Respostas dos Exercícios

PHP 5 - Conceitos, Programação e Integração com Banco de Dados

Capítulo 1

1. PHP é acrônimo de Hypertext Preprocessor (pré-processador de hipertexto), uma poderosa linguagem de programação open source, mundialmente utilizada, principalmente no ambiente web (apesar de existir a versão PHP-GTK para ambiente desktop).

2. # cd apache_2.0.x;

   # ./configure --prefix=/www

   # cd ../php-x.x.x

   # ./configure --with-mysql --with-apache=../apache_2.0.x --enable-ftp

   # make

   # make install

   # cd ../apache_2.0.x

   # ./configure --activate-module=src/modules/php4/libphp5.a

   # make

   # make install

   # cd ../php-x.x.x

   # cp php.ini-dist /usr/local/lib/php.ini

3. cgi.force_redirect, deve ser ajustado para 1.

4. phpinfo(). um programa simples: <?=phpinfo();?>

5. Linux: devemos compilar o PHP com a extensão desejada, utilizando o parâmetro de configuração apropriado a cada extensão (por exemplo, –with-mysqli para a extensão mysqli do PHP).

   Windows: habilitar a extensão desejada no arquivo php.ini, utilizando a sintaxe extension=nome_da_extensão, por exempolo extension-php_mysqli.dll para a habilitar a extensão mysqli.

Capítulo 2

1. <?php , <?, <%, <?= , <%= e <script language="PHP"> para iniciar um script PHP e ?> , %> e </script> para finalizar.
2. Sim, basta incluir os script PHP nas tags aceitas.
3. Basta utilizar os delimitadores /* e */.
4. Ponto-e-virgula (;). Devemos utilizá-lo sempre, apesar de não ser necessário o delimitador na última instrução do script, antes do delimitador de finalização do código PHP.

Capítulo 3

1. Oito tipos, dispostos em três grupos: escalares, compostos e especiais.
2. Arrays e objetos.
3. Não. Porque o PHP determina automaticamente o tipo da variável na primeira atribuição a ela, mas é possível forçar um tipo de dado por meio das funções (int), (string), (object), etc.
4. A função **gettype()** retorna o tipo de uma variável.
5. is_int
6. integer
   double
   float(5.0E+14)
7. Quatorze dígitos decimais.
8. Aspas duplas, aspas simples e a sintaxe herodoc (<<<). Aspas duplas permitem a substituição de variáveis em sua estrutura, além de admitirem caracteres especiais, já as aspas simples não fazem a interpolação de variáveis nem os caracteres especiais. A sintaxe herodoc permite a inclusão de textos logos, admite a interpolação de variáveis e a inclusão dos mesmos caracteres especiais de aspas duplas.
9. Arrays são mapas ordenados com índices numéricos ou alfanuméricos (chaves associativas).

```
$_array = Array(10,20,30,1);
$_array = Array("código"=>10,"nome"=>"José");
$_array = Array(1=>array(5,10,1),3=>array(1,1,1));
```

10. bool(TRUE)

11. O PHP tenta converter uma variável ou escalar em um tipo compatível com a operação que está sendo realizada, por exemplo, a soma de um inteiro com uma string, em que o PHP tenta converter a string ou parte dela (seu início) em um inteiro para que a operação seja possível.
12. NULL

Capítulo 4

1. Depois de definida, o valor de uma constante não pode ser alterado.
2. Define(nome,valor).
3. __LINE__, __FILE__, __CLASS__, __METHOD__, __FUNCTION__
4. get_defined_constants()
5. Variável é um elemento do PHP para armazenar valores dinamicamente. As regras de formação do nome de uma variável são:

   Deve iniciar por uma letra ou pelo símbolo subscrito (_).

   Pode conter letra, números ou subscrito (_).
6. São variáveis predefinidas disponíveis em qualquer lugar de um script PHP (mesmo dentro de funções e classes). Exemplos: $_SERVER, $_SESSION, $_COOKIE, $GLOBALS, $_GET, $_POST, $_FILES, $_ENV, $_REQUEST.
7. global nome_da_variavel. Sim, existe outra forma, basta acessarmos a variável superglobal $GLOBALS[“nome_da_vairavel”], por exemplo: podemos definir global $_var ou então acessá-la com $GLOBALS[“_var”].
8. $codigo = 1000
9. Esses comandos enviam dados para o browser. Exemplos:

   echo “PHP5”;

   echo date(“d/m/Y H:i:s”);

   print “Editora Érica”;
10. O argumento é tratado como inteiro e exibido como o caractere ASCII correspondente a esse inteiro. Exemplo

    printf(‘%c’,65); // A
11. 001900092004

Capítulo 5

1. São operadores que trabalham com dois argumentos, ,por exemplo o operador + (adição) ou o operador << (deslocamento à esquerda).

2. Adição(+), subtração(-), multiplicação(*), divisão (/) e módulo(%).

$_a + 10;

$_a - $_b;

$_a * 5;

$_a / $_c;

$_a % 2;

3. O += é um operador de atribuição com adição, fazendo com que não seja necessária a repetição no lado direito do operador da variável que irá receber o resultado, por exemplo, $_a += 10 é o mesmo que $_a = $_a + 10.

4. Esse operador realiza o deslocamento de bits à esquerda, ou seja, a expressão à esquerda do operador será deslocada do número de bits informados na expressão à direita do operador, por exemplo, 4 << 2 será igual a 8. 16 << 2 = 64.

5. Ambos verificam a igualdade, porém o operador === verifica além da igualdade de valor a igualdade de tipo. Exemplo

   $_a = 1;

   if($_a==TRUE) retorna verdadeiro.

   if($_a===TRUE) retorna falso, pois $_a não é do tipo booleano.

   $_a = FALSE;

   if($_a==FALSE) retorna verdadeiro.

   if($_a===FALSE) retorna verdadeiro.

6. expr ? expr_verdadeiro : expr_falso. Se expr for avaliada como verdadeiro, então expr_verdadeiro é avaliada; caso contrário, expr_falso é avaliada e é o mesmo que:

```
if(expr) {
      expr_verdadeiro
   } else {
      expr_falso
   }
```

7. O operador @ permite que os erros gerados por expressões não sejam exibidos no browser.

8. O PHP dispõe dos operadores ++ e – para incremento e decremento respectivamente. Esses operadores têm comportamentos diferentes Quando o resultado é atribuído a outra variável. Assim, caso o operador seja inserido à esquerda do operando ($_a = ++$_b;) ,

o PHP primeiro incrementa/decrementa o operando para depois atribuir seu resultado à variável, já se o operador estiver à direita ($_a = $_b++;), a atribuição é realizada antes de o operando ser incrementado/decrementado.

9. AND, OR, XOR, !, || e && (E, OU, Ou exclusivo, negação, OU e E respectivamente)
10. Basta utilizar parênteses, por exemplo, 7*2+8/4 = 16, mas 7*(2+8)/4 = 17,5.

Capítulo 6

1. A estrutura de controle if...elseif...else permite a execução condicional de trechos do script, ou seja, caso a expressão avaliada em if seja verdadeira, um grupo de comandos é executado, senão as expressões elseif existentes (se existirem) são avaliadas, e caso uma seja verdadeira, o grupo de comandos relacionado a esse elseif será executado (as expressões elseif são executadas seqüencialmente); caso contrário, o grupo de comandos dentro de else (caso exista) é executado. Além da forma padrão if(expressão) { comandos } elseif(expressão) { comandos} ..... else {comandos}, existe o formato:

```
if(expressão) :
   comandos
elseif(expressão) :
   comandos
....
else :
   comandos
endif
```

2. As estruturas while e do..while executam um grupo de comandos (ou apenas um comando) repetidas vezes, enquanto uma determinada condição for verdadeira, ou seja, a cada interação do loop a expressão fornecida é avaliada e caso seja verdadeira, uma nova rodada é executada; caso contrário (se for avaliada com falsa), a repetição é interrompida.
3. Sim, enquanto while avalia a expressão que condiciona o loop no início do bloco de comandos, do ... while avalia no final do bloco, desta forma while executa o loop 0 ou mais vezes, enquanto do..while executa o bloco pelo menos uma vez.
4. Deve-se utilizar a instrução **for**, cuja sintaxe é for(variavel;condição;expressão).
5. foreach. Exemplo foreach($_a as $k=>$v), em que $k conterá o índice e $v o valor contido no elemento do array.
6. O valor final é 6.
7. **break**: interrompe a execução do loop (nenhum comando contido no bloco de comandos após break será executado).

   continue: interrompe a iteração atual do loop (ou seja, nenhum comando do bloco após continue será executado) indo para a próxima. Ambos permitem que seja informado quantos níveis serão considerados, sendo o padrão 1, ou seja, somente o nível atual.

8. Ambos inserem um script PHP no script atual, porém **include_once** verifica antes se o mesmo script não foi inserido anteriormente (verifica o nome do script não seu conteúdo), evitando desta forma a duplicação de código, funções, classes, etc. Exemplo:

```
include(“cadastro.inc”);
include_once(“class_html.inc”);
```

Capítulo 7

1. Uma função é um subprograma que executa uma série de instruções e pode retornar ou não um valor como resultado.

   Por meio da definição de parâmetros de entrada da função ou do envio de parâmetros dinâmicos (não definidos na função, porém tratados por ela com as instruções **func_num_args**, **func_get_arg** e **func_get_args**).

2. Argumentos passados como valor pertencem ao escopo da função, ou seja, as alterações realizadas não refletem nas variáveis originais. Já a passagem por referência faz com que as alterações realizadas na função sejam refletidas na variável referenciada. Exemplos:

```
function soma(&$a,$b,$c) {
   $b += $c;
   $a  += $b;
   return $a;
}
$_v1 =10;
$_v2 = 5;
$_v3 = 2;
echo soma($_v1,$_v2,$_v3);
echo “<br/>_v1 = $_v1”;
```

   O resultado será: 17 e _v1=17

3. Basta defini-lo no formato argumento=valor_padrão.
4. O argumento $v2 precisa de um valor-padrão ou o argumento $v1 não deve ter um valor--padrão; caso contrário, o PHP aponta um erro.
5. **func_num_args**: retorna o número de argumentos enviados à função.

   func_get_arg: recupera um argumento enviado à função (deve ser informado qual argumento deve ser recuperado).

   func_get_args: retorna um array com todos os argumentos enviados à função.

