Apostila de PHP

Autor: Anderson Henrique R Maciel

Capítulo 1

PHP Tutorial

Com PHP você pode criar seu próprio sistema web.

Este tutorial ensina a introdução a linguagem PHP.

PHP é fácil de aprender - você vai gostar.

PHP Introdução

O que é PHP?

- PHP é uma linguagem de programação 100% voltada para a web.
- PHP significa PreProcessor Hipertext.
- O PHP é uma linguagem de programação interpretada.
- Podemos programar de forma estruturada ou orientado a objetos.

Extensões de arquivos

Abaixo temos os tipos de extensões de arquivos no PHP:

.php	Arquivo PHP contendo um programa.
.class.php	Arquivo contendo uma classe.
.inc.php	Arquivo PHP a ser incluído.

Delimitadores de código

O código de um programa em PHP deve estar contido entre os seguintes delimitadores:

<?php código; ?>

Cada linha do código deve finalizar com ponto-e-vírgula.

Comentando o código

O código de um programa PHP pode receber comentários seguindo a seguinte regra:

//	Para comentar uma única linha
#	Para comentar uma única linha
/* */	Para comentar diversas linhas
/** */	Para documentar o código

Comandos de saída

echo

É um comando que imprime uma ou mais variáveis no console.

```
 9    $l1 = 'a';
10    $l2 = 'b';
11    $l3 = 'c';
12    echo $l1, $l2, $l3;
```

print

É uma função que imprime uma string no console.

```
18    print('abc');
```

var_dump( )

Imprime o conteúdo de uma variável de forma explanativa, muito comum para se realizar debug.

```
25    $vetor = array('Palio','Corsa','Gol','Fiesta');
26    var_dump($vetor);
```

print_r( )

Imprime o conteúdo de uma variável de forma explanativa, mas em um formato mais legível para o programador, com os conteúdos alinhados e suprimindo os tipos de dados.

```
34    print_r($vetor);
```

Variáveis

São identificadores utilizados para representar valores mutáveis e voláteis, que só existem durante a execução do programa. Elas são armazenadas na memória RAM e seu conteúdo é destruído após a execução do programa. Para criar uma variável em PHP, precisamos atribuir-lhe um nome de identificação, sempre precedido pelo caractere **cifrão ($).**

```
12    $nome = "João";
13    $sobrenome = "da Silva";
14    echo "$sobrenome, $nome";
```

Podemos ter em nosso código-fonte nomes de variáveis que podem mudar de acordo com determinada situação. Neste caso, não só o conteúdo da variável é mutável, mas também seu nome. Para isso, utilizamos o conceito de variáveis variantes. Sempre que utilizarmos dois sinais de cifrão($$) precedendo o nome da variável, o PHP irá referenciar uma variável representada pelo conteúdo da primeira.

```
⚠    $variavel = 'nome';
⚠    $$variavel = 'Maria';
31    echo $nome;
```

Quando uma variável é atribuída a outra, sempre é criada uma nova área de armazenamento na memória. Apesar de $b receber o valor de $a, após qualquer modificação em $b, $a continua com o mesmo valor.

```
38    $a = 5;
      $b = $a;
      $b = 10;
41    echo $a, $b;
```

Para criar referência entre variáveis, ou seja, duas variáveis apontando para a mesma região da memória, a atribuição deve ser precedida pelo operador &. Assim qualquer alteração reflete na outra.

```
48    $c = 5;
      $d = &$c;
      $d = 10;
51    echo $c, $d;
```

Obs.: A linguagem PHP é case sensitive, ou seja, diferencia letras maiúsculas e minúsculas.

Tipos de variáveis

Tipo booleano

Expressa um valor lógico que pode ser verdadeiro ou falso. Para especificar um valor booleano, utilize as palavras-chave **TRUE** ou **FALSE**.

```
 8    $exibir_nome = TRUE;
 9
10    if($exibir_nome){
11        echo 'José da Silva';
12    }
```

Quando utilizamos os operadores relacionais, esta comparação também retorna um valor booleano (TRUE ou FALSE). O conteúdo da variável $vai_chover é um boolean.

```
19    $umidade = 91;
20    $vai_chover = ($umidade > 90);
21
22    if($vai_chover){
23        echo 'Está chovendo';
24    }
```

Tipo numérico

Números podem ser especificados em notação decimal (base 10), hexadecimal (base 16) ou octal (base 8), opcionalmente precedido de sinal (+ ou -).

```
33    //número decimal
34    $dec = 1234;
35    //número octal (equivalente a 83 em decimal)
36    $oct = 0123;
37    //número hexadecimal (equivalente a 26 em decimal)
38    $hex = 0x1A;
39    //ponto flutuante
40    $flo = 1.234;
41    //notação científica
42    $cie = 4e23;
```

Tipo string

É uma cadeia de caracteres alfanuméricos. Para declará-los podemos utilizar aspas simples '' ou aspas duplas "".

```
49    $str_s = 'Isto é um teste';
50    $str_d = "Isto é um teste";
```

Tipo array

É uma lista de valores armazenados na memória, os quais podem ser de tipos diferentes e podem ser acessados a qualquer momento, pois cada valor é relacionado a uma chave. Um array também pode crescer de forma dinâmica com a adição de novos itens.

```
59    $carros = array('Palio','Corsa','Gol');
60    echo $carros[1];
```

Tipo objeto

É uma entidade com um determinado comportamento definido por seus métodos (ações) e propriedades (dados). Para criar um objeto deve-se utilizar o operador new.

```
68  class Computador{
69      var $cpu;
70      function ligar(){
71          echo "Ligando computador a {$this->cpu}...";
72      }
73  }
74  $obj = new Computador;
75  $obj->cpu = "500Mhz";
76  $obj->ligar();
```

Tipo recurso

Recurso (resource) é uma variável especial que mantém referência de recurso externo. Recursos são criados e utilizados por funções especiais, como abrir um arquivo no modo leitura, retorna uma variável de referência do tipo recurso.

```
85    $ponteiro = fopen("06 - tipos_variaveis.php", "r");
86    var_dump($ponteiro);
```

Tipo NULL

A utilização do valor especial NULL significa que a variável não tem valor. NULL é o único valor possível do tipo NULL.

```
93    $nulo = NULL;
94    if(is_null($nulo)){
95        echo 'É do tipo NULL';
96    }
```

Constantes

Uma constante é um valor que não sofre modificações durante a execução do programa. Ela é representada por um identificador, assim como as variáveis, com a exceção de que só pode conter valores escalares (boolean, inteiro, ponto flutuante, string). Um valor escalar não pode ser composto de outros valores, como vetores ou objetos. Você pode definir uma constante utilizando a função define( ).

```
107   define("MAXIMO_CLIENTES", 100);
108   echo MAXIMO_CLIENTES;
```

Requisição de arquivo

Em linguagens de script como o PHP, frequentemente precisamos incluir dentro de nossos programas outros arquivos com definições de funções, constantes, configurações, ou mesmo carregar um arquivo contendo a definição de uma classe. Para atingir esse objetivo no PHP, podemos fazer uso de um dos seguintes comandos:

include <arquivo>	include_once <arquivo>
require <arquivo>	require_once <arquivo>

```
17    include 'quadrado.inc.php';
18    echo quadrado(4);
```

Capítulo 2

Manipuladores

Funções

Uma função é um pedaço de código com um objetivo específico, encapsulado sob uma estrutura única que recebe um conjunto de parâmetros e retorna um dado. Uma função é declarada uma única vez, mas pode ser utilizada diversas vezes. É uma das estruturas mais básicas para prover reusabilidade.

