MARCOS DE MELO

PHP – Criando Páginas Dinâmicas

2º Edição

Sumário

Aula 1 – Introdução a Linguagem PHP

Inicialmente, bem-vindo ao curso "PHP – Desenvolvendo páginas dinâmicas", um curso voltado para profissionais da área de tecnologia e desenvolvimento de Sites com conteúdo dinâmico e principalmente com interação a banco de dados. A forma didática do conteúdo deste livro, permite que usuários principiantes possam aprender os conceitos da linguagem PHP e como interagir com Banco de dados em servidores web, capacitando o profissional de programação a criar sites dinâmicos, proporcionando interatividade entre o usuário e o site.

É extremamente importante que você leia e compreenda as aulas deste livro na sequência, para o objetivo proposto.

O que é o PHP?

PHP é uma linguagem de programação utilizada na criação de páginas dinâmicas na WEB, significa ***Hypertex Reprocesso***, ou também conhecida como ***Personal Home Page***, sendo assim, essa linguagem nos possibilita uma maior interação com o usuário através de formulários, parâmetros da URL e links. O código PHP assim como outras linguagens de páginas dinâmicas, é executado no servidor, sendo enviado para o cliente apenas html do que foi processado no servidor, o PHP trabalha como **Server-Side Scripts**, ou seja, são scripts responsáveis pelas ações executadas no servidor.

Sendo assim é possível interagir com bancos de dados variados, como, por exemplo, MySQL, que abordaremos neste livro como o banco de dados utilizado na conexão com o PHP.

A linguagem PHP executa no servidor, devolvendo para o lado cliente somente o código HTML. O cliente receberia os resultados da execução desse script, mas não poderá ver como é o código fonte.

O php é uma linguagem extremamente simples, mas ao mesmo tempo oferecendo muitos recursos avançados.

O PHP é grátis

Sim, o PHP é grátis, ou seja, você não precisa pagar para utilizar a linguagem. O PHP foi desenvolvido em código aberto.

Mesmo assim, o desenvolvedor tem que entender que, a linguagem PHP depende de um servidor de páginas dinâmicas, hospedado em um servidor Web e este pode ser pago.

De qualquer maneira, o valor de um servidor de aplicações desenvolvidas em PHP é bem mais barato que um servidor de ASP.NET por exemplo.

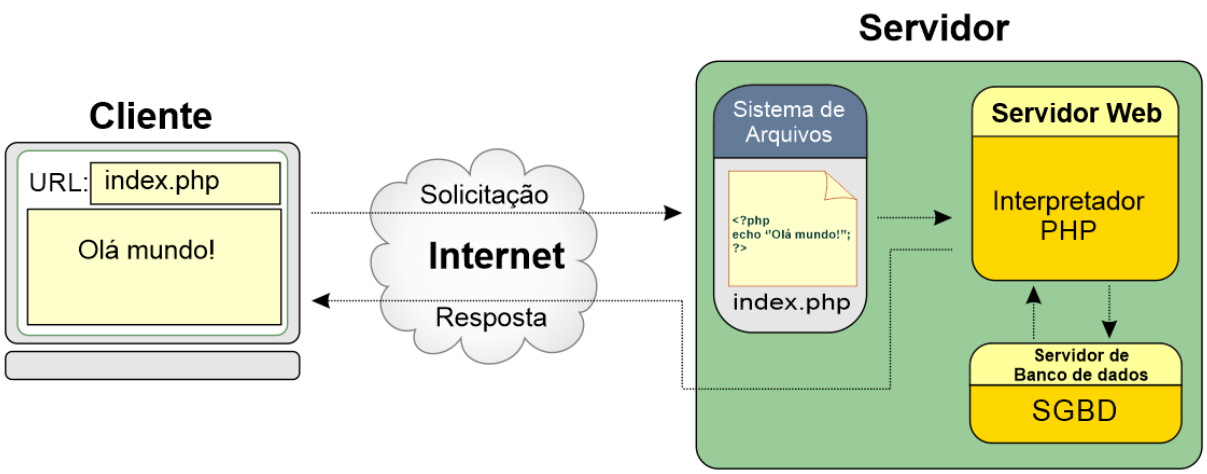
O que podemos fazer com PHP?

Basicamente, qualquer coisa como, coletar dados de um formulário e gravá-los em um banco de dados, gerar páginas dinamicamente ou enviar e receber *cookies*.

PHP também tem como uma das características mais importantes o suporte a um grande número de banco de dados, como dBase, Interbase, mSQL, MySQL, Oracle, Sybase, PostgreSQL e vários outros.

Além disso, a linguagem PHP tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, http. Ainda é possível abrir *sockets* e interagir com outros protocolos.

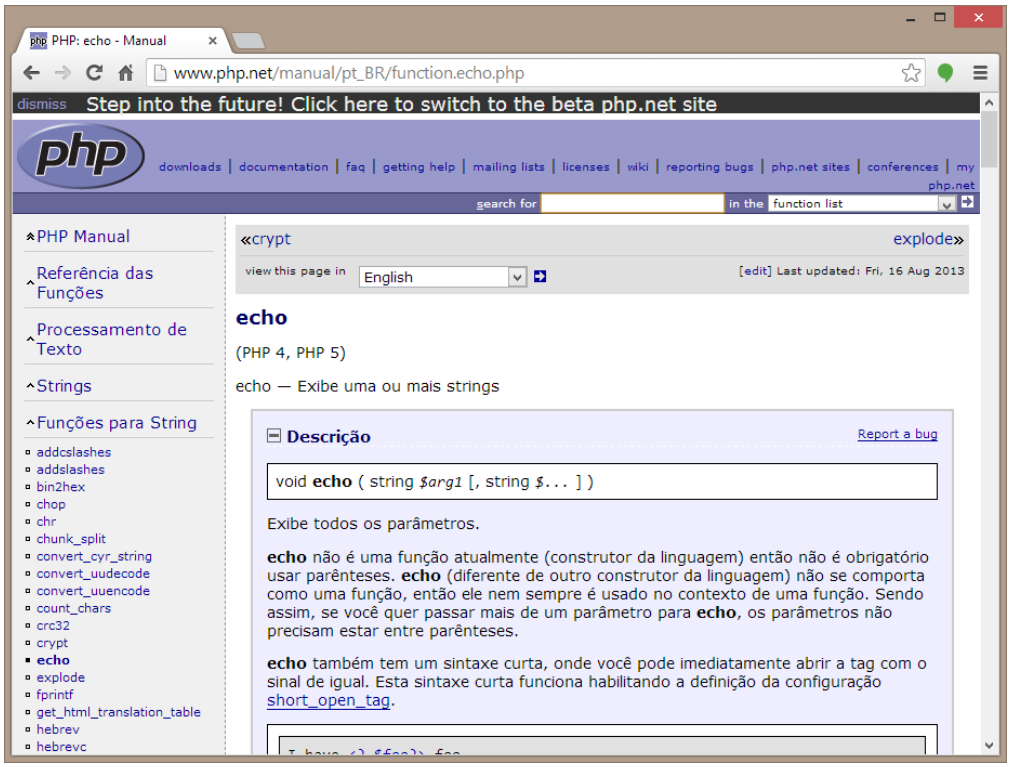
Observer na imagem abaixo como seria a comunicação com o php, quando um computador cliente solicita ao servidor web a exibição de uma página php.



Sites de referências sobre PHP

PHP.net

O site php.net é o site oficial do php e nele é possível consultar e obter suporte sobre a linguagem, os exemplos sobre cada comando da linguagem são muito bem explicativos e didáticos. Vários outros assuntos como, termos de uso e novidades, também podem ser explorados no site.



Link: www.php.net

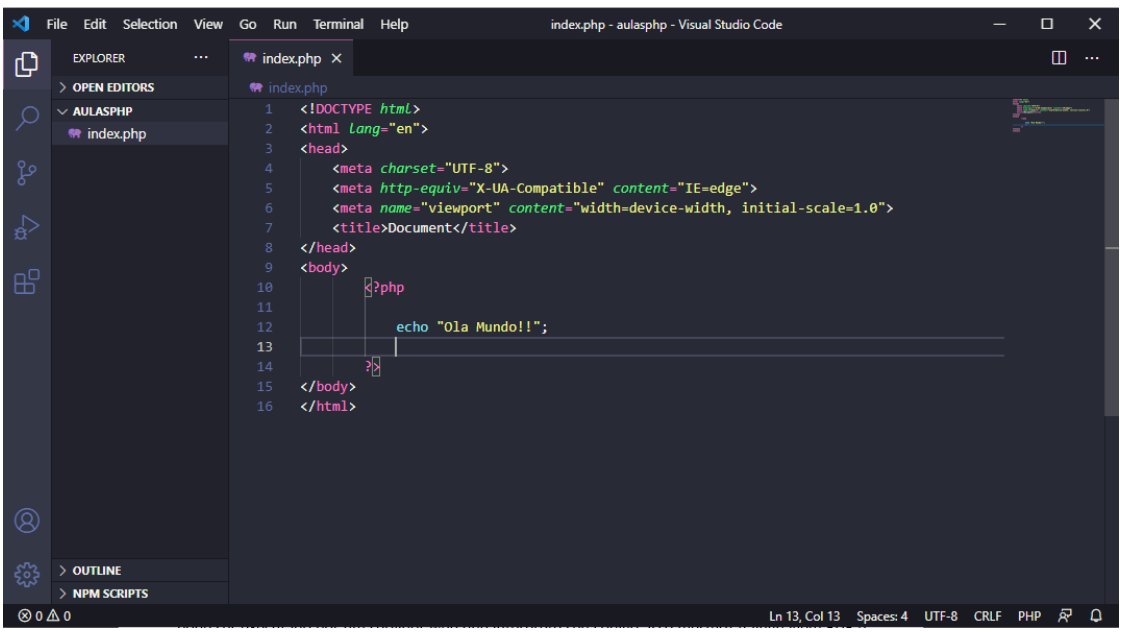
Ferramentas utilizadas

Para criar páginas dinâmicas PHP, basicamente precisamos de um editor de texto simples como, o famoso **Bloco de Notas**, que já vem instalado no Windows 8.

Existem várias outras ferramentas de uso profissional, utilizadas na criação de páginas dinâmicas em PHP. O Adobe Dreamweaver por exemplo, é a mais completa IDE de desenvolvimento de Sites

dinâmicos, mas é um software proprietário, ou seja, você deve pagar para usá-lo. Mas há outras opções profissionais de programas de desenvolvimento Web otimizados para a interpretação da linguagem PHP totalmente gratuitas como o Netbeans, Aptana, Eclipse e Notepad++, Sublime Text, Visual Studio Code.

Você pode usar entre os programas acima indicados, o que melhor se adequar as suas preferencias ao desenvolver os exercícios deste livro, mas recomendamos o uso do último programa indicado, o Visual Studio Code, pois este é o programa usado para criar os códigos dos exercícios deste livro e por ser um programa eficiente e simples de se usar.



Site de download do Visual Studio Code: https://code.visualstudio.com/Download

Instalando o servidor web e de banco de dados

Como já foi dito, o php como muitas outras linguagens de programação de páginas dinâmicas, só pode ser executado por um servidor web que interprete seu código. Isso mesmo! A linguagem PHP é interpretada e não compilada, com é o caso da linguagem JAVA. Por isso precisamos antes de começar a criar os códigos PHP e testá-los, precisamos instalar um servidor web que interprete o PHP. Entre os servidores capazes de interpretar a linguagem de script PHP, o mais conhecido é o Apache. O Apache é o servidor Web mais utilizado no mundo de uso livre.

Vamos manipular banco de dados através do PHP também, o banco de dados utilizado neste livro é o MySQL.

Como você deve ter percebido que precisamos de dois servidores distintos, um servidor web e o outro de banco de dados. Se você ainda não instalou estes programas que são necessários para continuar as aulas deste livro, vamos indicar um pacote de instalações que instala e configura estes programas. Utilizaremos o pacote de programas chamado XAMPP de uso livre.

O que é o XAMPP?

O XAMPP é um pacote de distribuição de programas de desenvolvimento Web. Ao baixar e instalar o XAMPP, ele instala e configura automaticamente o apache, servidor de páginas web dinâmicas como, o PHP e o mais importante para nós, o servidor de banco de dados MySQL, MariaDB.

Instalando o pacote de programas XAMPP

O download do pacote de programas XAMPP pode ser feito através da página oficial do XAMPP https://www.apachefriends.org/pt_br/index.html .

Acessando o link deste site, é possível baixar o XAMPP disponível para o sistema operacional que desejar. Como este curso foi desenvolvido na plataforma Windows vamos demonstrar como instalar o servidor no Windows.

Clique no sistema operacional Windows.

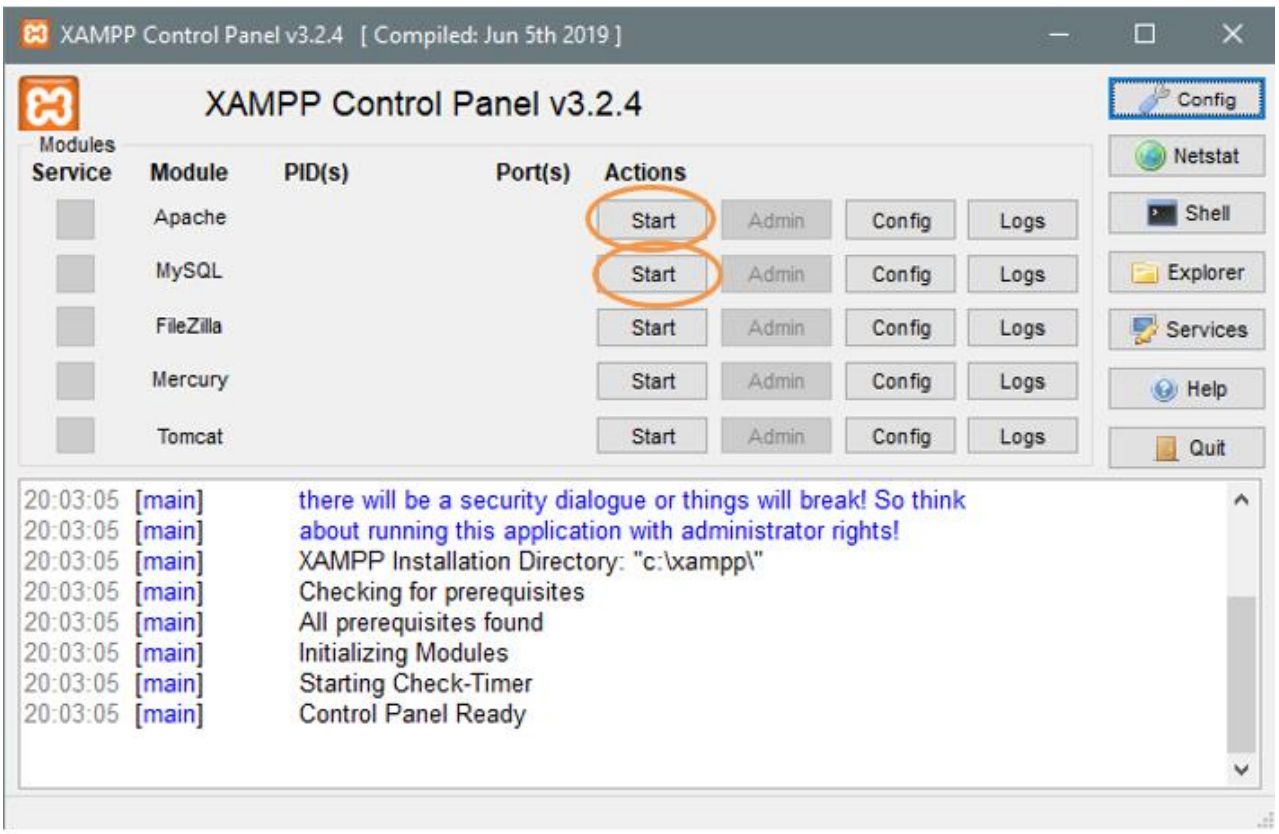
Aguarde o download do arquivo estar completo e em seguida execute o arquivo para instalá-lo.

A instalação é feita automaticamente sem a necessidade de maiores configurações.

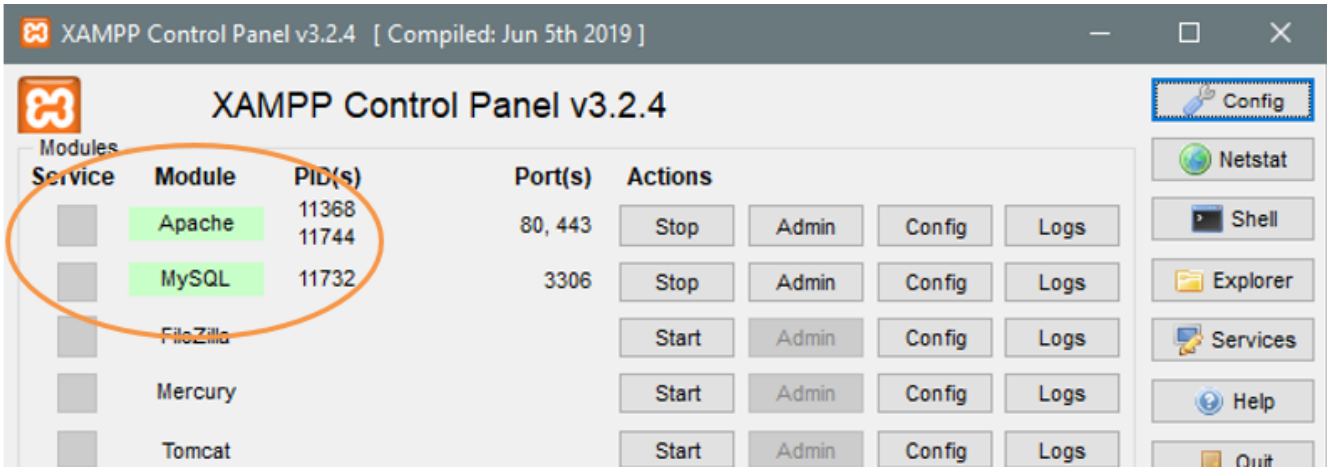
Executando XAMPP Control Panel

Após a instalação do XAMPP, o mesmo pode ser ativado pelo ícone “XAMPP Control Panel”, que executa o painel de controle do XAMPP para inicializar o servidor apache e MySQL.

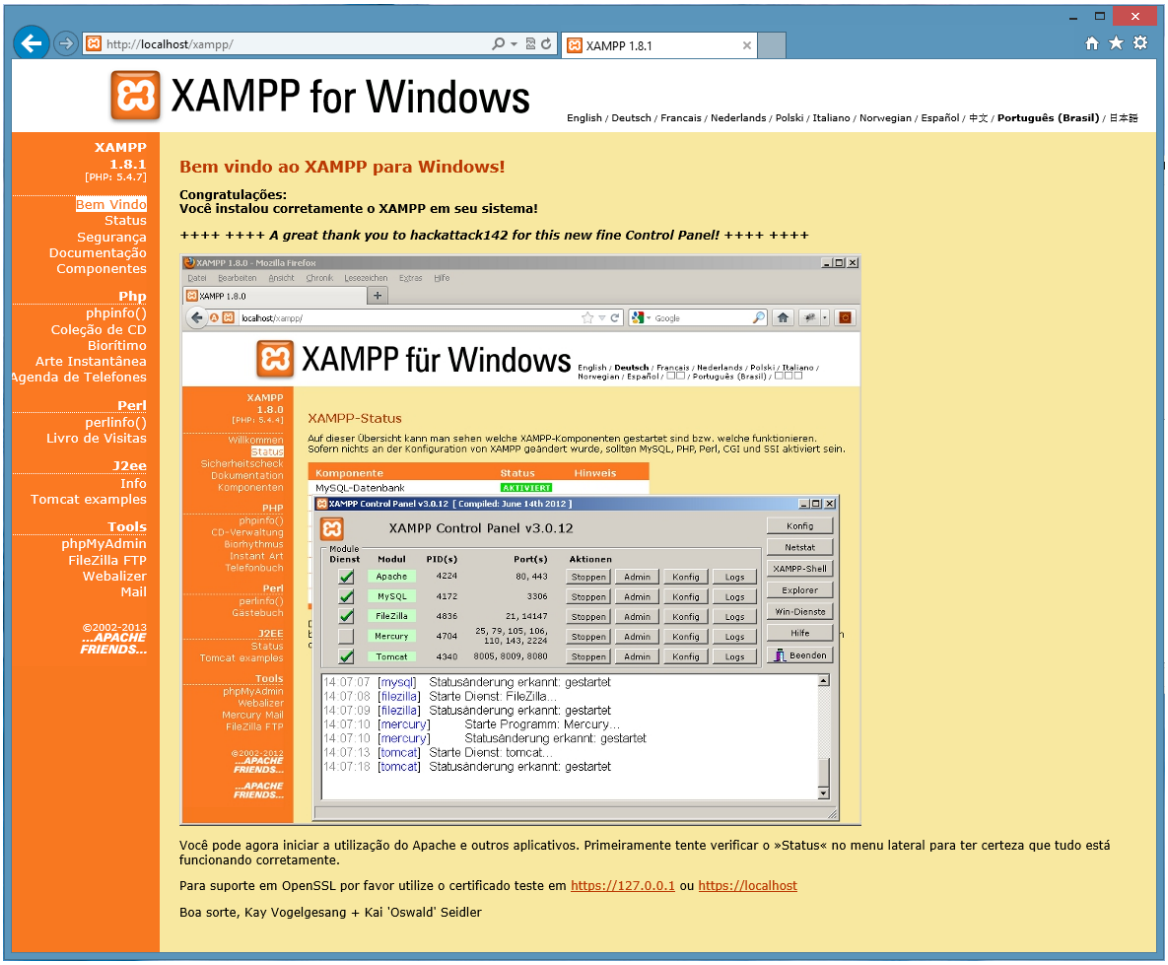
Pressione os botões de “Start” do servidor “Apache” responsável pela a interpretação das páginas PHP e o do servidor de banco de dados “MySQL” para a manipulação e interação do PHP com o banco de dados MySQL.



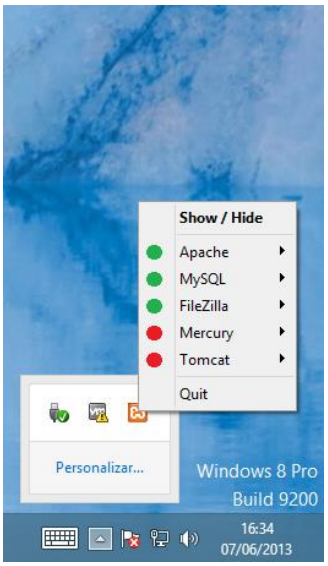
Após pressionar os botões starts devera mostrar na coluna Modulo na cor verde os servidores em funcionamento.



Para verificar se o servidor Apache está funcionando corretamente, basta digitar o link http://localhost no navegador (Browser). O site abaixo deverá aparecer.



Ao executar o programa, um ícone de controle do XAMPP fica disponível na barra de tarefas "Deskbar", onde podemos verificar os serviços inicializados e controlar os mesmos.

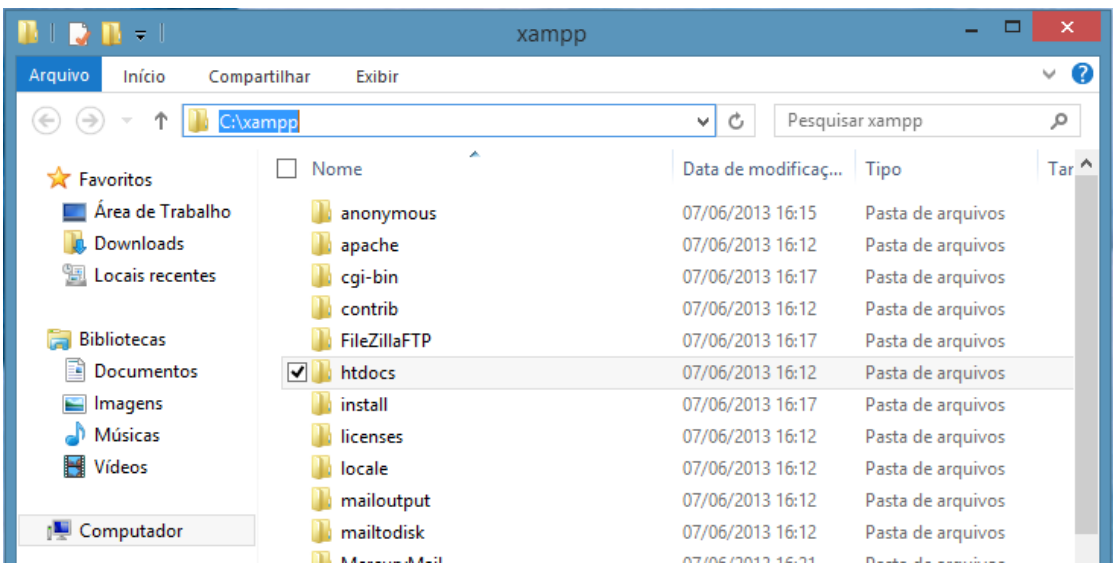


Onde criar os arquivos de suas páginas dinâmicas em PHP?