6. Sim é possível, mas se a função não tratar esses argumentos com as funções func_num_args, func_get_arg e func_get_args eles serão desprezados.

7. É uma função que chama a si mesma. O exemplo clássico é a função fatorial, em que fatorial(n) = n * fatorial(n-1).
8. Serve para a criação de funções anônimas (ou funções lambda). Exemplo: $_f = create_function('$a,$b','return pow($a,$b);');
9. function_exists.
10. A função **register_shutdown_function** permite que sejam registradas funções (uma ou mais) para execução após o término de um script. Sua sintaxe é:

```
register_shutdown_function(nome_da_função)
```

Capítulo 8

1. Esse parâmetro define a ação que será executada quando o formulário for submetido. Geralmente é um nome de um programa PHP que irá tratar os formulários.
2. A diferença entre GET e POST está basicamente na capacidade de envio de dados, uma vez que GET é limitado à cerca de 2KB de dados e POST não possui limite. Outra diferença está na forma de envio. GET envia os dados anexados ao nome do script informado em ACTION (utiliza-se para isso o símbolo? após o nome do script, seguido da relação de campos e seus respectivos valores separados por &). Já no método POST os dados são enviados no corpo da mensagem que será enviada ao servidor.
3. type="password"
4. SELECT permite exibir uma caixa com várias opções para que o usuário selecione uma ou mais (conforme parâmetro no próprio SELECT). As opções são informadas com <OPTION>. Sua sintaxe é:

```
<SELECT NAME="nome" SIZE="numero_linhas" MULTIPLE>
   <OPTION VALUE="valor">Opção</OPTION>
   ....
</SELECT>
```

5.

```
<FORM ACTION="verifica.php5"  METHOD="POST">
<TABLE>
<TR>
   <TD>Código:</TD>
   <TD>
      <INPUT TYPE="text" NAME="F_CODIGO" SIZE=10>
   </TD>
</TR>
<TR>
   <TD>Nome:</TD>
   <TD><INPUT TYPE="text" NAME="F_NOME" SIZE=30></TD>
</TR>
<TR>
```

```
      <TD>Faixa Etária:</TD>
      <TD>
         <SELECT NAME="F_ETARIA" SIZE=1>
            <OPTION VALUE=1>Até 10 anos</OPTION>
            <OPTION VALUE=2>Entre 10 e 15 anos</OPTION>
            <OPTION VALUE=3>Entre 15 e 20 anos</OPTION>
            <OPTION VALUE=4>Entre 20 e 30 anos</OPTION>
            <OPTION VALUE=5>Acima de 30 anos</OPTION>
         </SELECT>
      </TD>
   </TR>
   <TR>
      <TD COLSPAN=2 ALIGN="CENTER">
         <INPUT TYPE="SUBMIT" VALUE=" OK ">
      </TD>
   </TR>
</TABLE>
</FORM>
```

6.

```
   <html>
   <body>
   <table border="0" cellpadding="3" cellspacing="0" width="400">
            <tr>
           <td align="right" bgcolor="#8CDAFF">
             <?=date("d-m-Y H:i:s")?> 
           </td>
         </tr>
   </table>
   <?php
   $erro = Array(); // Array com os erros encontrados
   // Vamos verificar se existe algum campo não informado
   foreach($_POST as $chv => $vlr) {
      if($vlr=="" AND substr($chv,0,2)=="F_") {
         $erro[] = "O campo " . substr($chv,2) . " Não foi informado";
      }
   }
   echo  '<p align="left"><font color="navy"><b>';
   echo 'Dados informados:</b></font></p>';
   echo '<table border=0 cellpadding=1 cellspacing=2>';
   reset($_POST);
   foreach($_POST as $chv => $vlr) {
      if(substr($chv,0,2)=="F_") {
         if($chv=="F_ETARIA") {
            $chv = "F_Faixa Etária";
            switch($vlr) {
```

```
                case 1: $vlr = "Até 10 anos";
                      break;
                case 2:  $vlr = "Entre 10 e 15 anos";
                      break;
                case 3:  $vlr = "Entre 15 e 20 anos";
                      break;
                case 4:  $vlr = "Entre 20 e 30 anos";
                      break;
                case 5:  $vlr = "Acima de 30 anos";
                      break;
             }
          }
          echo  "<tr><td>" . ucfirst(strtolower(substr($chv,2))) .
                "</td> ";
          echo "<td><b>" . $vlr . "</b></td></tr>\n";
       }
    }
    echo '</table>';
    // Erros
    if(sizeof($erro)>0) {
       echo  '<p align="left"><font color="red"><b>';
       foreach($erro as $vlr) {
          echo "$vlr<br>";
       }
       echo '</b></font></p>';
       echo '<p><a href="exerc_8_5.php5">Voltar</a></p>';
    }
    ?>
    </body>
    </html>
```

7. ENCTYPE=”mixed/form-data”

8. **Parte 1 - Formulário:**

```
<html>
<body>
<table border="0" cellpadding="3" cellspacing="0" width="100%">
 <tr>
      <td align="right" bgcolor="#8CDAFF">
        <?=date("d-m-Y H:i:s")?> 
      </td>
 </tr>
</table>
<form name="arquivo" enctype="multipart/form-data"
    method="post" action="upload.php5">
 <table border="0" cellpadding="5" cellspacing="5">
```

```
    <?php
       for($_i=1;$_i<=5;$_i++) {
    ?>
    <tr>
      <td height="30"><b>Arquivo <?=$_i?>:</b></td>
      <td height="30" >
       <input type="hidden" name="MAX_FILE_SIZE"
    value="<?=100*1024?>">
       <input type="FILE" name="ARQUIVO[]" size="50">
      </td>
    </tr>
    <?php
       }
    ?>
    <tr>
      <td colspan=2 align>
       <input type="submit"  value=" Enviar o Arquivo">
      </td>
    </tr>
 </table>
</form>
</body>
</html>
```

Parte 2 – Tratamento dos arquivos enviados:

```
<html>
<body>
<table border="0" cellpadding="3" cellspacing="0" width="100%">
 <tr>
   <td height="30" bgcolor="#8CDAFF">
    <b>Upload de Arquivo - Resultado</b>
   </td>
   <td align="right" bgcolor="#8CDAFF">
     <?=date("d-m-Y H:i:s")?> 
   </td>
 </tr>
</table>
<?php
 // Definir os parâmetros de teste
 $tamanho_maximo = 100*1024; // em bytes(=100KB)
 $tipos_aceitos = array( "image/gif",   "image/pjpeg");
 $_num_arquivos = 5;
 // Validar o arquivo enviado
 $arquivo = $_FILES['ARQUIVO'];
 for($_i=0;$_i<$_num_arquivos;$_i++) {
```

```
    echo "Arquivo " . ($_i+1) . ":<br/>";
    if($arquivo['error'][$_i] != 0) {
       echo "<p><b><font color='red'>Erro no Upload do arquivo
    {$arquivo['erro'][$_i]}<br>";
       switch($arquivo['error'][$_i]) {
       case  UPLOAD_ERR_INI_SIZE:
                  echo 'O Arquivo excede o tamanho máximo permitido';
                  break;
       case UPLOAD_ERR_FORM_SIZE:
                  echo 'O Arquivo enviado é muito grande';
                  break;
       case  UPLOAD_ERR_PARTIAL:
                  echo 'O upload não foi completo';
                  break;
       case UPLOAD_ERR_NO_FILE:
                  echo 'Nenhum arquivo foi informado para upload';
                  break;
       }
       echo  '</font></b></p>';
       continue;
    }
    if($arquivo['size'][$_i]==0 OR $arquivo['tmp_name'][$_i]==NULL)     {
       echo '<p><b><font color="red">Nenhum arquivo enviado
             </font></b></p>';
       continue;
    }
    if(array_search($arquivo['type'][$_i],$tipos_aceitos)===FALSE) {
       echo '<p><b><font color="red">O Arquivo enviado não é do tipo
       (' . $arquivo['type'] . ') aceito para
             upload.</font></b></p>';
       continue;
    }
    // Exibir o Arquivo
    $_d =    dirname($arquivo['tmp_name'][$_i]) . "/" .
             $arquivo['name'][$_i];
    move_uploaded_file($arquivo['tmp_name'][$_i],$_d);
    echo "<img src='$_d'><br>";
 }
?>
</body>
</html>
```

Capítulo 9

1. htmlentities ou htmlspceialchars, porém htmlentities é mais completa que htmlspecialchars. O inverso é conseguido com html_entity_decode.

2. trim() retira os espaços e ltrim() retira os espaços somente à esquerda. A sintaxe das duas funções é:

```
trim/ltrim (string, conjunto_caracteres)
```

3. A função MD5 converte uma string em um código criptografado de 32 caracteres hexadecimais, conforme o algoritmo definido por RSA Data Security, Inc. em seu documento MD5 Message-Digest Algorithm.
4. **str_pad()**. Sua sintaxe é:

```
str_pad(string,tamanho,texto_complementar,tipo)
```

texto_complementar é opcional e se não informado, assume-se espaços.

Tipo é opcional, podendo ser um dos seguintes valores:

STR_PAD_RIGHT: complemento à direita.

STR_PAD_LEFT: complemento à esquerda.

STR_PAD_BOTH: complemento nos dois sentidos.

Se tipo não for informado, STR_PAD_RIGHT é utilizado.