Sintaxe de declaração de uma função:

```php
function nome_funcao($arg1, $arg2, $argN){
    $valor = $arg1 + $arg2 + $argN;
    return $valor;
}
```

Variáveis globais

Todas as variáveis declaradas dentro do escopo de uma função são locais. Para acessar uma variável externa ao contexto de uma função sem passá-la como parâmetro, é necessário declará-la como **global**. Uma variável global é acessada a partir de qualquer ponto da aplicação.

```php
$total = 0;
function km2mi($quilometros){
    global $total;
    $total += $quilometros;
    return $quilometros * 0.6;
}
echo 'percorreu ' . km2mi(100) . " milhas <br>";
echo 'percorreu ' . km2mi(200) . " milhas <br>";
echo 'percorreu no total ' . $total . " quilometros <hr>";
```

Variáveis estáticas

Dentro do escopo de uma função podemos armazenar variáveis de forma estática. Assim, elas mantém o valor que lhes foi atribuído na última execução.

```php
function percorre($quilometros){
    static $total;
    $total += $quilometros;
    echo "percorreu mais $quilometros do total de $total <br>";
}
percorre(100);
percorre(200);
percorre(50);
```

Passagem de parâmetros

Existem dois tipos de passagem de parâmetros: por valor (**by value**) e por referência (**by reference**). Por padrão, os valores são passados **by value** para as funções. Assim, o parâmetro

que a função recebe é tratado como variável local dentro do contexto da função, não alterando o seu valor externo.

```php
function incrementaByValue($variavel, $valor){
    $variavel += $valor;
}
$a = 10;
incrementaByValue($a, 20);
echo $a;

function incrementaByReference(&$variavel, $valor){
    $variavel += $valor;
}
$b = 10;
incrementaByReference($b, 20);
echo $b;
```

Parâmetros com valor padrão

O PHP permite definir valores default para parâmetros. Assim, se o programador executar a função sem especificar o parâmetro, será assumido o valor predefinido.

```php
function incrementaParamDefault(&$variavel, $valor = 40){
    $variavel += $valor;
}
$c = 10;
incrementaParamDefault($c);
echo $c;
```

Argumentos variáveis

O PHP permite definir uma função com o número de argumentos variáveis, ou seja, permite obtê-los de forma dinâmica, mesmo sem saber quais são ou quantos são. Para obter quais são, utilizamos a função **func_get_args**( ); para obter a quantidade de argumentos, utilizamos a função **func_num_args**( )

```php
function Ola(){
    $argumentos = func_get_args();
    $quantidade = func_num_args();

    for($n=0;$n < $quantidade; $n++){
        echo '<br> Olá ' . $argumentos[$n];
    }
}
Ola('João','Maria','José','Pedro');
```

Recursão

O PHP permite chamada de funções recursivamente. No caso a seguir criaremos uma função para calcular um fatorial de um número.

```php
function Fatorial($numero){
    if($numero == 0 || $numero == 1){
        return 1;
    }else{
        return $numero * Fatorial($numero - 1);
    }
}
echo Fatorial(5);
echo Fatorial(7);
```

Strings

Uma string é uma cadeia de caracteres alfanuméricos. Para declarar uma string podemos utilizar aspas simples '' ou aspas duplas "".

```php
$variavel = 'Isto é um teste';
$variavel = "Isto é um teste";
```

A diferença é que todo conteúdo contido dentro de aspas duplas é avaliado pelo PHP. Assim, se a string contém uma variável, esta variável será traduzida pelo seu valor.

```php
$fruta = 'maçã';
print "como $fruta"; //resultado 'como maçã'
print 'como $fruta'; //resultado 'como $fruta'
```

Também podemos declara uma string literal com muitas linhas observando a sintaxe a seguir, na qual escolhemos uma palavra-chave para delimitar o início e o fim da string.

```php
$texto = <<<CHAVE
        Aqui nesta área
        você poderá escrever
        textos com múltiplas linhas
CHAVE;
```

Concatenação

Para concatenar strings, pode-se utilizar o operador "." ou colocar múltiplas variáveis dentro de strings com aspas duplas "", uma vez que seu conteúdo é interpretado.

```php
$fruta = 'maçã';

echo $fruta.' é a fruta de adão';
echo "{$fruta} é a fruta de adão";
```

O PHP realiza automaticamente a conversão entre tipos. Como neste exemplo de concatenação entre uma string e um número:

```php
$a = 1234;

echo 'O salário é '. $a . '<br>';
echo "O salário é $a <br>";
```

Caracteres de escape

Dentro de aspas duplas podemos utilizar controles especiais interpretados diferentemente pelo PHP, que são os caracteres de escape (\). Veja a seguir os mais utilizados:

\n	Nova linha, proporciona quebra de linha.
\r	Retorno de carro.
\t	Tabulação.
\\	Barra invertida é o mesmo que "\".
\"	Aspas duplas.
\$	Símbolo de $.

```php
echo "seu nome é \"Paulo \".";
echo 'seu nome é "Paulo".';
echo 'seu salário é $650,00';
echo "seu salário é \$650,00";
```

As funções a seguir formam um grupo cuja característica comum é a manipulação de cadeias de caracteres (strings), como conversões, transformações, entre outras funcionalidades.

strtoupper

Transforma uma string para maiúsculo. Retorna a string com todos os caracteres alfabéticos convertidos para maiúsculo.

```php
echo strtoupper("Convertendo para maiusculo");
```

strtolower

Transforma uma string para minúsculo. Retorna a string com todos os caracteres alfabéticos convertidos para minúsculo.

```php
echo strtolower("CONVERTENDO PARA MINUSCULO");
```

substr

Retorna parte de uma string. Retorna uma porção de conteúdo, começando em início, contendo comprimento em caracteres. Se comprimento for negativo, conta n caracteres antes do final.

```php
$rest = substr("América", 1);
echo "$rest <br>";
$rest = substr("América", 1,3);
echo "$rest <br>";
$rest = substr("América", 0,-1);
echo "$rest <br>";
$rest = substr("América", -2);
echo "$rest <br>";
```

str_pad

Preenche uma string com uma outra string, dentro de um tamanho específico.

```php
$texto = "The Beatles";
print str_pad($texto, 20) . "<br>";
print str_pad($texto, 20,"*",STR_PAD_LEFT) . "<br>";
print str_pad($texto, 20,"*",STR_PAD_BOTH) . "<br>";
print str_pad($texto, 20,"*",STR_PAD_RIGHT) . "<br>";
```

str_repeat

Repete uma string uma certa quantidade de vezes.

```php
$txt = ".oOOOo.";
print str_repeat($txt, 5) . "<br>";
```

strlen

Retorna o comprimento de uma string.

```php
$com = "O Rato roeu a roupa do rei de roma";
print 'O comprimento é: '.strlen($com). "<br>";
```

str_replace

Substitui uma string por outra em um dado contexto.

```php
$rep = "O Rato roeu a roupa do rei de roma";
print str_replace("Rato", "Leão", $rep) . "<br>";
```

strpos

Encontra a primeira ocorrência de uma string dentro de outra.

```php
$minha_string = "O Rato roeu a roupa do rei de roma";
$encontrar = 'roupa';
$posicao = strpos($minha_string, $encontrar);
if($posicao){
    echo "String encontrada na posição $posicao";
}else{
    echo "String não encontrada";
}
```

sha1

Função utilizada para criptografar senhas, pode ser utilizada de forma encadeada, retorna 40 caracteres hexadecimais.

```
151    $senha = '123abc';
152    echo 'Criptografada (40 caracteres): '.sha1($senha)."<br>";
```

md5

Função utilizada para criptografar senhas, pode ser utilizada de forma encadeada, retorna 32 caracteres hexadecimais.

```
159    echo 'Criptografada (32 caracteres): '.md5($senha)."<hr>";
```

Números

O PHP nos permite facilmente formatar números, definindo quantidade de casas decimais, o separador de decimais, separador de milhares utilizando a função **number_format( )**.

