válida para PHP < 5.4.0

session_register() accepts a variable number of arguments, any of which can be either a string holding the name of a variable or an array consisting of variable names or other arrays. For each name, **session_register()** registers the global variable with that name in the current session.

bool **session_unregister** (string name)

session_unregister() unregisters (forgets) the global variable named *name* from the current session. This function returns **TRUE** when the variable is successfully unregistered from the session.

string **session_encode** (void)

session_encode() returns a string with the contents of the current session encoded within.

■ Gestão de sessões:

bool session_decode (string data)

session_decode() decodes the session data in *data*, setting variables stored in the session.

session_cache_expire -- Return current cache expire

session_cache_limiter -- Get and/or set the current cache limiter

session_get_cookie_params -- Get the session cookie parameters

session_module_name -- Get and/or set the current session module

session_readonly -- Begin session - reinitializes freezed variables, but no writeback on request end

session_set_cookie_params -- Set the session cookie parameters

session_set_save_handler -- Sets user-level session storage functions

session_unset -- Free all session variables

session_write_close -- Write session data and end session

■ Gestão de sessões:

```
<?php
// Initialize the session.
// If you are using session_name("something"), don't forget it now!
session_start();
// Unset all of the session variables.
session_unset();
// Finally, destroy the session.
session_destroy();
?>
```

```
<?php
// set the session name to WebsiteID
$previous_name = session_name("WebsiteID");
echo "The previous session name was $previous_name<p>";
?>
```

■ Gestão de sessões:

```
<?php  
session_start();  
$var_sessao = "exemplo de var sessao";  
session_register ("var_sessao");  
?>
```

Noutra página

```
<?php  
session_start();  
echo "O valor de var sessao é $var_sessao";  
?>
```

■ Gestão de sessões:

```
<?php  
session_start();  
session_register("contador");  
$contador++;  
echo "A página foi acedida <b>$contador</b> vezes.";  
?>
```


■ Gestão de sessões:

Outras formas de criar variáveis de sessão.

```
<?php
$barney = "A big purple dinosaur.";
session_register("barney");

$HTTP_SESSION_VARS["zim"] = "An invader from another planet.";

# the auto-global $_SESSION array was introduced in PHP 4.1.0
$_SESSION["spongebob"] = "He's got square pants.";
?>
```

■ Gestão de sessões:

ex_cor.php

```
<?php
session_start();
session_register("cfundo");
?>

<html>
<body>
Esta página exemplifica a utilização de variáveis de sessão.<br>
<form action=ex_cor_regista.php method=post>
Escolha a cor de fundo da sua página.<br>
<select name="cfundof">
    <option value="#ffffff">Branco
    <option value="#ff0000">Vermelho
    <option value="#0000ff">Azul
    <option value="#00ff00">Verde
</select>
</form>
<br>
<a href="ex_cor_pag1.php">página de teste 1.</a><br>
<a href="ex_cor_pag2.php">página de teste 2.</a><br>
</body>
</html>
```

■ Gestão de sessões:

ex_cor_regista.php

```
<?php
session_start();
$cfundo=$cfundof;
printf ("A variável de sessão foi actualizada.<br>
Voltar à página anterior. <a href=\"ex_cor.php\"<a>");
?>
```

ex_cor_pag1.php

```
<?php
session_start();
printf ("<html><body> bgcolor=%s> Página de teste 1</b><br><br> Voltar à página anterior.
<a href=\"ex_cor.php\"<a></body></html>", $cfundo);
?>
```

■ Gestão de sessões:

ex_cor_pag2.php

```
<?php
session_start();
printf("<html><body bgcolor=%s>Página de teste 2</b><br><br> Voltar à página anterior.
<a href='\"ex_cor.php\"'<a></body></html>", $fundo);
?>
```

■ Gestão de sessões:

login.php

```
<?php
ob_start();
$username = $_POST["uname"];
$password = $_POST["pass"];

$ligacao = mysql_connect ("localhost", "root", "root")
    or die ("Não foi possível efectuar a ligação com a BD!!!");

$sql = "select uname, passwd from utilizador where uname='" . $username . "' and passwd='" . $password . "'";
$resultado = mysql_db_query ("lpi", $sql, $ligacao);

if ($resultado) {
    session_start();
    $_SESSION["uname"] = $username ;
    $_SESSION["passwd"] = $password ;
    header ("success.php");
}

else
    print ("Invalid password <a href='index.html'>try again</a>");

?>
```

■ Gestão de sessões:

sucess.php

```
<?php
    session_start();
    if (($_SESSION["uname"] == NULL) || (!isset($_SESSION["uname"]))) {
?>
You are not logged in<br/>
<a href="index.html">Please Login</a>
<?php
    }
    else {
?>
Welcome <?php echo $_SESSION["uname"]?>
<a href='logout.php'>Log out</a>
<?php
    }
?>
```

logout.php

```
<?php
    ob_start();

    session_start();
    $_SESSION["uname"] = null;
    unset($_SESSION["uname"]);
    session_destroy();
    header("index.html");
?>
```