Após instalar o pacote de programas XAMPP, o mesmo por padrão, é instalado dentro de uma pasta chamada "XAMPP" na unidade C:\.

C:\xampp

Todas as páginas que você criar devem estar dentro de uma pasta especial chamada "**htdocs**" para funcionarem no navegador. A pasta htdocs fica dentro da pasta C:\xampp.

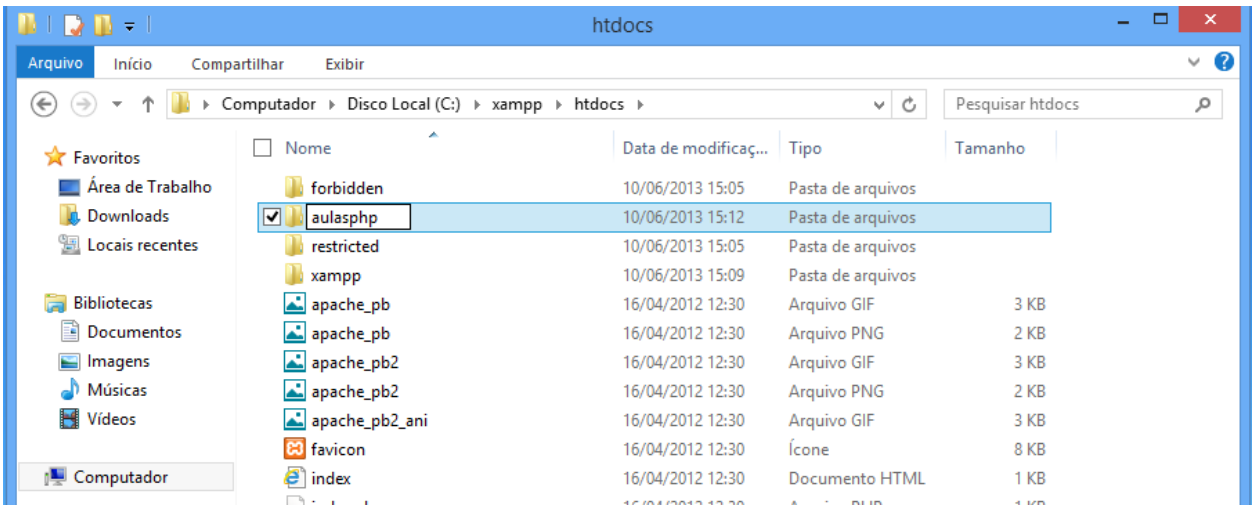


Dentro da pasta htdocs, você deve criar uma “pasta” para cada aplicação desenvolvida em php.

A pasta de cada aplicação PHP é denominada “pasta raiz do site”.

Os nomes das pastas raízes e os arquivos de páginas não devem conter espaços em nomes compostos, para que funcione corretamente ao digitar o endereço no navegador. Procure também, digitar o nome das pastas e arquivos somente em minúsculo.

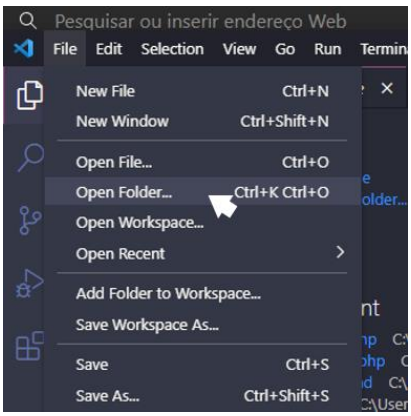
Para começar a criar nossas primeiras páginas dinâmicas em PHP, vamos criar uma pasta raiz padrão chamada “**aulasphp**” dentro da pasta htdocs, esta pasta será a pasta raiz do nosso site onde vamos criar arquivos PHP de exemplos da aula.



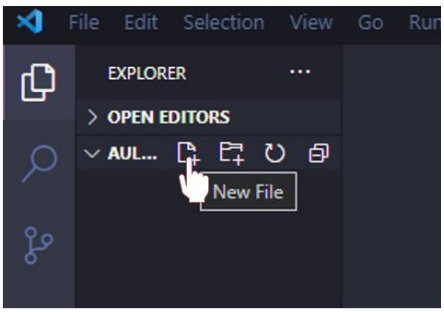
Primeira Página em PHP

Vamos criar nossa primeira aplicação (página php) utilizando o editor de texto Visual Studio Code para o desenvolvimento.

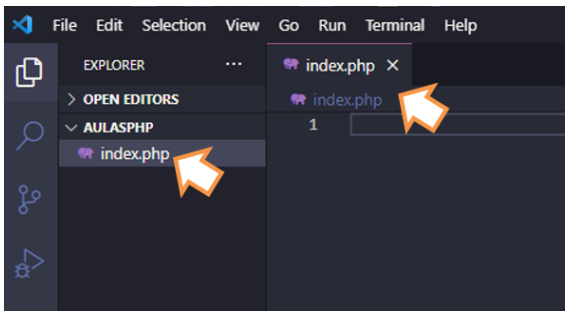
Ao abrir o Visual Studio Code, clique em **File ➔ Open Folder**, para poder selecionar a pasta aulasphp, assim o Visual Studio Code poder gerenciar todos os arquivos criados dentro desta pasta.



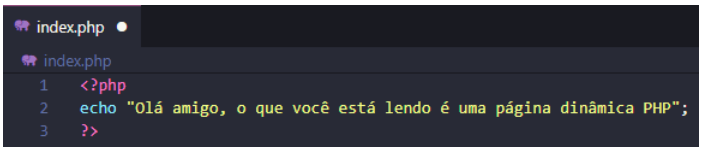
Dentro da guia Explorer na pasta clique no ícone New File, para gerar num novo arquivo exatamente dentro da pasta aulasphp.



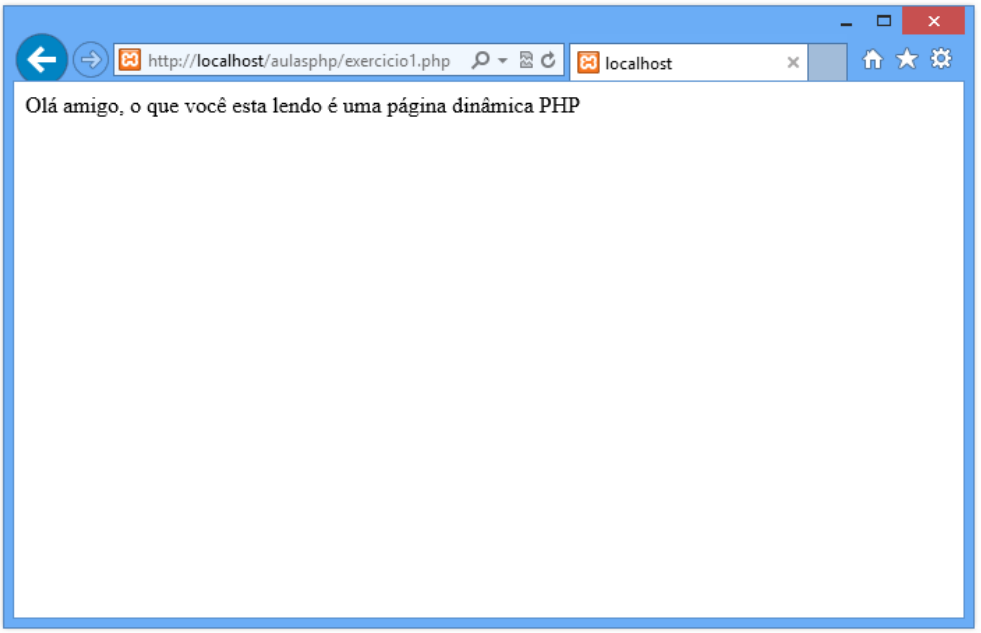
Nomeio o arquivo com o nome **index.php**



Digite o código php a seguir no arquivo index.php



Abra o navegador de internet e digite na barra de endereço o link http://localhost/aulasphp/index.php para visualizar a página PHP deste código.



Entendendo o código:

Após ter digitado o código do programa, ter salvo na pasta e executado no navegador, vamos entender as 3 linhas de código do programa **index.php**.

Para que o servidor interpreto o código php, o mesmo deve estar dentro de uma tag especial exclusiva do php "**<?php ..... ?>**". Portanto, na linha 1 do código, é iniciado o código com "**<?php** " e finalizado na linha 3 com " **?>**".

Na linha 2 é executado um dos códigos mais utilizados do php, o comando ECHO que tem a função de escrever na tela do navegador, ou seja, devolve em formato HTML o código php processado no servidor.

Orientação para os próximos exercícios

Como você pode ter reparado no exercício 1 (Um) acima, o mesmo encontra-se digitado em cores variadas que distingue os códigos PHP claramente, para melhorar o entendimento do código descrito. Os demais exercícios também se apresentam da mesma forma.

Todos os arquivos de exemplo desse livro estão devidamente nomeados e em ordem.

As linhas do código de cada exemplo php desse livro estão todas numeradas, para orientá-lo na localização de determinada linha de código que será explicada. Lembre-se, você não deve digitar estes números no início de cada linha.

Aplicando comentários no código

Os "comentários", como o próprio nome já diz, são basicamente anotações deixadas pelo programador, para se orientar no código do programa que está construindo. Os comentários não tem nenhuma influência no código. A seguir é mostrado as opções de comentários;

Comentário de uma linha

// ➔ Comentário de uma linha, todo texto contido nesta linha será ignorado pelo php.

➔ Comentário de uma linha somente no estilo Shell, todo texto contido nesta linha será ignorado pelo php.

Exemplo 2:

Nome do arquivo: aula01ex02.php

aula01ex02.php

```
1  <?php
2      // Isto é um comentário de uma linha somente no estilo C++.
3      # Isto é um comentário de uma linha somente no estilo Shell.
4
5      echo "O que você está lendo é uma página dinâmica PHP.";
6  ?>
```

Resultado:

O que você está lendo é uma página dinâmica PHP.

Explicando código:

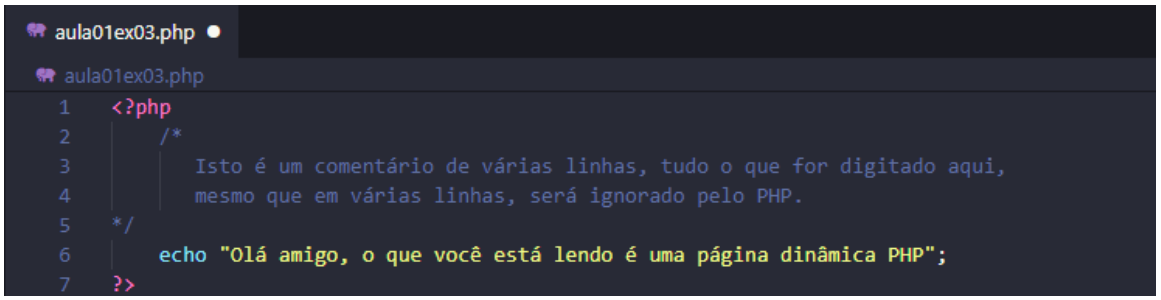
O código do exercício 2 ao ser visualizado no navegador, exibe somente a *String* descrita no comando *echo* na linha 5. O conteúdo das linhas 2 e 3 são interpretados como comentários no código e consequentemente, são ignorados pelo PHP.

Comentário de várias linhas

/* ..... */ ➔ Comentário de várias linhas, podemos digitar várias linhas de comentário desde que estejam entre **/*** e ***/**.

Exemplo 3:

Nome do arquivo: aula01ex03.php



Resultado:

Olá amigo, o que você está lendo é uma página dinâmica PHP

Explicando o código:

O código do exercício 3 ao ser visualizado no navegador, exibe somente a *String* descrita no comando *echo* na linha 6. O conteúdo das linhas 2 até 5 são interpretados como um comentário digitado em mais de uma linha no código e consequentemente, são ignorados pelo PHP.

Misturando o código PHP com HTML

Como já vimos no código php, tudo o que estiver após o comando *"echo"* será exibido no corpo do documento no navegador em forma de código HTML. Mas isso não quer dizer que temos que criar todo o código HTML dentro do comando echo. Podemos misturar o código PHP dentro de uma página HTML.

Exemplo 4:

Nome do arquivo: aula01ex04.php

aula01ex04.php

```php
<?php
    $texto = "<p>Este é o quarto parágrafo</p>"; // variável php.
?>
<html>
    <head>
        <title> Exercício 4 de PHP </title>
    </head>
    <body>
        <h1>Exercício 4 de PHP – Misturando códigos</h1>
        <p>Este é o primeiro parágrafo. </p>
        <p><?php echo "Este é o segundo parágrafo."; ?></p>
        <p>Este é o terceiro parágrafo</p>
        <?php echo $texto; ?>
    </body>
</html>
```

Resultado:

Explicando o código:

No código do exercício 4 na linha 1 até a 3 é definido a primeira parte do código php, onde na linha 2 é definida uma variável e atribuído um valor na mesma.

Em seguida na linha 4 é iniciado o código html padrão da página.

Nas linha 11 e 13, uma segunda parte do código php é definido, repare que cada parte do código php inicia na tag **<?php** e finaliza com **?>** , assim o navegador saberá distinguir o html padrão do código PHP e ao enviar as requisições do código para o servidor, recebera do mesmo a resposta em html na mesma posição em que se encontra o código.

Trabalhando com Constantes

Diferente de uma variável, uma constante ao ser definida no início do programa, após ser atribuído o seu valor, ele não se altera durante toda a execução do programa.

As constantes são "Case-Sensitive", ou seja, detectam a diferença entre letras maiúsculas e minúsculas. Por definição, os nomes de constantes são sempre em maiúsculas.

Exemplo 5:

Nome do arquivo: aula01ex05.php

aula01ex05.php

```php
<?php
// Nomes de constantes válidas
define("BOLA1","FUTEBOL");
define("BOLA2","BASQUETE");
define("BOLA_3","PING-PONG");
define("bola4","Ténis"); // Tudo em minúsculo, funciona, mas recomenda-se não usar.

echo BOLA1 . "<br>";
echo BOLA2 . "<br>";
echo BOLA_3;

// Nomes de constantes inválidas
define("5BOLA", "Bola murcha"); //Constante invalida, pois começa com um número.
?>
```

Resultado:

```
FUTEBOL
BASQUETE
PING-PONG
```

Trabalhando com Variáveis

Em linguagem de programação, as variáveis são objetos que criamos dentro do programa para armazenar valores que serão utilizados no programa, valores estes, que podem variar no decorrer do programa. Em uma variável, podemos guardar um valor por vez, ou seja, se quisermos guardar um

novo valor, em uma variável, que já possui um valor, este valor anterior será excluído, dando lugar ao novo valor. Por isso, é dado origem a expressão Variável, porque os valores variam dentro dela.

Estrutura das variáveis

Como em todas as outras linguagens de programação, o PHP tem algumas regras para criar e usar variáveis, a começar pelo nome. O nome de uma variável em PHP, deve iniciar com o caractere especial "**$**" (Cifrão), e o próximo caractere após o cifrão, deve ser uma letra, após a letra é possível colocar números na composição do nome da variável.

Exemplo 6:

Nome do arquivo: aula01ex06.php

aula01ex06.php

```
1  <?php
2      $valor1;  // Esta é uma variável válida, pois começa com letras após o caractere $.
3      $100valor;  // esta é uma variável inválida, pois começa com números após o caractere $.
4      echo $valor1;
5      echo $100valor;
6  ?>
```

Explicando o código:

O código do exercício 6 não executara corretamente, pois apresentara um erro na linha 3, consequência da variável invalida $100valor.

Case sensitive

As variáveis do PHP são "Case-sensitive", ou seja, detectam a diferença entre letras maiúsculas e minúsculas. Portanto uma variável chamada $valor1 é diferente de uma outra chamada $VALOR1.

Exemplo 7:

Nome do arquivo: aula01ex07.php

aula01ex07.php

```php
<?php
    $valor1 = 100;// esta variável é diferente de todos as outras por estar em caixa baixa.

    $Valor1 = 200;// esta variável é diferente de todos as outras por Iniciar a primeira letra em maiúsculo e o reste em caixa baixa.

    $VALOR1 = 300;//esta variável é diferente de todos as outras por estar em caixa alta.

    echo "$valor1 - $Valor1 - $VALOR1";
?>
```

Resultado:

100 – 200 - 300

Nomes compostos

Geralmente devemos nomear as variáveis com palavras que representam o conteúdo (“valor”) que ela irá guardar. Ex: Se o conteúdo (“valor”) a ser guardado em uma variável for a idade de uma pessoa, o nome da variável deveria ser $idade. Mas, pode ser necessário especificar o nome de uma variável por mais de uma palavra, o que caracteriza um nome composto, porém uma variável com nome composto, não pode ter espaços. Uma boa técnica para se nomear uma variável com nome composto, é digitar a primeira palavra em caixa baixa (minúscula) e a segunda palavra a seguir, sem espaços com a Primeira Letra em caixa alta (maiúscula). Observe o exemplo a seguir;

Exemplo 8:

Nome do arquivo: aula01ex08.php

aula01ex08.php

```php
<?php
$valorUm            = 1500;
$valorDois      = 1100;

$idadeFuncionario    = 36;
$idadeCliente        = 26;

echo "A idade do cliente é $idadeCliente e a do funcionário é $idadeFuncionario.<br>";
echo "O valor que o funcionário vende o produto é $valorUm.<br>";
echo "Mas o valor que o cliente quer pagar é $valorDois.";
?>
```

Resultado:

A idade do cliente é 26 e a do funcionário é 36.
O valor que o funcionário vende o produto é 1500.
Mas o valor que o cliente quer pagar é 1100.

Tipos de dados no PHP

Agora que você já sabe como criar variáveis e constantes no PHP, e sabe exibir os dados no browser iremos aprofundar nos diferentes tipos de dados suportados pelas variáveis do PHP.

O PHP suporta oito tipos de dados primitivos divididos em três grupos:

Quatro tipos básicos, os dados escalares

- integer
- float (número de ponto flutuante, ou também double)
- string
- boolean

Dois tipos compostos

- array
- object

Dois tipos especiais:

- resource
- NULL

O tipo integer, inteiro no PHP

Um inteiro é qualquer número sem casas decimais, positivo ou negativo. Englobando todos os números do conjunto Z (os números inteiros).

Exemplo 9:

Nome do arquivo: aula01ex09.php

aula01ex09.php

```php
<?php

// Declaração de uma variável como inteiro
$idadeAluno = 18;

?>
<html>
    <head>
    <title>Tipo Inteiro no PHP</title>
    </head>

    <body>
        <p>O aluno tem <?php echo $idadeAluno; ?> de idade.</p>
    </body>
</html>
```

O tipo float ou double, número de ponto flutuante, no PHP

Float, Double ou ainda números de ponto flutuante são os números reais.

Exemplo 10:

Nome do arquivo: aula01ex10.php

aula01ex10.php

```php
<?php
    // Declaração de uma variável como float, armazenando o valor de pi
    $pi = 3.14159265;
?>
<html>
    <head>
        <title>Tipo float, double ou número de ponto flutuante no PHP</title>
    </head>

    <body>
        <p>O valor de pi é <?php echo $pi; ?></p>
    </body>
</html>
```

Resultado:

O valor de pi é: 3.14159265

O tipo String no PHP

Uma String é uma série de caracteres, um texto por exemplo. Para declararmos Strings podemos utilizar as **aspas simples** (apóstrofos) e as **aspas duplas**.

String com aspas simples (apóstrofos) no PHP

Com aspas simples, você tem uma String realmente como ela foi declarada, independentemente de qualquer caractere especial (com exceção do caractere de escape – \n\t\r, etc).

Para definirmos uma String com aspas simples basta delimitarmos o texto com aspas simples (').

Exemplo 11:

Nome do arquivo: aula01ex11.php

aula01ex11.php

```php
<?php
// Declaração de uma variável com String com aspas simples
$texto = 'O PHP é uma linguagem server-side.';

?>
<html>
    <head>
        <title>String com aspas simples no PHP</title>
    </head>

    <body>
        <p><?php echo $texto; ?></p>
    </body>
</html>
```

Resultado:

O PHP é uma linguagem Server-side.

Mas como o início e fim do texto são indicados pela aspa simples, palavras que contenham aspas simples não serão aceitas, gerando erro no código do php. Veja o exemplo a seguir, esta situação;

Exemplo 12:

Nome do arquivo: aula01ex12.php

aula01ex12.php

```php
<?php
// Declaração de uma variável com String com aspas simples
$texto = 'A caixa d'agua está transbordando.';

?>

<html>
    <head>
        <title>String com aspas simples no PHP</title>
    </head>

    <body>
        <p><?php echo $texto; ?></p>
    </body>
</html>
```

Resultado:

Erro na linha 4

Explicando o código:

O exemplo acima gera um erro de sintaxe na palavra d'agua, isto porque a aspa simples da palavra, é entendia pelo código, como sendo o fim do texto da String, e na verdade não é. Para especificar uma aspa simples (apóstrofo) no meio de uma String, você precisará "escapá-la" com uma contra barra (\), veja a seguir.

Exemplo 13:

Nome do arquivo: aula01ex13.php

```php
<?php

// Declaração de uma variável com String com aspas simples
$texto = 'A caixa d\'agua está transbordando.';

?>
<html>
<head>
<title>String com aspas simples no PHP</title>
</head>

<body>
<p><?php echo $texto; ?></p>
```

```
</body>
</html>
```

Resultado:

A caixa d'agua está transbordando.

String com aspas duplas no PHP

Aspas duplas são iguais as aspas simples, porém as aspas duplas interpretam variáveis, matriz (quando estiver entre {}) e comandos de texto (\n (quebra de linha), \t (tabulação), \r (retorno de carro) comprometendo a velocidade do programa já que o interpretador tem que percorrer toda a String em busca de variáveis, matriz e comandos de texto.

Exemplo 14:

Nome do arquivo: aula01ex14.php

```
<?php
// Declaração de um produto
$produto = 'pizza';
// Declaração de uma variável com String com aspas duplas
$texto = "Ele \"comprou\" uma $produto no Joey's.";

?>
<html>
<head>
<title>String com aspas duplas no PHP</title>
</head>
<body>
<p><?php echo $texto; ?></p>
</body>
</html>
```

Resultado:

Ele "comprou" uma pizza no Joey's.

Explicando o código:

Observe que a variável `$produto` foi interpretada dentro da String declarada na variável `$texto`. O que não ocorre nas variáveis declaradas com aspas simples. Por fim, a palavra comprou que precisava de um destaque com aspas duplas, precisou de um caractere de escape para não conflitar com as aspas duplas do início e fim da String.

Veja outro exemplo a seguir:

Exemplo 15:

Nome do arquivo: aula01ex15.php

```
<?php
// Declaração de um produto
$produto = 'pizza';
// Declaração de uma variável com String com aspas duplas
$texto = 'Ele comprou uma ' . $produto . 'no Joey\'s.';
?>
<html>
<head>
<title>String com aspas duplas no PHP</title>
</head>

<body>
<p><?php echo $texto; ?></p>
</body>
</html>
```

Resultado:

Ele comprou uma pizza no Joey's.

Explicando o código:

Assim como tivemos que inserir uma contra barra (\) para exibir aspas simples no texto quando declaramos uma String entre aspas simples, devemos utilizar o mesmo caractere de escape, contra barra (\), para podemos inserir as aspas duplas dentro de uma String declarada com aspas duplas.

Podemos observar também na linha 5 que, para colocar uma variável dentro de uma String declarada com aspas simples, a mesma será interpretada como texto normal, ou seja, não exibira o seu valor. Para resolver este problema, devemos encerrar as partes da String com (') e adicionar o sinal de ponto (.) para concatenar (juntar) a String com a variável

Outra grande diferença entre as aspas simples e as aspas duplas é que temos uma sequência de caracteres de controle que podem ser inseridos em nossa String, veja a tabela a seguir:

Sequência	Significado
\n	Fim de linha
\r	Retorno de carro(carriage return)
\t	Tab horizontal
\v	Tab vertical

\f	form feed
\\	Contra barra ou barra invertida
\$	Sinal de cifrão
\"	Aspas

O tipo boolean, booleano, no PHP

O tipo booleano é muito simples pois aceita apenas os valores **verdadeiro(TRUE)** ou **falso(FALSE)**.