Define quantidade de casas decimais para 02

```
6    $dec = 25.87499;
7    echo number_format($dec, 2) . "<br>";
```

Define o caractere utilizado para separar decimais (vírgula)

```
13   $n = 8.89;
14   echo number_format($n, 2, ",", NULL) . "<br>";
```

Define o caractere utilizado para separar milhares (ponto)

```
19   $s = 32578.97;
20   echo number_format($s, 2, ",", ".") . "<br>";
```

Retorna o valor da constante matemática PI

```
25    echo pi() . "<br>";
```

Retorna a raiz quadrada de um número

```
30   echo sqrt(81) . "<br>";
```

Retorna a potência de um número pow(base, potencia)

```
35    echo pow(2, 3) . "<br>";
```

Retorna um número randômico dentro de um intervalo definido

```
40    echo rand(1, 10) . "<br>";
```

Arredonda número decimal para o primeiro número inteiro acima

```
45   echo ceil(1.01) . "<br>";
```

Converte dado binário em hexadecimal

```
50    echo bin2hex(001) . "<br>";
```

Converte dado binário em decimal

```
55    echo bindec(1001) . "<br>";
```

Converte hexadecimal em dado binário

```
60    echo hex2bin("1C") . "<br>";
```

Converte hexadecimal em decimal

```
65    echo hexdec("B45") . "<br>";
```

Converte decimal em dado binário

```
70    echo decbin(9) . "<br>";
```

Converte decimal em octo

```
75    echo decoct(9) . "<br>";
```

Converte decimal em hexadecimal

```
80    echo dechex(1010) . "<br>";
```

Capítulo 3

Arrays

É, sem dúvida, um dos recursos mais poderosos da linguagem. O programador que assimilar bem esta parte terá muito mais produtividade no seu dia-a-dia. Isto porque os arrays no PHP servem como verdadeiros contêineres, servindo para armazenar números, strings, objetos, dentre outros, de forma dinâmica. Além disso, o PHP nos oferece uma gama enorme de funções para manipulá-los, as quais serão vistas a seguir.

Criando um array

Arrays são acessados mediante uma posição, como um índice numérico. Para criar um array, pode-se utilizar a função array([chave] => valor,...)

```
    $cores = array('vermelho','azul','verde','amarelo');
    $cores = array(0 => 'vermelho', 1 => 'azul', 2 => 'verde',
20                                                  3 => 'amarelo');
```

Outra forma de criar um array é simplesmente adicionando-lhe valores com a seguinte sintaxe:

```
25    $nomes[] = 'maria';
26    $nomes[] = 'joão';
27    $nomes[] = 'carlos';
28    $nomes[] = 'josé';
```

De qualquer forma, para acessar o array indexado, basta indicar o seu índice entre colchetes:

```
35    echo $cores[0] . "<br>";
36    echo $cores[1] . "<br>";
37    echo $cores[2] . "<hr>";
38
39    echo $nomes[0] . "<br>";
40    echo $nomes[1] . "<br>";
41    echo $nomes[2] . "<hr>";
```

Arrays associativos

Os arrays no PHP são associativos pois contêm uma chave de acesso para cada posição. Para criar um array, pode-se utilizar a função array([chave] => valor).

```
48    $hex = array('vermelho' => 'FF0000', 'verde' => '00FF00',
49                                          'azul' => '0000FF');
```

Outra forma de criar um array associativo é simplesmente adicionando-lhes valores com a seguinte sintaxe:

```
56    $pessoa['nome'] = 'Maria da Silva';
57    $pessoa['rua'] = 'São João';
58    $pessoa['bairro'] = 'Cidade Alta';
59    $pessoa['cidade'] = 'Porto Alegre';
```

De qualquer forma, para acessar o array, basta indicar sua chave entre colchetes:

```
64    echo $hex['vermelho'] . "<br>";
65    echo $hex['azul'] . "<br>";
66    echo $hex['verde'] . "<hr>";
67
68    echo $pessoa['nome'] . "<br>";
69    echo $pessoa['rua'] . "<br>";
70    echo $pessoa['bairro'] . "<hr>";
```

Iterações

Os arrays podem ser iterados no PHP pelo operador FOREACH, percorrendo cada uma das posições do array. Exemplo:

```
78    $frutas['cor'] = 'vermelha';
79    $frutas['sabor'] = 'doce';
80    $frutas['formato'] = 'redonda';
81    $frutas['nome'] = 'maçã';
82
83    foreach ($frutas as $chave => $fruta){
84        echo "$chave => $fruta <br>";
85    }
```

Acesso

As posições de um array podem ser acessadas a qualquer momento, e sobre elas operações podem ser realizadas.

```
93    $minha_multa['carro'] = 'Pálio';
94    $minha_multa['valor'] = 178.00;
95
96    $minha_multa['carro'] .= ' ED 1.0';
97    $minha_multa['valor'] += 20.00;
98
99    print "<pre>";
100   var_dump($minha_multa);
101
102   $comidas[] = 'Lazanha';
103   $comidas[] = 'Pizza';
104   $comidas[] = 'Macarrão';
105
106   $comidas[1] = 'Pizza Calabreza';
107
108   var_dump($comidas);
```

Arrays multidimensionais

Também conhecidas como matrizes são arrays nas quais algumas de suas posições podem conter outros arrays de forma recursiva. Um array multidimensional pode ser criado pela função array().

```
117   $veiculos = array(
118       'Palio' => array(
119           'cor'=>'azul',
120           'potência'=>'1.0',
121           'opcionais'=>'Ar Cond.'
122           ),
123       'Corsa' => array(
124           'cor'=>'cinza',
125           'potência'=>'1.3',
126           'opcionais'=>'MP3'
127           ),
128       'Gol' => array(
129           'cor'=>'branco',
130           'potência'=>'1.0',
131           'opcionais'=>'Metálica'
132           )
133   );
```

Para realizar iterações em um array multidimensional é preciso observar quantos níveis ele possui. No exemplo a seguir, realizamos a iteração para o primeiro nível do array(veiculos) e, para cada iteração, realizamos uma nova iteração, para imprimir suas características.

```php
foreach ($veiculos as $modelo => $caracteristicas){
    echo "=> Modelo: $modelo <br>";
    foreach ($caracteristicas as $caracteristica => $valor){
        echo "característica $caracteristica => $valor <br>";
    }
}
```

A seguir veremos uma série de funções utilizadas exclusivamente para manipulação de arrays, funções de ordenação, intersecção, acesso, dentre outras.

array_push

Adiciona elementos ao final de um array.

```php
$a = array('verde','azul','vermelho');
array_push($a, 'amarelo');
print_r($a);
```

array_pop

Remove um valor do final de um array.

```php
array_pop($a);
print_r($a);
```

array_shift

Remove um elemento do início de um array.

```php
array_shift($a);
print_r($a);
```

array_unshift

Adiciona um elemento no início de um array.

```php
array_unshift($a, 'laranja');
print_r($a);
```

array_pad

Preenche um array com um dado valor, determinada quantidade de posições.

```php
$cli = array('ana','marcos','simone');
$cli = array_pad($cli, 6, 'luiza');
print_r($cli);
```

array_reverse

Recebe um array e retorna-o na ordem reversa.

```
195    $num[0] = 1;
196    $num[1] = 2;
197    $num[2] = 3;
198    $num[3] = 4;
199    $rev = array_reverse($num,TRUE);
200    print_r($rev);
```

array_merge

Mescla dois ou mais arrays. Um array é adicionado ao final de outro. O resultado é um novo array. Se ambos arrays tiverem conteúdo indexado pela mesma chave, o segundo irá sobrepor ao primeiro.

```
208    $times_paulistas = array('Palmeiras','Corinthians');
209    $times_cariocas = array('Vasco da Gama','Botafogo');
210    $times_juntos = array_merge($times_paulistas,$times_cariocas);
211    print_r($times_juntos);
```

array_keys

Retorna as chaves de um array. Se o segundo parâmetro for indicado, a função retornará apenas índices que apontam para um conteúdo igual ao parâmetro.

```
218    $aluno = array('matricula'=>'0012','nome'=>'liliane','idade'=>14);
219    $indices = array_keys($aluno);
220    print_r($indices);
```

array_values

Retorna um array contendo os valores de um outro array.

```
226    $valores = array_values($aluno);
227    print_r($valores);
```