Caso o tamanho seja menor ou igual ao tamanho do texto original, nenhum complemento é realizado.

```
<?php
   $texto = "PHP";
   echo str_pad($texto,10) . "!";
?>
```

5.

```
<?php
   $s1 = str_word_count($texto);
   echo "Total de palavra no texto: $s1 <br>";
?>
```

6.

```
<?php
   // Exercicio 9.6
   // $_str = String de entrada
   //$_str =  "teste de troca de / que deve aparece / varias / vezes";
   // a) Tamanho da string
   $_t = strlen($_str);
   // b) primeira ocorrencia de /
   $_pb = strpos($_str,"/");
   // c) ultima ocorrencia de /
   $_ub = strrpos($_str,"/");
   // d) troca de / por \
   $_nt = strtr($_str,"/","\\");
```

```
   echo "String Original: $_str <br>";
   echo "Tamanho da String : $_t <br>";
   echo "Primeira ocorrência de / : $_pb <br>";
   echo "Última ocorrência de / : $_ub <br>";
   echo "Novo Texto : $_nt <br>";
?>
```

7.

```
<?php
   // Exercicio 9.7
   // $_valor = String de entrada no formato R$ 99.999,99999
   // $_valor = "R$ 11245,9887";
   // a) Retirar os caracteres R $ e .
   $_a = Array("R"=>"","r"=>"","$"=>"","."=>"");
   $_v = strtr($_valor,$_a);
   // b) Troca de , por .
   $_v = strtr($_v,",",".");
   // c) Mostrar o valor Resultante
   echo "String Original: $_valor <br>";
   echo "String Formatada : " . number_format($_v,2,",",".") ."<br>";
?>
```

Capítulo 10

1. Um array, também conhecido como vetor ou lista, é uma das formas elementares de estrutura de dados presentes nas linguagens de programação. No PHP, um array pode conter elementos de qualquer tipo (inteiro, string, booleano, etc.), diferentemente de outras linguagens que exigem a definição do array para um único tipo de dado.
2. É um array que contém outros arrays, por exemplo: Array(1=>Array(10,20,30),2=>Array(5,15)).
3. É possível criar arrays por meio construtor de arrays **Array()** ou da criação implícita. Exemplos:

```
Contrutor:   $_a = Array(1,2,3,4,5);
Implícita:      $_a[0] = 1; $_a[1] = 2;
```

4. **unset()**. Exemplo: unset($_a[1]);
5. **array_change_key__case()**. Sua sintaxe é:

```
array_change_key_case(Array,CASE_LOWER);
```

6. A função array_flip() retorna um array com os valores do array informados como índices e os índices do array informados como valores. Os valores existentes devem ser candidatos válidos para índices (inteiros ou strings).
7. **array_key_exists()**.
8. array_pop() e array_shift(). Exemplos:

```
<?php
            $_a = Array(10,20,30,40,50);
            echo array_pop($_a);
            echo array_push($_a);
            print_r($_a);
      ?>
```

9. Caso seja necessário classificar um determinado array em ordem ascendente, porém mantendo os índices desse array (isso geralmente é necessário quando temos índices não-numéricos, isto é, chaves associativas), devemos utilizar a função asort().

10.

```
<?php
   // Exercicio 10.10
   function fatorial($_num) {
      if($_num<0) {
         return 0;
      } elseif($_num==0 || $_num==1) {
         return 1;
      }  else {
         return $_num * fatorial($_num-1);
      }
   }

   // Array Exemplo
   $_a = Array(5,7,9,13,15,22,50);
   // Execução da função fatorial em todos os elementos do array
   echo "<PRE>";
   print_r(array_map("fatorial",$_a));
   echo "</PRE>";

?>
```

11.

```
<?php
   // Exercicio 10.11
   // Retorna maior valor do array, menor valor e média
   // Array
   $_a = Array(-1,10,0,20,23,35.7,50,-2.57);
   // Maior e Menor Valor
   $_maior = NULL;
   $_menor = NULL;
   foreach($_a as $_v) {
      if($_v>$_maior || $_maior===NULL) {
         $_maior = $_v;
      }
      if($_v<$_menor || $_menor===NULL) {
         $_menor = $_v;
      }
```

```
   }
   // média
   $_total = array_sum($_a);
   $_media = $_total / sizeof($_a);

   echo "Maior valor: $_maior <br/>";
   echo "Menor valor: $_menor <br/>";
   echo "Média: $_media";
?>
```

12.

```
<?php
   /*    Exercicio 10.12
      Filtra o Array conforme o critério
      o elemento deve conter a palavra PHP
      e deve ser maior que 3 caracteres, ou seja
      não deve conter apenas a palavra PHP
   */
   // Função para filtrar
   function filtra($vlr) {
      global $_result;
      if(strpos($vlr,"PHP")!==FALSE && strlen($vlr)>3) {
         $_result[] = $vlr;
         return FALSE;
      } else {
         return TRUE;
      }
   }

   // Array de exemplo
   $_array = Array("PHP",1,10,"PHP5","PHP4", "Livro de PHP",2);

   // Aplicar a função ao Array
   $_r = array_filter($_array,"filtra");

   echo "<pre>";
   print_r($_result);
   print_r($_r);
   echo "</pre>";
 ?>
```

Capítulo 11

1. A função **time()** retorna o número de segundos desde a data base (1º de janeiro de 1970 00:00:00 GMT).

2. 1102042799 ou 02/12/2004 23:59:59. Como 32 não é um dia válido, a função calcula com base no mês informado o número de dias excedentes (neste caso 2) e faz os ajustes necessários, que neste caso foi somar um ao mês e considerar o dia como 2.
3. **checkdate()**.
4. Algo como: 19-9-2004 13:32:06, pois **j** devolve o dia do mês, **n** o mês do ano (com um ou dois dígitos), **Y** o ano no formato de quatro dígitos, **G** a hora do dia (1 a 23)., **i** os minutos e **s** os segundos.
5.

```
<?php
   // Exercicio 11.5
   // Calculo da diferença entre datas
   // $_data1 e $_data2
   $_data1 = "01/09/2004 15:00:00";
   $_data2 = time();

   function converte($_data) {
      if(!is_int($_data)) {
         // Converter para timestamp
         $_dia = substr($_data,0,2);
         $_mes = substr($_data,3,2);
         $_ano = substr($_data,6,4);
         $_hr  = substr($_data,11,2);
         $_mn  = substr($_data,14,2);
         $_seg = substr($_data,17,2);
         return mktime($_hr,$_mn,$_seq,$_mes,$_dia,$_ano);
      } else {
         return $_data;
      }
   }

   $_data1 = converte($_data1);
   $_data2 = converte($_data2);

   // Diferença
   if($_data1<=$_data2) {
      $_dif = $_data2-$_data1;
   } else {
      $_dif = $_data1-$_data2;
   }
   $_dias = floor($_dif / 86400); // 86400 = 24*60*60
   $_dif -= $_dias * 86400;

   echo "Diferença entre " . date("d/m/Y H:i:s",$_data1) .
       " e " . date("d/m/Y H:i:s",$_data2) .
       " = $_dias dias ,  " . date("H",$_dif) . " Horas, " .
```

```
        date("i",$_dif) . " minutos  e " . date("s",$_dif) .
      " segundos";
?>
```

6. **mkdir()**.No PHP5 podemos utilizar a função mkdir() tanto em diretórios locais (/usr/local, por exemplo) quanto em alguns endereços URL (ftp://...).

   Se o PHP estiver sendo executado no modo seguro (safe mode), ele verifica se o diretório no qual será criado o novo diretório (ou no caminho inteiro conforme o parâmetro diretório) possui o mesmo dono (UID) do script que está executando a função.

7. Essa classe possui os seguintes atributos:

   path: contém o diretório informado.

   handle: ponteiro para o diretório.

   A classe disponibiliza ainda os seguintes métodos:

   read(): retorna a entrada atual do diretório e move o ponteiro para a próxima entrada. Caso não exista mais nenhuma entrada, retorna falso (FALSE).

   Rewind(): retorna o ponteiro para o início do diretório.

   Close(): encerra o uso da classe, desta forma não é possível executar nenhuma outra operação na classe.Explique a classe **dir.** Descreva seus métodos.

8. readdir() retorna a entrada atual do diretório aberto com opendir() e move o ponteiro para a próxima entrada. Caso não exista nenhuma entrada para o diretório, retorna falso (FALSE).
9. Será o nome do script que está sendo executado neste momento.
10. **disk_total_space()** e **disk_free_space()**.
11. Basta utilizar a função **file_exists(nome_arquivo)** e se a função retornar verdadeiro (TRUE), o arquivo existe; caso contrário, não existe.
12. A função **file()** lê o arquivo informado e retorna um array. Cada elemento do array contém uma linha do arquivo (incluindo o caractere de salto de linha \n \r\n \r, dependendo do sistema operacional).
13. **parse_ini_file().**
14. **unlink**
15. **w** Abre somente para escrita. O arquivo é truncado (seu conteúdo é excluído). O ponteiro indica o início do arquivo. Caso o arquivo não exista, tenta criá-lo.

    w+ Abre para leitura e escrita. O arquivo é truncado (seu conteúdo é excluído). O ponteiro indica o início do arquivo. Caso o arquivo não exista, tenta criá-lo.

16. **feof()**. Exemplo:

```
<?php
$_a= fopen("http://static.php.net/www.php.net/images/php.gif","r");
if($_a!==FALSE) {
   while(!feof($_a)) {
      echo fgets($_a,4096);
```

```
   }
}
fclose($_a);
?>
```

17.

```
<?php
   // Exercicio 11.17
   // Lê um arquivo Texto e troca todas as ocorrencias de > por &gt;
   //$_arq = "exerc_8_5.php5";
   $_f = file_get_contents($_arq);
   $_f1 = str_replace(">","&gt;",$_f);
   echo "<pre>";
   echo $_f;
   echo $_f1;
?>
```

18.

```
<?php
   /* Exercicio 11.18
    Lê um arquivo Texto e executa os seguintes processos
    Localiza todas as ocorrencias da palavra PHP
    Cria um array com número da linha e posição da ocorrencia
    imprime o array
   */
   $_arq = "exerc_10_12.php5";
   $_f = file($_arq);
   foreach($_f as $_k=>$_l) {
      $_p = strpos($_l,"PHP");
      while($_p!==FALSE) {
         $_a[] = Array($_k,$_p);
         $_p = strpos($_l,"PHP",$_p+3);
      }
   }

   echo "<pre>";
   echo "Texto Original: ";
   print_r($_f);
   echo "Ocorrências da palavra PHP:<br>";
   foreach($_a as $_c) {
      echo "Linha {$_c[0]}, posiçao {$_c[1]}<br>";
   }
   echo "</pre>";
?>
```