Exemplo 16:

Nome do arquivo: aula01ex16.php

```
<?php
// Declaração de uma variável booleana com o valor verdadeiro
$sim      = TRUE;
// Declaração de uma variável booleana com o valor falso
$nao      = FALSE;
?>
```

Explicando o código:

Você deve ter reparado que não exibimos valores na tela, isto porque o tipo booleano expressa um valor verdade (verdadeiro ou falso). Quando chegarmos às estruturas de controle, aí sim o valor booleano fará muito mais sentido.

O tipo booleano é case-insensitive, ou seja, não diferencia maiúsculas de minúsculas. Isto é true, TrUe e TRUE são iguais assim como false, FaLsE e FALSE são iguais.

Mesmo assim, prefira sempre utilizar: TRUE e FALSE ou true e false.

O tipo NULL no PHP

Outro tipo de dado muito simples, o valor especial **NULL**, representa que a variável não tem valor. NULL é o único valor possível do tipo NULL.

Exemplo 17:

Nome do arquivo: aula01ex17.php

```
<?php
// Declaração de uma variável NULL
$nulo = NULL;
?>
```

O tipo NULL é case-insensitive, assim como o tipo booleano, ou seja, não diferencia maiúsculas de minúsculas. Isto é null, NulL e NULL são iguais.

Mesmo assim, prefira sempre utilizar: NULL ou null.

Atividades

Nesta atividade vamos criar uma página PHP simples que armazena vários dados em variáveis diversas, e em seguida os exibam na tela pelo comando *echo*.

Nome do arquivo: cadastro.php

```
<?php
// Definindo variáveis
  define("NOMECLIENTE","Marcos de Melo");
  $endereco       = 'Rua das Violetas, 320';
  $bairro         = 'Jd. Callegari';
  $estado         = 'SP';
  $cep            = '13.181-659';
  $idade          = 18;
  define("RG", "28290355-x");
  $fone           = '(19) 8888-9696';
?>
<html>
  <head>
        <title>Trabalhando com Variáveis</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-
  8">
  </head>
  <body>
        <h1>Informações do Contato</h1>
        <p>Nome completo: <?php echo NOMECLIENTE;?> </p>
        <p>Endereço: <?php echo $endereco;?> </p>
        <p><?php echo "$bairro - $estado - CEP: $cep";?> </p>
        <p> Telefone: <?php echo $fone;?></p>
        <h2>Documentos</h2>
        <?php echo "<p>Idade: $idade | RG: " . RG . "</p>";?>
  </body>
</html>
```

Questionário

1) Quais são as tags principais para que o Navegador identifique conteúdo PHP em uma página.

a) ( ) <?php e ?>
b) ( ) <html> e </html>
c) ( ) <head> e </html>
d) ( ) <title> e </title>

2) Assinale a alternativa correta que define uma variável.

a) ( ) $100moral;
b) ( ) $valor 1;
c) ( ) $ valorProduto1;
d) ( ) $casal20;

3) Assinale a alternativa correta que define uma constante.

a) ( ) define("20VER","Cheguei olhei e não vi nada");
b) ( ) define("100VALOR", "0");
c) ( ) define("CPFCLIENTE", "196635589-25");
d) ( ) define("RG CLIENTE", "28568365-X");

Aula 2 - Operadores

O PHP possui basicamente os mesmos operadores de qualquer outra linguagem de programação como: Aritméticos, de Strings, de Atribuição, Bit a Bit, Lógicos, Comparação e de Incremento e Decremento.

Aritméticos

Só podem ser utilizados quando os operandos são números (`integer` ou `float`). Se forem de outro tipo, terão seus valores convertidos antes da realização da operação.

+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo

Operador de adição

Exemplo 1:

Nome do arquivo: aula02ex01.php

```
<?php
    $valor1             = 100;
    $valor2             = 550;
    $resultado    = $valor1 + $valor2;
    echo "A soma do valor {$valor1} com o valor {$valor2} é: {$resultado}";
?>
```

Resultado:

A soma do valor 100 com valor 550 é: 650

Operador de Subtração

Exemplo 2:

Nome do arquivo: aula02ex01.php

```
<?php
    $valor1             = 1000;
    $valor2             = 550;
    $resultado    = $valor1 - $valor2;
    echo "A subtração do valor 2 do valor1 é: $resultado";
?>
```

Resultado:

A subtração do valor 550 do valor 1000 é: 450

Operador de Multiplicação

Exemplo 3:

Nome do arquivo: aula02ex03.php

```
<?php
   $valor1      = 1000;
   $valor2      = 3;
   $resultado   = $valor1 * $valor2;
   echo "A Multiplicação do valor 1 pelo valor 2 é: $resultado";
?>
```

Resultado:

A Multiplicação do valor 1 pelo valor 2 é: 3000

Operador de Divisão

Exemplo 4:

Nome do arquivo: aula02ex4.php

```
<?php
$valor1   = 1000;
$valor2   = 2;
$resultado      = $valor1 / $valor2;
echo "A divisão do valor 1 por valor2 é: $resultado";
?>
```

Resultado:

A divisão do valor 1 por valor 2 é: 500

Operador de Módulo

Exemplo 5:

Nome do arquivo: aula02ex05.php

```php
<?php
    $valor1      = 5;
    $valor2      = 2;
    $resto       = $valor1 % $valor2;
    echo "O resto da divisão do valor 1 pelo valor 2 é: $resto";
?>
```

Resultado:

```
O resto da divisão do valor 1 pelo valor 2 é: 1
```

Aritméticos unários

Os operadores unários utilizam apenas um operador para incrementar ou decrementar uma variável.

-oper	Troca o sinal do operador
++oper	Pré-incremento.
--oper	Pré-decremento.
oper++	Pós-incremento.
oper--	Pós-decremento.

Exemplo 6:

Nome do arquivo: aula02ex06.php

```php
<?php
    $operador++; // A variável foi incrementada para mais 1 (um).
    echo $operador . "<br>";
    $operador++;
    echo $operador; // A variável já guarda o valor 1 e foi incrementada para mais 1 (um)
?>
```

Resultado:

```
1
2
```

Strings

No PHP só há um operador exclusivo para Strings.

.	Concatenação

O operador (.) “concatenação” é utilizado quando queremos juntar uma cadeia de String com outra cadeia de String.

Exemplo 7:

Nome do arquivo: aula02ex07.php

```php
<?php
  $string1 = "Programando";
  $string2 = "em PHP.";
  $stringFinal = $string1 . " " . $string2;
  // string1 junta com "espaço" que junta com string2
  echo($stringFinal);
?>
```

Resultado:

Programando em PHP.

Atribuição

Existe um operador básico de atribuição o igual (=) e diversos derivados, sempre retornam o valor atribuído. No caso dos operadores derivados de atribuição, a operação é feita entre os dois operandos, sendo atribuído o resultado para o primeiro. A atribuição é sempre por valor, e não por referência.

=	Atribuição simples
+=	Atribuição com adição
-=	Atribuição com subtração
*=	Atribuição com multiplicação
/=	Atribuição com divisão
%=	Atribuição com módulo
.=	Atribuição com concatenação

Exemplo 8:

Nome do arquivo: aula02ex08.php

```php
<?php
echo "<h2>Atribuição simples</h2>";
$valor1 = 100; // Operador atribui o valor 100 à variável $valor1
echo "Valor 1 é: $valor1 <br>"; //100

echo "<h2>Atribuição com adição</h2>";
$valor1 += 50; // Igual a valor1 = valor1 + 50
echo "Valor 1 é: $valor1 <br>"; // 150

echo "<h2>Atribuição com subtração</h2>";
$valor1 -= 25; // Igual a valor1 = valor1 - 25
echo "Valor 1 é: $valor1 <br>"; // 125

echo "<h2>Atribuição com multiplicação</h2>";
$valor1 *= 2; // Igual a valor1 = valor1 * 2
echo "Valor 1 é: $valor1 <br>"; // 250

echo "<h2>Atribuição com divisão</h2>";
$valor1 /= 2; // Igual a valor1 = valor1 / 2
echo "Valor 1 é: $valor1 <br>"; // 125

echo "<h2>Atribuição com módulo</h2>";
$valor1 %= 3; // Retorna o resto da divisão de 125 por 3
echo "Valor 1 é: $valor1 <br>"; // 2

echo "<h2>Atribuição com módulo</h2>";
$valor1 .= 9999; // Junta o valor "2" com "9999" ficando 29999.
```

```
echo "Valor 1 é: $valor1 <br>"; // 29999
?>
```

Resultado:

Atribuição simples
Valor 1 é: 100
Atribuição com adição
Valor 1 é: 150

Atribuição com subtração
Valor 1 é: 125

Atribuição com multiplicação
Valor 1 é: 250

Atribuição com divisão
Valor 1 é: 125

Atribuição com módulo
Valor 1 é: 2

Atribuição com módulo
Valor 1 é: 29999

Lógicos

Utilizados para inteiros representando valores booleanos, ou seja, “Verdadeiro ou “Falso”.

AND	“e” lógico
OR	“ou” lógico
XOR	ou exclusivo
!	não (inversão)
&&	“e” lógico
||	“ou” lógico

Existem dois operadores para “e” e para “ou porque eles têm diferentes posições na ordem de precedência. Mais adiante é mostrada a tabela de precedência dos operadores.

Operador AND

O operador lógico AND avalia de uma expressão a sua esquerda e também a sua direita são verdadeiras. Se ambas forem verdadeiras, a condição será VERDADEIRA, caso contrário FALSA.

Exemplo 9:

Nome do arquivo: aula02ex09.php

```
<?php
$valor1 = true;
$valor2 = true;

if($valor1 and $valor2)// Avalia se valor1 e também valor2 são verdadeiros
   echo 'Verdadeiro'; // Se ambos os valores forem verdadeiros realiza esta expressão
else
   echo 'Falso'; // Caso um ou todos os valores sejam falsos realiza esta expressão
?>
```

Resultado:

```
Verdadeiro
```

Explicando o código:

No exemplo acima, duas variáveis $valor1 e $valor2, possuem o valor booleano “true” e são comparadas pelo operador lógico AND e se ambas retornarem um valor verdadeiro, a condição no comando IF será verdadeira, realizando a expressão da linha 6. Caso uma das variáveis tenha o valor False, a condição no comando IF retornara False e realizara em caso falo, a expressão da linha 8.

Operador OR

Para que o operador OR (Ou) retorne “Verdadeiro” no teste lógico, basta que uma das expressões retorne o valor verdadeiro. Se as duas expressões forem falsas retornara “Falso”.

Exemplo 10:

Nome do arquivo:aula02ex10.php

```
<?php
$valor1 = true;
$valor2 = false;

if($valor1 or $valor2)// Verdadeiro se pelo menos um valor for verdadeiro
   echo 'Verdadeiro'; // Se um ou todos os valores forem verdadeiros, realiza esta
expressão
else
   echo 'Falso'; // Caso todos os valores forem falsos, realiza esta expressão
?>
```

Resultado:

```
Verdadeiro
```

Operador XOR

O operador XOR é exclusivo, oi seja, entre as expressões avaliadas, somente uma deve retornar o valor verdadeiro, para que a condição retorne “Verdadeiro”. Se todas as expressões forem verdadeiras ou todas falsas, a condição retornara “Falso”.

Exemplo 11:

Nome do arquivo:aula02ex11.php

```
<?php
$valor1 = true;
$valor2 = true;

if($valor1 xor $valor2)// Verdadeiro se somente um valor for verdadeiro
   echo 'Verdadeiro'; // Se um valor somente for verdadeiro, realiza esta expressão.
else
```

```
    echo 'Falso'; // Caso todos os valores forem falsos ou verdadeiros, realiza esta
expressão.
?>
```

Resultado:

Falso

Operador !(NOT)

O operador !(not) avalia o valor retornado de uma expressão somente e sua função é inverter o valor retornado por esta expressão, ou seja, se a expressão retornar o valor Verdadeiro, este valor é invertido para Falso no resultado da condição.

Exemplo 12:

Nome do arquivo: aula02ex12.php

```
$valor1 = true;

if(!$valor1)// Verdadeiro se o valor for Falso.
   echo 'Verdadeiro';
else
   echo 'Falso';
?>
```

Resultado:

Falso

A tabela a seguir mostra os resultados obtidos com os operadores lógicos.

Expressão 1	Expressão 2	Resultados com cada operador			
		AND	OR	XOR	!(not)
V	V	V	V	F	
V	F	F	V	V	
F	V	F	V	V	
F	F	F	F	F	
F					V
V					F

Ternário

É uma maneira abreviada de usar o comando IF. Basicamente, se uma condição for avaliada como verdadeira, um valor é atribuído à variável, caso contrário, se for falso, um outro valor é atribuído.

Exemplo 13:

Nome do arquivo: aula02ex13.php

```
<?php
    $nota = 4;
    $resultado = ($nota >= 7)? ($resultado="Aprovado"):($resultado="Reprovado");
    echo "Resultado da nota: $resultado";
?>
```

Resultado:

```
Resultado da nota: Reprovado
```

Comparação

As comparações são feitas entre os valores contidos nas variáveis, e não as referências. Sempre retornam um valor booleano.

==	Igual a
!=	Diferente de
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a

Ordem de precedência dos operadores

A tabela a seguir mostra a ordem de precedência dos operadores no momento de avaliar as expressões.

Precedência	Associatividade	Operadores
1.	Esquerda	,
2.	Esquerda	or
3.	Esquerda	xor
4.	Esquerda	and
5.	Direita	print
6.	Esquerda	= += -= *= /= .= %= &= != ~= <<= >>=
7.	Esquerda	? :
8.	Esquerda	||
9.	Esquerda	&&
10.	Esquerda	|
11.	Esquerda	^
12.	Esquerda	&
13.	Não associa	== !=
14.	Não associa	< <= > >=
15.	Esquerda	<< >>
16.	Esquerda	+ - .
17.	Esquerda	* / %
18.	Direita	! ~ ++ -- (int) (double) (string) (array) (object) @
19.	Direita	[
20.	Não associa	new

Atividades

1) Desenvolva um programa em PHP com variáveis e constantes para receber os valores abaixo e em seguida mostra-los na tela.

Valores pra variáveis;

Nome	Valores
Nome do aluno	Maria Fernanda
Endereço	Rua Violeta, 3025
Cidade	Campinas
Estado	São Paulo
CEP	13.536-459
Telefone	(19) 8182-9977
Celular	(19) 3854-3695

Valores para Constantes;

Nome	Valores
RG	28.569.569-X
CPF	123.653.659-98
PIS	8452368888

2) Crie um programa que seja utilizado um dos operadores aritméticos.

Aula 3 - Vetores e Matrizes

Vamos compreender nesta aula o conceito de vetores e matrizes no PHP. Será demonstrado como, inserir dados em um vetor ou em uma matriz e as diferenças entre vetores e matrizes.

Vetores

Os vetores, também conhecidos como arrays, são semelhantes as variáveis, porém a diferença entre vetor e variável é que, uma variável guarda somente um valor por vez em seu interior e um vetor pode guardar vários.

Para explicar melhor quando utilizar vetores no lugar de variáveis, imagine a necessidade de se guardar 50 valores semelhantes, Ex: Nomes de Produtos, se fosse feito em variáveis, faríamos assim, $prod1="banana", $prod2=" Maçã", $prod3="Mamão",... teríamos então que, criar 50 variáveis, já em vetores, este processo seria bem mais fácil, pois criaríamos somente uma variável de array (vetor) para guardar os 50 valores de produtos de uma vez só.

Criando vetores

Antes de começarmos a criar vetores, você deve entender que, um vetor assim como uma variável são espaços na memória (RAM) em tempo de execução, ou seja, se desligarmos o computador os vetores criados serão eliminados.

Podemos criar um vetor atribuindo a uma variável as propriedades de um vetor pela função array(). A regra para a criação dos nomes de vetores é a mesma utilizada em variáveis. Veja o exemplo a seguir;

Exemplo 1:

Nome do arquivo: aula03ex01.php

```
<?php
   $vetor = array();
?>
```

Explicando o código:

No exemplo acima, a variável $vetor já é um array, porém ainda não definimos posições para a atribuição de valores, podendo definir posteriormente.

Agora vamos declarar o mesmo vetor novamente, já com valores atribuídos.

Exemplo 2:

Nome do arquivo: aula03ex02.php

```
<?php
   $vetor = array("Fulano", "Ciclano", "Beltrano");
   echo $vetor[1];
?>
```

Resultado:

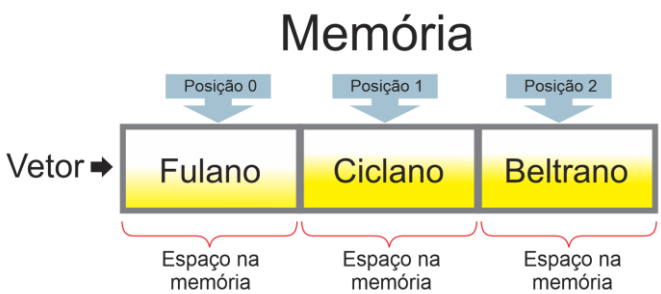
```
Ciclano
```

Explicando o código:

Na linha 2, a variável $vetor foi convertida para um vetor e no momento de sua declaração, já foi atribuído três valores. Podemos adicionar qualquer tipo de valor separados por vírgula(,).

Cada valor inserido possui uma posição relacionada a um índice(index) numérico no vetor que por padrão inicia no valor 0 (zero).

Os índices são indicados por colchetes ([]) logo após o nome do vetor. Portanto, se quisermos atribuir, modificar ou acessar algum valor em uma posição especifica do vetor, colocaríamos o número da posição entre colchetes. Na linha 3 do código acima, acessamos o valor da posição 1 que seria "Ciclano".



Arrays Associativos

Vetores associativos, ao contrário de usarmos números como índices, usa-se Strings (nomes). Para usarmos o array associativo basta apenas substituir o número do índice por uma String. Veja o exemplo abaixo.

Exemplo 3:

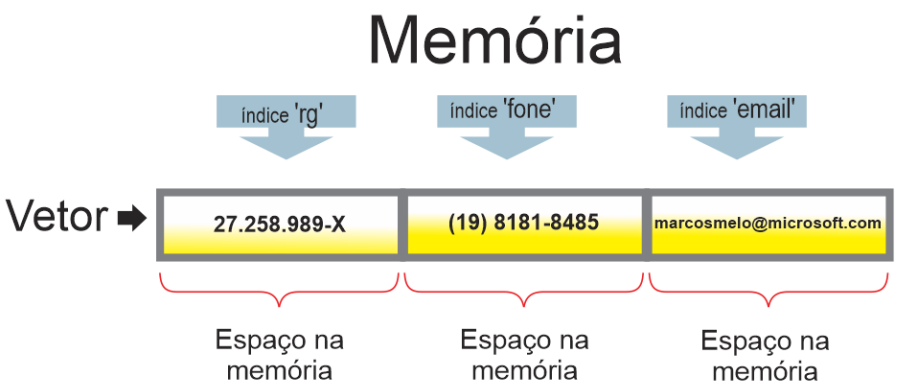
Nome do arquivo: aula03ex03.php

```
<?php
    $doc = array();
    $doc['rg']          = "27.258.989-X";
    $doc['fone']        = "(19) 8181-8485";
    $doc['email']       = "marcosmelo@microsoft.com";

    echo $doc['email'];
?>
```

Explicando o código:

No código acima um vetor chamado $doc foi criado na linha 2 e logo após nas linhas 3,4 e 5, atribuímos três valores a esse vetor por índices associativos.

Na linha 7, para acessar o valor armazenado no índice associativo email, referenciamos o índice entre colchetes.



Podemos criar um vetor, definir índices de chaves associativas e atribuir valores ao mesmo tempo utilizando o operador "=>". Veja o exemplo abaixo.

Exemplo 4:

Nome do arquivo: aula03ex04.php

```
<?php
    $doc = array("rg"=>"27.258.989-X","fone"=> "(19) 8181-
    8485","mail"=>"marcosmelo@microsoft.com");
    echo $doc["rg"];
?>
```

Resultado:

```
27.258.989-X
```

Matrizes

Matrizes são basicamente vetores dentro de outros vetores. Ou seja, em cada posição de um vetor(array), possui outro vetor.

Quando utilizamos arrays multidimensionais (matrizes) existem mais índices numa mesma variável.

- Array comum**: $vetor []**
- Array bidimensional**: $vetor [][]**
- Array tridimensional**: $vetor [][][] e assim por diante.**

Em matrizes podemos trabalhar com o conceito de tabelas, ou seja, linhas e colunas manipuláveis onde podemos armazenar valores diversos.

Exemplo 5:

Nome do arquivo: aula03ex05.php

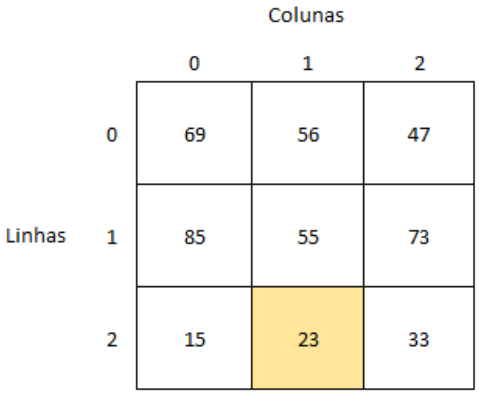
```php
<?php
   //Matriz 3x3 (três linhas por três colunas)

   $Matriz = array(array(69,56,47), array(85,55,73), array(15,23,33));

   echo $Matriz[2][1];
?>
```

Resultado:

```
23
```

Explicação do código:

Se formos visualizar a tabela gerada na matriz ela seria assim.



A linha 6 imprime na tela o valor da referência das coordenadas, linha de índice 2 e coluna de índice 1 retornando o valor 23. Lembrando que a contagem do índice começa do zero, ou seja, a linha de índice 2 é a linha 3 e a coluna de índice 1 é a coluna 2.

Neste formato o primeiro colchete define a coluna e o segundo colchete define a linha.

Veja outro formato associativo;

Exemplo 6:

Nome do arquivo: aula03ex06.php

```php
<?php

// Uma matriz associativa
$valor = array(
    'Cor'               => array( 'Vermelho', 'Branco', 'Prata' ),
    'Capacidade'        => array( '4GB', '16GB', '32GB' ),
    'Interface'         => array( 'Nokia', 'iPhone', 'HTC' ),
);

// Esta linha imprimira 'Vermelho' na tela
echo $valor['Cor'][0];

//Coluna 'cor' e o item com linha zero '0'.

?>
```

Colunas com chaves associativas

Linhas	Cor	Capacidade	Modelo
0	Vermelho	4GB	Nokia
1	Branco	16GB	iPhone
2	Prata	32GB	HTC

É possível especificar uma tabela em matriz por índices numéricos, veja o exemplo a seguir;

Exemplo 7:

Nome do arquivo: aula03ex07.php

```php
<?php

//Dados coluna 0 - Código
$dadosProduto[0][0] = 1;
$dadosProduto[1][0] = 2;
$dadosProduto[2][0] = 3;
$dadosProduto[3][0] = 4;
$dadosProduto[4][0] = 5;

// Dados coluna 1 - Produto
$dadosProduto[0][1] = "Banana";
$dadosProduto[1][1] = "Maçã";
$dadosProduto[2][1] = "Mamão";
$dadosProduto[3][1] = "Melão";
$dadosProduto[4][1] = "Caju";

// Dados coluna 2 - Valor
$dadosProduto[0][2] = 10.50;
$dadosProduto[1][2] = 15.99;
$dadosProduto[2][2] = 25.50;
$dadosProduto[3][2] = 30.50;
$dadosProduto[4][2] = 45.99;

echo "<h2>Detalhes do produto</h2>";
echo 'Código: ' . $dadosProduto[4][0] . ' <br>';
echo 'Produto: ' . $dadosProduto[4][1] . ' <br>';
echo 'Valor: ' . $dadosProduto[4][2] . ' <br>';

?>
```

Resultado:

Detalhes do produto
Código: 5
Produto: Caju
Valor: 45.99

Atividades

1 – Descreva o código php, para criar o vetor associativo, especificado graficamente abaixo.

Código	Livros	Autor	Páginas
8282	PHP - Páginas dinâmicas	Marcos de Melo	360

2- Descreva o código php para criar a matriz especificada graficamente abaixo.

	codigo	livros	autor	paginas
0	8282	PHP	Marcos de Melo	360
1	8283	HTML5	Ernandes Silva	266
2	8284	jQuery	Nicoly Almeida	159
3	8285	CSS	Juliana Aparecid	123

Aula 4 – Funções internas

O PHP vem por padrão com muitas funções, nesta aula vamos compreender o uso de algumas dessas várias funções, funções essas que vamos utilizar em exemplos contidos no livro.