array_slice

Extrai uma porção de um array.

```
233    $color[0] = 'green';
234    $color[1] = 'yellow';
235    $color[2] = 'red';
236    $color[3] = 'blue';
237    $color[4] = 'gray';
238    $color[5] = 'white';
239    $fatia = array_slice($color, 2, 3);
240    print_r($fatia);
```

count

Retorna a quantidade de elementos de um array.

```
246    $bebidas = array('refrigerante','suco','vinho','cerveja','vodka');
247    echo 'O array $bebidas contém '.count($bebidas).' elementos <hr>';
```

in_array

Verifica se um array contém um determinado valor.

```
253  if(in_array('café', $bebidas)){
254      echo 'café foi encontrado <hr>';
255  }else{
256      echo 'café não foi encontrado <hr>';
257  }
```

sort

Ordena um array pelo seu valor, não mantendo associação de índices.

```
263    sort($bebidas);
264    print_r($bebidas);
```

rsort

Ordena um array em ordem reversa pelo seu valor, não mantendo a associação de índices.

```
271    rsort($bebidas);
272    print_r($bebidas);
```

asort

Ordena um array pelo seu valor, mantendo a associação de índices. Para ordenar de forma reversa, use o arsort().

```
279    asort($bebidas);
280    print_r($bebidas);
```

ksort

Ordena um array pelos seus índices. Para ordem reversa, utilize o krsort().

```
286    $automovel['potência'] = '1.0';
287    $automovel['cor'] = 'branco';
288    $automovel['modelo'] = 'celta';
289    $automovel['opcionais'] = 'ar quente';
290    ksort($automovel);
291    print_r($automovel);
```

explode

Converte uma string e um array, separando os elementos por meio de um identificador.

```
297    $data = "31/12/2018";
298    print_r(explode("/", $data));
```

Implode

Converte um array em uma string, unindo os elementos da string por meio de um identificador.

```
305    $padrao = array('Maria','Paulo','Joana','Loiane');
306    $resultado = implode(' - ', $padrao);
307    print_r($resultado);
```

Capítulo 4

Data, hora e arquivo

Na linguagem PHP, facilmente conseguimos manipular informações locais e regionais como data, hora. Para isso, contamos com funções nativas da linguagem, as quais utilizaremos neste capítulo.

Função date( )

Retorna a data atual, para isto precisamos informar em qual formato essa data deverá ser retornada:

Ano -> Y retorna no padrão 4 dígitos (2019).
Ano -> y retorna no padrão 2 dígitos (19).

```
11    echo date("d/m/Y") . "<br>";
```

Retorna a hora atual, para isto precisamos informar em qual formato essa hora deverá ser retornada:

Hora -> H retorna no padrão 24 horas (18:30)
Hora -> h retorna no padrão 12 horas (06:30 PM)

```
20    echo date("H:i:s") . "<br>";
```

Parâmetro dia (Day)

d -> representa o dia do mês utilizando 2 dígitos (25)
j -> representa o dia do mês não utiliza o dígito 0 (7)
D -> representa o dia da semana utilizando 3 caracteres (Mon)
l -> representa o dia da semana utilizando texto completo (Monday)
N -> representa numericamente o dia da semana (1-7)
z -> representa o dia do ano (0 - 365)

```php
echo date("d") . "<br>";
echo date("j") . "<br>";
echo date("D") . "<br>";
echo date("l") . "<br>";
echo date("z") . "<br>";
```

Parâmetro mês (Month)

F -> representa o mês utilizando texto completo (February)
m -> representa o mês numericamente utilizando o dígito 0 (01-12)
M -> representa o mês utilizando 3 caracteres (Feb)
n -> representa o mês numericamente sem o dígito 0 (1-12)
t -> representa o número de dias do mês (inclui 28 e 30)

```php
echo date("F") . "<br>";
echo date("m") . "<br>";
echo date("M") . "<br>";
echo date("n") . "<br>";
echo date("t") . "<br>";
```

Timezone

Dependendo da nossa localização e do continente e país que nos encontramos, sabemos que o fuso horário pode interferir na data e hora dos nossos sistemas, para isso o PHP nos permite trabalhar facilmente com o Timezone.

Retorna o timezone utilizado como padrão

```php
echo date_default_timezone_get();
```

Permite definir um timezone

```php
date_default_timezone_set("Europe/Lisbon");
```

Objeto DateTime

Converte uma string (válida) em um objeto do tipo DateTime( )

```php
$strData = "2019-02-25";
$objDateTime = date_create($strData);
```

Define um formato de data em um objeto do tipo DateTime( )

```php
echo date_format($objDateTime, "d/m/Y");
```

Arquivo e Diretório

Na linguagem PHP, podemos trabalhar com arquivos de forma fácil, para isso, utilizamos a variável global **$_FILES[ ][ ]** para que o arquivo que está sendo carregado no formulário seja recebido em uma variável.

Neste exemplo, utilizaremos um formulário para enviar o arquivo para um código PHP que se responsabilizará pela manipulação do mesmo.

```
<h3>Envio de Arquivo</h3>
<form method="post" action="upload.php"
                    enctype="multipart/form-data">
    <input type="file" name="arquivo" required>
    <input type="submit" value="Upload">
</form>
```

Observe que o formulário possui um atributo chamado **enctype** que possui o valor **multipart/form-data**, sem ele não é possível o envio de arquivos via formulário. O atributo **action** chama o arquivo **upload.php**, este terá o código necessário para a manipulação do arquivo que será enviado.

A variável global $_FILES[ ][ ], em sua sintaxe, entre colchetes receberá no primeiro colchete o valor do atributo **name** do campo input, e no segundo poderá receber três valores diferentes como nos mostra a tabela abaixo:

atributo	o que significa
name	Pega o nome do arquivo e extensão (carro.jpg)
size	Pega o tamanho do arquivo (medido em bytes)
type	Pega o mimetype do arquivo (image/jpg)

Abaixo segue o código do arquivo **upload.php**:

Pega o nome do arquivo e sua extensão.

```
$arquivo = $_FILES["arquivo"]["name"];
```

Pega o tamanho do arquivo (bytes).

```
$tamanho = $_FILES["arquivo"]["size"];
```

Pega o mimetype do arquivo.

```
$tipo = $_FILES["arquivo"]["type"];
```

Verifica se a localização se refere a um arquivo, retorna booleano.

```
if(is_file($_FILES["arquivo"]["tmp_name"])){
    echo "É arquivo <br>";
}else{
    echo "Não é arquivo <br>";
}
```

Cria um array de mimetypes.

```
$mimeType = array("image/jpg",
                  "image/jpeg",
                  "image/gif",
                  "image/png",
                  "image/bmp");
```

Verifica se o array contém o valor especificado, retorna booleano.

```php
if(in_array($tipo, $mimeType)){
    echo "Tipo de arquivo válido <br>";
}else{
    echo "Formato de arquivo inválido <br>";
}
```

Move o arquivo para um diretório especificado, retorna booleano.

```php
if(move_uploaded_file($_FILES["arquivo"]["tmp_name"], $dir.$arquivo)){
    echo "Arquivo movido para diretório: $dir <br>";
}else{
    echo "Erro ao mover arquivo para diretório <br>";
}
```

Verifica se o arquivo/diretório já existe, retorna booleano.

```php
if(file_exists("upload/$arquivo")){
    echo "Esse arquivo já existe <br>";
}else{
    echo "Esse arquivo ainda não existe <br>";
}
```

Apaga um arquivo de um diretório, retorna booleano.

```php
if(unlink("upload/$arquivo")){
    echo "Arquivo foi apagado! <br>";
}else{
    echo "Erro ao apagar o arquivo <br>";
}
```

A seguir veremos uma série de funções utilizadas exclusivamente para manipulação de arquivos, como abertura, leitura, escrita e fechamento dos mesmos.

fopen

Abre um arquivo e retorna um identificador.

```php
$filename = "upload.php";
$ponteiro = fopen($filename, "r");
var_dump($ponteiro);
```