19.

```
<?php
      /* Exercício 11.19
      Alteração das funções le_diretorio() e
```

```
detalhe(), para exibir o tamanho do arquivo
e sua cota no disco.
Para isto foi incluída a função converte()
*/
function converte($_t) {
 $_1k = 1024;
 $_1m = pow($_1k ,2);
 $_1g = pow($_1k,3);
  return ($_t<$_1k
             ? $_t . " Bytes"
          : ($_t < $_1m
             ? round($_t/$_1k,2) . " KB"
             :  ($_t < $_1g
                  ? round($_t/$_1m,2) . " MB"
                  : round($_t/$_1g,2) . " GB"
                )
           )
      );
 }

function detalhe($_arq) {
   if(is_file($_arq)) {
         $_ret.= "<td>Arquivo</td>";
      $_ret.= "<td>".(is_link($_arq)? "*":"-")."</td>";
      $_ret.= "<td>".(is_executable($_arq)? "*":"-")."</td>";
         $_ret.= "<td>".(is_readable($_arq)? "*":"-")."</td>";
         $_ret.= "<td>".(is_writable($_arq)? "*":"-")."</td>";
      // Tamanho
      $_t = filesize($_arq);
      $_s = disk_total_space(dirname($_arq);
      $_ret.= "<td align='right'>" . converte($_t) . "</td>";
      // %uso
      $_ret.= "<td>". number_format($_t/$_s)*100,5,",",".") .
               "%</td>";
      return $_ret;
   }
   else {
      return "<td><b>Diretório</b></td><td>-
      </td><td>- </td><td>-</td><td>-</td>";
   }
}

function le_diretorio($_dir) {
         $_ret = "<table border=1 cellpadding=2>\n";
      $_ret.= "<tr><td><b>Arquivo</td>";
      $_ret.= "<td>Tipo</td>";
      $_ret.= "<td>Link Simbólico</td>";
```

```
            $_ret.= "<td>Executável</td>";
            $_ret.= "<td>Permissão leitura</td>";
            $_ret.= "<td>Permissão escrita</td>";
            $_ret.= "<td>Tamanho</td>";
            $_ret.= "<td>% Uso</td>";
            $_ret.= "</tr>";
            foreach(scandir($_dir) as $_arq) {
               if(substr($_arq,0,1)=="." AND strlen($_arq)<=2) {
                  continue;
               }
               $_ret.=  "<tr align='center'><td align='left'> " .
                        "$_dir\\$_arq</td>" .
                        detalhe("$_dir/$_arq") .
                        "</tr>\n";
            }
            $_ret.= "</table>\n";
            echo $_ret;
      }

   ?>
```

20. a função header() deve ser utilizada para enviar um ou vários cabeçalhos http (conforme a especificação HTTP/1.1).
21.

Parâmetro	**Descrição**
Nome	Nome dado ao cookie, podendo ter qualquer formato aceito para uma string (se o parâmetro register_globals do PHP estiver on, o nome dado ao cookie deve respeitar ainda o formato aceito pelo PHP para nomes de variáveis).
Valor	O valor que será armazenado no computador do usuário.
Exp	Data, no formato timestamp, na qual o cookie irá expirar (caso não seja informada, o cookie será descartado quando o usuário fechar o browser).
Caminho	Define o caminho no servidor para o qual o cookie estará definido (limitando a área do site na qual o cookie será visível).
Domínio	Define o domínio para o qual o cookie estará disponível.
Seguro	Define se o cookie deve ser enviado por uma conexão segura (Https) ou não. Aceita os valores 0 e 1; 0 permite o envio por um servidor comum (http) e 1 define que o envio deve ser feito por um servidor seguro(https).

22. **dechex()**.
23. Converte o número dado de uma base qualquer em outra base (as bases devem estar entre 2 e 36). Exemplo:

    base_convert(1024,10,2);

24. **floor()**.
25. Para cálculo da potência, conforme a base e o expoente informados. Exemplo: pow(2,5) = 32.
26. **max**: retorna o maior valor encontrado nos argumentos informados. É possível avaliar uma seqüência de valores ou então um array. Sua sintaxe é:

```
max(valor_1,valor_2,valor_3,....)
```

Não existe um limite de valores a comparar. Se o primeiro argumento for um array, retorna o maior valor encontrado no array.

min: retorna o menor valor encontrado em uma série de valores informados. É possível avaliar uma seqüência de valores ou então um array. Sua sintaxe é:

```
min(valor_1,valor_2,valor_3,....)
```

Não existe um limite de valores a comparar. Se o primeiro argumento for um array, retorna o menor valor encontrado no array.

27.

```
<?php
   // Exercicio 11.27
   // Retorna maior valor do array, menor valor e média
   // Array
   $_a = Array(-1,10,0,20,23,35.7,50,-2.57);
   // Maior e Menor Valor
   $_maior = max($_a);
   $_menor = min($_a);
   // média
   $_total = array_sum($_a);
   $_media = $_total / sizeof($_a);

   echo "Maior valor: $_maior <br/>";
   echo "Menor valor: $_menor <br/>";
   echo "Média: $_media";
?>
```

28. Utilizando a função **extension_loaded()**, a qual retorna verdadeiro (TRUE) caso a extensão esteja carregada.
29. Essa função devolve o valor de um parâmetro de configuração do PHP. Sua sintaxe é:

```
ini_get(nome_parametro)
```

Caso o parâmetro informado não exista, a função retorna uma string vazia.

30. A função ini_set() altera (quando permitido) o valor de uma opção do PHP. Sua sintaxe é:

```
ini_set(opção,valor)
```

O resultado dessa função é o valor antigo da opção informada. O novo valor será válido enquanto o script estiver sendo executado (voltando ao valor original no seu término).

31. **phpversion()** ou por meio da constante PHP_VERSION.

32. Essa função permite que o tempo máximo, em segundos, de execução de um script seja alterado (o valor-padrão é definido no parâmetro max_executaion_time do arquivo php.ini, e na ausência deste o valor-padrão será 30 segundos).
33. Os caracteres informados devem aparecer pelo menos uma vez. Exemplo: 'a+b' procura a letra **a** (deve aparecer pelo menos uma vez) seguida de b (ab,aab, aaab, porém não b somente).
34. a*bc{2,3} procura a ocorrência de a (zero ou mais vezes) seguida por bc no mínimo duas vezes e no máximo três vezes.
35. A função ereg() realiza uma busca conforme a expressão regular informada, retornando verdadeiro(TRUE) se a busca for bem-sucedida (ou seja, se a expressão informada for encontrada na string) ou falso (FALSE) caso contrário. Opcionalmente é possível armazenar o resultado da busca (somente os elementos entre parênteses) em um array informado.
36. A expressão valida a variável $_usr, retornando verdadeiro (TRUE) caso ela contenha somente letras e/ou número. Seu tamanho mínimo é quatro caracteres e o máximo dez.
37. Essa função retorna o endereço IP da URL informada.
38. **eval(string).**
39. A função **highlight_file()** avalia o arquivo enviado como um script PHP e marca as tags e comandos PHP, conforme o padrão de cores definido pelo próprio PHP.

Capítulo 12

1. O arquivo php.ini contém os parâmetros de configuração do PHP, os quais ditam seu comportamento-padrão em várias situações (por exemplo, o controle de erros ou quais extensões devem ser carregadas na inicialização do PHP no Windows).
2. **error_reporting =** E_ERROR | E_WARNING | E_PARSE
3. **log_errors= On** e **error_log=rel_erros.txt**
4. **Set_error_handler()** permite que seja definida uma função para tratar os erros gerados na execução de um script no PHP.
5. **trigger_error()**. Essa função gera um erro de usuário (a função tem um alias de nome user_error()), podendo ser utilizada com o gerenciador padrão de erros do PHP ou por uma função definida em set_error_handler(). Sua sintaxe é:

   trigger_error(mensagem,tipo_erro)
6. A função debug_backtrace() gera um array com informações úteis para a depuração de um script, podendo ser inserida em qualquer lugar que seja necessária uma verificação mais detalhada (por exemplo, se estamos em dúvida sobre possíveis erros no script).
7.

```
<?php
        // Exercicio 12.7
        // Permitir gravar as informações de debug
```

```
  // $_modo: 0 -> Envia para o Browser, 1 -> Grava no arquivo
      informado
function debug($_modo=0,$_arq="log_erros.txt") {
   $_trc = debug_backtrace();
   array_shift($_trc);
   if($_modo==0) {
      $_res = '<pre>';
   }
   foreach($_trc as $arr)  {
      if (isset($arr['class'])) {
            $_res.= $arr['class'] . '->';
      }
      $_res.= "Função: ";
      $args = array();
      if(!empty($arr['args'])) {
         foreach($arr['args'] as $v) {
            if (is_null($v)) {
               $args[] = 'Null';
            } elseif(is_array($v)) {
               $args[] = 'Array('  .  implode($v) . ')';
            } elseif(is_object($v)) {
               $args[] = 'Objeto: ' . get_class($v);
            } elseif (is_bool($v)) {
               $args[] = 'Bool: ' . ($v ? 'V(TRUE)' : 'F(FALSE)');
            } else {
               $v = (string) $v;
               $str = htmlspecialchars($v);
               $args[] = "\"".$str."\"";
            }
         }
      }
      $_res.= $arr['function'] . '(' .  implode(', ',$args) . ')';
      $_l = (isset($arr['line']) ?  $arr['line'] : "-");
      $_f = (isset($arr['file']) ?  $arr['file'] : "-");
      $_res .= "Linha $_l , Arquivo: $_f";
      if($_modo==0) {
               $_res.= "<br/>";
            } else {
                  $_res.= "\n";
         }
   }
      if($_modo==0) {
         $_res.= '</pre>';
            echo $_res;
            return TRUE;
      } else {
            $_f = fopen($_arq,"a");
```

```
                fwrite($_f,$_res);
                fclose($_f);
                return TRUE;
            }
        }
    ?>
```

Capítulo 13

1. A versão 5 está mais aderente ao modelo de Orientação a Objetos, disponibilizando, entre outras coisas: visibilidade de atributos e métodos (privativos, protegidos e públicos), classes e métodos abstratos, controle de exceções(try..catch / throw), definição de interfaces.
2. Uma classe é a abstração de um objeto, ou seja, é um modelo a partir do qual os objetos podem ser criados. Enquanto um objeto representa uma coisa real e somente uma coisa em um determinado instante, uma classe é a generalização deste, mostrando quais devem ser suas características (dados) e quais as suas capacidades (funcionalidades).
3. Atributo: é cada uma das peças de dados de uma classe, ou seja, a coleção de atributos representa o que uma classe sabe (os dados que a classe possui). Em termos de programação, os atributos são as variáveis que definimos para a classe e que são utilizadas apenas na classe.