Trabalhando com Datas

No PHP existem diversas funções para o tratamento de datas. Você pode exibir a data atual de várias formas. Usando a função **date()** do PHP é possível formatar a data atual de diversas formas, vamos a alguns parâmetros mais comuns que podem ser usados:

Caractere de *Formatação*	Descrição	Valores retornados
Day	---	---
D	Dia do mês, 2 dígitos com preenchimento de zero	*01* até *31*
D	Uma representação textual de um dia, três letras	*Mon* até *Sun*
J	Dia do mês sem preenchimento de zero	*1* até *31*
***l* ('L' minúsculo)**	A representação textual completa do dia da semana	*Sunday* até *Saturday*
N	Representação numérica ISO-8601 do dia da semana (adicionado no PHP 5.1.0)	*1* (para Segunda) até *7* (para Domingo)
S	Sufixo ordinal inglês para o dia do mês, 2 caracteres	*st*, *nd*, *rd* ou *th*. Funciona bem com *j*
W	Representação numérica do dia da semana	*0* (para domingo) até *6* (para sábado)
Z	O dia do ano (começando do 0)	*0* through *365*
Semana	---	---
W	Número do ano da semana ISO-8601, semanas começam na Segunda (adicionado no PHP 4.1.0)	Exemplo: *42* (the 42nd week in the year)
Mês	---	---
F	Um representação completa de um mês, como January ou March	*January* até *December*
M	Representação numérica de um mês, com leading zeros	*01* a *12*
M	Uma representação textual curta de um mês, três letras	*Jan* a *Dec*
N	Representação numérica de um mês, sem leading zeros	*1* a *12*
T	Número de dias de um dado mês	*28* through *31*
Year	---	---

L	Se está em um ano bissexto	*1* se está em ano bissexto, *0* caso contrário.
O	Número do ano ISO-8601. Este tem o mesmo valor como *Y*, exceto que se o número da semana ISO (*W*) pertence ao anterior ou próximo ano, o ano é usado ao invés. (Adicionado no PHP 5.1.0)	Exemplos: *1999* ou *2003*
Y	Uma representação de ano completa, 4 dígitos	Exemplos: *1999* ou *2003*
Y	Uma representação do ano com dois dígitos	Exemplos: *99* ou *03*
Tempo	---	---
A	Antes/Depois de meio-dia em minúsculo	*am* or *pm*
A	Antes/Depois de meio-dia em maiúsculo	*AM* or *PM*
B	Swatch Internet time	*000* até *999*
G	Formato 12-horas de uma hora sem preenchimento de zero	*1* até *12*
G	Formato 24-horas de uma hora sem preenchimento de zero	*0* até *23*
H	Formato 12-horas de uma hora com zero preenchendo à esquerda	*01* até *12*
H	Formato 24-horas de uma hora com zero preenchendo à esquerda	*00* até *23*
i	Minutos com zero preenchendo à esquerda	*00* até *59*
S	Segundos, com zero preenchendo à esquerda	*00* até *59*
U	Milisegundos (adicionado no PHP 5.2.2)	Exemplo: *54321*
Timezone	---	---
E	Identificador de Timezone (adicionado no PHP 5.1.0)	Exemplos: *UTC*, *GMT*, *Atlantic/Azores*
***I* (capital i)**	Se a data está ou não no horário de verão	*1* se horário de verão, *0* caso contrário.
O	Diferença para Greenwich time (GMT) em horas	Exemplo: *+0200*
P	Diferença para Greenwich time (GMT) com dois pontos entre horas e minutos (adicionado no PHP 5.1.3)	Exemplo: *+02:00*
T	Abreviação de Timezone	Exemplos: *EST*, *MDT* ...
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	*-43200* até *50400*
Full Date/Time	---	---
C	ISO 8601 date (adicionado no PHP 5)	2004-02-12T15:19:21+00:00
R	» RFC 2822 formatted date	Exemplo: *Thu, 21 Dec 2000 16:01:07 +0200*

U	Segundos desde a Época Unix (January 1 1970 00:00:00 GMT)	Veja também time()

Nota: A data e hora que a função date() retorna, é a do servidor web php e como nosso servidor web, o Apache, está instalado localmente, a data fornecida é a de seu próprio Computador, porém se a aplicação estiver em um servidor remoto, a data e hora podem variar.

Exemplo 1:

Nome do arquivo: aula04ex01.php

```
<?php
echo date('d/m/Y'); // Resultado: 12/03/2009
echo date('H:i:s'); // Resultado: 03:39:57
echo date('d/m h:i A'); // Resultado: 12/03 03:39 AM
?>
```

Datas no formato certo para o banco de dados MySQL

No MySQL existem os tipos de coluna DATE e DATETIME que são para armazenar datas e datas com hora respectivamente. O formato de entrada de um campo DATE é **AAAA-MM-DD** e do DATETIME é **AAAA-MM-DD HH:MM:SS**. Pra usar a função **date()** e gerar datas no formato de entrada do MySQL é só fazer assim:

Exemplo 2:

Nome do arquivo: aula04ex02.php

```
<?php
$data = date('Y-m-d');        // Formato DATE: 2009-03-12
$data = date('Y-m-d H:i:s');  // Formato DATETIME: 2009-03-12 03:39:57

echo time();                          // Mostra o timestamp atual: 1236844015

$timestamp = time();          // Salva o timestamp atual numa variável
echo date('d/m/Y H:i:s', $timestamp); // Exibe DD/MM/YYYY HH:MM:SS em função de um
timestamp
?>
```

Exibindo a data por extenso

O formato data por extenso retornada no php é mostrada em inglês.

Exemplo 3:

Nome do arquivo: aula04ex03.php

```
<?php
// Suponde que a data atual no servidor seja 15/09/2013.
echo date('l, F d, Y');
?>
```

Resultado:

Friday, June 21, 2013

Portanto, para exibir a data do servidor por extenso em português, é necessário utilizar de algumas técnicas com vetores. Utilizaremos também a função getdate() para extrairmos as informações de data para utilizar como índices nos vetores. Veja a seguir maiores detalhes da função getdate();

getdate()

getdate — Obtém informações da data e hora do servidor

Sintaxe

array **getdate** ([int `$timestamp`])

Retorna um array associativo contendo a informação da data do *`timestamp`*, ou a hora corrente local se nenhum *`timestamp`* é dado.

Parâmetros

`Timestamp`

O parâmetro opcional timestamp é um integer Unix timestamp cujo padrão é a hora local se timestamp não for dado. Em outras palavras, o padrão é o valor de time().

Valor Retornado

Retorna um vetor associativo de informação sobre o *`timestamp`*.

Vejas os elementos de chaves associativas do vetor e seus valores a seguir;

Elementos chave de retorno do vetor associativo		
Chave	**Descrição**	**Exemplo dos valores retornados**

"seconds"	Retorna numericamente os segundos	*0* a *59*
"minutes"	Retorna numericamente os minutos	*0* a *59*
"hours"	Retorna numericamente as horas	*0* a *23*
"mday"	Retorna numericamente o dia do mês	*1* a *31*
"wday"	Retorna numericamente o dia da semana	*0* (para Domingo) a *6* (para Sábado)
"mon"	Retorna numericamente o mês	*1* a *12*
"year"	Retorna numericamente o ano completo (4 dígitos)	Exemples: *1999* ou *2003*
"yday"	Retorna numericamente a quantidade de dias corridos até a data atual do dia em um ano.	*0* a *366*
"weekday"	Retorna textualmente o dia da semana (sem abreviação)	*Sunday* a *Saturday*
"month"	Retorna textualmente o mês da semana (sem abreviação), tal como January ou March	*January* a *December*
0	Retorna os segundos desde a época UNIX, similar aos valores retornados pela time() e usados por date().	Dependente do sistema, tipicamente *-2147483648* à *2147483647*.

Analisando a tabela de informações que podem ser retornadas pela função getdate(), vamos listar as informações necessárias para acessar os vetores que iram guardar os dias da semana e meses do ano em linguagem "Português Brasil".

Exemplo 4:

Nome do arquivo: aula04ex04.php

```
<?php
// Supondo que a data atual no servidor seja 15/09/2013.

/* Criamos um vetor com 7 (sete) posições com índices numéricos de 0 à 6,
sendo 0 para Domingo (primeiro dia da semana) indo até 6 Sabado (último
dia).
*/
//Dias da semana por extenso armazenados no vetor $vetorSemana[].

$vetorSemana[0]      = "Domingo";
$vetorSemana[1]      = "Segunda-feira";
$vetorSemana[2]      = "Terça-feira";
$vetorSemana[3]      = "Quarta-feira";
$vetorSemana[4]      = "Quinta-feira";
$vetorSemana[5]      = "Sexta-feira";
$vetorSemana[6]      = "Sábado";
```

```
/* Criamos um vetor com 12 posições que guarda o valor dos meses por
extenso, iniciando em índice 1 até 12
*/

// Meses do ano armazenados no vetor $vetorMeses[ ]
$vetorMeses[1]        = "Janeiro";
$vetorMeses[2]        = "Fevereiro";
$vetorMeses[3]        = "Janeiro";
$vetorMeses[4]        = "Fevereiro";
$vetorMeses[5]        = "Janeiro";
$vetorMeses[6]        = "Junho";
$vetorMeses[7]        = "Janeiro";
$vetorMeses[8]        = "Fevereiro";
$vetorMeses[9]        = "Janeiro";
$vetorMeses[10]       = "Fevereiro";
$vetorMeses[11]       = "Janeiro";
$vetorMeses[12]       = "Fevereiro";


$dataAtual            = getdate();// Cria o vetor associativo com informações do
timestamp.

// Recuperação dos dados necessários no vetor
$numDiaSemana  = $dataAtual["wday"];   // 0 a 6 - Domingo a Segunda
$diaDoMes             = $dataAtual["mday"];   // 1 a 31
$numDoMes             = $dataAtual["mon"];    // 1 a 12
$ano                  = $dataAtual["year"];   // Ano em 4 dígitos

$diaSemana            = $vetorSemana[$numDiaSemana];
$mes                  = $vetorMeses[$numDoMes];

echo "Hoje é  $diaSemana, $diaDoMes de $mes de $ano.";

?>
```

Resultado baseado na data atual 15/09/2013 – Sexta-feira:

```
Hoje é Sexta-feira, 25 de Outubro de 2013.
```

Funções de validação e formatação de caracteres

Aprenda neste tópico algumas funções internas do PHP para formatação e validação de caracteres de entrada no PHP, inseridos por exemplo em formulários.

Localizando a primeira ocorrência de

strstr()

A função strstr() localiza a primeira entrada de um caractere especificado no parâmetro.

Sintaxe:

```
string strstr ( string $haystack , mixed $needle [, bool $before_needle ] )
```

É muito importante quando desejamos evitar que o usuário digite espaços em branco entre os caracteres, como campos de texto de senhas em formulários. Geralmente não utilizamos espaços em nomes de usuários e senhas, e caso o usuário digite algum espaço devemos localizar a existência desses espaços para alertar o usuário. Observe o exemplo abaixo, como resolver esta situação;

Exemplo 5:

Nome do arquivo: aula04ex05.php

```
<?php
   $usuario     = "marcos melo";
   $senha       = "php 123";

   if(strstr($usuario,' ') or strstr($senha,' ')){
          echo "O nome de usuário ou a senha contém espaços, remova os espaços e tente
   novamente.";
}
```

Resultado:

O nome de usuário ou a senha contém espaços, remova os espaços e tente novamente.

Explicando o código:

Neste código aplicamos a função interna strstr() nas variáveis $usuario e $senha para localizar o caractere espaço. As funções estão dentro de uma instrução condicional IF e se alguma delas encontrar o caractere espaço a condição será considerada verdadeira executando o trecho de código pedindo para o usuário remover os espaços.

Substituindo valores

str_replace()

A função str_replece() basicamente localiza um valor especificado em uma String e ao ser encontrado o substitui por outro também especificado na função.

Sintaxe:

```
str_replace([valor procurado],[valor que o substituirá],[string de pesquisa])
```

A função pode ser usada com a String direto na função.

Exemplo 6:

Nome do arquivo: aula04ex06.php

```php
<?php
   $nova   = str_replace('m', 'M', 'marcos de melo');
   echo $nova;
?>
```

Resultado:

Marcos de Melo

Ou utilizando variáveis para armazenar a String.

Exemplo 7:

Nome do arquivo: aula04ex07.php

```php
<?php
   $string = 'Senado aprova projeto que libera governo para gastar R$ 62 bilhões';
   $nova   = str_replace('Senado', 'Camara', $string);
   echo $nova;

?>
```

Resultado:

Camara aprova projeto que libera governo para gastar R$ 62 bilhões

number_format()

Em programação sabemos que a separação decimal de números monetários, é o ponto(.), que é um formato padrão americano. Ex: “155.99”, e que é também o formato padrão usado nos sistemas de armazenamento de banco de dados como o MySQL, que veremos mais adiante. Nós sabemos desta informação, mas o usuário final, que irá digitar as informações em um formulário web, pode não saber e digitar o valor monetário no formato brasileiro, ou seja, usar a virgula (,) como separador de casas decimais.

Para resolver este problema podemos utilizar a função interna number_format() que converte esta pontuação de uma para outra conforme a necessidade.

Sintaxe:

```
number_format(<valor a converter>,<n° de casas decimais>,'<pontuação procurada>',
'<pontuação nova>')
```

Exemplo 8:

Nome do arquivo: aula04ex08.php

```php
<?php
$dolar = 87946.00;
$real     = number_format($dolar,2,',','.');
echo 'R$ '. $real;
?>
```

Resultado:

```
R$ 87946,00
```

Explicando o código:

A pontuação para casas decimais monetária para números da moeda do dólar($) também é o ponto(.) e usamos este exemplo como motivo claro para usar a função number_format() convertendo a pontuação para a moeda Real (R$).

Basicamente a variável $dolar guarda o valor em formato americano, logo em seguida a mesma é convertida pela função trocando a pontuação ponto(.) para virgula(,) e armazenando o valor convertido na variável $real. O parâmetro <nº de casas decimais> indica quantas casas decimais deveram ser exibidas no formato após a pontuação.

Atividades

1 – Escreva o resultado a ser exibido na execução do programa php abaixo:

```
<?php
   $string   = 'curso de WEB DESIGN';
   $string   = str_replace('c', 'C', $string);
   $string   = str_replace('EB', 'eb', $string);
   $string   = str_replace('ESIGN', 'esing', $string);
   $string   = str_replace(' ', '-', $string);
   echo $string;
?>
```

Resposta: _____

Aula 5 - Condicionais

Nesta aula vamos compreender a utilização dos comandos condicionais do PHP. Os comandos condicionais basicamente servem para tomada de decisões onde determinada instrução só será realizada se antes uma condição atendida, for considerada verdadeira e caso a condição seja considerada falsa, o programa tomara um rumo diferente, realizando outra instrução de comandos.

Condicional IF, ELSE e ELSEIF

Estrutura IF simples

Usamos o comando condicional `If` simples (Sem o else) para executar todo o código da instrução que esteja entre as chaves do comando após a condição, caso o resultado da avaliação seja verdadeiro, caso contrário nada será executado.

Exemplo 1:

Nome do arquivo: aula05ex01.php

```
<?php
    $nota = 10;
    if( $nota >= 7 ){
      echo 'O aluno foi aprovado';
    }
?>
```

Resultado:

O aluno foi aprovado

Explicando o código:

IF traduzindo para o português significa “SE” que é uma Conjunção subordinativa condicional. No código acima, podemos observar o uso do `IF` (SE) para expressar uma condição;

SE nota for maior ou igual a 7; então escrever “O aluno foi aprovado”.

ELSE o desvio condicional composto no PHP

O comando else é utilizado logo após a estrutura de controle `if` ou `elseif`. Sempre que uma condição if ou `elseif` retornar false, o comando `else` será executado.

Exemplo 2:

Nome do arquivo: aula05ex02.php

```
<?php

   $nota = 5;

   if( $nota >= 7 ){

     echo 'O aluno foi aprovado';
   }
   else{
     echo 'O aluno foi reprovado';
   }
?>
```

Resultado:

O aluno foi reprovado

Explicando o código:

Veja como funciona o código do programa acima;

Se (if) na nota for maior ou igual a 7; então escrever “O aluno foi aprovado”;

Senão (else); escrever “O aluno foi reprovado”.

ELSEIF outro desvio condicional composto no PHP

O `elseif` é uma estrutura de controle que pode ser adicionado entre a estrutura `if` e else, avaliando mais uma condição e consequentemente realizando outra instrução de código para caso sua condição seja verdadeira. No código do programa a estrutura `elseif` só será realizada caso a condição do `if` seja falsa. Podemos inserir mais de um `elseif`.

Entenda melhor como seu uso funcionamento no exemplo abaixo;

Exemplo 3:

Nome do arquivo: aula05ex03.php

```
<?php
    $nota = 4;

    if($nota >= 7 ){

      echo 'O aluno foi aprovado';
    }
    elseif($nota >= 5 && $nota < 7){
          echo 'O aluno está de recuperação';
    }
    else{
          echo 'O aluno está reprovado';
    }
?>
```

Resultado:

O aluno está reprovado

Explicando o código:

Veja como funciona o código do programa acima;

Se(if):

A nota for maior ou igual a 7; então escrever “O aluno foi reprovado”;

Senão Se (elseif):

A nota for maior ou igual a 5 e nota menor que 7; então escrever “O aluno está de recuperação”;

Senão(else):

Escrever “O aluno esta reprovado”.

Condicional Switch, Case e Default

O Switch é um comando semelhante ao comando if, pois ambos testam o valor de uma variável várias vezes. A única diferença é que o Switch testa a variável basicamente com o operador de

igualdade enquanto que o if testa com todos os operadores. A motivação em usar o Switch em vez do if é quando queremos testar o valor de uma variável com uma quantidade maior de opções.

`Switch` trabalha usando casos (`case`) que são valores definidos.

A estrutura do switch é: **`switch`** `(variável) {case valor: código a ser executado}`

Sintaxe:

```
Switch{variável)
{
Case <valor>: <código a ser executado>
Case <valor>: <código a ser executado>
Case....
}
```

Utilizando o Switch, economizamos tempo de trabalho evitando usar várias estruturas if-else aninhadas para verificar um dado específico.

Exemplo 4:

Nome do arquivo: aula05ex04.php

```php
<?php

$numero_premio = 3;

switch ( $numero_premio ){
  case 1:
    $premio = 'uma casa';
    break;
  case 2:
    $premio = '40 mil reais em barras de ouro';
    break;
  case 3:
    $premio = 'um carro zero';
    break;
  case 4:
    $premio = 'uma bicicleta';
    break;
  case 5:
    $premio = 'um conjunto de panelas';
    break;
  default:
    $premio = 'nada, infelizmente';
}
```

```
echo " Seu prêmio é: $premio.";
?>
```

Resultado:

Seu prêmio é: um carro zero.

Explicando o código:

No código acima o valor de uma variável chamada $numero_premio é comparado com outros valores, contidos nas instruções case. Por exemplo, se o valor for igual a 3, o prêmio a ser armazenado na variável $premio, será “um carro zero”, porém todos os outros casos após, também seriam executados. Para impedir isso, usamos o comando break após a instrução de código de cada case, evitando assim, que os demais “cases” após o case avaliado, sejam executados.

O comando Default (padrão) é usado no final, após todas as instruções cases, se nenhum case for igual ao valor contido na variável $numero_premio, a instrução do comando Default será executado automaticamente.

Atividade

1 – Crie um programa usando a instrução condicional IF e ELSE que atenda a seguinte condição abaixo:

Em uma casa noturna somente pessoas maiores de 18 anos podem entrar, a condição deve ser a seguinte;

Se: a idade for maior ou igual a 18 anos; então escrever “Pode entrar!”;

Senão: “Proibido a entrada de menores de 18 anos”;

R:

Aula 6 - Estruturas de Repetição

Nesta aula será compreendido exclusivamente, os comandos que envolvem estruturas de repetição conhecidas como, laços e loops nas linguagens de programação.

Nos comandos de repetição, basicamente uma instrução de comandos é realizada e repetida sempre que uma condição avaliada for verdadeira. Deve-se observar que, caso o bloco de instruções nunca modifique o estado da condição, a estrutura será executada para sempre, uma situação chamada laço infinito.

Estruturas repetição for

O comando FOR é usado para criar estruturas de repetição de instruções de código. É um dos comandos mais usados no PHP e em todas as linguagens de programação conhecidas.

Sintaxe:

```
for (<inicialização>; <condição>;<incrementação>) {instruções}
```

Se no começo da execução do código a condição no código FOR for falso, nada será executado.

O comando FOR possui três parâmetros que devem ser informados para que funcione. O primeiro parâmetro definimos uma variável numérica e indicamos um valor inicial para ser incrementado ou decrementado. O segundo parâmetro fornecemos a c

Exemplo 1:

Nome do arquivo: aula06ex01.php

```php
<?php
for ($i = 0; $i < 10; $i++) {
    echo "O loop atual é: $i";
}
?>
```

Resultado:

```
O loop atual é: 0
O loop atual é: 1
O loop atual é: 2
O loop atual é: 3
O loop atual é: 4
```

```
O loop atual é: 5
O loop atual é: 6
O loop atual é: 7
O loop atual é: 8
O loop atual é: 9
```

Explicando o código:

Esse script imprime números de 0 até 9, de acordo com a segunda expressão do for.

Traduzindo: PARA i igual a zero, i menor que 10, incremente o i (some 1) e faça.

Estruturas repetição while

A estrutura While são as mais simples estruturas de repetição. Seu comportamento é idêntico aos seus parentes em C, por exemplo.

Sintaxe:

```
while (<condição>){ instruções}
```

Essa instrução é executada ENQUANTO a expressão de controle for verdadeira(true), e esse teste é feito no começo do Loop, assim sendo, se o resultado da primeira verificação for false, as instruções não são executadas nenhuma vez. Se a condição for avaliada como falsa (`False`) o processamento da rotina é desviado para fora do loop.

Exemplo 2:

Nome do arquivo: aula06ex02.php

```php
<?php
    $i=0;
    while($i<=10){
    echo $i "<br>";
    $i++;
}
?>
```

Resultado:

```
0
1
```

```
2
3
4
5
6
7
8
9
```

10

Explicando o código:

Na linha 2 do código a variável $i é criada para atribuirmos a ela um valor que servira como parâmetro de início do processo de repetição (loop).

Na linha 3 aplicamos o comando while onde é definida a condição para que o loop continue é definida. Neste caso a condição é;

Enquanto a variável $i for menor que o valor 10 a instrução é repetida.

Perceba na linha 5 que a variável $i é incrementada para mais um, assim o loop não repetira infinitamente, por quando a variável $i valer 10 a mesma não atendera mais a condição do comando.

Estruturas repetição do while

A estrutura `do…while` é quase idêntica a `while` com a diferença que o teste da variável é feito no final da estrutura, ou seja, o código é executado pelo menos uma vez.

Exemplo 3:

Nome do arquivo: aula06ex03.php

```php
<?php
   $i = 5;
   do {
        echo $i;
   } while ($i >= 10);
?>
```

Resultado:

```
5
```

Explicando o código:

Nesse caso a variável $i seria impressa ao menos uma vez, mesmo que o resultado da verificação seja false logo no começo e não voltar a executar o código.

Exemplo 4:

Nome do arquivo: aula06ex04.php

```
<?php
   $i=0;
   do {
      echo "i vale $i <br>";
      $i++;
   } while ($i<=10);
?>
```

Resultado:

```
i vale 0
i vale 1
i vale 2
i vale 3
i vale 4
i vale 5
i vale 6
i vale 7
i vale 8
i vale 9
i vale 10
```

Explicando o código:

No exemplo acima, temos a mesma estrutura que no exemplo anterior só que há um teste na variável de controle de repetição, e SE o resultado do mesmo for true, a mensagem se repetirá ENQUANTO i for menor ou igual a 10, conforme a condição do laço de repetição.

Estrutura de repetição foreach

O ***foreach*** construção fornece uma maneira fácil de iterar sobre matrizes. ***foreach*** funciona somente com arrays e objetos, e irá emitir um erro quando você tentar usá-lo em uma variável com um tipo de dados diferente ou uma variável não inicializada. Existem duas sintaxes:

Sintaxe:

```
foreach (expressao_array as $value)
    afirmação
foreach (expressao_array as $ key => $ value)
    afirmação
```

A primeira forma faz um loop sobre a matriz dada por ***expressao_array*** . Em cada iteração, o valor do elemento corrente é atribuído a ***$ valor*** e o ponteiro interno da matriz é avançado em uma posição (assim, na próxima iteração, você estará olhando para o próximo elemento).

Exemplo 5:

Nome do arquivo: aula06ex05.php

```
1.  <?php
2.  $vetor = array(1,2,3,4,5);
3.  foreach($vetor as $v)
4.  {
5.    echo "O valor atual do vetor é $v <br>";
6.  }
7.  ?>
```

Resultado:

O valor atual do vetor é 1
O valor atual do vetor é 2
O valor atual do vetor é 3
O valor atual do vetor é 4
O valor atual do vetor é 5

A segunda forma, adicionalmente, atribui a chave do elemento atual para a ***chave $*** variável em cada iteração.