Parâmetro	Modo de abertura
r	Abre o arquivo em modo leitura, ponteiro no início do arquivo.
r+	Abre o arquivo em modo leitura e escrita, ponteiro no início do arquivo.
w	Abre o arquivo em modo escrita, se arquivo não existir o cria.
w+	Abre o arquivo em modo leitura e escrita, se arquivo não existir o cria.
a	Abre o arquivo em modo escrita, ponteiro inicia no final do arquivo.
a+	Abre o arquivo em modo leitura e escrita, ponteiro inicia no final do arquivo.
x	Cria e abre o arquivo em modo escrita.
x+	Cria e abre o arquivo em modo leitura e escrita.

feof

Verifica se um determinado identificadorde arquivo (criado pela função fopen( )) está no fim do arquivo (End Of File). Retorna booleano.

fgets

Lê uma linha de um arquivo. Retorna uma string com até (tamanho – 1) bytes lidos do arquivo apontado pelo identificador de arquivo. Se nenhum tamanho for informado, o default é 1 Kb ou 1024 bytes.

fclose

Fecha o arquivo aberto apontado pelo identificador de arquivo. Retorna booleano.

```
14  while(!feof($ponteiro)){
15
16      $buffer = fgets($ponteiro, 4096);
17
18      echo "$buffer <br>";
19  }
20
21  fclose($ponteiro);
```

fwrite

Grava uma string (conteúdo) no arquivo apontado pelo identificador de arquivo. Se o argumento comprimento é dado, a gravação irá parar depois que comprimento bytes for escrito ou o fim da string conteúdo for alcançado. Retorna o número de bytes gravados. Se arquivo não existir será criado.

```
23  $ponteiro2 = fopen("gravar.txt", "w");
24
25  fwrite($ponteiro2, "Linha 1 \n");
26  fwrite($ponteiro2, "Linha 2 \n");
27  fwrite($ponteiro2, "Linha 3 \n");
```

file_put_contents

Grava uma string em um arquivo independente se apontado pelo identificador de arquivo. Retorna a quantidade de bytes gravados. Se arquivo não existir será criado.

```
29  echo file_put_contents("escrever.txt", "Este é \n o conteúdo gravado");
```

file_get_contents

Lê o conteúdo de um arquivo independente se apontado pelo identificador de arquivo e retorna o conteúdo em forma de string.

```
31  echo file_get_contents("upload.php");
```

file

Lê um arquivo e retorna um array com todo o seu conteúdo, de modo que cada posição do array representa uma linha lida do arquivo.

```php
$arrayLeitura = file("delete.php");

foreach ($arrayLeitura as $linha){
    echo "$linha <br>";
}
```

copy

Copia um arquivo para outro (diretório) local/nome. Retorna booleano. Antes de efetuarmos a cópia precisamos garantir que o diretório existe, para isso utilizamos a função **mkdir** para criar o diretório e a função **is_dir** para verificar se o caminho apontado é um diretório.

```php
$origem = "upload.php";
$destino = "bkp/upload.php";

if (mkdir("bkp")) {
    echo "Diretório foi criado! <br>";
}
$dir = "bkp";

if (is_dir($dir)) {
    if (copy($origem, $destino)) {
        echo "Cópia efetuada!";
    } else {
        echo "Erro ao copiar!";
    }
}
```

rename

Altera a nomenclatura de um arquivo ou diretório. Retorna booleano.

```php
$novo = "bkp/bkp_upload.php";
$velho = "bkp/upload.php";

if(rename($velho, $novo)){
    echo "Arquivo renomeado!";
}else{
    echo "Erro ao renomear!";
}
```

getcwd

Retorna o diretório corrente.

```php
echo "O diretório atual é " . getcwd();
```

rmdir

Apaga um diretório se o mesmo estiver vazio, diretórios que contém arquivos não são apagados. Retorna booleano.

opendir

Abre um diretório e retorna um identificador.

```php
$dir = "cookies";

$identificador = opendir($dir);
var_dump($identificador);
```

readdir

Realiza a leitura do conteúdo de um diretório por meio do identificador criado pela função opendir( ).

closedir

Libera um recurso alocado pela função opendir( ). Fecha o diretório.

```php
if(is_dir($dir)){
    $ident = opendir($dir);
    while($arquivo = readdir($ident)){
        echo "$arquivo <br>";
    }
}
closedir($ident);
```

Capítulo 5

Persistência dos dados (cookies e session)

Se você precisa manter informações sobre seus usuários enquanto eles navegam pelo seu sistema, ou até quando eles saem do sistema, é importante que você saiba lidar com cookies e sessões. Cookies são arquivos-texto que podem ser armazenados no computador do usuário para serem recuperados posteriormente pelo servidor no qual seu sistema está hospedado.

Sessões são recursos que podemos utilizar para manter uma conexão com o usuário, enquanto ele estiver navegando no site. Entre algumas utilidades de cookies e sessões, podem ser citadas:

- **Autenticação de usuários**: criação de um sistema envolvendo login, autenticação e logout, o que garante o acesso do conteúdo somente aos usuários autorizados.
- **Carrinho de compras:** utilizando nos sites de comércio eletrônico para armazenar todos os produtos já selecionados pelo cliente para compra, enquanto ele navega pelo site da loja.
- **Exibição de anúncios ou imagens:** para não exibir mais de uma vez um mesmo anúncio ou uma imagem para o usuário é necessário manter informações sobre as imagens que já foram exibidas.
- **Personalização de páginas:** por exemplo, uma livraria virtual poderia exibir o anúncio de um livro de culinária, caso o usuário tivesse feito uma pesquisa pela palavra “culinária” na última vez que acessou o site.

Utilizando cookies

Um cookie é formado pelo par nome/valor, ou seja, possui um nome pelo qual ele é referenciado e um valor associado a esse nome. Podem ser utilizados em qualquer aplicação que necessite

compartilhar dados entre diferentes páginas ou até entre diferentes acessos (em dias e horários distintos).

O PHP nos oferece a função **setcookie**, que envia cookies para o computador do usuário. Essa função é usada tanto para definir um cookie como para excluí-lo. Sua sintaxe é a seguinte:

setcookie(NOME_COOKIE,VALOR_COOKIE);

```
6    setcookie("Login", "Admin");
7    setcookie("Senha", "123abc");
```

Criando cookie válido por 2 dias, utilizando a função time( ) para obter o tempo atual e somamos 172.800 segundos, que equivalem a 48 horas. Função time( ) trabalha com o padrão UNIX (desde 1 de janeiro de 1970, à 0h).

```
14    setcookie("Autenticado", TRUE, time()+172800);
```

Obtendo o valor do cookie

O array superglobal **$_COOKIE** no PHP moderno não é mais utilizado por questões de segurança, agora utilizamos filter_input(INPUT_COOKIE, NOME_COOKIE), recuperando o valor dessa forma:

```
22    echo "Login: " . filter_input(INPUT_COOKIE, "Login") . "<br>";
23    echo "Senha: " . filter_input(INPUT_COOKIE, "Senha") . "<br>";
```

```
25  if(filter_input(INPUT_COOKIE, "Autenticado")){
26      echo "Usuário foi autenticado! <br>";
27  }else{
28      echo "Usuário não foi autenticado! <br>";
29  }
```

Excluindo cookies

Basta definirmos novamente com o mesmo nome identificador e não passar valores no segundo parâmetro, ou passar valor NULL.

```
36    setcookie("Login", NULL);
37    setcookie("Senha", NULL);
```

Criando sessões e utilizando

Para utilizarmos sessões, primeiramente devemos inicializar utilizando a função **session_start()**, criamos a sessão utilizando a variável superglobal **$_SESSION[Identificador]**.