   Método: os métodos definem o que as classes sabem fazer, ou seja, eles podem alterar os atributos e realizar funções inerentes à classe (como, por exemplo, incluir um novo funcionário, alterar seu salário, etc.). Em termos de programação podemos pensar em métodos como funções na classe, as quais podem retornar dados (o salário do funcionário) ou não conforme suas características funcionais.
4. É a capacidade de uma classe em herdar os atributos e métodos de uma outra classe (chamamos de superclasse a classe mãe, aquela que fornece os atributos e métodos à subclasse que os herda).
5. Uma classe altamente coesa representa e manipula um e somente um tipo de objeto.
6. É de difícil entendimento; sua reusabilidade fica prejudicada; é de difícil manutenção; qualquer mudança afeta vários pontos da classes, tornando-a complexa de lidar; possui mais responsabilidades do que deveria, uma vez que assume funções que são de outras classes (incluir um novo departamento na classe de funcionários por exemplo).Cite pelo menos dois problemas de uma classe com baixa coesão

7.
```
Class nome_classe <opções> {
            Atributos
            Métodos
      }
```

7. Os atributos e métodos definidos como private(privativo) são visíveis apenas na classe que os criou, ou seja, subclasses ou o script que contêm a classe não podem acessar esses atributos ou métodos.
8. Para referenciar o próprio objeto dentro de seus métodos.

9. **__construct** e **__destruct.**
10. A é a superclasse e B uma herança de A (ou seja, B é uma subclasse).
11. Não, pois o atributo é privativo, ou seja, só pode ser manipulado pela classe à qual pertence (neste caso A).
12. É uma classe que define um ou mais métodos para serem implementados pela suas subclasses (os métodos abstratos da classe abstrata). Uma classe abstrata não pode ser instanciada, somente as subclasses podem.
13. Uma interface permite definir quais métodos públicos uma classe deve implementar, sem ser necessário ter que definir como os métodos devem ser construídos. Exemplo:

```
interface Itabela {
 public function busca($_codigo);
 public function setBD($_bd);
}
```

14. Resultado: 200
    Resultado: 10000
15.

```
Class [  class B extends A ] {
  @@ C:\Inetpub\wwwroot\PHP5\Exercicios\exerc_13_16.php5 13-23

  - Constants [0] {
  }

  - Static properties [0] {
  }

  - Static methods [0] {
  }

  - Properties [1] {
    Property [  protected $_nome ]
  }

  - Methods [4] {
    Method [   public method __construct ] {
      @@ C:\Inetpub\wwwroot\PHP5\Exercicios\exerc_13_16.php5 14 - 16
    }

    Method [  public method setNome ] {
      @@ C:\Inetpub\wwwroot\PHP5\Exercicios\exerc_13_16.php5 17 - 19

      - Parameters [1] {
        Parameter #0 [ $_n ]
      }
    }

    Method [  public method getnome ] {
      @@ C:\Inetpub\wwwroot\PHP5\Exercicios\exerc_13_16.php5 20 - 22
    }

    Method [  public method setCodigo ] {
      @@ C:\Inetpub\wwwroot\PHP5\Exercicios\exerc_13_16.php5 9 - 11

      - Parameters [1] {
```

```
        Parameter #0 [ $_c ]
      }
    }
  }
}
```

16. A classe **ReflectionMethod** realiza a engenharia reversa em métodos de classes.
17. Encerra o método executado imediatamente e lança a exceção gerada para o sistema. Exemplo:

```
Throw new exception(exceção)
```

18. Permite o controle de exceções (erros) gerados na execução de scripts PHP. Qualquer comando dentro do bloco try que gerar uma exceção será enviado para o bloco catch.
19. Qualquer exceção gerada será pega pelo primeiro bloco catch. O correto é deixar esse bloco como o último do tratamento de exceções.
20. **get_declared_classes**.
21. Através da função **instanceof**, exemplo $_class instanceof classeA
22. **method_exists**.

Capítulo 14

1. Uma sessão é basicamente um meio de mantermos dados durante a navegação por várias páginas de um site.
2. Para evitar alguns tipos de ataque ao site, por exemplo através da simulação de session-id (pois a identificação pode ser anexada à URL), pode-se estabelecer o controle de sessões apenas por cookies (o que dependerá da aceitação dos usuários, pois eles podem desabilitar a gravação de cookies em suas máquinas). O valor-padrão é 0, o que significa que tanto cookies quanto propagação por URL são aceitos. Para habilitar essa funcionalidade, informe 1 nesse parâmetro.
3. Inicia uma sessão.
4. $_SESSION. Para remover, devemos usar a função **unset()**.
5. **session_unset(), session_destroy()**.
6. A função **session_regenerate_id()** gera um novo indicador de sessão, mantendo as variáveis atualmente registradas
7. O objetivo dessa função é forçar a gravação dos dados da sessão e o seu encerramento
8. Através do parâmetro session.cookie_lifetime ou da função session_set_cookies_params (porém somente são válidos para sessões que utilizam cookies).
9. Primeiro crie um atributo $_arquivo:

```
private $_arquivo;
```

   Em seguida crie os métodos setArquivo, getArquivo

```
public function setArquivo($_arq) {
     $this->_arquivo = $_arq;
  }

  public function getArquivo() {
     return $this->_arquivo;
  }
```

E finalmente o método gravaSessao:

```
public function gravaSessao() {
     if($this->_arquivo==NULL || $this->_arquivo=="") {
        return FALSE;
     }
     $_f = fopen($this->_arquivo,"w+");
     foreach($_SESSION as $_k=>$_v) {
        fwrite($_f,"$_k=$_v \n");
     }
     fclose($_f);
  }
```

10.

```
  public function getNovoID() {
     session_regenerate_id();
     $this->_id = session_id();
  }

  public function gravar($_inicia=TRUE) {
     session_write_close();
     if($_inicia===TRUE) {
        $this->start();
     }
  }
```

Capítulo 15

1. GD. Está disponível em http://www.boutell.com/gd/ .
2. Através do parâmetro de compilação --with-gd ou da extensão extension=php_gd2.dll do arquivo php.ini.
3. Para criar uma imagem. A diferença está no fato de que **imagecreatetruecolor** criará uma imagem com o formato true color (disponibilizando todo o espectro de cores).
4. **imagecreatefrompng**().
5. **imagecolorallocate()**.

6. **imageline**(). Exemplo: imageline($_img,0,0,100,100,$_cor).
7. **imagerectangle**(). Exemplo: imagerectangle($_img,10,10,50,50,$_cor).
8. Essas funções desenham um polígono de n lados, **imagepolygon**() desenha um polígono vazado (somente com a borda) e **imagefilledpolygon**() desenha um polígono preenchido com a cor informada.
9. **imageellipse**() e **imagefilledellipse**().
10. Teremos o desenho apenas do arco ou da corda, sem preenchimento.
11. **imagestring()** desenha apenas textos simples e sempre na horizontal, já **imagettftext()** permite que o texto seja desenhado em várias posições (ângulos), além de permitir o uso de fontes true color.
12. As formas possíveis de salvar uma imagem são: imagepng(nome_arquivo), imagejpeg(nome_arquivo), imagewbmp(nome_arquivo) e imagexbmp(nome_arquivo). É possível apenas enviar a imagem para o browser, basta não informar o parâmetro nome_arquivo nas funções imagepng, imagejpeg, imagewbmp e imagexbmp (mas lembre-se de que é preciso enviar o cabeçalho correto para o browser através da função header).
13. Classe grafico_linhas:

```
<?php
class grafico_linhas extends imagem {
   function desenha_grafico($_v,$_t=Array("p"=>"","x"=>"","y"=>""),
                            $_l=Array(),$_bc=Array(255,255,255)) {
      // Montar o gráfico de barras com Linhas (eixo y duplo)
      $l_spc = 10;
      $_space_opt = 5;
      $_left = 60;
      $_bottom = imagesy($this->_img)-60;
      $_footer = imagesy($this->_img)-40;
      $spc_total = imagesx($this->_img);

      // Espaço Total necessário
      $_spc_total_lines = $l_spc+sizeof($_v)+(sizeof($_v)-1)*$_space_opt;
      if($_spc_total_lines>$spc_total) {
         $_space_opt = ($spc_total-$l_spc-sizeof($_v))/(sizeof($_v)-1);
      }
      if($_space_opt<=2) $_space_opt=2;

      $this->alocaCor("cor1",Array(0xFF,0xF0,0xE6));
      $this->alocaCor("preto",Array(0,0,0));
      $this->alocaCor("cinza",Array(204,204,204));
      $this->alocaCor("cinza-claro",Array(224,224,224));
      $this->alocaCor("_cf",$_bc);
      $this->alocaCor("linha",Array(0x33,0x00,0xff));
      imagefilledrectangle($this->_img,0,0,imagesx($this->_img)-1,
                           imagesy($this->_img)-1,$this->getCor("_cf"));
      $lim = -1;
      $base = 10;
```

```
$maximo = max($_v);
$base = round($maximo/10,1);
if($base<=0.1) $base = 0.1;
elseif($base<=0.3) $base = 0.3;
elseif($base<=0.4) $base = 0.4;
elseif($base<=0.5) $base = 0.5;
elseif($base<=0.7) $base = 0.7;
elseif($base<=1) $base = 1;
elseif($base<=3) $base = 3;
elseif($base<=5) $base = 5;
elseif($base<=10) $base = 10;
elseif($base<=100) $base = round($base/10,0)*10;
elseif($base<=1000) $base = round($base/100,0)*100;
elseif($base<=10000)$base = round($base/1000,0)*1000;
elseif($base<=100000) $base = round($base/10000,0)*10000;
elseif($base<=1000000) $base = round($base/100000,0)*100000;
elseif($base<=10000000) $base = round($base/1000000,0)*1000000;
else $base = round($base/1000000,0)*1000000;

if($base>500000 AND $base<1000000) {
  $base = 1000000;
}

if($maximo>($base*10)) {
     $base*=1.5;
  if($maximo>($base*10)) {
    $base*=2;
  }
}

if($base>=10000) {
  $_fator = 10000;
}
if($base>=100000) {
  $_fator = 100000;
}
if($base>=1000000) {
  $_fator = $base;
}

$_spc_grad = 15;

$_cores[0] = "navy";
$_cores[1] = "yellow";

if($base<1) {
  $_num = 1;
```

```
}
else {
  $_num = 0;
}

// titulo
imagestring($this->_img,3,20,10,$_t["p"],$this->getCor("preto"));

// Coordenadas

imageline($this->_img,$_left,$_bottom-13,$_left,
            35,$this->getCor("cinza"));
imageline($this->_img,$_left+1,$_bottom-13,$_left+1,
            35,$this->getCor("cinza"));
imageline($this->_img,$_left,$_bottom-13,imagesx($this->_img)-5,
            $_bottom-13,$this->getCor("cinza"));
imageline($this->_img,$_left,$_bottom-12,imagesx($this->_img)-5,
            $_bottom-12,$this->getCor("cinza"));

$_spc_grad = 25;
for($i=1;$i<=20;$i++) {
   $vlr = $base*$i;
   if($base>=10000) {
      $vlr /= $_fator;
   }
   if(($_bottom-10-$i*$_spc_grad)<40) {
      break;
   }
   imagestring($this->_img,2,$_left-30,
               $_bottom-20-$i*$_spc_grad,
               str_pad($vlr,4," ",STR_LEFT_PAD),
               $this->getCor("cinza"));
   imageline($this->_img,$_left-2,$_bottom-10-$i*$_spc_grad-3,
               $_left,$_bottom-10-$i*$_spc_grad-3,
               $this->getCor("cinza"));
   imageline($this->_img,$_left+2,$_bottom-10-$i*$_spc_grad-3,
               imagesx($this->_img)-5,$_bottom-10-$i*$_spc_grad-3,
               $this->getCor("cinza-claro"));
}

$pos = $_left+$l_spc;
for($i=0;$i<sizeof($_v);$i++) {
   // Linha
   $top = round($_v[$i]/$base,2);
   $_fcor = $_cores[0];
   if($_v[$i]==0) $_fcor = $_cores[1];
   // Primeiro uma bolha, depois as linhas...
   imagefilledellipse($this->_img,$pos,
                      $_bottom-10-$top*$_spc_grad-3,3,3,
                      $this->getCor("preto"));
```

```
            // Se for maior que 0, então devemos ligar os pontos...
            if($i>0) {
               imageline($this->_img,$_pos_ant,$_top_ant,$pos,
                           $_bottom-10-$top*$_spc_grad-3,
                           $this->getCor("linha"));
            }
            $_top_ant = $_bottom-10-$top*$_spc_grad-3;
            $_pos_ant = $pos;
            $pos+= $_space_opt;
         }
      }
   }
   ?>
```