Exemplo 6:

Nome do arquivo: aula06ex06.php

```
<?php
$vetor = array(1,2,3,4,5);
foreach ($vetor as $key => $value)
{
echo "Chave: $key – Valor: $value <br>";
```

```
}
?>
```

Resultado:

```
Chave: 0 – Valor: 1
Chave: 1 – Valor: 2
Chave: 2 – Valor: 3
Chave: 3 – Valor: 4
Chave: 4 – Valor: 5
```

Atividade

1 – O programa abaixo utiliza o comando While para realizar uma estrutura de repetição onde, enquanto a variável $i for menor que 20 as instruções são repetidas. O programa exibe na tela o valor da variável $i incrementada a cada laço de repetição.

```
<?php
$i=1;
while($i<=20){ // Enquanto $i menor ou igual a 20 faça..
   if($i>=3) // Se $i maior ou igual a 3 faça..
   {
          echo "$i <br>"; // Exibe o valor de $i

          if($i%2 == 0) // Se o resto da divisão de $i por 2 for igual a zero faça...
          {
                $i=$i+3; // Incrementa ao valor de $i mais 3
          }
          else
          {
                $i++;// Incrementa ao valor de $i mais 1
          }

   }else
   {
          $i++; // Incrementa ao valor de $i mais 1
   }

}
?>
```

Responda quais números são exibidos na tela pelo programa.

Resposta: _____

2 – O programa abaixo usa o comando FOR para fazer uma determinada quantidade de loops executando uma instrução em seu laço.

```
<?php
For($i=0;$i<50;$i=$i+5)
{
echo "O valor de I é: $i <br>";
}
?>
```

Analise o código e responda;

Quantas voltas (loops) o comando realizou?

R: _____

Quais números a variável $i exibiu na página?

R:_____

Aula 7 - Funções

Nesta aula vamos compreender como e quando utilizar funções no php, um recurso muito útil quando se deseja automatizar tarefas a serem realizadas no programa. Por exemplo, supondo que, em uma determinada parte do código de um programa, você tenha que realizar mais de um tarefa sequencialmente, você então teria que descrever o código de cada tarefa. Até ai, tudo bem, pois você tem mesmo que descrever o código das tarefas. Mas agora vamos ao problema, em outra parte do código, é necessário executar novamente, as mesmas tarefas e sem o uso de funções, você seria obrigado a digitar todos os códigos de cada tarefa outra vez, consequentemente demoraria muito mais tempo para concluir o desenvolvimento do programa, sem contar que, o mesmo, ficaria muito poluído, com códigos em demasia. Com o uso de Funções podemos evitar esta repetição de código.

Para que serve uma função?

As funções são métodos de economizar tempo e trabalho para ações que irão se repetir. Quando se tem, uma sequência de códigos de tarefas a serem executadas, mais de uma vez, para não ter que repetir os mesmos códigos a cada vez que precisarmos das ações desses códigos, digitamos o código somente uma vez, dentro de uma função criada com um nome que a identifique. Essa função poderá ser executada quantas vezes for necessário.

Sintaxe:

```
function <nome_da_função>( <parâmetros>,...){

Instruções
}
```

Criar funções é muito importante em sistemas grandes que requerem a execução de tarefas rotineiras como, por exemplo, fazer um mesmo calculo com números diferentes.

Exemplo 1:

Nome do arquivo: aula07ex01.php

```php
<?php

function calcula_o_dobro($numero) {
    // Faz o calculo do dobro do $numero
    $resultado = $numero * 2;
    return $resultado;
```

```
}
echo "O dobro de 3 é: " . calcula_o_dobro(3) . "<br>";
echo "O dobro de 4 é: " . calcula_o_dobro(4);
?>
```

Explicando o código:

Neste programa, uma função php é criada com a tarefa de calcular o dobro de um numero, fornecido por parâmetro.

A função com o nome calcula_o_dobro(<parâmetro>) é criada na linha 3, uma variável chamada $numero é definida como parâmetro para armazenar o numero a ser calculado.

```
function calcula_o_dobro($numero)
```

Na linha 5 do programa, o número armazenado na variável $numero é multiplicado por 2, e o resultado é atribuído na variável local $resultado.

```
$resultado = $numero * 2;
```

Na linha 6 utilizamos o comando return para retornar o valor armazenado na variável $resultado para a função.

```
return $resultado;
```

A função calcula_o_dobro($numero) é chamada (executada) duas vezes, calculando dois números diferentes passados por parâmetros nas linhas 9 e 11. Resumindo, a mesma função foi utilizada duas vezes para calcular o dobro de dois números diferentes passados por parâmetros.

Não há limite de argumentos que uma função pode receber, por exemplo, vamos fazer uma função que calcule um número elevado a outro:

```
function potenciacao($numero, $potencia) {
    return $numero ^ $potencia;
}

echo "2 elevado a 5ª potência é: " . potenciacao(2, 5);
// 2 elevado a 5ª potência é: 32
```

Atividades

1 - Crie uma função que some o valor de dois números e mostre o resultado.

Resposta:

Aula 8 – Passando informações para o PHP

Nesta aula vamos demonstrar como fazer o PHP interagir com o usuário através de formulários dinâmicos. Você lembra daqueles formulários que você criava em html e ao digitar informações neles, os dados digitados não iam pra lugar algum, te deixando frustrado. Pois bem, eles não enviavam os dados coletados pelo usuário ao digitar neles, porque não havia uma página dinâmica, ou seja, uma página PHP que receba estes dados para fazer alguma coisa com eles.

Entendendo o funcionamento de formulários

Para que serve um formulário? Um formulário serve para o usuário interagir de alguma forma com o site, enviando informações, escolhendo ou selecionando certas funções do site ou entrar em contato com o criador ou dono do site.

Como já foi dito, o formulário por si só não consegue enviar os dados coletados para lugar algum, é necessário existir uma página dinâmica após o envio dos dados. Mas esta página dinâmica só recebera os dados mediante configurações especificas no formulário.

Para que um formulário envie os dados corretamente para uma página PHP, duas propriedades devem ser configuradas. São elas, a propriedade `action` e `method`.

Exemplo 1:

Nome do arquivo: cadastro.php

```
<html>
<head>
   <title>Cadastro de Contato</title>
</head>
<body>
   <h1> Cadastro de contato</h1>
         <form name="cadastro" action="recebe-dados.php" method="post">
               Nome: <input type="text" name="nome"><br>
               E-mail:<input type="text" name="email"><br>
         <input type="submit" value="Envidar dados">
   </form>
</body>
</html>
```

Resultado:

A Tag <form>

No código do exemplo de um formulário acima, na linha 7 até a linha 10 iniciamos e finalizamos a existência de um formulário. A tag `<form>` é a principal tag para a criação de um formulário. Sempre que for fazer um formulário, todos os componentes do formulário devem ficar entre as tags `<form>` e`</form>`. Dentro da tag inicial <form>, configuramos determinadas propriedades para que o formulário envie os dados coletados para a página dinâmica de destino. Na linha 7 podemos observar estas propriedades.

São elas, name, action e method.

```
<form name="cadastro" action="recebe-dados.php" method="post">
```

Action

A propriedade action informa para onde os dados do formulário deve ser transportado. O valor desse atributo é o endereço, estático ou relativo, de onde está a página que vai tratar esse formulário.

Exemplo:

```
action="recebe-dados.php"
action="www.empresa.com.br/recebe-dados.php"
```

Method

A propriedade `method` especifica o método de envido dos dados contidos no formulário. Nesta propriedade podemos especificar os métodos post e get como valor. Explicaremos mais detalhadamente sobre estes dois tipos de métodos e suas diferenças logo a frente.

Se você não especificar o método (propriedade method), o padrão é o GET.

```
method="post"
method="get"
```

Name

Essa propriedade nomeia o formulário `<form>`, fazendo ele poder ser identificado por folhas de estilo e scripts (como JavaScript). Não tem influência no envio dos dados.

Dados de entrada nos formulários

A tag input

O formulário possui variados tag de entrada de dados como, caixas de texto, caixa de listagens, caixas de seleção e opções entre outros. A mais comum é a caixa de texto, a tag que representa esta caixa de texto. Sua sintaxe é: `<input type="text" name="nome">`, onde input é a tag, `type` é o tipo de entrada.

A propriedade `name` é a mais importante e não deve ser esquecida, pois esta propriedade, define o nome de cada item de dados transportado separadamente. Outro fator importante que devemos seguir sobre a propriedade name, é o nome dado como valor, ou seja, o nome de cada campo de entrada de dados, que deve seguir um padrão especifico, os mesmos, não devem conter espaços nem caracteres especiais, assim como do lado servidor (PHP) o mesmo é case-sensitive, tome cuidado ao definir nomes em caixa alta ou baixa (maiúscula ou minúsculas), pois o que for definido, do lado servidor tem que ser igual. Recomenda-se que seja tudo em minúsculo por padrão.

Como funciona o Método GET

O método GET utiliza a própria URL para enviar dados ao servidor, quando enviamos um formulário pelo método GET, o navegador pega as informações do formulário e coloca junto com a URL de onde o formulário vai ser enviado e envia, separando o endereço da URL dos dados do formulário por um “?” que definem o início da coleção de variáveis passadas pelo formulário com os

dados. O símbolo de = define o valor de cada variável e cada variável com seu valor são separadas uma da outra pelo símbolo &. Veja o exemplo dos dados do formulário de exemplo criado anteriormente passadas para a URL.

Exemplo:

http://www.seusite.com.br/recebe-dados.php?nome=Marcos&email=mmelo@yahoo.com

Vantagens e desvantagens ao utilizar o Método GET

O uso do método GET possui vantagens e desvantagens para quando for usado.

Vantagens

As vantagens em utilizar o método GET é quando queremos reaproveitar os dados coletados, podemos fazer isso porque os dados são expostos na URL após o nome da página php.

Outra vantagem deste método, é que podemos informar dados por links.

Desvantagens

Quanto ao que pode ser enviado, como os dados vão ser enviados pela URI, só poderão ser enviados caracteres aceitos na URL, se você quer enviar dados binários (como arquivo, imagens e outros), não poderá usar GET.

A quantidade de caracteres permitida por este método também é limitada, baseando-se pela quantidade disponível na caixa de endereços de URL no navegador que chega aos 2000 aproximadamente. Portanto se for enviar um formulário com muitas informações que ultrapasse esta cota, não é aconselhável usar este método.

Como funciona o Método POST

O método POST envia os dados colocando-os no corpo da mensagem. Ele deixa a URL separada dos dados que serão enviados e com isso podemos enviar qualquer tipo de dados por esse método.

Quando Utilizar o Método POST

O método POST envia praticamente todo o tipo de dados ou sempre que queremos enviar dados que não podem ser enviados pelo método GET, como arquivos.

A diferença é simples, sempre que for buscar ou apenas consultar alguma coisa, utilize GET e se for fazer alguma alteração com a requisição, envio de arquivo ou os dados forem muitos, utilize POST.

Recuperando os dados do lado do Servidor PHP

Agora que compreendemos como os dados são enviados pelo formulário, vamos entender como recupera-los no lado servidor.

Enviando os dados do formulário

No arquivo php cadastro.php que contém o formulário de envio, definimos que o mesmo enviara duas informações nome e email pelo método POST. Na propriedade `Action` está definida a página php "recebe_dados.php". Reveja o código do formulário de cadastro com estas propriedades destacadas.

```
<html>
<head>
<title> Cadastro de Contato</title>
</head>
<body>
<h1> Cadastro de contato</h1>
<form name="cadastro" action="recebe-dados.php" method="post">
Nome: <input type="text" name="nome"><br>
E-mail:<input type="text" name="email"><br>
<input type="submit" value="Envidar dados">
</form>
</body>
```

Recuperando os dados

Após o formulário ter enviado os dados para a página `recebe-dados.php`, precisamos fazer com que esta página receba estes dados. O PHP disponibiliza dois arrays superglobais, para envio pelos métodos GET e POST. São eles:

$_POST["<campo>"];

$_GET["<campo>"];

No formulário os nomes nas propriedades name, serão utilizados como chave associativa para armazenar os dados digitados pelo usuário. Por exemplo, estamos enviando pelo formulário dados nos campos **nome** e **email**, pelo método POST, então ficaria assim a capitação destes dados;

$_POST[“nome”];

$_POST[“email”];

É interessante atribuir seu valor a variáveis php para uma melhor organização. Ficaria assim:

```
$nome  = $_POST["nome"];
$email = $_POST["email"];
```

Veja abaixo o código complete do arquivo recebe-dados.php;

Exemplo 2:

Nome do arquivo: recebe-dados.php

```
<?php

if(isset($_POST["nome"]) or isset($_POST["email"])){

    $nome  = $_POST["nome"];
    $email = $_POST["email"];

    echo "<h2>Dados recebidos com sucesso!</h2>";
    echo "Usuário: $nome <br>";
    echo "E-Mail: $email";
}

?>
```

Resultado:

Explicando o código:

Antes de recuperar os dados e atribui-los a variáveis do php é necessário checar se eles chegaram na página. Para verificar isso, usamos o comando `isset()`, este comando simplesmente avalia se uma variável ou vetor foi definido no programa e retorna VERDADEIRO se existe ou FALSO se não.

Na linha 7 do código está checagem é feita nos dois campos dentro de uma condição IF. Se a condição retornar VERDADEIRO, então a instrução if será realizada recuperando e atribuindo os dados em variáveis PHP.

Atividades

1 – Crie a página php que contém um formulário de cadastro de visitante de um site com as informações abaixo, em seguida desenvolva uma página php que recebe estes dados e os exiba na tela.

Nome:

E-Mail:

Telefone:

Endereço:

Aula 9 – Banco de dados MySQL – Parte 1

Nesta aula vamos compreender como a linguagem PHP interage e manipula com banco de dados.

Sempre que você for desenvolver alguma aplicação PHP que tenha que armazenar e manipular informações variadas, é necessário ter um servidor de banco de dados que gerencie esses dados.

O PHP pode se conectar a variados tipos de banco de dados, neste livro em especial, usaremos o banco de dados MySQL, por ser bastante prático e de fácil entendimento que utiliza a linguagem SQL. O MySQL é um sistema de banco de dados muito eficiente de uso gratuito e é também um dos bancos de dados mais populares no mundo para o desenvolvimento de sistemas online e web sites.

Conexão com o servidor MySQL

Basicamente para o php se conectar a um servidor de banco de dados, o mesmo só precisa de três informações, são elas; endereço do servidor, usuário e senha. Mas estas três informações servem somente para se comunicar com o servidor. Um servidor de banco de dados pode conter várias bases de dados e precisamos selecionar o banco que desejamos manipular, por tanto precisamos também informar o nome do banco de dados.

O aluno aprendera através do PHP a conectar com o banco de dados, realiza consultas, cadastros de dados através de formulários dinâmicos, atualização e exclusão de dados.

Criando a base de dados MySQL

Antes de começarmos a trabalhar com banco de dados no PHP, precisamos inicialmente criar um banco de dados com tabelas já definidas e com alguns dados já inseridos para que possamos realizar a conexão.

Observações

Não detalharemos como configurar, executar ou manusear o servidor MySQL para a criação do banco de dados utilizado, supondo que o aluno já saiba fazer isso, tendo realizado o módulo de Banco de dados MySQL anterior a este. Caso o aluno tenha dúvidas ao realizar estes procedimentos, reveja as explicações do livro Banco de dados MySQL.

Atividade

Execute o script SQL abaixo para criar o banco de dados que iremos utilizar neste livro;

```
/* Cria o banco de dados dbLocadora */
create database dbLocadora;

/* Seleciona o banco dbLocadora */
use dblocadora;

/* Cria a tabela */
create table tbFilmes(
idFilme                 int           NOT NULL AUTO_INCREMENT PRIMARY KEY,
tituloFilme             varchar(100)  NOT NULL,
duracaoFilme            varchar(10)   NOT NULL,
valorLocacao            decimal(10,2) NOT NULL,
idCategoria             int           NOT NULL);

/* Cria a tabela tbCategorias */
create table tbCategorias(
idCategoria             int           NOT NULL AUTO_INCREMENT PRIMARY KEY,
nomeCategoria           varchar(30)   NOT NULL);

/* Insere dados na tabela tbFilmes */
INSERT INTO tbfilmes (
idFilme,tituloFilme,duracaoFilme,valorLocacao,idCategoria)
VALUES
(1,'Exterminador do Futuro','1:30',3.50,1),
(2,'Indiana Jones','2:00',3.00,2),
(3,'Rambo 2','2:30',3.00,1),
(4,'Star Wars','1:45',3.00,3),
(5,'Sexta-feira 13','2:10',3.00,4),
(6,'Jornada nas Estrelas','1:60',3.00,3),
(7,'O silêncio dos Inocentes','2:00',1.50,5),
(8,'Freddy krueger','2:00',1.50,4),
(9,'Comando para Matar','2:00',1.50,1),
(10,'Connan o Barbaro','1:90',1.50,1),
(11,'Missão: Impossivel','1:90',2.00,1),
(12,'O Chamado','1:80',2.00,4),
(13,'Hellboy','1:85',3.00,1),
(14,'O sexto sentido','1:47',2.00,5),
(15,'Cisne Negro','1:43',2.50,5),
(16,'O senhor dos Anéis','3:20',3.00,2);

/* Insere dados na tabela tbCategorias */
INSERT INTO tbCategorias(
idCategoria, nomeCategoria)
VALUES
(NULL , 'Ação'),
(NULL , 'Aventura'),
(NULL , 'Ficção Cientifica'),
(NULL , 'Terror'),
(NULL , 'Suspense');
```

Funções php de manipulação de banco de dados

O PHP possui funções internas de conexão e manipulação a vários bancos de dados, inclusive o MySQL. Essas funções funcionam por meio de extensões dos bancos de dados fornecidas para a linguagem PHP.

As extensões dos tipos de bancos de dados disponíveis a qual o PHP pode se comunicar, são arquivos DLL que ficam armazenadas dentro pasta de instalação do PHP em uma pasta de extensões chamada “ext”.

Ex: c:\php\ext

Nota:

O local desta pasta pode mudar dependendo do tipo de instalação do PHP.

Conectando ao Banco de dados dbLocadora

Criamos anteriormente um banco de dados chamado “dblocadora” e armazenamos alguns dados no mesmo, agora vamos criar uma página PHP para conectar a este banco e em seguida nas próximas aulas manipular os dados.

Abaixo temos a página php com o código comentado para se conectar ao banco de dados dblocadora e em seguida a explicação detalhada.

Exemplo 1:

Nome do arquivo: conexao.php

```
<?php
// ------ Dados para conexão com o servidor MySQL ---------------------------
$servidor       = "localhost";      // Endereço do servidor (DNS ou IP direto)
$usuario        = "root";           // Nome do usuário de acesso ao servidor
$senha                  = "";               // Senha do usuário de acesso ao servidor
$banco                  = "dblocadora";     // Nome do banco de dados que será manipulado

// ------ Executa a conexão com o servidor ----------------------------------
$conexao = mysqli_connect($servidor,$usuario,$senha,$banco) or die ("Não foi possível
conectar-se ao servidor. Erro: ". mysqli_connect_error());

// ------ Seleciona o banco de dados especifico no servidor -----------------
```

```
/* Verifica se houve conexão com o banco de dados no servidor e mostra na tela.*/
if(isset($conexao)){
echo "Banco de dados selecionado com sucesso.";
}
?>
```

Resultado:

Banco de dados selecionado com sucesso.

Explicando o código:

Nas linhas 3 até a 6, são definidas variáveis para atribuirmos as informações de conexão com o servidor. São 4 informações principais; endereço do servidor, usuário de acesso, senha de acesso e o nome do banco de dados a ser selecionado no servidor.

```
$servidor       = "localhost";
$usuario        = "root";
$senha          = "";
$banco          = "dblocadora";
```

Na linha 9 criamos uma variável chamada "`$conexao`" e atribuímos a ela a função de conexão com o banco de dados MySQL `mysqli_connect()`. Observe que passamos por parâmetros na rodem correta a seguir, os dados de conexão: Servidor, usuário e senha.

```
$conexao = mysqli_connect($servidor,$usuario,$senha,$banco) or die ("Não foi possível
conectar-se ao servidor. Erro: ". mysqli_connect_error());
```

Ainda na mesma linha 9 após a função `mysqli_connect()`, a função `die()` exibe uma mensagem de erro caso a conexão falhe e a função `mysqli_connect_error()` mostra o motivo do erro.

A variável $banco guarda o nome do banco que queremos selecionar. O resultado desta seleção, retorna para a variável `$conexao`, o número 1 verdadeiro (true) ou não retorna nada para caso falso.

Na linha 15 até a linha 18 uma pequena checagem sobre a seleção do banco de dados é feita através do comando condicional IF, onde se houver seleção do banco de dados a frase "Banco de dados selecionado com sucesso" é mostrada na página, indicando o sucesso da conexão e seleção do banco de dados. Se a seleção do banco de dados tiver êxito a variável $conexao será definida recebendo o

valor 1, e para descobrir se a variável foi definida usamos a função `isset(<variável>)` que retorna `true` (verdadeiro) se estiver definida ou `false` (falso) se não foi .

Selecionando registros na tabela do banco de dados

Agora que já aprendemos a conectar ao servidor de banco de dados MySQL e também a selecionar o banco de dados que queremos manipular, vamos então realizar algumas consultas ao banco de dados dblocadora, na tabela tbfilmes para retornar todos os registros na mesma. Já temos alguns registros de filmes cadastrados, inseridos já na criação do banco.

Realizando consultas SQL no banco de dados

Até agora compreendemos como conectar ao servidor MySQL utilizando a função `mysqli_connect()`. Mas agora, vamos realizar consultas SQL no banco de dados, mais especificamente na tabela do banco, onde fica armazenado os dados (registros). Podemos executar diversas consultas SQL no banco de dados, dentre elas, quadro principais, consultas de seleção de registros, de inserção de dados, de atualização e exclusão.

Para realizar consultas SQL diretamente no banco de dados MySQL pelo PHP, utilizamos a função `mysqli_query(<conexão>,<Consulta SQL>)`.

Vamos demonstrar como utilizar a função em um exemplo muito simples de código PHP onde desejamos exibir todos os dados da tabela tbFilmes. Portanto, vamos realizar uma consulta de seleção de registros no banco.

Importante!

É importante deixar claro que manipular banco de dados é necessário ter conhecimento em linguagem SQL para realizar as consultas ao banco. Caso o tenha dúvidas sobre a linguagem SQL, reveja o livro de Banco de dados MySQL que explica os detalhes da linguagem SQL.

Exemplo 2:

Nome do arquivo: lista-filmes.php

Observe o código abaixo;

```
<?php
// ------ Dados para conexão com o servidor MySQL ---------------------------
$servidor             = "localhost";        // Endereço do servidor (DNS ou IP direto)
$usuario              = "root";             // Nome do usuário de acesso ao servidor
$senha        = "neon321";          // Senha do usuário de acesso ao servidor
```

```
$banco          = "dblocadora";        // Nome do banco de dados que será manipulado

// ------ Executa a conexão com o servidor ---------------------------------
$conexao = mysqli_connect($servidor,$usuario,$senha,$banco) or die ("Não foi possível conectar-
se ao servidor. Erro: ". mysqli_connect_error());


echo "<h1> Selecionando registros usando FOR </h1>";

$sqlRegistros  = mysqli_query($conexao,"select * from tbfilmes") or die("Não foi possivel
consultar os filmes." . mysqli_error($conexao));

$dados          = mysqli_fetch_array($sqlRegistros);

$idFilme                = $dados["idFilme"];
$tituloFilme    = $dados["tituloFilme"];
$duracaoFilme   = $dados["duracaoFilme"];
$valorLocacao   = $dados["valorLocacao"];

echo "Código: $idFilme | Titulo: $tituloFilme | Duração: $duracaoFilme | Valor: $valorLocacao
<br>";

?>
```

Resultado:

Selecionando registros usando FOR

Código: 1 | Titulo: Exterminador do Futuro | Duração: 1:30 | Valor: 3.50

Explicando o código:

Antes de qualquer manipulação no banco de dados é preciso que haja sempre uma conexão com o servidor e uma seleção do banco de dados, portanto, nas linhas 2 a 6 está conexão é feita, como já tínhamos explicado antes.

Na linha 14 uma variável chamada $sqlRegistros é definida para atribuirmos a ela o resultado da consulta a ser realizada no banco de dados. A função `mysqli_query()` executa como parâmetro a consulta SQL "`select * from tbFilmes`".

```
$sqlRegistros  = mysqli_query($conexao,"select * from tbfilmes");
```

A variável `$sqlRegistros` passa ser um array multidimensional (Matriz) com todos os registros selecionados pela consulta SQL.