No próximo exemplo, iremos criar duas sessões para exemplificar o uso em nossos sistemas:

Inicializa a sessão

```
6    session_start();
```

Criando sessões

```
11    $_SESSION["Usuario"] = "Jair da Silva";
12    $_SESSION["Autenticado"] = true;
```

Obtendo o valor da sessão utilizando o array superglobal $_SESSION[ ]

```php
if($_SESSION["Autenticado"]){
    echo "O usuário {$_SESSION['Usuario']} foi autenticado! <br>";
}else{
    echo "O usuário {$_SESSION['Usuario']} não foi autenticado! <br>";
}
```

Para finalizar, podemos apagar todas as variáveis da sessão, e ou destruir os dados de uma sessão, garantindo assim que os dados não serão mais acessados, pois a persistência foi perdida.

```php
session_unset();
session_destroy();
```

Capítulo 6

Integrando o MySQL com o PHP

Como vimos anteriormente, as variáveis armazenam dados de forma temporária, assim que o programa termina sua execução a informação é destruída da memória RAM. Para resolver este problema, podemos armazenar as informações em um BD através de SGDB confiável. Neste capítulo, iremos trabalhar com funções que nos auxiliam na integração da linguagem com o SGDB MySQL.

Função	Para que serve
mysqli_connect( )	Estabelece conexão com o servidor web.
mysqli_select_db( )	Seleciona um banco de dados específico.
mysqli_error( )	Retorna um erro, caso ocorra.
mysqli_query( )	Executa uma instrução do SQL (DDL, DML, DQL).
mysqli_close( )	Encerra conexão com o servidor web.
mysqli_affected_rows( )	Retorna quantidade de registros afetados.
mysqli_num_rows( )	Retorna quantidade de registros encontrados.
mysqli_fetch_assoc( )	Armazena registros em array associativo.
mysqli_fetch_all( )	Armazena registros em array numérico.
mysqli_fetch_array( )	Armazena registros em arrays (associativo, numérico, misto)
mysqli_fetch_object( )	Armazena registros em um objeto dinâmico.
mysqli_fetch_field( )	Retorna o próximo campo em um resultado.
mysqli_fetch_fields( )	Retorna um array de objetos representa os campos em um resultado.

Acesse o phpmyadmin, e na opção SQL, digite o código abaixo para criar o banco de dados e a tabela:

```sql
CREATE DATABASE php_integrado;

USE php_integrado;

CREATE TABLE pessoa(

    id              int(11)     PRIMARY KEY AUTO_INCREMENT NOT NULL,
    nome            varchar(20) NOT NULL,
    sobrenome       varchar(20) NOT NULL,
    data_nascimento date        NOT NULL,
    genero          char(1)     NULL,
    estado_civil    varchar(10) NULL
)
```

Obs.: Aumentar o tamanho do campo **estado_civil** para 15.

Arquivo de conexão com o servidor

Para estabelecermos a conexão com o servidor, precisamos saber quais os parâmetros de configuração necessários para que isto aconteça. Vamos utilizar a função mysqli_connect( ) e a função mysqli_select_db( ).

Parâmetro	Valor
HOST (Servidor)	localhost ou 127.0.0.1
USER (Usuário)	root
PASS (Senha)	----------
PORT (Porta)	3306
DBNAME (Nome do Banco de Dados)	php_integrado

Com estas informações, vamos criar um arquivo PHP chamado **conecta.php**, guardar esses parâmetros dentro das constantes, estabelecer a conexão e selecionar o banco de dados.

```php
    define("HOST", "localhost");
    define("USER", "root");
    define("PASS", "");
    define("PORT", "3306");
    define("DBNAME", "php_integrado");
```

Linha código que estabelece conexão com o servidor, abaixo:

```php
    $conn = mysqli_connect(HOST, USER, PASS, NULL, PORT, NULL)
                                        or die(mysqli_error());
```

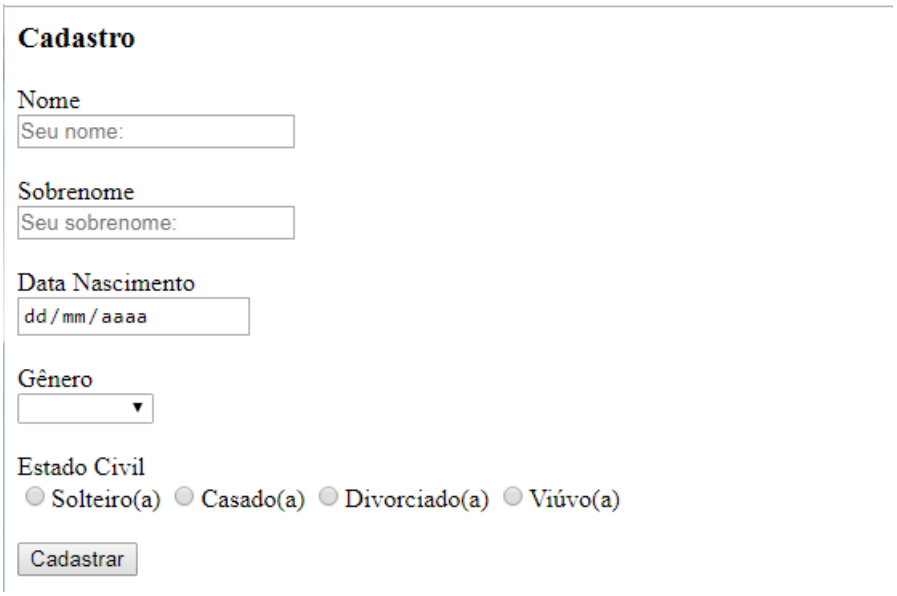
Linha código que seleciona um banco de dados, abaixo:

```php
    mysqli_select_db($conn, DBNAME) or die(mysqli_error());
```

Comando DML (INSERT INTO) com o PHP

Agora que nossa conexão foi estabelecida e o banco de dados selecionado, teremos que criar um arquivo (**form_inserir.php**) que terá um formulário para o cadastro, assim utilizaremos o DML **INSERT INTO** na tabela pessoa:

```
<h3>Cadastro</h3>
<form method="post" action="insert.php">
    Nome<br>
    <input type="text" name="nome" required placeholder="Seu nome:"><br><br>
    Sobrenome<br>
    <input type="text" name="sobrenome" required placeholder="Seu sobrenome:"><br><br>
    Data Nascimento<br>
    <input type="date" name="data" required><br><br>
    Gênero<br>
    <select name="genero">
        <option selected disabled></option>
        <option value="M">Masculino</option>
        <option value="F">Feminino</option>
    </select><br><br>
    Estado Civil<br>
    <input type="radio" name="es" value="Solteiro(a)" required>Solteiro(a)
    <input type="radio" name="es" value="Casado(a)" required>Casado(a)
    <input type="radio" name="es" value="Divorciado(a)" required>Divorciado(a)
    <input type="radio" name="es" value="Viúvo(a)" required>Viúvo(a)<br><br>
    <input type="submit" value="Cadastrar">
</form>
```



Perceba que o atributo **action** na TAG form está chamando o arquivo **insert.php**. Este arquivo contém o código que utilizará o DML necessário para efetuar o cadastro na tabela do banco de dados.