Um exemplo:

```
<?php
   // Exercicio 15.13
   include("class_imagem.inc");
   include("class_linhas.inc");
   $_l = new grafico_linhas();
   $_l->novaImagem(400,400);
   for($c=0;$c<=180;$c++) {
      $_v[] = round(mt_rand(100,1000),1);
   }
   $_l->desenha_grafico($_v,Array("p"=>"Gráfico de Linhas",
                        "x"=>"Valores X","y"=>"Valores Y"));
   $_l->mostra();
   $_l->destroi();
?>
```

Capítulo 16

1. simpleXML e DOM.
2. simplexml_load_file() e simplexml_load_string().
3. O método attributes() retorna todos os atributos existentes dentro das tags XML.
4. O método xpath da classe simpleXML executa uma busca nos nós do documento XML, procurando os filhos que satisfaçam o critério de busca informado no método. Esse método retorna um array de objetos da classe simpleXML ou falso (FALSE) caso não exista nenhum nó do documento que satisfaça o critério de busca (ou seja, caso o nó informado não exista).

   O formato do parâmetro de busca deve ser: "/no_1/no_2/.../no_n"

   É possível realizar uma busca absoluta, ou seja, informando todos os níveis que se deseja encontrar, bastando informar, conforme mostrado anteriormente, todos os nós que queremos encontrar.

5.

```
function filhos($_no,$_n=0,$_criterio) {
   foreach($_no->children() as $_f=>$_v) {
      echo str_repeat(" ",$_n*5);
      echo "<span";
      if(trim($_f)==$_criterio) {
         echo " style='background-color: yellow;'>";
      } else {
         echo ">";
      }
      echo $_f . (trim($_v)!="" ? " = $_v" : "") . "</span><br/>";
      filhos($_v,$_n+1,$_criterio);
   }
}
```

6. DomDocument.
7. Carrega um documento XML contido em um arquivo.
8. CreateElement(), seguido de AppendChild().
9. Retorna uma lista dos nós que satisfaçam o critério de busca (para buscar todos os nós, basta informar * como critério).
10. Utilizar **getElementsbyTagName()**, e para varrer todos os nós de um documento, devemos acessar childNodes do documento Dom e navegar por todos os níveis existentes no documento.
11. O valor 1 indica um nó, o valor 9 o documento e 3 um texto.
12. Verifica se o elemento possui atributos, em caso afirmativo retorna verdadeiro (TRUE).
13. **replacechild().**
14. Retorna todos os nós “item” do documento XML (não importando o nível) e que contenham o atributo “id” com valor igual a 2.
15. O atributo formatOutput indica para a função saveXML() que a endentação entre os nós do documento, bem como as quebras de linhas, deve ser respeitada, produzindo uma saída mais compreensível.

Capítulo 17

1. A configuração do arquivo php.ini consiste em ajustar três parâmetros para o PHP rodando em ambiente Windows e apenas um parâmetro no Linux.

   Os parâmetros necessários no Windows são:

   SMTP = nome_servidor_stmp

   Informe aqui o nome do servidor SMTP (algo como servidor_stmp.com.br) ou seu IP (192.168.0.1, por exemplo).

smtp_port = porta_servidor_stmp

Deve-se informar aqui o número da porta do servidor SMTP. O valor-padrão (que é o utilizado na maioria dos servidores SMTP) é 25.

sendmail_from= nome_dono_mensagem

Informa o usuário-padrão para o envio do e-mail (pode-se alterar isso no envio de e-mails por meio da função mail()). Geralmente está no formato usuário@servidor.

No Linux precisamos apenas informar o caminho para o SendMail (ou outro software de gerenciamento de e-mails, como o qmail):

sendmail_path = caminho_sendmail

Deve-se informar aqui. O PHP tenta encontrar na instalação o caminho do SendMail, mas caso não consiga, é possível informá-lo manualmente nesse parâmetro (no formato path/executável, por exemplo /usr/sbin/sendmail).

Esse parâmetro também funciona no Windows, e caso seja informado, o PHP ignora os demais parâmetros (mostrados anteriormente).

Temos ainda uma outra diretiva no php.ini, relacionada ao sendmail, que informa se desejamos inserir parâmetros extras na chamada do Sendmail:

mail.force_extra_paramaters = parâmetros_extras

Essa diretiva força o envio dos parâmetros extras na função mail(),sobrepondo o parâmetro correspondente na função.

2. mail(destinatario,assunto,mensagem,header,param_extras).
3. O parâmetro header deve ser utilizado para informar cabeçalhos para a mensagem (tais como: content-type, From, etc.). Exemplo “Content-Type: text/html”
4. Basta informá-los no parâmetro destinatário, separando-os com vírgula (,). Outra opção é utilizar o header para informar os endereços (cc e cco).
5. Devemos alterar o content-type para “multipart/mixed” e criar os separadores de cada anexo (boundaries), além disso precisamos converter os anexos em um formato que possa ser processado na mensagem, como, por exemplo, base64 e particioná-lo no tamanho-padrão (76) com a função chunk_split (devemos informar dentro do boundary Content-Transfer-Encoding: base64).
6. {Servidor[:porta][flags]}[Nome_caixa]

   Os flags possíveis são:

```
/service=nome_serviço
/pop3 o mesmo que /service=pop3
/nntp o mesmo que /service=nntp
/imap o mesmo que /service=imap (padrão)
/tls  força a utilização de sessão codificada (TLS)
/notls   não força a utilização de sessão codificada
```

   A opção [Nome_caixa] indica a caixa de e-mail que deve ser lida (por exemplo, INBOX, que é a caixa de entrada padrão).
7. Neste código é tentada a conexão com um servidor pop3 (serv_pop3), sem a utilização de sessão codificada(notls), com usuário “usr” e senha “snh”, além disso é determinado que a

caixa-padrão é INBOX. Caso a conexão seja bem-sucedida $_pop3, conterá um ponteiro para esse servidor.

8. Uma vez conectado ao servidor POP3, podemos verificar as mensagens disponíveis por meio da função **imap_headerinfo**, ou seu alias **imap_header**. O resultado dessa função é um objeto com todas as informações referentes ao cabeçalho das mensagens de e-mail existentes. **imap_headers** retorna um array com todos os cabeçalhos de mensagens existentes no servidor.
9. **imap_body** retorna o corpo da mensagem sem nenhuma formatação, ou seja, se a mensagem contiver anexos, eles serão mostrados como parte do corpo da mensagem.
10. **imap_fetchstructure()** e **imap_fetchbody()**.

Capítulo 18

1. www.mysql.com
2. 4.0, 4.1 e 5.0
3. mysqladmin create database <nome_bd> ou através do gerenciador mysql (comando create database <nome_banco>).
4. mysql e mysqli. A primeira deve ser utilizada para versão menores que 4.1.3 e a segunda (improved MySQL) para versões 4.1.3 ou maiores, pois aproveita melhor os recursos disponíveis nessa versão, além de ser totalmente orientada a objetos.
5. mysql_connect deve ser utilizada para a conexão com o servidor mysql, já mysql_select_db para selecionar o banco de dados padrão. A seqüência é mysql_connect e em seguida mysql_select_db.
6. **mysql_query**.
7. Através da função **mysql_num_rows**.
8. **mysql_affected_rows**.
9. Retorna um array, com índices tanto numéricos quanto como chaves associativas, do registro atual da consulta.Os resultados possíveis são um array com os índices numéricos, ou um array com as chaves associativas ou ambos (índices numéricos e chaves associativas).
10. Para retornar o valor do último campo do tipo auto_increment inserido no banco de dados atual.
11. A função mysql_real_escape_string() deve ser utilizada na maioria das execuções de comandos SQL no banco de dados, principalmente de dados vindos de formulários ou outras fontes externas, evitando ataques por injeção de códigos SQL nos campos do formulário.
12. mysqli.
13. Executa uma consulta (comando SQL) no banco de dados aberto.