Na linha 9 definimos uma variável chamada $dados, esta variável servira como um array comum (vetor) para receber registros por linha através da função mysqli_fetch_array().

```
$dados    = mysqli_fetch_array($sqlRegistros);
```

Nas linhas 11 até 14 definimos variáveis para cada campo do registro que queremos exibir na página php.

```
$idFilme          = $dados["idFilme"];
$tituloFilme      = $dados["tituloFilme"];
$duracaoFilme     = $dados["duracaoFilme"];
$valorLocacao     = $dados["valorLocacao"];
```

A Função `mysqli_fetch_array()` é um array (vetor) que pode ser associativo ou de índices números armazenando valor de campo separadamente. Usamos a opção associativa para recuperar os valor de cada campo do vetor $dados[<índice associativo>]. Por exemplo, para armazenar o valor do campo idFilme na variável $idFilme digitamos;

```
$idFilme = $dados["idFilme"];
```

Onde o texto entre aspas “” é o índice associativo para recuperar o valor do campo idFilme.

É possível recuperar o mesmo valor pelo índice numérico, mas por este método você teria que descobrir a posição exata do campo no vetor.

Por exemplo; para recuperar o mesmo valor do campo idFilme ficaria assim;

```
$idFilme = $dados[0];
```

Por fim, na linha 16 os dados são exibidos na tela.

Exibindo todas as linhas de registros consultados

Você deve ter percebido que somente o primeiro registro foi exibido no programa acima, mas a tabela possui 6 registros de filmes cadastrados ao todo. Então porque que os outros 5 registros não foram exibidos? A resposta é simples, isso acontece porque a função mysqli_fetch_array() recupera somente um registro por vez num vetor, ou seja, a função extrai o registro completo da primeira linha (linha 0) e não pula para a segunda linha (linha 1) para recuperar o próximo registro e assim por diante.

Para exibir todos os registros, devemos adicionar ao código um processo de repetição, como por exemplo o comando FOR ou WHILE.

Acrescente ao código do programa anterior o código em destaque vermelho na posição correta conforme descrito abaixo;

```
<?php

$conexao = mysqli_connect("localhost","root","123"); // Executa a conexão com o
servidor.


   echo "<h1> Selecionando registros usando FOR </h1>";

   $sqlRegistros = mysqli_query($conexao,"select * from tbfilmes");
   $num_linhas = mysqli_num_rows($sqlRegistros); // Retorna para a variável o número
total de linhas contendo registros.

   for($i=0;$i<$num_linhas;$i++){

         $dados              = mysqli_fetch_array($sqlRegistros);

         $idFilme            = $dados["idFilme"];
         $tituloFilme        = $dados["tituloFilme"];
         $duracaoFilme       = $dados["duracaoFilme"];
         $valorLocacao       = $dados["valorLocacao"];

   echo "Código: $idFilme | Titulo: $tituloFilme | Duração: $duracaoFilme | Valor:
$valorLocacao <br>";
   }
?>
```

Resultado:

Código: 1 |Titulo: Exterminador do Futuro | Duração: 1:30 | Valor: 3.50
Código: 2 | Titulo: Indiana Jones | Duração: 2:00 | Valor: 3.00
Código: 3 | Titulo: Rambo 2 | Duração: 2:30 | Valor: 3.00
Código: 4 | Titulo: Star Wars | Duração: 1:45 | Valor: 3.00
Código: 5 | Titulo: Sexta-feira 13 | Duração: 2:10 | Valor: 3.00
Código: 6 | Titulo: Jornada nas Estrelas | Duração: 1:60 | Valor: 3.00

Explicando o código:

Antes de aplicarmos o comando FOR para navegarmos por todas as linhas de registros é preciso descobrir o número total de registros retornados pela consulta, isso é necessário porque o comando for precisa saber em seu parâmetro condicional quantas vezes ele executara o loop, ou seja, a quantidade de vezes que ele repetira a instrução será a quantidade de registros encontrados pela tabela.

Na linha 10 do código, é criada uma variável e é atribuído a ela o valor numérico da quantidade de registros retornados pela consulta recuperado pela função `mysqli_num_rows()`.

```
$num_linhas = mysqli_num_rows($sqlRegistros);
```

Na linha 12 é aplicado o comando FOR que repete a instrução sempre que a variável de inicialização $i for menor que o valor da variável `$num_linhas`.

Agora o a função `mysqli_fetch_array()` roda por todas as linhas da variável `$sqlRegistros`, ou seja, a variável $dados recebe os dados da primeira linha escreve na página, o comando FOR roda novamente fazendo a variável $dados receber os dados do próximo registro e assim por diante até que todos os registros sejam exibidos.

Atividades

1 – Com base no exercício desta aula, crie uma página php que mostre todos os registros da tabela tbCategorias do banco de dados dbLocadora. Nomeie o arquivo com o nome "**lista-categorias.php**".

Aula 10 - Banco de dados MySQL – Parte 2

Na aula anterior você aprendeu a exibir todos os registros de uma tabela em uma página PHP através da consulta SQL usando a instrução SELECT. Pois bem, nesta aula vamos criar outras consultas utilizando a instrução SELECT novamente, porém desta vez, vamos especificar na instrução, quais dados poderão ser exibidos, ou seja, em vez de mostrar todos os registros, mostraremos somente os que desejarmos.

Usando a função include()

A função `include()` do php é muito utilizada pelos desenvolvedores para facilitar o desenvolvimento e manipulação de um website. Sua funcionalidade é bem básica, o mesmo inclui, no conteúdo de uma página php, o script do código de outra página php. A vantagem em utilizar esta função é que determinados códigos que normalmente se repetem em várias páginas, podem ser criados em páginas únicas e incluídos em todas as páginas que necessitarem dos mesmos. Assim quando houver a alteração destes códigos, basta fazer uma vez somente.

Sintaxe:

```
include("<caminho / nome do arquivo>");
```

Exemplo:

Pagina de inclusão:

Nome do arquivo: arquivo-geral.php

```
<?php
echo "Olá eu sou o código do arquivo de inclusão.";
?>
```

Pagina 1:

Nome do arquivo: pagina1.php

```
<?php
echo "Esta página inclui em seu conteúdo o código do arquivo-geral.php <br>";
include("arquivo-geral.php");
?>
```

Resultado:

Esta página inclui em seu conteúdo o código do arquivo-geral.php

```
Olá eu sou o código do arquivo de inclusão
```

Caso ocorra algum problema na inclusão de um arquivo, será apresentado um **Warning** (aviso) que não foi possível incluir o arquivo e continuará a exibição da página normalmente sem a inclusão do arquivo.

A partir desta aula, vamos utilizar a função `include()` para incluir um código que se repetira muitas vezes nas páginas que criaremos.

Algumas das páginas a seguir terão a inclusão da página `conexao-db.php` criado na aula 9 para conectar ao servidor e selecionar o banco de dados `dblocadora`. Vamos conectar a este banco mais vezes, portanto, ter uma página exclusiva com o código desta conexão incluso em cada página, será muito útil.

Exibindo todos os registros de filmes em tabela

Compreendemos na aula anterior, como exibir todos os registros de filmes contidos na tabela tbFilmes, exibindo cada um por linhas usando o comando de repetição `FOR`, porém o layout não ficou muito bom, deixando os dados desalinhados. Geralmente, quando queremos exibir dados tabulares em grande quantidade, aplicamos estes dados dentro de tabelas, para um melhor alinhamento e visualização dos dados.

Colocar os dados tabulares dentro de uma tabela não tem nada de complicado, os dados continuam sendo exibidos através de interações em "loop" como o comando `FOR` que usamos como exemplo. A técnica para colocar estes dados dentro de uma tabela é saber qual parte da estrutura da tabela devemos colocar ou não colocar dentro do laço, pois tudo aquilo que você colocar dentro do laço se repetira, e nem todos os elementos da tabela devem se repetir. Por exemplo, a tag html que inicia e finaliza a criação de uma tabela é `<table>` e `</table>`, se as mesmas forem colocadas erroneamente dentro de um loop que roda 30 vezes, em vez de ser criado uma tabela somente, será criado 30 tabelas.

O correto é a tag `<table>` ser posicionada antes do laço e a tag de fechamento ser posicionada após o laço, deste jeito, somente as tag de conteúdo `<th><tr><td>` se repetira. Ou seja, as linhas e colunas onde os dados dinâmicos serão colocados, porém se a primeira linha da tabela, for o cabeçalho dos dados contidos em cada coluna, esta linha também deverá ficar acima do laço de repetição. O

mesmo acontece com a última linha, caso a mesma seja colocado informações de rodapé. Veja o exemplo a seguir que exemplifica esta técnica;

Exemplo 1:

Nome do arquivo: aula10ex01.php

```
<?php
    echo "<table border='1'>
            <tr><th>Cabeçalho</th><th>Cabeçalho</th></tr>";
// ------- Conteúdo do loop ------------------------------
    for($i=1;$i<=4;$i++){
    echo "<tr> <td>Conteúdo $i</td> <td>Conteúdo $i</td> </tr>";
    }
// ------- fim do conteúdo do loop -----------------------
    echo "<tr><th>Rodapé</th><th>Rodapé</th></tr>
    </table>";
?>
```

Resultado:

Cabeçalho	Cabeçalho
Conteúdo 1	Conteúdo 1
Conteúdo 2	Conteúdo 2
Conteúdo 3	Conteúdo 3
Conteúdo 4	Conteúdo 4
Rodapé	Rodapé

Com base no exemplo acima, criamos mais uma página PHP exibindo os dados de todos os filmes, mas desta vez, dentro de uma tabela.

Exemplo 2:

Nome do arquivo: aula10ex02.php

```
<?php
    include("conexao.php");// Inclusão do arquivo de conexão com o servidor e seleção do banco.
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Lista de Filmes - Locadora MC</title>
</head>
<body>

<h1>LISTA DE FILMES</h1>
<?php
    $sqlRegistros = mysqli_query($conexao,"select * from tbfilmes");
    $num_linhas = mysqli_num_rows($sqlRegistros);
    // ----------Inicio da tabela -----------------------------------
    echo "<table border='1'>";
    // ---------- Linhas de cabeçalhos da tabela ---------------------
```

```
    echo "<tr><th>Código</th><th>Título</th><th>Duração</th><th>Valor da
Locação</th><th>Cod.Categoria</th></tr>";
    // ------------------------------------------------------------
    for($i;$i<$num_linhas;$i++){

            $dados          = mysqli_fetch_array($sqlRegistros);

            $idFilme        = $dados["idFilme"];
            $tituloFilme    = $dados["tituloFilme"];
            $duracaoFilme   = $dados["duracaoFilme"];
            $valorLocacao   = $dados["valorLocacao"];
            $idCategoria    = $dados["idCategoria"];
    // ----------- linhas da tabela contendo os registros -----------
    echo "
        <tr>
        <td> $idFilme</td>
        <td>$tituloFilme</td>
        <td>$duracaoFilme</td>
        <td>$valorLocacao</td>
        <td>$idCategoria</td>
        </tr>";
    // ------------------------------------------------------------
    }
    echo "</table>"; // fim da tabela
?>
</body>
</html>
```

Resultado:

Código	Título	Duração	Valor da Locação	Cod.Categoria
3	Rambo 2	2:30	3.00	1
4	Star Wars	1:45	3.00	3
5	Sexta-feira 13	2:10	3.00	4
6	Jornada nas Estrelas	1:60	3.00	3
7	O sicêncio dos Inocentes	2:00	1.50	5
8	Freddy krueger	2:00	1.50	4
9	Comando para Matar	2:00	1.50	1
10	Connan o Barbaro	1:90	1.50	1
11	Missão: Impossivel	1:90	2.00	1
12	O Chamado	1:80	2.00	4
13	Hellboy	1:85	3.00	1

Exibindo os dados de tabelas relacionadas

Vamos continuar mostrando dados da tabela tbFilmes, porém desta vez, vamos exibir campos da tabela relacionada tbCategorias. Você deve ter reparado no exercício anterior que, a coluna que informa a categoria do filme, exibe somente o código da categoria, ficando vaga a informação para o usuário, já que o mesmo não tem a obrigação de ficar decorando códigos de categorias de filmes.

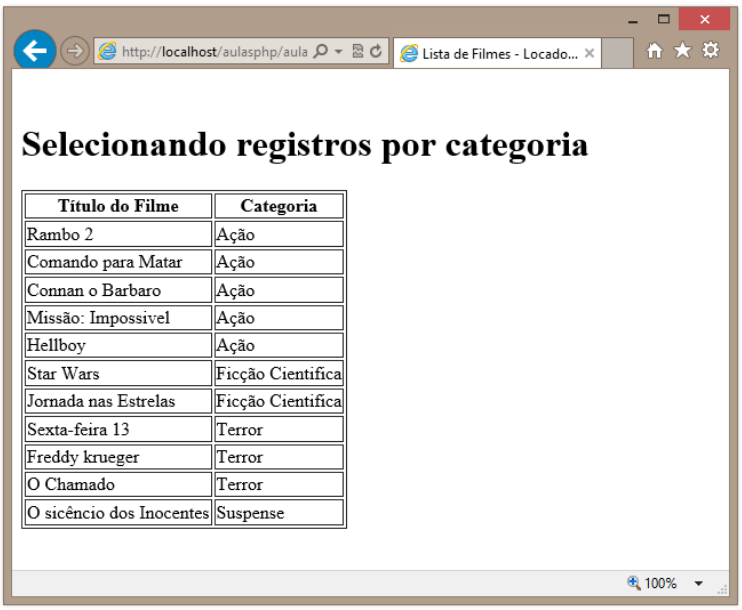
A solução é, exibir as informações relacionadas ao código da categoria que está presente nas duas tabelas. As informações complementares relacionadas entre as duas tabelas são obtidas através da consulta SQL com a inclusão do parâmetro `INNER JOIN` que tem esta função de selecionar as informações relacionadas. Observe o código da página php a seguir, que une estas informações;

Exemplo 3:

Nome do arquivo: aula10ex03.php

```
<?php
    include("conexao.php");
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Lista de Filmes - Locadora MC</title>
</head>
<body>

<h1>Selecionando registros por categoria</h1>
<?php
    $sqlRegistros = mysqli_query($conexao,"select tituloFilme, nomeCategoria from tbfilmes inner join
tbcategorias on tbfilmes.idCategoria = tbCategorias.idCategoria");
    $num_linhas = mysqli_num_rows($sqlRegistros);
    echo "<table border='1'>";
    echo "<tr><th>Título do Filme</th><th>Categoria</th></tr>";
    for($i;$i<$num_linhas;$i++){

            $dados          = mysqli_fetch_array($sqlRegistros);

            $tituloFilme    = $dados["tituloFilme"];
            $nomeCategoria  = $dados["nomeCategoria"];

    echo "<tr> <td>$tituloFilme</td><td>$nomeCategoria</tr>";
    }
    echo "</table>";
?>
</body>
</html>
```

Resultado:



Explicando o código:

O código desta página é bastante semelhante ao do exercício anterior, mas com algumas modificações para exibir somente as informações que queremos.

Na linha 14 é executada a consulta SQL que extrai os dados na tabela relacionada tbCategoria.

```
$sqlRegistros = mysqli_query($conexao,"select tituloFilme, nomeCategoria from tbfilmes inner join tbcategorias on tbfilmes.idCategoria = tbCategorias.idCategoria");
```

Repare que após a instrução `SELECT` não usamos mais o caractere * (asterisco) cuja a função é mostrar todos os campos, desta vez, descrevemos exatamente quais campos devem ser selecionados, ou seja, tituloFilme e nomeCategoria separados por vírgula (,). Observe também que, cada campo em questão, pertence a uma das tabelas, o campo tituloFilme pertence a tabela tbFilmes e nomeCategoria pertence a tabela tbCategorias. A instrução INNER JOIN junta as informações relacionadas entre as tabelas pelo campo idCategoria.

Observações

Como já falamos, não vamos detalhar o uso de consultas SQL neste livro. Qualquer dúvida sobre consultas SQL e usabilidade de banco de dados relacionais, reveja o livro de banco de dados MySQL.

Exibindo todos os filmes de uma categoria especifica

Após termos demonstrado como aplicar a instrução JOIN no exemplo anterior, onde conseguimos exibir as informações da tabela relacionada tbCategoria, vamos aplicar em uma nova página php, filtros na consulta SQL, para que somente determinados dados sejam visualizados no navegador, por exemplo, exibir filmes, cuja a categoria seja "Ação". O código neste novo arquivo é basicamente o mesmo do anterior, porém com uma modificação na consulta SQL. Usaremos a clausula WHERE para impor uma condição para os resultados que deveram ser retornados na consulta.

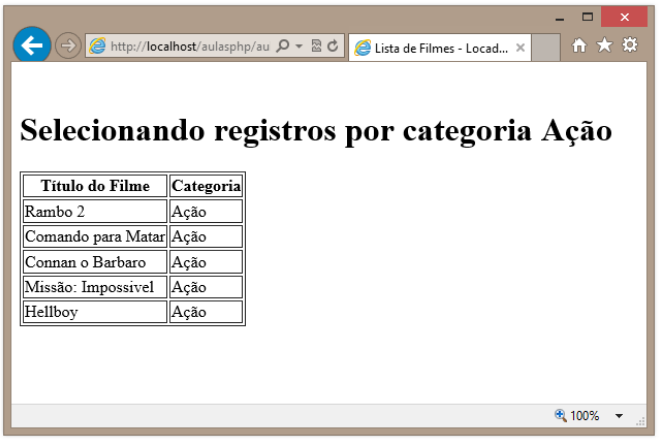
Observe o código abaixo onde abordamos o uso da clausula Where;

Exemplo 4:

Nome do arquivo: aula10ex04.php

```
<?php
    include("conexao.php");
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Lista de Filmes - Locadora MC</title>
</head>
<body>

<h1>Selecionando registros por categoria Ação</h1>
<?php
    $sqlRegistros = mysqli_query($conexao,"select tituloFilme, nomeCategoria from tbfilmes inner join
tbcategorias on tbfilmes.idCategoria = tbCategorias.idCategoria where tbfilmes.idCategoria = 1");
    $num_linhas = mysqli_num_rows($sqlRegistros);
    echo "<table border='1'>";
    echo "<tr><th>Título do Filme</th><th>Categoria</th></tr>";
    for($i;$i<$num_linhas;$i++){

            $dados          = mysqli_fetch_array($sqlRegistros);

            $tituloFilme    = $dados["tituloFilme"];
            $nomeCategoria  = $dados["nomeCategoria"];

    echo "<tr> <td>$tituloFilme</td><td>$nomeCategoria</tr>";
    }
    echo "</table>";
?>
</body>
</html>
```

Resultado:



Explicando o código:

Com já comentamos, o código descrito neste tópico é o mesmo do exercício anterior. Informações relacionadas entre duas tabelas são consultadas, a única modificação foi aplicada na linha 14, onde na consulta SQL acrescentamos a instrução Where (onde), para filtrar somente registros cujo o campo idCategoria seja igual a 1, que é o código da categoria "Ação".

```
$sqlRegistros = mysqli_query($conexao,"select tituloFilme, nomeCategoria from tbfilmes inner join tbcategorias on tbfilmes.idCategoria = tbCategorias.idCategoria where tbfilmes.idCategoria = 1");
```

Criando formulários de pesquisa por título do filme

Um recurso muito utilizado pelos programados de sistemas é a consulta dinâmica por dados no banco de dados. Neste tipo de consulta, os dados pesquisados não são definidos no código da página, os mesmos são definidos pelo usuário através de formulários. A cada nova informação fornecida pelo usuário, a página fornece um novo resultado.

A vários tipos de consultas SQL que retornam resultados variados, entre os métodos possíveis, o mais comum para pesquisar por String, é usando o operador Like, associado ao comando Where.

Quando não se conhece o valor exato a procurar, mas temos uma ideia aproximada, podemos utilizar o operador LIKE que permite selecionar linhas que concordem com um dado padrão de

caracteres. A cadeia de caracteres usada como padrão de pesquisa, pode utilizar dois símbolos especiais:

SIMBOLO	REPRESENTA
%	Qualquer cadeia com nenhum ou vários caracteres
_	Um caractere qualquer

```
SELECT <campo> FROM <nome_da_tabela> WHERE <campo> LIKE '%<string>%';
```

Exemplo de consulta no Workbench:

```
SELECT * FROM tbFilmes WHERE tituloFilme LIKE '%inocentes%';
```

Resultado:

idFilme	tituloFilme	duracaoFilme	valorLocacao	idCategoria
7	O sicêncio dos Inocentes	2:00	1.50	5
NULL	NULL	NULL	NULL	NULL

Perceba que a única informação que o usuário digitou, é uma parte do texto do nome completo do texto, isto porque a informação que o usuário informou, está antes e depois do símbolo “%” que procura dentro de uma String, em qualquer parte da mesma.

Agora observe o exercício abaixo, onde criamos um formulário para que o usuário informe o nome do título do filme que deseja pesquisar. O usuário poderá digitar partes do texto ou até mesmo.

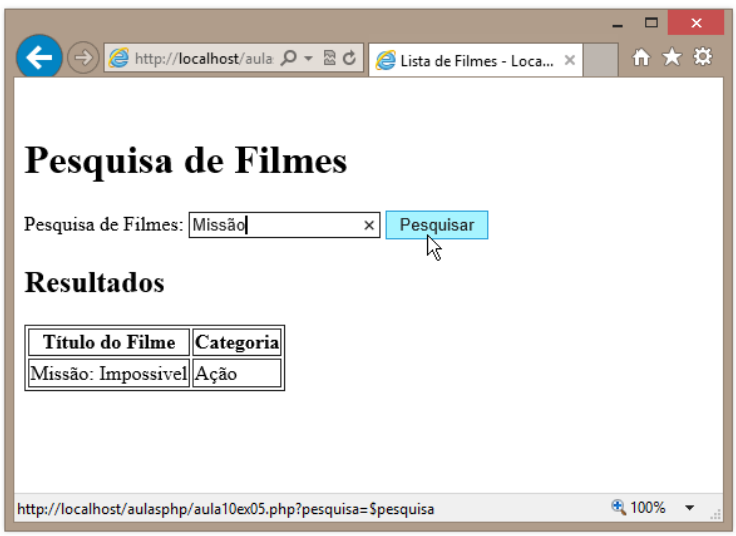
Exemplo 5:

Nome do arquivo: pesquisa-filmes.php

```
<?php
   include("conexao.php");
   $pesquisa = $_GET["pesquisa"];
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Lista de Filmes - Locadora MC</title>
</head>
<body>

<h1>Pesquisa de Filmes</h1>
<form name="pesquisa" action="pesquisa-filmes.php" method="get">
<label>Pesquisa de Filmes:</label> <input type="text" name="pesquisa">
<input type="submit" value="Pesquisar">
```

```
</form>
<h2>Resultados</h2>
<?php
   $sqlRegistros = mysqli_query($conexao,"select tituloFilme, nomeCategoria from
tbfilmes inner join tbcategorias on tbfilmes.idCategoria = tbCategorias.idCategoria
where tituloFilme like '%$pesquisa%'");
   $num_linhas = mysqli_num_rows($sqlRegistros);
   echo "<table border='1'>";
   echo "<tr><th>Título do Filme</th><th>Categoria</th></tr>";
   for($i;$i<$num_linhas;$i++){

         $dados              = mysqli_fetch_array($sqlRegistros);

         $tituloFilme        = $dados["tituloFilme"];
         $nomeCategoria      = $dados["nomeCategoria"];

   echo "<tr> <td>$tituloFilme</td><td>$nomeCategoria</tr>";
   }
   echo "</table>";
?>
</body>
</html>
```

Resultado:



Explicando o código:

Neste código da linha 14 até 17, através de um formulário, o usuário informa o que deseja pesquisar. Perceba que os dados digitados são enviados na propriedade `action` para a própria página, assim um novo resultado é mostrado.

```
<form name="pesquisa" action="pesquisa-filmes.php" method="get">
<label>Pesquisa de Filmes:</label> <input type="text" name="pesquisa">
<input type="submit" value="Pesquisar">
</form>
```

Após os dados serem enviados, esta informação é recuperada pela variável $pesquisa na linha 3, este valor é adicionado na consulta SQL, após a instrução LIKE na linha 20.

Aula 11 - Banco de dados MySQL – Parte 3

Na aula anterior, já aprendemos a conectar ao servidor, selecionar o banco de dados e até mostramos como retornar dados do banco de dados na página PHP. Nesta aula, vamos manipular mais ainda o banco de dado MySQL. Desenvolveremos um pequeno e simples sistema de cadastro de filmes em uma locadora, para compreendermos como gravar, atualizar e excluir dados no banco de dados, através de páginas php.