Requisição do arquivo de conexão

```
include_once '11 - integrando_mysql_e_php.php';
```

Utilizando a função filter_input() para pegar os valores digitados e enviados através do método POST, o PHP recebe como atributos deste método (METODO, NOME_CAMPO), nome do campo está no atributo name da TAG input.

```php
$nome       = filter_input(INPUT_POST, "nome");
$sobrenome  = filter_input(INPUT_POST, "sobrenome");
$data       = filter_input(INPUT_POST, "data");
$genero     = filter_input(INPUT_POST, "genero");
$es         = filter_input(INPUT_POST, "es");
```

Armazena o DML INSERT INTO dentro da variável sqlInsert.

```php
$sqlInsert = "INSERT INTO pessoa(nome,sobrenome,data_nascimento,
                genero,estado_civil)VALUES('$nome','$sobrenome',
                '$data','$genero','$es')";
```

A função mysqli_query retorna valor booleano, portanto estamos utilizando uma estrutura condicional composta, se retornar TRUE o cadastro foi efetuado, senão algum erro ocorreu e não foi possível efetuar o cadastro.

```php
if(mysqli_query($conn, $sqlInsert)){
    echo "Cadastro efetuado com sucesso!";
}else{
    echo "Erro ao efetuar o cadastro";
}
```

Encerramos a conexão com o servidor.

```php
mysqli_close($conn);
```

Vamos efetuar alguns cadastros, pois em seguida criaremos uma página que irá listar todas as pessoas da tabela. Abaixo segue a imagem dos cadastros que foram efetuados na tabela **pessoa** do banco de dados **php_integrado**.

	id	nome	sobrenome	data_nascimento	genero	estado_civil
Editar Copiar Remover	1	Anderson	Henrique	1978-09-28	M	Solteiro(a
Editar Copiar Remover	2	Fabiana	de Souza	1982-03-12	F	Solteiro(a
Editar Copiar Remover	3	Diogo	da Silva	1999-01-15	M	Casado(a)
Editar Copiar Remover	4	Viviane	MourÃ£o	1974-08-22	F	Casado(a)

Comando DQL (SELECT FROM) com o PHP

Iremos criar uma página chamada **select.php** onde utilizaremos o DQL SELECT FROM para listar todos os registros da tabela. Segue o código abaixo:

Tabela que armazenará os resultados

```php
<h3>Listar</h3>

<table style="width: 95%; text-align: center">
    <tr style="background-color: #333; color: #FFF">
        <td>Nome</td>
        <td>Sobrenome</td>
        <td>Data Nascimento</td>
        <td>Gênero</td>
        <td>Estado Civil</td>
    </tr>
    <?php
```

Requisição do arquivo de conexão

```
13      include_once '11 - integrando_mysql_e_php.php';
```

Armazena o DQL SELECT FROM na variável sqlSelect

```
16      $sqlSelect = "SELECT * FROM pessoa";
```

Executa a instrução SQL e armazena o resultado, caso ocorra algum erro, o código finaliza a execução nessa linha

```
19      $result = mysqli_query($conn, $sqlSelect) or die(mysqli_error());
```

Retorna a quantidade de registros (linhas) afetadas pela instrução SQL

```
22      echo "Total de Registros Afetados: " . mysqli_affected_rows($conn)
23                                           . "<br>";
```

Retorna o total de registros (linhas) encontradas

```
26      echo "Total de registros: " . mysqli_num_rows($result) . "<br>";
```

Retorna o próximo campo da tabela

```
29      print_r(mysqli_fetch_field($result));
```

```
stdClass Object
(
    [name] => id
    [orgname] => id
    [table] => pessoa
    [orgtable] => pessoa
    [def] =>
    [db] => php_integrado
    [catalog] => def
    [max_length] => 1
    [length] => 11
    [charsetnr] => 63
    [flags] => 49667
    [type] => 3
    [decimals] => 0
)
```

Retorna um array de objetos que representam os campos da tabela

```
32      print_r(mysqli_fetch_fields($result));
```

```
Array
(
    [0] => stdClass Object
        (
            [name] => id
            [orgname] => id
            [table] => pessoa
            [orgtable] => pessoa
            [def] =>
            [db] => php_integrado
            [catalog] => def
            [max_length] => 1
            [length] => 11
            [charsetnr] => 63
            [flags] => 49667
            [type] => 3
            [decimals] => 0
        )

    [1] => stdClass Object
        (
            [name] => nome
            [orgname] => nome
            [table] => pessoa
            [orgtable] => pessoa
            [def] =>
            [db] => php_integrado
            [catalog] => def
            [max_length] => 8
            [length] => 20
            [charsetnr] => 8
            [flags] => 4097
            [type] => 253
            [decimals] => 0
        )
```

Retorna registro (linha) em um array associativo

```
35          print_r(mysqli_fetch_assoc($result));
```

```
Array
(
    [id] => 1
    [nome] => Anderson
    [sobrenome] => Henrique
    [data_nascimento] => 1978-09-28
    [genero] => M
    [estado_civil] => Solteiro(a
)
```

Retorna todos os registros (linhas) em um array numérico

```
38          print_r(mysqli_fetch_all($result));
```

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => Anderson
            [2] => Henrique
            [3] => 1978-09-28
            [4] => M
            [5] => Solteiro(a
        )

    [1] => Array
        (
            [0] => 2
            [1] => Fabiana
            [2] => de Souza
            [3] => 1982-03-12
            [4] => F
            [5] => Solteiro(a
        )
```

Retorna registro (linha) em arrays (associativo, numérico, misto)

```
41          print_r(mysqli_fetch_array($result, MYSQLI_ASSOC));
```

Retorna registro (linha) em objeto dinâmico

```
44          print_r(mysqli_fetch_object($result));
```

```
stdClass Object
(
    [id] => 1
    [nome] => Anderson
    [sobrenome] => Henrique
    [data_nascimento] => 1978-09-28
    [genero] => M
    [estado_civil] => Solteiro(a
)
```

Optamos retornar os registros em um array associativo, para isto, utilizamos um laço de repetição para buscar todos os registros e armazenar em linhas da nossa tabela. Cada volta que o laço dá, cria uma nova linha na tabela e exibe os valores das colunas da tabela em uma TAG <**td**>.

Para exibir o resultado, utilizamos a sintaxe do php conhecida como modo resultado **<?= ?>**

```
21          while($linha = mysqli_fetch_assoc($result)){
22          ?>
23          <tr>
24              <td><?=$linha["nome"]?></td>
25              <td><?=$linha["sobrenome"]?></td>
26              <td><?=$linha["data_nascimento"]?></td>
27              <td><?=$linha["genero"]?></td>
28              <td><?=$linha["estado_civil"]?></td>
29          </tr>
30          <?php } ?>
31      </table>
```

Listar

Nome	Sobrenome	Data Nascimento	Gênero	Estado Civil
Anderson	Henrique	1978-09-28	M	Solteiro(a
Fabiana	de Souza	1982-03-12	F	Solteiro(a
Diogo	da Silva	1999-01-15	M	Casado(a)
Viviane	Mourão	1974-08-22	F	Casado(a)

Comando DML (DELETE) com o PHP

Vamos alterar o arquivo **select.php** adicionando uma coluna de título na nossa tabela e coluna no resultado contendo um link para excluir cada registro.

```
9           <td>Estado Civil</td>
10          <td>#</td>
11      </tr>
```

```
32          <td><?=$linha["estado_civil"]?></td>
33          <td>
34              <a href="delete.php?id=<?=$linha["id"]?>">excluir</a>
35          </td>
36      </tr>
```

Observe que o link chama o arquivo **delete.php** sendo que inserimos o sinal de (?), o que indica a passagem de um parâmetro na URL, esse parâmetro armazenará o valor do id (chave_primária) de cada registro (linha) na nossa tabela.

Gênero	Estado Civil	#
M	Solteiro(a	excluir
F	Solteiro(a	excluir
M	Casado(a)	excluir
F	Casado(a)	excluir

Se o ponteiro do mouse estiver sobre o link **excluir**, a barra de status do navegador apresentará na URL o nome do parâmetro seguido do valor que este recebe dinamicamente do array. Esta passagem de parâmetro se dá pela URL, portanto, o método que utilizamos para recuperar esse valor é o **GET**.

localhost/curso_php/delete.php?id=4

Abaixo temos o código do arquivo **delete.php**, utilizando a conexão com o servidor, buscando o valor do parâmetro id, executando a instrução sqlDelete e retornando o resultado:

Requisição do arquivo de conexão

```
4   include_once '11 - integrando_mysql_e_php.php';
```

Pega o valor do parâmetro 'id' da URL e armazena

```
7   $id = filter_input(INPUT_GET, "id");
```

Armazena o DML DELETE FROM na variável sqlDelete

```
10  $sqlDelete = "DELETE FROM pessoa WHERE id='$id' ";
```

A função **mysqli_query** retorna valor booleano, portanto estamos utilizando uma estrutura condicional composta, se retornar TRUE o registro foi excluído, senão algum erro ocorreu e não foi possível excluir o registro.