14. **field_count**.
15. Atributo name do método **fetch_field_direct**.
16.

```
<?php
   $_con  = new mysqli("localhost","root","","BD_PHP5");
   if(!$_con) {
      echo "Não foi possivel conectar ao MySQL. Erro #" .
          mysqli_connect_errno() . " : " . mysql_connect_error();
      exit;
   }

   $_sql = "SELECT * FROM Usuario WHERE userNivel=1";
   $_res = $_con->query($_sql);
   if($_res===FALSE) {
      echo "Erro na consulta... " . $_con->error . "<br/>";
   } else {
      $_nr = $_res->num_rows;
      if($_nr>0) {
         echo "<table border=1>";
         echo "<tr bgcolor='#f0f0f0'>";
         for($_i=0;$_i<$_res->field_count;$_i++) {
            $_f = $_res->fetch_field_direct($_i);
            echo "<td>" . $_f->name . "</td>";
         }
         echo "</tr>";
         while($_row=$_res->fetch_assoc()) {
            echo "<tr>";
            foreach($_row as $_vlr) {
               echo "<td>$_vlr</td>";
            }
            echo "</tr>";
         }
         echo "</table>";
      }
   }

   $_con->close();
?>
```

17. Sim (mas somente na extensão mysqli), basta utilizar o método multi_query do mysqli.
18. **more_results()** verifica se existem mais resultados da consulta múltipla, e **next_result()** busca o próximo resultado disponível (caso não exista, retorna falso).
19. SELECT @@autocommit
20.

```
<?php
   $_con  = new mysqli("localhost","root","","BD_PHP5");
```

```
   if(!$_con) {
      echo "Não foi possivel conectar ao MySQL. Erro #" .
          mysqli_connect_errno() . " : " . mysql_connect_error();
      exit;
   }

   $_sql = "SELECT * FROM Usuario ORDER BY userEmail";
   $_res = $_con->query($_sql);
   if($_res===FALSE) {
      echo "Erro na consulta... " . $_con->error . "<br/>";
   } else {
      $_nr = $_res->num_rows;
      if($_nr>0) {
         echo "<table border=1>";
         echo "<tr bgcolor='#f0f0f0'>";
         for($_i=0;$_i<$_res->field_count;$_i++) {
            $_f = $_res->fetch_field_direct($_i);
            echo "<td>" . $_f->name . "</td>";
         }
         echo "</tr>";
         while($_row=$_res->fetch_assoc()) {
            echo "<tr>";
            foreach($_row as $_vlr) {
               echo "<td>$_vlr</td>";
            }
            echo "</tr>";
         }
         echo "</table>";
      }
   }

   $_con->close();
?>
```

21. A classe **mysqli_stmt** deve ser utilizada quando desejamos trabalhar com consulta pré-preparadas no mysql , pois é possível estabelecer os parâmetros e as variáveis que receberão os resultados da consulta.
22. **prepare**(), **bind_param**(), **bind_result**() e **execute**().
23.

```
<?php
   $_con  = new mysqli("localhost","root","","BD_PHP5");
   if(!$_con) {
      echo "Não foi possivel conectar ao MySQL. Erro #" .
            mysqli_connect_errno() . " : " . mysql_connect_error();
      exit;
   }
```

```
    // definir a clausula SQL
    $_sql  = "UPDATE Usuario SET userName = ?,";
    $_sql .= "userLogin = ?, userPassw = ?, ";
    $_sql .= "userNivel = ?, userEmail = ? ";
    $_sql .= "WHERE userId = ?";

    // Preparar a clausula
    $_q = $_con->prepare($_sql);

    // Informar quais variáveis serão utilizadas como parâmetros
    $_q->bind_param('sssisi',$_nome,$_login,$_senha,$_nivel,$_email,$_id);

    // Executar a alteração
    if($_q->execute()) {
       echo "Usuário $_id alterado<br/>";
    }

    $_q->close();
    $_con->close();
?>
```

Capítulo 19

1. www.postgresql.org
2. createdb
3. **pg_connect**()
4. **pg_query()** executa um comando SQL no banco de dados conectado, informado como primeiro parâmetro da função.
5. Através da função **pg_num_rows()**.
6. **pg_affected_rows()**.
7. Retorna um objeto cujos atributos são as colunas (nomes) do resultado. A função retorna a linha atual do resultado, ou opcionalmente uma linha informada na função.
8. Sim, através da função **pg_send_query()**, juntamente com a função **pg_get_result()** que é responsável por recuperar os diversos resultados da consulta múltipla.
9. Essa função retorna a estrutura de uma tabela do banco de dados. O resultado é um array multidimensional, que contém em seu primeiro nível os nomes das colunas da tabela como seu índice (chave associativa) e cada um desses índices aponta para um outro array com as definições da coluna.
10. Sim, através das cláusulas SQL “BEGIN”, “COMMIT” (ou “END”) e “ROLLBACK” (ou “ABORT”).
11.

```
<?php

   $_con = pg_connect("host=localhost dbname=BD_PHP5 user=usuario
                       password=senha");

   $_sql = "SELECT * FROM Usuario WHERE usernivel=2";
   $_res = pg_query($_con,$_sql);
   if($_res===FALSE) {
      echo "Erro na consulta... " . pg_result_error($_res) . "<br/>";
   } else {
      $_nr = pg_num_rows($_res);
      echo "A consulta retornou " . (int) $_nr . " registro(s)<br/>";
      if($_nr>0) {
         // Primeiro o cabeçalho com os campos da tabela
         echo "<table border=1 cellspacing=3>";
         echo "<tr bgcolor='#f0f0f0'>";
         for($_i=0;$_i<pg_num_fields($_res);$_i++) {
            echo "<td>" . pg_field_name($_res,$_i) . "</td>";
         }
         echo "</tr>";
         // Agora o resultado
         while($_row=pg_fetch_assoc($_res)) {
            echo "<tr>";
            foreach($_row as $_vlr) {
               echo "<td>$_vlr</td>";
            }
            echo "</tr>";
         }
         echo "</table>";
      }
   }
   pg_close($_con);
?>
```

Capítulo 20

1. Não. Porque a bd é uma classe abstrata e classes abstratas não podem ser instanciadas, devem ser herdadas por outras classes.
2. Ajusta os parâmetros de conexão (servidor, porta, banco, usuário, senha e opções extras) e conecta o banco de dados conforme seu tipo (mysql, postgresql ou mysqli).
3. A classe consulta é um subclasse da classe bd (ou seja, consulta herda bd).
4. O método **moveponteiro()** da classe consulta não pode ser acessado diretamente nem pelo objeto nem por outras classes que herdem a classe consulta, pois esse método é privativo (private), ou seja, só pode ser acessado por métodos internos da classe.

5. Precisamos incluir os seguintes métodos na classe consulta:

```
// Executa uma consulta com múltiplos comandos SQL
   public function executa_multi($_sql,$_tipo_res=NULL) {
      if($this->_connection===NULL) {
         return FALSE;
      }
      if($this->_bdtype==0) {
         return FALSE; // mysql não aceita consultas multiplas...
      }
      switch($this->_bdtype) {
         case 1:  $_cont = 0;
                  while(pg_connection_busy($this->_connection)) {
                     $_cont++;
                     sleep(1);
                     if($_cont>=10) {
                        return FALSE;
                     }
                  }
                  $this->_res = pg_send_query($this->_connection,$_sql);
                  break;
         case 2:  $this->_res = $this->_connection->multi_query($_sql);
                  break;
      }
      if($this->_res===FALSE) {
         return FALSE;
      } else {
         switch($this->_bdtype) {
            case 1:  $this->_res = pg_get_result($this->_connection);
                     break;
            case 2:  $this->_res = $this->_connection->store_result();
                     break;
         }
         if($this->_res!==FALSE) {
            $this->setAffectedRows();
            $this->setNumRows();
            $this->primeiro();   // Retorna o primeiro elemento
         } else {
            return FALSE;
         }
      }
   }

   // Verifica o status da consulta -> somente para mysqli
   public function status_multi() {
      if($this->_connection===NULL) {
```

```
            return FALSE;
        }
        if($this->_bdtype==0 || $this->_bdtype==1) {
            return FALSE; // mysql e postgresql
        }
        return $this->_connection->more_results();
    }

    // Retorna os próximos resultados da consulta múltipla
    public function busca_resultado() {
        if($this->_connection===NULL) {
            return FALSE;
        }
        if($this->_bdtype==0) {
            return FALSE; // mysql
        }
        switch($this->_bdtype) {
            case 1:  $this->_res = pg_get_result($this->_connection);
                     break;
            case 2:  if($this->_connection->next_result()) {
                        $this->_res = $this->_connection->store_result();
                     } else {
                        $this->_res = FALSE;
                     }
                     break;
        }
        if($this->_res!==FALSE) {
            $this->setAffectedRows();
            $this->setNumRows();
            $this->primeiro();
        } else {
            return FALSE;
        }
    }
```