Organizando os arquivos

Bom, já que vamos criar um sistema, é interessante criar uma pasta raiz exclusiva para ele, portanto crie uma pasta com o nome "vídeo_locadora" para ser a pasta raiz do site.

Observações

Lembre-se, a pasta deve ser criada na área de criação de sites do servidor web, ou seja, se o servidor de testes for o XAMPP, deve-se criar dentro da pasta "**htdocs**".

Arquivos de inclusão

Neste sistema utilizaremos 2 (dois) arquivos PHP de inclusão nas demais páginas.

conexao.php

O Primeiro arquivo é o responsável por realizar a conexão com o servidor de banco de dados. Este arquivo de inclusão será adicionado em todas as páginas que se conectam ao banco de dados.

Nome do arquivo: conexao.php

```
<?php
   $conexao = mysqli_connect("localhost","root","123"); // Executa a conexão com o
servidor (Servidor, usuário, senha).
?>
```

menu.php

O arquivo menu.php é como o nome sugere, um MENU, que também estará presente em todas as páginas, para servir de navegação entre as páginas.

Nome do arquivo: menu.php

```
<p><a href="index.php">Home</a> | <a href="lista-filmes.php">Lista de Filmes</a> | <a
href="cadastrar-filme.php">Adicionar Filme</a></p>
```

Home

A primeira página do site do sistema, a index.php, basicamente possui um código php para incluir o arquivo menu.php e uma imagem personalizada logo abaixo.

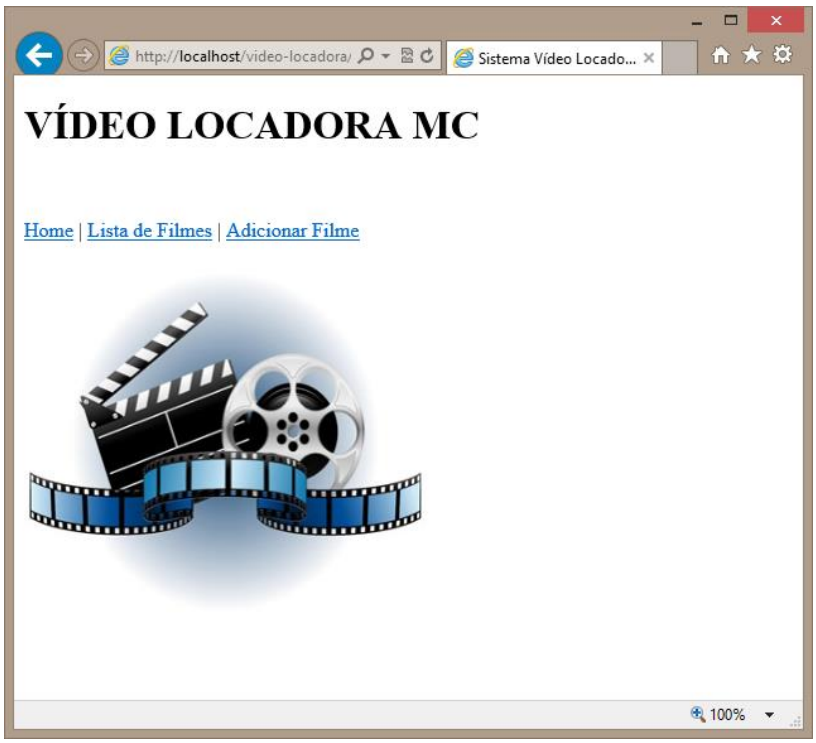
Observações:

Você pode usar qualquer imagem, da internet por exemplo mas renomeia com o nome “filme.jpg”

Através da página principal podemos acessar as demais páginas do sistema.

Nome do arquivo: index.php

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Sistema Vídeo Locadora MC</title>
</head>

<body>
<h1>VÍDEO LOCADORA MC</h1>

<?php
include("menu.php");
?>
<p><img src="filme.jpg" width="319" height="289"  alt=""/></p>
</body>
</html>
```



Listando dados

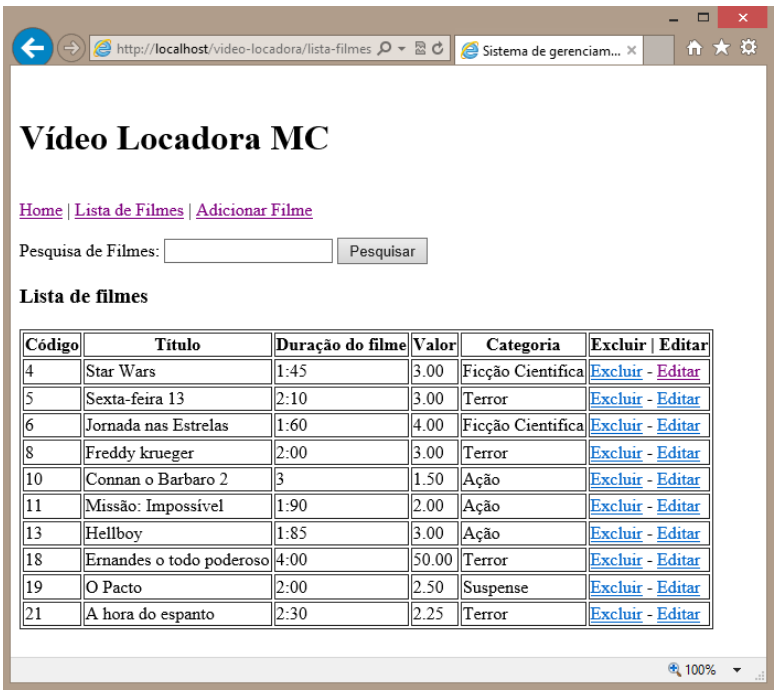
A página lista-filmes.php possui um sistema de pesquisa que retorna o resultado pesquisado. Já criamos esta página anteriormente, porem desta vez, adicionamos a cada linha de registro, os links Excluir e Editar, justamente para que possa ser possível excluir ou editar o registro em questão.

Nome do arquivo: lista-filmes.php

```
<?php
include("conexao.php");// include do arquivo de conexão com o banco de dados
// --- Verifica se dados da pesquisa existe, e se existir, atribui o mesmo na variável
$pesquisa
if(isset($_GET["pesquisa"])){
$pesquisa = $_GET["pesquisa"];
}
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Sistema de gerenciamento de filmes</title>
</head>
<body>
```

```
<h1>Vídeo Locadora MC</h1>
<?php
include("menu.php");// Inclui o arquivo do menu
?>
<form name="pesquisa" action="lista-filmes.php" method="get">
<label>Pesquisa de Filmes:</label> <input type="text" name="pesquisa">
<input type="submit" value="Pesquisar">
</form>

<h3>Lista de filmes</h3>
<table border="1">
<tr>
   <th>Código</th><th>Título</th><th>Duração do
filme</th><th>Valor</th><th>Categoria</th><th> Excluir | Editar</th>
</tr>
<?php
//------------- Consulta pesquisa por títulos ------------------------------------
   $sqlRegistros = mysqli_query($conexao,"select idFilme,tituloFilme, duracaoFilme,
valorLocacao, nomeCategoria from tbfilmes inner join tbcategorias on
tbfilmes.idCategoria = tbCategorias.idCategoria where tituloFilme like '%$pesquisa%'
order by idFilme");
   $num_linhas = mysqli_num_rows($sqlRegistros);

   for($i;$i<$num_linhas;$i++){

         $dados              = mysqli_fetch_array($sqlRegistros);

         $idFilme            = $dados["idFilme"];
         $tituloFilme        = $dados["tituloFilme"];
         $duracaoFilme       = $dados["duracaoFilme"];
         $valorLocacao       = $dados["valorLocacao"];
         $nomeCategoria      = $dados["nomeCategoria"];
?>
   <tr>
   <td><?php echo $idFilme;?> </td><td><?php echo $tituloFilme;?> </td><td><?php echo
$duracaoFilme;?> </td><td><?php echo $valorLocacao;?> </td><td> <?php echo
$nomeCategoria;?> </td>
    <td>
   <a href="excluir-filme.php?idFilme=<?php echo $idFilme;?>">Excluir</a> -
   <a href="editar-filme.php?idFilme=<?php echo $idFilme;?>">Editar</a>
       </td>
       </tr>
   <?php
         }
   ?>
</body>
</html>
```

Resultado:



Código	Título	Duração do filme	Valor	Categoria	Excluir | Editar
4	Star Wars	1:45	3.00	Ficção Cientifica	Excluir - Editar
5	Sexta-feira 13	2:10	3.00	Terror	Excluir - Editar
6	Jornada nas Estrelas	1:60	4.00	Ficção Cientifica	Excluir - Editar
8	Freddy krueger	2:00	3.00	Terror	Excluir - Editar
10	Connan o Barbaro 2	3	1.50	Ação	Excluir - Editar
11	Missão: Impossível	1:90	2.00	Ação	Excluir - Editar
13	Hellboy	1:85	3.00	Ação	Excluir - Editar
18	Ernandes o todo poderoso	4:00	50.00	Terror	Excluir - Editar
19	O Pacto	2:00	2.50	Suspense	Excluir - Editar
21	A hora do espanto	2:30	2.25	Terror	Excluir - Editar

Explicando o código:

Observando o resultado da página no navegador, podemos verificar que o menu de navegação está presente, graças ao código `include("menu.php")` na linha 17.

Nas linhas 19 à 22, foi adicionado um formulário, onde é possível inserir dados de pesquisa por título do filme. Os dados do formulário são enviados pelo método "get" por uma variável especificada através da propriedade `name="pesquisa"` a ser recuperado seu valor no servidor.

Na propriedade Action está definido o endereço da página de destino dos dados enviados pelo método GET.

```
<form name="pesquisa" action="lista-filmes.php" method="get">
<label>Pesquisa de Filmes:</label> <input type="text" name="pesquisa">
<input type="submit" value="Pesquisar">
</form>
```

As configurações deste formulário, mais importantes são justamente os links "editar" e "excluir" nas linhas 47 e 48. No endereço dos links parâmetros da variável $idFilme são enviados pelo método GET à página php de destino que recebera estas informações.

```
<a href="excluir-registro.php?idFilme=<?php echo $idFilme;?>">Excluir</a> -
<a href="editar-registro.php?idFilme=<?php echo $idFilme;?>">Editar</a>
```

Gravando dados enviados por formulários

Já vimos na aula 8 que, para o usuário enviar dados ao banco de dados através de páginas PHP, devesse utilizar formulários, pois bem, vamos criar a página php com este formulário para cadastrar os filmes da locadora no banco de dados na tabela tbFilmes, porém após o envio dos dados pelo formulário, é necessário também criar a página php receptora, responsável pelo processamento dos dados, ou seja, a página que gravara os dados no servidor de banco de dados.

Formulário de cadastro

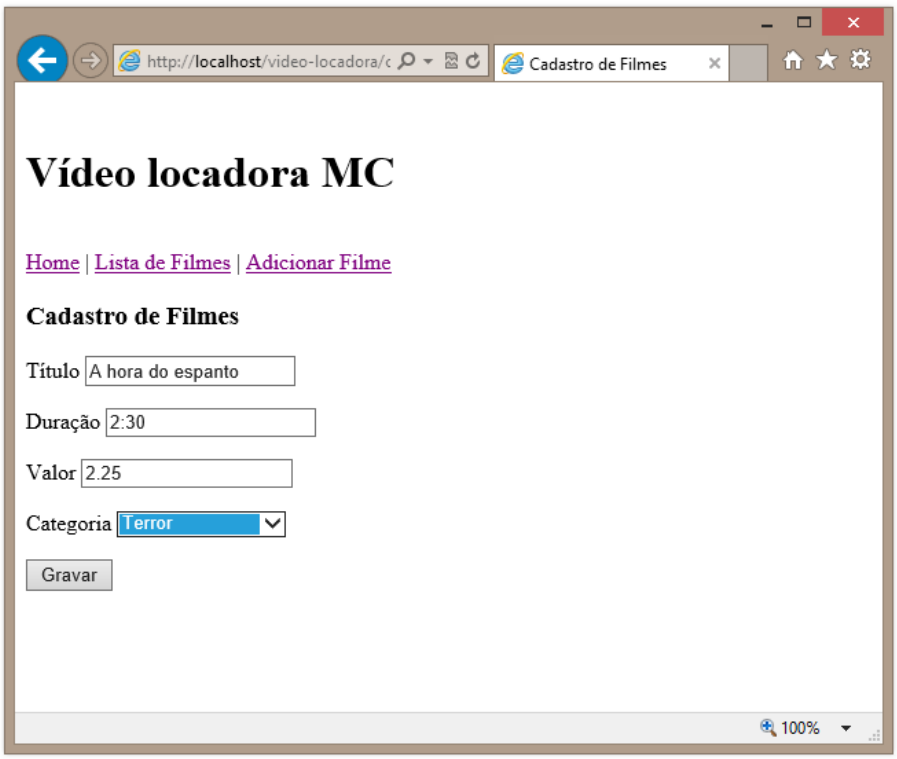
O arquivo que contém o formulário para inserir os dados pelo usuário é um arquivo muito simples, se o mesmo tem o objetivo de receber os dados e envia-los para o destino, não há a necessidade de programação php. Mas este formulário que vamos desenvolver, o campo categoria recebe informações dinâmicas relacionadas vindas da tabela tbCategoria no servidor de dados, portanto vamos aplicar um código php neste campo para carregar a lista de categorias. Acompanhe o código completo desta página e a explicação logo a seguir.

Nome do arquivo: cadastrar-filme.php

```
<?php
include("conexao.php");
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Cadastro de Filmes</title>
</head>

<body>
<h1>Vídeo locadora MC</h1>
<?php
include("menu.php");
?>

<h3>Cadastro de Filmes</h3>
<form name="cadFilmes" action="gravar-filme.php" method="post">
<p><label>Título </label><input type="text" name="tituloFilme"></p>
<p><label>Duração </label><input type="text" name="duracaoFilme"></p>
<p><label>Valor </label><input type="text" name="valorLocacao"></p>
<p><label>Categoria </label>
<select name="idCategoria">
<?php
//Carrega a lista de Categorias -------------------------------
$rsCategoria = mysqli_query($conexao,"select * from tbCategorias");
$linhas = mysqli_num_rows($rsCategoria);
```

```
for($i=0;$i<$linhas;$i++){
$dados = mysqli_fetch_array($rsCategoria);
$idCategoria = $dados["idCategoria"];
$nomeCategoria = $dados["nomeCategoria"];
echo "<option value='$idCategoria'>$nomeCategoria</option>";
}
//Fim da lista de categorias ------------------------------------
?>
</select>
</p>
<p><input type="submit" value="Gravar"></p>
</form>
</body>
</html>
```

Resultado:



Explicando o código:

Na linha 2 aplicamos um código php com a função `include()` para incluir o código de conexão com o banco de dados, pois vamos precisar consultar o banco para fornecer a lista de categorias relacionadas na tabela tbCategorias.

Esta página possui basicamente um formulário html com os campos de textos respectivos aos campos contidos no banco de dados MySQL na tabela tbFilmes. Após o usuário digitar os dados de um novo filme, ao pressionar o botão “Gravar” os dados são enviados para a página gravar-filme.php definida na propriedade action do formulário.

No banco de dados dbLocadora na tabela tbFilmes, o campo idCategoria recebe um valor numérico que corresponde ao código da categoria do filme que está relacionado na tabela tbCategoria, porem na hora de cadastrar o usuário não é obrigado a saber o código de cada categoria, assim uma saída pra resolver esta situação é criar uma lista dinâmica que consulta o banco de dados e retorna o código e o nome de cada categoria.

O elemento de entrada de dados de formulário responsável por gerar a lista de opções de itens para seleção é:

```
<select>
   <option value="1">Item 1</option>
   <option value="2">Item 2</option>
</select>
```

Cada item da lista aparece entra <option> e </option>, mas o valor que é enviado para o banco é o que está na propriedade value.

Utilizamos a estrutura de loop FOR para consultar a tabela tbCategorias e retornar a lista de categorias.

```
<p><label>Categoria </label>
<select name="idCategoria">
<?php
//Carrega a lista de Categorias --------------------------------
   $rsCategoria = mysqli_query($conexao,"select * from tbCategorias");
   $linhas = mysqli_num_rows($rsCategoria);
   for($i=0;$i<$linhas;$i++){
   $dados = mysqli_fetch_array($rsCategoria);
   $idCategoria = $dados["idCategoria"];
   $nomeCategoria = $dados["nomeCategoria"];
   echo "<option value='$idCategoria'>$nomeCategoria</option>";
}
//Fim da lista de categorias -----------------------------------
?>
</select>
</p>
```

Gravando os dados enviados pelo formulário

Após enviar os dados pelo formulário da página cadastrar-filme.php, precisamos criar a página que recebe estes dados e os grava no banco de dados na tabela tbFilmes.

A tarefa deste arquivo php é muito simples, o mesmo recebe os dados, ou seja, recupera os dados vindos do formulário de cadastro de filmes, executa uma consulta SQL de inserção de dados na tabela e em seguida, retorna para a página que lista todos os filmes inclusive o que foi inserido. Veja abaixo o código deste arquivo;

Nome do arquivo: gravar-filme.php

```
<?php
include("conexao.php");

    $tituloFilme          = $_POST["tituloFilme"];
    $duracaoFilme         = $_POST["duracaoFilme"];
    $valorLocacao         = $_POST["valorLocacao"];
    $idCategoria          = $_POST["idCategoria"];
    $sqlGravarFilme       = mysqli_query($conexao,"insert into
tbFilmes(tituloFilme,duracaoFilme,valorLocacao,idCategoria)
value('$tituloFilme','$duracaoFilme','$valorLocacao','$idCategoria')") or die("Erro ao gravar
registro. " . mysqli_error($conexao));
    header('Location: lista-filmes.php');
?>
```

Explicando o código:

Na linha 2 é feita a conexão com o servidor de banco de dados MySQL e a seleção do banco dblocadora.

Nas 4 até 7 são criadas as variáveis correspondentes a cada campo de um registro na tabela. A cada variável é atribuído um valor recuperado pelas variáveis superglobais $_GET[<campo>].

Por fim, na linha 8 os dados são inseridos no banco através da consulta SQL usando a instrução INSERT.

Insert

A instrução INSERT não tem muita complexidade, basicamente é um comando padrão de inserção de novos valores em uma tabela de banco de dados.

Sintaxe;

```
INSERT INTO <nome_da_tabela> (<campo1>,<campo2>,....)
VALUES( <valor_campo1> , <valor_campo2>,.....)
```

Exemplo:

```
INSERT INTO tbCurso (nomeCurso, valorCurso)VALUES('Português' , 100.00)
```

Atualizando dados no banco de dados

Após cadastrar um novo filme, o mesmo constara na lista de filmes cadastrados, porém se acidentalmente digitar alguma informação errada do filme, como o título por exemplo ou até mesmo o valor da locação, será necessário editar os dados do registro desse filme em questão.

O processo de edição ou melhor, atualização das informações de um registro no banco de dados através de uma página dinâmica PHP é bem semelhante ao de inserção (Gravação) de dados. Ambos necessitam de uma página com um formulário para que o usuário possa fornecer os dados que deseja inserir ou atualizar, porém enquanto o formulário de cadastro grava novas informações, o outro atualiza informações já existentes.

Para atualizar as informações de um registro precisaremos de dois arquivos, um para recuperar em um formulário os dados que serão modificados pelo usuário e em seguida ao enviar os dados novos, o arquivo que recebe estes novos dados e os atualiza no banco de dados.

Veja a seguir os dois arquivos descritos e devidamente explicados;

Formulário de edição

Você deve ter reparado no arquivo lista-filmes.php que em cada linha de registro de filme no final da linha, há dois links; editar e excluir. Ambos direcionam para uma página especifica, o link "Editar" direciona para a página editar-filme.php, porém após o nome da página um parâmetro é adicionado que informa a variável idFilme que guarda o código do filme que desejamos atualizar. Este código é recuperado na página editar-filme.php para consulta-lo no banco para retornar as informações do filme no formulário de edição.

Exemplo:

```
<a href="editar-filme.php?idFilme=<?php echo $idFilme;?>">Editar</a>
```

Nome do arquivo: editar-filme.php

```
<?php
   include("conexao.php");
   $idFilme            = $_GET["idFilme"];
   $sqlRegistros       = mysqli_query($conexao,"select tituloFilme, duracaoFilme,
valorLocacao, tbFilmes.idCategoria, nomeCategoria from tbFilmes inner join tbCategorias
on tbFilmes.idCategoria = tbCategorias.idCategoria where idFilme=$idFilme") or die("Erro
na execusão da consulta" . mysqli_error($conexao));

   $dados              = mysqli_fetch_array($sqlRegistros);

   $tituloFilme        = $dados["tituloFilme"];
   $duracaoFilme       = $dados["duracaoFilme"];
   $valorLocacao       = $dados["valorLocacao"];
   $idCategoria        = $dados["idCategoria"];
   $nomeCategoria      = $dados["nomeCategoria"];
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Gerencia Registro</title>
</head>

<body>
<h1>Vídeo Locadora MC</h1>
<?php
include("menu.php");
?>
<form name="form1" action="atualizar-filme.php?" method="post">
  <input type="hidden" name="idFilme" value="<?php echo $idFilme;?>">
  <p>
    <label>Tidulo:</label>
    <input type="text" name="tituloFilme" value="<?php echo $tituloFilme; ?>">
  </p>
  <p>
    <label>Duração:</label>
    <input type="text" name="duracaoFilme" value="<?php echo $duracaoFilme; ?>">
  </p>
    <p>
      <label>Valor: R$ </label>
      <input type="text" name="valorLocacao" value="<?php echo $valorLocacao; ?>">
      </p>
<p><label>Categoria </label>
<select name="idCategoria">
<option value='<?php echo $idCategoria;?>'><?php echo $nomeCategoria;?></option>
<?php
$rsCategoria = mysqli_query($conexao,"select * from tbCategorias");
$linhas = mysqli_num_rows($rsCategoria);
for($i=0;$i<$linhas;$i++){
   $dados              = mysqli_fetch_array($rsCategoria);
```

```
        $idCategoria        = $dados["idCategoria"];
        $nomeCategoria      = $dados["nomeCategoria"];
        echo "<option value = '$idCategoria'>$nomeCategoria</option>";
    }
    ?>

    </select>
    </p>
      <p>
        <input type="submit" name="opcao" value="Atualizar">
      </p>
    </form>
    </body>
    </html>
```

Resultado:

Para testar esta página é preciso acessar a página lista-filmes.php e clicar no link “Editar” de um dos registros de filme listados.

Explicando o código:

Na linha 2 usamos a função `include()` para conectar ao banco de dados.

```
include("conexao.php");
```

Na linha 3 recuperamos o código do filme enviado pelo método GET e o atribuímos a variável $idFilme.

```
$idFilme = $_GET["idFilme"];
```

Na linha 4 até a 12 uma consulta SQL de seleção que retorna para o formulário os dados do filme a ser atualizado.

```
  $sqlRegistros       = mysqli_query($conexao,"select tituloFilme, duracaoFilme,
valorLocacao, tbFilmes.idCategoria, nomeCategoria from tbFilmes inner join tbCategorias
on tbFilmes.idCategoria = tbCategorias.idCategoria where idFilme=$idFilme") or die("Erro
na execusão da consulta" . mysqli_error($conexao));

  $dados              = mysqli_fetch_array($sqlRegistros);

  $tituloFilme        = $dados["tituloFilme"];
  $duracaoFilme       = $dados["duracaoFilme"];
  $valorLocacao       = $dados["valorLocacao"];
  $idCategoria        = $dados["idCategoria"];
  $nomeCategoria      = $dados["nomeCategoria"];
```

Perceba que, para preencher cada campo do formulário, é preciso aplicar um código PHP em cada um, utilizando o comando echo para exibir cada variável como valor da propriedade value.

```
<input type="text" name="tituloFilme" value="<?php echo $tituloFilme; ?>">
```

Após os dados exibidos no formulário, serem editados, ao clicar no botão Atualizar, os dados novos são enviados para a página php de destino responsável pelo processamento da atualização dos dados no servidor. Esta página está especificada na linha 26 na propriedade action do form.