```
13  if(mysqli_query($conn, $sqlDelete)){
14      echo "Registro excluído com sucesso!";
15  }else{
16      echo "Erro ao excluir o registro";
17  }
```

Excluímos o registro do Diogo da Silva

Nome	Sobrenome	Data Nascimento	Gênero
Anderson	Henrique	1978-09-28	M
Fabiana	de Souza	1982-03-12	F
Viviane	Mourão	1974-08-22	F

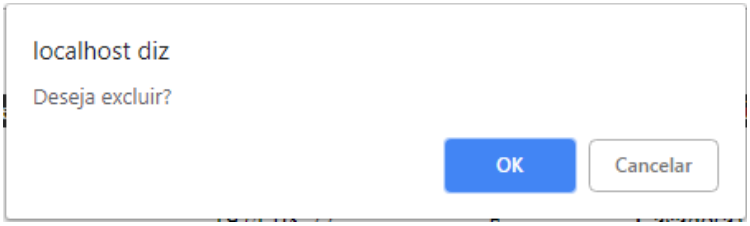
Capítulo 7

Integrando MySQL com PHP

Neste capítulo vamos aprender a utilizar o **DML UPDATE SET**, para isso teremos que criar um novo link na tabela que se encontra no arquivo **select.php**, conforme nos mostra a imagem abaixo:

```
30          <td>
31              <a href="delete.php?id=<?=$linha["id"]?>"
32                 onclick="return confirm('Deseja excluir?')">excluir</a>
33              <a href="form_alterar.php?id=<?=$linha["id"]?>">alterar</a>
34          </td>
```

Fizemos uma pequena alteração no link **excluir**, adicionamos o atributo **onclick** que chama um **Javascript** com uma janela de confirmação, assim evitamos excluir um registro de forma acidental.



Gênero	Estado Civil	#
M	Solteiro(a	excluir alterar
F	Solteiro(a	excluir alterar
F	Casado(a)	excluir alterar

Se o ponteiro do mouse estiver sobre o link **alterar**, a barra de status do navegador apresentará na URL o nome do parâmetro seguido do valor que este recebe dinamicamente do array. Esta passagem de parâmetro se dá pela URL, portanto, o método que utilizamos para recuperar esse valor é o **GET**.

localhost/curso_php/form_alterar.php?id=4

A alteração de valores em um registro de uma tabela se dá em dois passos, primeiro precisamos selecionar todos os campos do registro específico, estes são apontados pelo valor da chave-primária, e inserir esses valores em um formulário de edição.

Abaixo segue o código do arquivo **form_alterar.php**:

Requisição do arquivo de conexão

```
13      include_once '11 - integrando_mysql_e_php.php';
```

Armazena o DQL SELECT FROM na variável sqlSelect

```
 9   $sqlSelect = "SELECT * FROM pessoa WHERE id='$id' ";
```

Executa a instrução SQL e armazena o resultado, caso ocorra algum erro, o código finaliza a execução nessa linha

```
19      $result = mysqli_query($conn, $sqlSelect) or die(mysqli_error());
```

Retorna registro em um array associativo armazenando-o na variável

```
15   $linha = mysqli_fetch_assoc($result);
```

Perceba que o registro foi armazenado em um array que possui índice [chave] associativo, onde cada índice representa uma coluna da nossa tabela 'pessoa' no banco de dados.

```
Array
(
    [id] => 1
    [nome] => Anderson
    [sobrenome] => Henrique
    [data_nascimento] => 1978-09-28
    [genero] => M
    [estado_civil] => Solteiro(a
)
```

id	nome	sobrenome	data_nascimento	genero	estado_civil
1	Anderson	Henrique	1978-09-28	M	Solteiro(a
2	Fabiana	de Souza	1982-03-12	F	Solteiro(a
4	Viviane	MourÃ£o	1974-08-22	F	Casado(a)

Precisamos inserir os valores de cada índice no atributo **value** de cada campo respectivo, como nos mostra a imagem abaixo:

```
19      Nome<br>
20      <input type="text" name="nome" required placeholder="Seu nome:"
21          value="<?=$linha["nome"]?>"><br><br>
22      Sobrenome<br>
23      <input type="text" name="sobrenome" required placeholder="Seu sobrenome:"
24          value="<?=$linha["sobrenome"]?>"><br><br>
25      Data Nascimento<br>
26      <input type="date" name="data" required
27          value="<?=$linha["data_nascimento"]?>"><br><br>
```

Para o campo gênero, temos um **select** com duas opções, para resolver este problema precisamos verifica se o valor retornado foi M ou F, se M inserir um **option** com o valor Masculino já selecionado, senão inserir um **option** com o valor Feminino já selecionado.

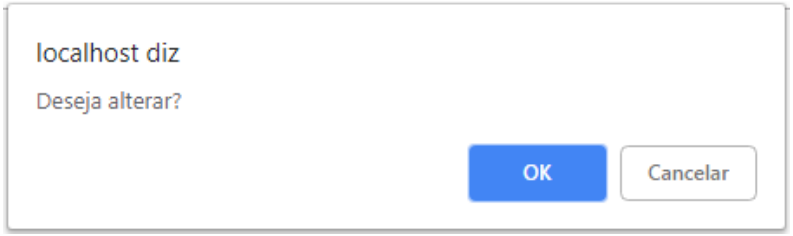
```
29      <select name="genero">
30          <?php
31              if($linha["genero"] == "M"){
32                  echo "<option selected value='M'>Masculino</option>";
33                  echo "<option value='F'>Feminino</option>";
34              }else{
35                  echo "<option value='M'>Masculino</option>";
36                  echo "<option selected value='F'>Feminino</option>";
37              }
38          ?>
39      </select><br><br>
```

Para o campo **estado_civil**, precisamos verificar qual o valor retornado, dependendo do valor o botão rádio já deverá ser marcado automaticamente.

```
41      <input type="radio" name="es" value="Solteiro(a)" required
42          <?=($linha["estado_civil"] == "Solteiro(a)")? "checked" : ""?>>Solteiro(a)
```

Quando clicarmos no botão **Editar**, as informações serão enviadas para o arquivo que se encontra no atributo **action** do formulário, é o **update.php**. Sendo que o valor da chave-primária do registro que será editado deve ser passado em um campo oculto do nosso formulário, como nos mostra a imagem abaixo, e uma janela de confirmação deverá aparecer:

```
50      <input type="hidden" name="id" value="<?=$linha["id"]?>">
51
52      <input type="submit" value="Editar"
53              onclick="return confirm('Deseja alterar?')">
```



Segue abaixo código do arquivo **update.php**, muito semelhante ao código utilizado para inserir registros na tabela do banco de dados:

Requisição do arquivo de conexão

```
13      include_once '11 - integrando_mysql_e_php.php';
```

Utilizando a função filter_input() para pegar os valores digitados e enviados através do método POST, o PHP recebe como atributos deste método (METODO, NOME_CAMPO), nome do campo está no atributo name da TAG input.

```php
$id        = filter_input(INPUT_POST, "id");
$nome      = filter_input(INPUT_POST, "nome");
$sobrenome = filter_input(INPUT_POST, "sobrenome");
$data      = filter_input(INPUT_POST, "data");
$genero    = filter_input(INPUT_POST, "genero");
$es        = filter_input(INPUT_POST, "es");
```

Armazena o DML UPDATE SET na variável sqlUpdate

```php
$sqlUpdate = "UPDATE pessoa SET nome='$nome', sobrenome='$sobrenome',
                data_nascimento='$data',genero='$genero',
                estado_civil='$es' WHERE id='$id' ";
```

A função **mysqli_query** retorna valor booleano, portanto estamos utilizando uma estrutura condicional composta, se retornar TRUE o registro foi editado, senão algum erro ocorreu e não foi possível editar o registro.

```php
if(mysqli_query($conn, $sqlUpdate)){
   echo "Cadastro editado com sucesso!";
}else{
    echo "Erro ao editar o cadastro";
}
```

Encerra a conexão com o servidor

```php
mysqli_close($conn);
```