Capítulo 21

1. Sim, pois os comentários HTML utilizam o formato <!-- comentários --> que é diferente do formato definido na classe (precisaríamos alterar o método toHTML para tratar esse tipo especial).
2. Inicia um tag no array HTML, ou seja, inclui a tag no array, incrementa o contador de tags, inclui atributos (caso eles tenham sido informados) e caso a tag não exija um finalizador, ajusta o flag apropriado de fechamento da tag.
3.

```
    public function toHTML() {
        $_nivel = 0;
```

```
$_f = FALSE;
if($this->_tagid==-1) {
   return FALSE;
} else {
      $htmltext = "";
      foreach($this->html as $key=>$vlr) {
         if($vlr[0]==FALSE) {
         $htmltext = "<p>ERRO na Estrutura.. Falta fechar a tag.." .
                     $this->tipostag[$vlr[1]] . " (id #" . $key .
                     ")</p>";
         return $htmltext;
      }
      if($vlr[2]==0) {
         // Start
         if($_f===FALSE&&$key>0) {
            $htmltext.= "\n";
         }
         $htmltext.= str_repeat(" ",$_nivel*3);
         $htmltext.= "<" . $vlr[1];
         if(is_Array($this->att[$key])) {
               foreach($this->att[$key] as $att=>$vlr) {
                  if($att=="_") {
                     $htmltext.= " " . $vlr;
                  } else {
                        $htmltext.= " " . $att . "=\"" . $vlr . "\"";
                  }
               }
         }
         $htmltext.= ">";
         if($this->text[$key]!="") {
               $htmltext.= $this->text[$key];
         }
         $_nivel++;
         $_f = FALSE;
      }
      if($vlr[2]==1) {
         // End
         $_nivel--;
         if($this->_flgend[$vlr[1]]===TRUE) {
            $htmltext.= str_repeat(" ",$_nivel*3);
               $htmltext.= "</" . $vlr[1] . ">";
         }
         if($vlr!="IMG") $htmltext.="\n";
         $_f = TRUE;
         }
      }
      return $htmltext;
```

```
        }
            return FALSE;
    }
```

4. Adiciona uma linha de cabeçalho à tabela.
5.

```
/* Classe Formulário */
class FORMULARIO extends HTML {
    private $_fnome = NULL;
    private $_facao = NULL;
    private $_fmetodo = "POST";
    private $_fenctype = NULL;
    private $_fsubmit = NULL;
    private $_fid = NULL;

    /* Construtor */
    function __construct() {
        parent::__construct();
    }

    /* métodos públicos */
    public function AddForm($_nome=NULL,$_acao=NULL,$_metodo=NULL,
                            $_enctype=NULL,$_submit=NULL) {
        $this->_fid = $this->AddTag("FORM");
        $this->setAcao($_SERVER["PHP_SELF"]);
        $_att = Array();
        if($_nome!==NULL) {
            $this->setNome($_nome);
        }
        if($_acao!==NULL) {
            $this->setAcao($_acao);
        }
        if($_metodo!==NULL) {
            $this->setMetodo($_metodo);
        }
        if($_enctype!==NULL) {
            $this->setEncType($_enctype);
        }
        if($_submit!==NULL) {
            $this->setSubmit($_submit);
        }
        $this->AddFormAtt();
        return $this->_fid;
    }

    public function AddFormAtt() {
```

```
   if($this->_fid===NULL) {
      return FALSE;
   }
   if($this->_fnome!==NULL) {
      $this->AddAtt($this->_fid,Array("NAME"=>$this->_fnome));
   }
   $this->AddAtt($this->_fid,Array("ACTION"=>$this->_facao));
   $this->AddAtt($this->_fid,Array("METHOD"=>$this->_fmetodo));
   if($this->_fenctype!==NULL) {
      $this->AddAtt($this->_fid,Array("ENCTYPE"=>$this->_fenctype));

   }
   if($this->_fsubmit!==NULL) {
      $this->AddAtt($this->_fid,Array("SUBMIT"=>$this->_fsubmit));

   }
}
public function setNome($_nome) {
   $this->_fnome = $_nome;
}

public function setAcao($_acao) {
   $this->_facao = $_acao;
}

public function setMetodo($_metodo) {
   $this->_fmetodo = $_metodo;
}

public function setEncType($_enc) {
   $this->_fenctype = $_enc;
}

public function setSubmit($_submit) {
   $this->_fsubmit = $_submit;
}

public function AddCampo(  $_nome,$_campo="INPUT",$_tipo="TEXT",
                           $_texto="",$_tam=NULL,$_outros=NULL) {
   if($this->_fid===NULL) {
      return FALSE;
   }
   $_id = $this->AddTAG($_campo);
   $_att["NAME"] = $_nome;
   if($_campo!="SELECT" && $_campo!="OPTION") {
      $_att["TYPE"] = $_tipo;
   }
```

```
        if($_tam!==NULL) {
           $_att["SIZE"] = $_tam;
        }
        if($_outros!==NULL) {
           $_att["_"] = $_outros;
        }
        $this->AddAtt($_id,$_att);
        $this->AddText($_id,$_texto);
        return $_id;
     }
  }
```

Capítulo 22

1.

```
     public function Incluir() {
        if($this->validar()) {
           // Ok podemos incluir o usuário
           $_sql = "INSERT INTO usuario(";
           $_sql.= "userName,userLogin,userPassw,";
           $_sql.= "userEmail,userNivel) VALUES(";
           $_sql.= "'{$this->_userName}',";
           $_sql.= "'{$this->_userLogin}',";
           $_sql.= "'{$this->_userPassw}',";
           $_sql.= "'{$this->_userEmail}',";
           $_sql.= "{$this->_userNivel})";
           if($this->executaSQL($_sql)) {
              // Enviar email
              $_msg = "Olá <b>{$this->_userName}</b>, <br>";
              $_msg.= "Bem-vindo! A partir de agora você é um usuário ";
              $_msg.= "registrado e contará com todos os privilégios ";
              $_msg.= "que nossos usuários possuem.<br>";
              $_msg.= "Se login é: {$this->_userLogin}<br>";
              $_msg.= "e sua senha é: {$this->_userPassw}<br><br>";
              $_msg.= "Atenciosamente, <br>";
              $_msg.= "Equipe do Site";
              $this->email("Seu Cadastro",$_msg);
              return TRUE;
           } else {
              return FALSE;
           }
        } else {
           return FALSE;
        }
     }
```

2. Classe usuario (incluído o método enviar_email) :

```
public function enviar_email($_assunto,$_mensagem) {
   $_sql = "SELECT userName, userEmail ";
   $_sql.= "FROM Usuario";
   $this->_bd->executa($_sql);
   while(($_dados = $this->_bd->getDados())!==FALSE) {
      $this->_userEmail = $_dados["userEmail"];
      $_msg = "Olá {$_dados["userName"]}<br>";
      $_msg.= $_mensagem;
      $this->email($_assunto,$_msg);
   }
}
```

Script man_usuario.php5.

1º Incluir o formulário de e-mail na opção default (listagem):

```
default:
        $_html->AddTag("P",NULL,TRUE,"<b>Relação de Usuários</b>");
        $_id[2] = $_html->AddTag("DIV");
        $_html->addText($_id[2],$_usr->listar());
        $_html->EndTag($_id[2]);
        // enviar Email para todos
        $_id[2] = $_html->AddTag("DIV");
        $_html->AddTag("P",NULL,TRUE,"<B>Enviar email para todos os
                                      usuários</b>");
        $_f = $_html->AddTag("FORM",Array("name"=>"email",
                        "method"=>"POST",
                        "action"=>$_SERVER["PHP_SELF"]));
        $_tab = $_html->AddTag("TABLE",Array("border"=>0,
                              "cellspacing"=>0,
                              "cellpadding"=>2,
                              "width"=>500,
                              "valign"=>"top"));
        // Assunto
        $_tr = $_html->AddTag("TR");
        $_html->AddTag("TD",NULL,TRUE,"Assunto:");
        $_html->EndTag($_td);
        $_html->EndTag($_tr);
        $_tr = $_html->AddTag("TR");
        $_td = $_html->AddTag("TD");
        $_html->AddTag("INPUT",Array("name"=>"AssuntoEmail",
                           "SIZE"=>75),TRUE);
        $_html->EndTag($_td);
        $_html->EndTag($_tr);
        // Mensagem
        $_tr = $_html->AddTag("TR");
        $_html->AddTag("TD",NULL,TRUE,"Mensagem:");
        $_html->EndTag($_td);
```

```
            $_html->EndTag($_tr);
            $_tr = $_html->AddTag("TR");
            $_td = $_html->AddTag("TD");
            $_html->AddTag("TEXTAREA",Array("name"=>"TextoEmail",
                                "ROWS"=>20,
                                "COLS"=>"75"),TRUE);
            $_html->EndTag($_td);
            $_html->EndTag($_tr);
            // Botão OK
            $_tr = $_html->AddTag("TR");
            $_td = $_html->AddTag("TD", Array("colspan"=>2,
                                              "align"=>"right"));
            $_html->AddTag("INPUT",Array("name"=>"ok",
                                "type"=>"submit",
                                "value"=>"Enviar Email"),TRUE);
            $_html->AddTag("INPUT",Array("name"=>"opcao",
                                "value"=>"EMAIL",
                                "type"=>"hidden"),TRUE);
            $_html->EndTag($_td);
            $_html->EndTag($_tr);
            $_html->EndTag($_tab);
            $_html->EndTag($_f);
            $_html->EndTag($_id[2]);
}
```

2º Tratar a opção EMAIL:

```
// Opção escolhida pelo Usuário (I,A ou E) ou (CA, CI ou CE)
if(isset($_GET["opt"])) {
   $_opcao = $_GET["opt"];
} elseif(isset($_POST["opcao"])) {
   $_opcao =  $_POST["opcao"];
}

switch($_opcao) {
   case 'EMAIL': // Enviar Email
            set_time_limit(0);
            $_usr->enviar_email( $_POST["AssuntoEmail"],
                                 $_POST["TextoEmail"]);
            $_opcao = "";
            break;
   case 'CA':  // Alteração dos dados
            if($_usr->buscar((int) $_POST["userid"])===FALSE) {
               echo "<script>alert('Usuário não Cadastrado');</script>";
               $_opcao = "";
   ... <<restante do programa man_usuario.php5>>
```

Capítulo 23

1. Retorna verdadeiro se um script estiver autorizado para determinado nível de acesso. Será utilizado pelos scripts para verificar se o usuário que está tentando acessá-lo tem permissão.
2. Não, pois na inclusão é feita uma chamada ao método **validar()** que faz uma chamada ao método **verifica()** com o nome do script que está sendo incluído, e caso o método **verifica()** retorne verdadeiro, o método **validar()** retorna falso, impedindo a inclusão.
3. O método **validar()**, como o nome diz, valida os dados que estão sendo incluídos ou alterados (permDesc e permScript), verifica ainda se o script informado existe ou não na tabela e caso exista e seja uma inclusão, ou caso exista e seja uma alteração e o código da permissão seja diferente da que está sendo alterada, retorna falso.