```
<form name="form1" action="atualizar-filme.php?" method="post">
```

Atualizando os dados recebidos pelo formulário

Após enviar os dados que deverão ser atualizados, vamos compreender a programação da página atualizar-filme.php.

Nome do arquivo: atualizar-filme.php

```
<?php
include("conexao.php");

$idFilme        = $_POST["idFilme"];
$tituloFilme          = $_POST["tituloFilme"];
$duracaoFilme   = $_POST["duracaoFilme"];
$valorLocacao   = $_POST["valorLocacao"];
$idCategoria          = $_POST["idCategoria"];

$sql = "update tbfilmes set tituloFilme =
'$tituloFilme',duracaoFilme='$duracaoFilme',valorLocacao='$valorLocacao',
idCategoria=$idCategoria where idFilme = $idFilme";
mysqli_query($conexao,$sql) or die("Erro ao atualizar registro. " .
mysqli_error($conexao));
header('Location: lista-filmes.php');
?>
```

Explicando o código:

Neste arquivo, basicamente os valores são recuperados, atribuídos a variáveis e introduzidos na consulta SQL que usa a instrução UPDATE para atualizar os dados de um registro.

Update

A instrução UPDATE é utilizada para atualizar valores em determinados campos de um registro ou de vários registros ao mesmo tempo. A instrução UPDATE deve ser usada cuidadosamente com a clausula WHERE, pois sem a clausula where todas as linhas de registros serão atualizadas.

Uma instrução UPDATE permite alterar:

- Uma coluna de uma linha da tabela;
- Várias colunas de uma linha;
- Uma coluna em várias linhas;
- Várias colunas em várias linhas;

Para obter o que foi descrito acima usamos:

- A cláusula SET para escolher a(s) coluna(s);
- A cláusula WHERE para escolher a(s) linha(s);

Sintaxe;

```
UPDATE <nome_da_tabela> SET <campo1> = <valor1>,<campo2> = <valor2>,..
WHERE <condição>
```

Exemplo 1:

No exemplo abaixo, o campo `salarioFuncionario` é atualizado para 5000, mas com a condição imposta pela cláusula WHERE que atualiza somente o registro do funcionário cujo o campo `idFuncionario` seja igual a 1.

```
UPDATE tbFuncionarios SET salarioFuncionario = 5000 WHERE idFuncionario = 1
```

Após a execução da consulta UPDATE, o funcionário Bernardo Castro terá seu salário atualizado para 5000,00.

Observações:

Caso a cláusula `WHERE` não fosse informada a atualização do campo `salarioFuncionario` aconteceria em todos os registros de funcionários.

Exemplo 2:

É possível também, atualizar dados numéricos de um campo em cada registro selecionado para o resultado de um cálculo matemático em tempo de execução.

No exemplo a seguir, querem acrescentar R$ 100,00 reais de bonificação ao salário de todos os funcionários. Para realizar esta consulta Update, adicionaremos na consulta SQL um pequeno cálculo a seguir.

```
UPDATE tbFuncionarios SET salarioFuncionario = salarioFuncionario + 100
```

Entendendo claramente o que acontece neste cálculo;

Vamos supor que o campo salário já tenha 5000, então a atualização seria assim.

salarioFuncionario = salarioFuncionario + 100
salarioFuncionario = 5000+100

Excluído dados no banco de dados

Excluir um registro através de uma página php é muito fácil, simplesmente porque, o comando SQL para excluir um registro na tabela também é muito simples.

No arquivo lista-filmes.php a um link em cada linha de registro que direciona para a página excluir-filme.php. Junto com a página, é informado como parâmetro, o id do Filme, assim o código de exclusão saberá qual registro excluir.

Nome do arquivo: excluir-filme.php

```
<?php
include("conexao.php");

   $idFilme = $_GET["idFilme"];
   $sqlDeleta = mysqli_query($conexao,"delete from tbFilmes where idFilme = $idFilme")
or die("Erro ao deletar arquivo. " . mysqli_error($conexao));
   header('Location: lista-filmes.php');
?>
```

Delete

A instrução DELETE exclui linhas de registros em uma tabela. Aplicando junto com a cláusula WHERE podemos especificar qual ou quais linhas de registros serão excluídas.

Sintaxe;

```
DELETE FROM <nome_da_tabela> WHERE <condição>
```

Exemplo:

```
DELETE FROM tbCursos WHERE idCurso = 3
```

No exemplo acima somente o registro cujo idCurso seja igual a 3 será excluído.

Explicando o código:

No código do arquivo “excluir-filme.php” na linha 4 obtemos o código do filme informado por parâmetro na URL e o colocamos na variável $idFime. Assim sabemos qual registro de filme excluir.

Na linha 5 através da função `mysqli_query()` executamos a consulta de exclusão do registro informado.

Na linha 6, após ter excluído, a função `header('Location: <link para onde quer ir>');` redireciona para a página informada em Location.

Aula 12 – Trabalhando com Cookies e Sessões

Nesta aula compreenderemos o uso dos mecanismos cookies e sessões oferecidos pelo PHP para controle de informações entre páginas no navegador. Será explicado suas principais funcionalidades e também será mostrado como criar um sistema de acesso restrito controlado por sessões.

O que é um cookie?

Um cookie é um pequeno arquivo de texto que o servidor insere no computador do usuário para que ele possa recupera-lo posteriormente. Geralmente é utilizado para o acompanhamento ou identificação de usuários que retornam. Os cookies podem ser usados para várias implementações como o recurso de auto completar de um formulário, verificar se o usuário já visitou o site mais de uma vez, capitar informações de preferências do usuário e muito mais.

Como criar um cookie?

Podemos criar cookies através da função setcookie().

Sintaxe basica;

```
setcookie(nome, valor,vencimento);
```

Quando você cria um cookie, usando a função *setcookie*, você deve especificar três argumentos principais. Estes argumentos são *setcookie (nome, valor, vencimento)*:

Nome : O nome do seu cookie. Você vai usar esse nome para mais tarde recuperar o seu cookie.

Valor : O valor que está armazenado em seu cookie.

Vencimento: A data em que o cookie irá expirar, e ser apagado. Se você não definir essa data de validade, então ele vai ser tratado como um cookie de sessão e ser removido quando o navegador for reiniciado.

Nota:

A função setcookie () deve aparecer antes do tag <html> para que funcione.

Exemplo 1:

No exemplo abaixo, vamos criar um cookie chamado "usuário" e atribuir o valor "Marcos de Melo" para ele.

```
<?php
setcookie("usuario", "Marcos de Melo");
?>
<html>
.....
```

O cookie criado acima, existira enquanto o navegador estiver aberto, ou seja, se fechar o navegador, o cookie será destruído automaticamente, isto porque não definimos um tempo para o mesmo expirar. Para determinar um tempo para o cookie expirar e então ser eliminado, podemos usar a time do php.

Veja o mesmo exemplo anterior descrito abaixo, mas com o parâmetro vencimento. O cookie chamado “usuário”, será destruído após 1 dia (86400 segundos).

Exemplo 2:

```
<?php
setcookie("usuario", "Marcos de Melo",time()+86400);
?>
<html>
.....
```

Dica!

Uma dica: cada hora tem 3600 segundos e cada dia tem 86400 segundos.

Como recuperar um valor de cookie?

A variável $ _COOKIE PHP é usada para recuperar um valor de cookie. No exemplo abaixo, podemos recuperar o valor do cookie chamado "usuário" e exibi-lo em uma página:

```
<?php
echo $_COOKIE["usuario"];
?>
```

No exemplo a seguir, usamos o isset() para descobrir se um cookie foi definido:

```
<html>
<body>
<?php
if (isset($_COOKIE["usuario"]))
```

```
echo "Bem vindo " . $_COOKIE["usuario"] . "!<br>";
else
  echo "Usuário novo, seja bem-vindo!<br>";
?>
</body>
</html>
```

Nota:

Os cookies ao serem criados em uma página só podem ser recuperados nas próximas páginas, ou seja, você não pode utiliza-los na mesma página que os criou.

Como excluir um cookie?

Ao excluir um cookie, basta usar a função setcookie() novamente, mas somente fornecendo o parâmetro nome.

Exemplo:

```
<?php
setcookie("usuario");
?>
```

O que é uma sessão?

Sessão é um mecanismo que nos permite guardar informações em variáveis, mas do lado do servidor. Quando acessamos uma página que cria uma sessão em seu código guardando um determinado valor e sua variável, esta sessão existira e poderá ser acessada em várias páginas enquanto o navegador estiver aberto. Se o navegador for fechado a mesma será destruída automaticamente.

Sessões são bastante utilizadas para compartilhar informações entre páginas no navegador enquanto o mesmo estiver aberto. Exemplo claros que frisão bem a necessidade de usar sessões, são sites de e-commerce onde você navega por diversas páginas de listas de produtos em um determinado site de vendas, onde em cada página é possível guardar a informação de um produto que pretendemos comprar, no “carrinho de compras”. É possível ver os produtos que estão no carrinho e voltar para a lista de produtos sem perder as informações de produtos no carrinho.

Outro exemplo significativo do uso de sessões são sistema de acesso restrito a páginas por meio de controle de usuários cadastrados.

Criando uma sessão

Para criar uma sessão manualmente ou até mesmo restaurar dados da mesma em uma página, devesse primeiro usar a função "session_start()" para iniciar um ambiente de sessão. Esta função não possui parâmetros e deve ser chamada no início da página dentro do código antes de qualquer código php, senão será gerado um erro.

Sintaxe:

session_start()

Exemplo 1:

```
<?php
session_start();
....< demais códigos> .....
?>
```

Criando variáveis em sessões

Após iniciar uma sessão podemos criar variáveis e ao mesmo tempo atribuir valores para as mesmas através do array superglobal $_SESSION

Sintaxe:

```
$SESSION['<nome da variável>']
```

Exemplo 2:

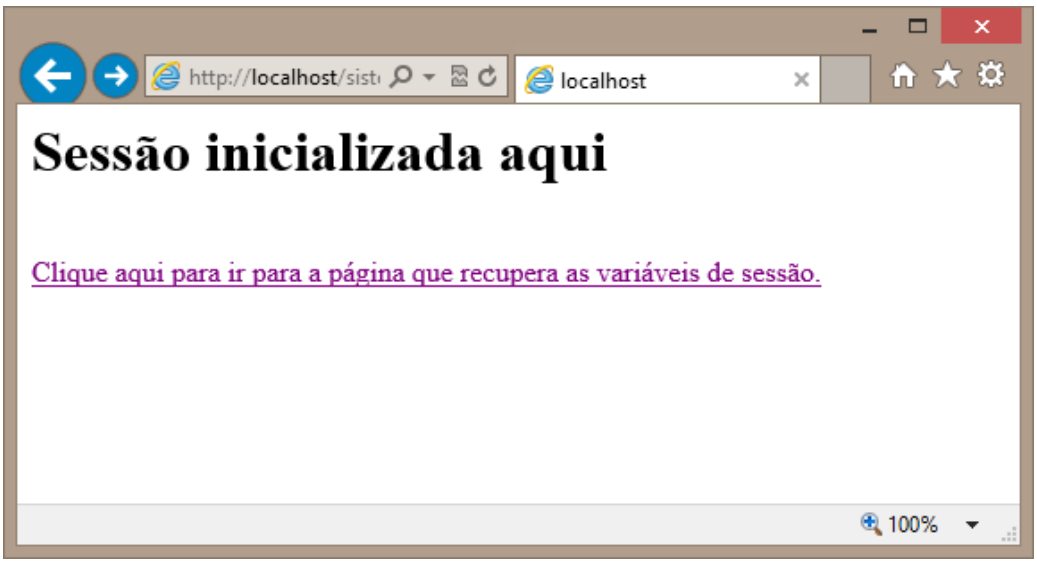
O arquivo a seguir cria uma sessão, onde são criadas variáveis de sessão, para que as mesmas sejam recuperadas e utilizadas por quaisquer páginas enquanto o navegador estiver aberto.

Nome do arquivo: pagina-cria-session.php

```
<?php
session_start(); // iniciamos a sessão
// variaveis criadas em sessão.
$_SESSION['nome']= 'Marcos de Melo';
$_SESSION['email']= 'contato@marcosdemelo.com.br';
echo "<h1>Sessão inicializada aqui</h1>";
```

```
echo "<br> <a href='pagina-recupera-session.php'> Clique aqui para ir para a página que
recupera as variáveis de sessão</a>";
?>
```

Resultado:



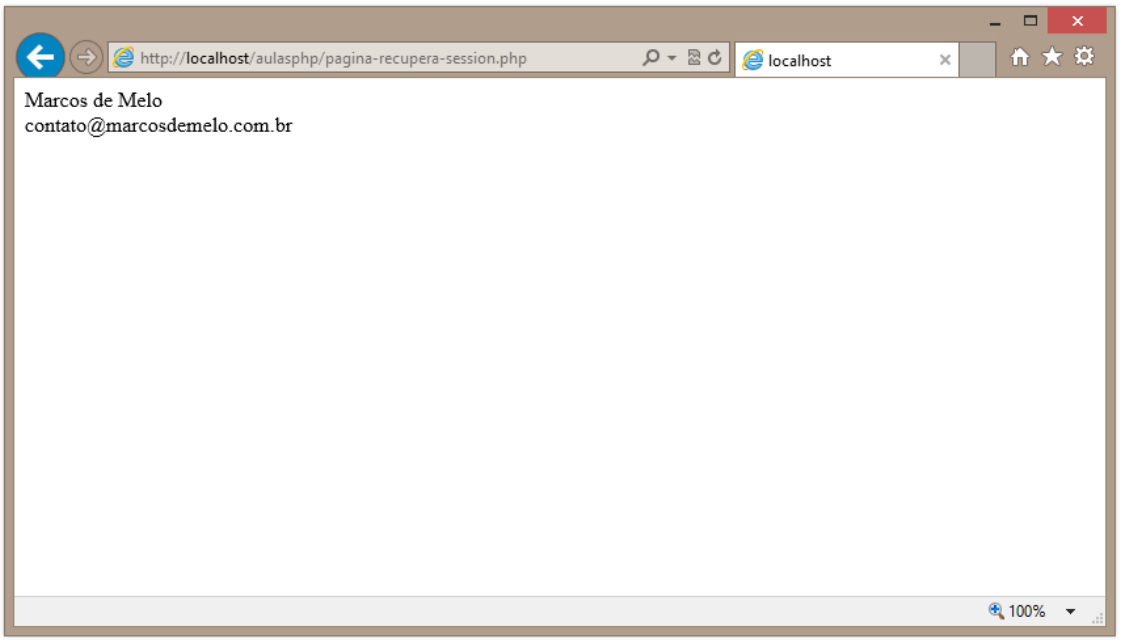
Exemplo 3:

O arquivo abaixo recupera os valores das variáveis de sessão criadas anteriormente. Este arquivo recupera os valores e os atribui a variáveis do php.

Nome do arquivo: pagina-recupera-session.php

```
<?php
session_start();
$nome = $_SESSION['nome'];
$email = $_SESSION['email'];
echo $nome ."<br>";
echo $email . "<br>";
?>
```

Resultado:



Excluir uma sessão

Já sabemos que, ao encerrar a navegação, ou seja, fechar todas as janelas do navegador, todas as sessões em aberto serão encerradas automaticamente. Mas, há uma maneira de eliminar uma sessão sem precisar fechar o navegador. Para excluir uma sessão e os dados de variáveis associados a ela, diretamente no código, usamos a função **session_destroy()**. Ao ser executada esta função, a sessão é eliminada.

Exemplo 4:

Nome do arquivo: sair.php

```
<?php
session_start();
session_destroy();
echo "Sessão finalizada";
?>
```

Atividades

Vamos desenvolver nesta atividade um sistema de controle de acesso por usuários, a páginas restritas de um site, utilizando sessões.

Este sistema validara usuários cadastrados em um banco de dados, se o usuário existir e fornecer o login e senha correto, o mesmo terá acesso as páginas restritas, e se não, não poderá acessar as páginas restritas, retornando para a página de login até que digite o login e senha corretos.

Este sistema após concluído com sucesso, poderá ser adaptado em qualquer site que você queira restringir acesso.

Criando o banco de dados

Precisamos criar um banco de dados com uma tabela de usuário que possua os campos login e senha, pois serão estes campos que serão validados no sistema.

Entre o no programa Workbench e crie o banco de dados “db_cadastro”. Digite o código abaixo na área de Script SQL.

```
CREATE DATABASE  IF NOT EXISTS `db_cadastro`;
USE `db_cadastro`;

CREATE TABLE `tb_usuarios` (
  `login_user` varchar(15) NOT NULL,
  `senha_user` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`login_user`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
INSERT INTO `tb_usuarios` VALUES ('marcos','456'),('nicoly','123');
```

conexao.php

Arquivo de conexão ao banco de dados MySQL.

```
<?php
$servidor       = "localhost";
$usuario        = "root";
$senha          = "neon321";
$banco          = "db_cadastro";

$conexao = mysqli_connect($servidor, $usuario, $senha,$banco);
?>
```

login.php

```
<?php
session_start();

if(isset($_POST["login"])  && isset($_POST["senha"])){

    $login_user = $_POST["login"];
    $senha_user = $_POST["senha"];
    if(!(empty($login_user) or empty($senha_user)))
    {
          include("conexao.php");
          $sql="select * from tb_usuarios where login_user = '$login_user' and
senha_user='$senha_user'";

          $res = mysqli_query($conexao,$sql);
          $linha = mysqli_num_rows($res);

          if($linha==0)
                {
                      session_destroy();
                      echo "Login ou senha incorretos!";
                      exit;
                }
          else
                {
                      $_SESSION["login_user"] = $_POST["login"];
                      $_SESSION["senha_user"] = $_POST["senha"];
                      header("Location: pagina-restrita1.php");
                }
    }
    else
    {
                      session_destroy();
                      echo "Você não efetuou o login!";
                      exit;
    }
}
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Sistema - Controle de acesso.</title>
</head>

<body>
<h1>Digite seu login e senha para entrar nas páginas restritas</h1>
<form id="form1" name="form1" method="post" action="login.php">

  <label for="login">Login:</label>
  <input type="text" name="login" id="login">

  <label for="senha">Senha:</label>
  <input type="text" name="senha" id="senha">
```

```
    <input type="submit" name="button" id="button" value="Entrar">
</form>
</body>
</html>
```

verifica_usuario.php

Este arquivo será o responsável por restringir o acesso nas demais páginas do site. Este arquivo será adicionado como include em todas as páginas que se deseja restringir acesso.

```
<?php
session_start();
if(isset($_SESSION['login_user']))
    $login_user = $_SESSION["login_user"];

if(isset($_SESSION["senha_user"]))
    $senha_user = $_SESSION["senha_user"];

if(!(empty($login_user) or empty($senha_user)))
{
    include("conexao.php");
    $res = mysqli_query($conexao,"select * from tb_usuarios where login_user =
'$login_user' and senha_user='$senha_user'");
    $linha = mysqli_num_rows($res);
    if($linha==0)
    {
            session_destroy();
            echo "Você não efetuou o login!";
            exit;
    }
}
else
{
            header("Location: login.php");
            exit;
}
?>
```

pagina_restrita1.php

Este arquivo é uma página que tem seu conteúdo restrito a usuários cadastrados no sistema. Repare que, para a página só terá seu conteúdo restrito por causa do código php que inclui o arquivo “verifica_usuario.php”.

```
<?php
    include("verifica_usuario.php");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Pagina de Acesso Restrito - 1</title>
</head>

<body>
<h1>Esta é a página 1 de acesso restrito a usuários cadastrados no sistema</h1>
<p>Você entrou!</p>
<p><a href="sair.php">Sair</a></p>
</body>
</html>
```

sair.php

Este arquivo é o responsável por destruir a sessão, ao ser carregado no navegador executa o script php com a função session_destroy() eliminando a sessão.

```
<?php
session_start();
session_destroy();
header("Location: login.php");
?>
```

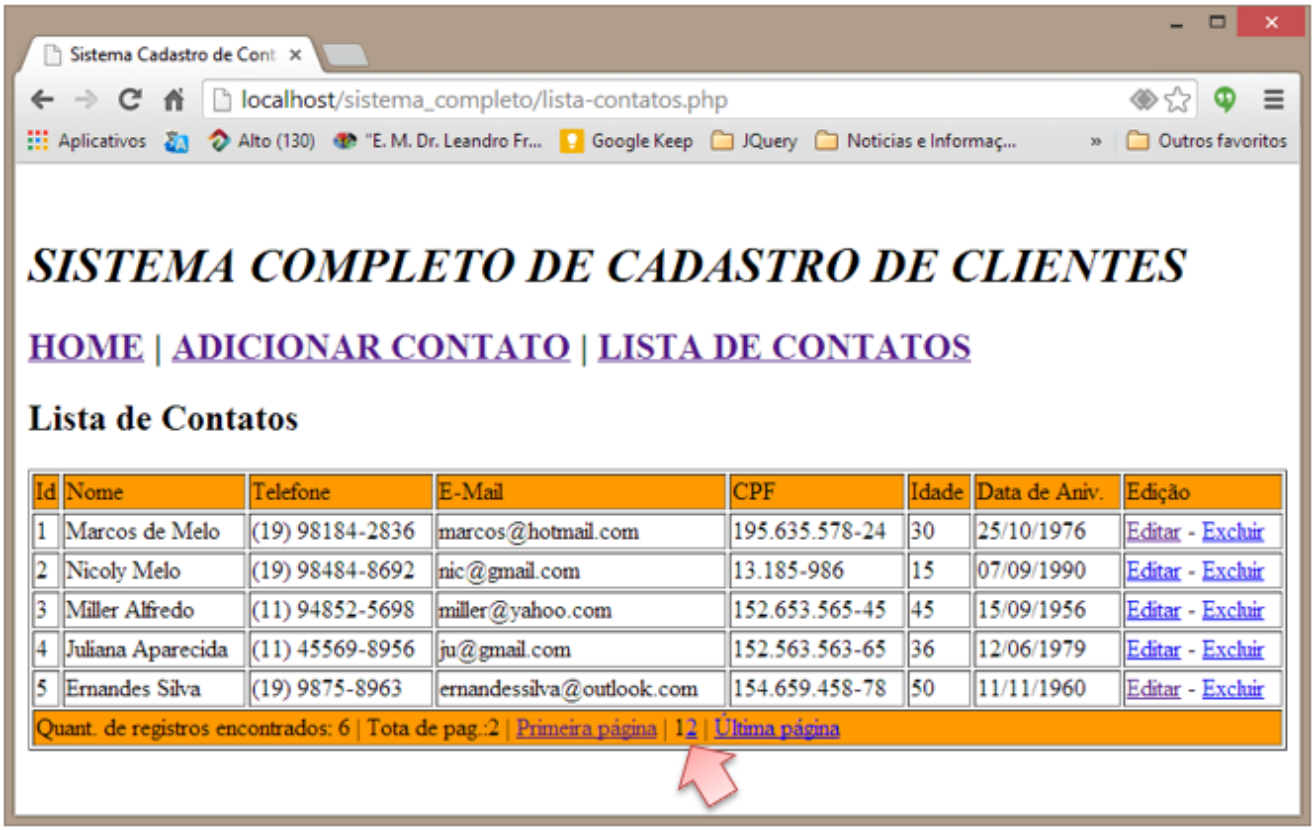
Aula 13 – Sistema completo

Está na hora de revisar todos as aulas desenvolvendo um sistema de cadastro de contatos, abordado todos os recursos das aulas anteriores deste livro.

Criaremos um sistema simples de cadastro de pessoas contendo algumas informações relacionadas como nome, telefone, e-mail, entre outras para serem armazenadas em banco de dados.

O desenvolvimento deste sistema será passo-a-passo, onde o código de cada página PHP será descrito e devidamente explicado para uma perfeita compreensão.

Um controle de paginação para controlar o número de registros por página, também é aplicado no projeto.


Sistema Cadastro de Cont
localhost/sistema_completo/lista-contatos.php
Aplicativos
Alto (130)
"E. M. Dr. Leandro Fr...
Google Keep
JQuery
Noticias e Informaç...
Outros favoritos
SISTEMA COMPLETO DE CADASTRO DE CLIENTES
HOME | ADICIONAR CONTATO | LISTA DE CONTATOS
Lista de Contatos
Id | Nome | Telefone | E-Mail | CPF | Idade | Data de Aniv. | Edição
1 | Marcos de Melo | (19) 98184-2836 | marcos@hotmail.com | 195.635.578-24 | 30 | 25/10/1976 | Editar - Excluir
2 | Nicoly Melo | (19) 98484-8692 | nic@gmail.com | 13.185-986 | 15 | 07/09/1990 | Editar - Excluir
3 | Miller Alfredo | (11) 94852-5698 | miller@yahoo.com | 152.653.565-45 | 45 | 15/09/1956 | Editar - Excluir
4 | Juliana Aparecida | (11) 45569-8956 | ju@gmail.com | 152.563.563-65 | 36 | 12/06/1979 | Editar - Excluir
5 | Ernandes Silva | (19) 9875-8963 | ernandessilva@outlook.com | 154.659.458-78 | 50 | 11/11/1960 | Editar - Excluir
Quant. de registros encontrados: 6 | Tota de pag.:2 | Primeira página | 1 2 | Última página

Banco de dados

Em um site desenvolvido com páginas dinâmicas como por exemplo, o php, e que interagem com banco de dados, a primeira coisa a planejar e desenvolver é a criação do Banco de dados. Nesta primeira etapa, vamos criar um banco de dados simples com uma única tabela de dados para cadastros de pessoas.

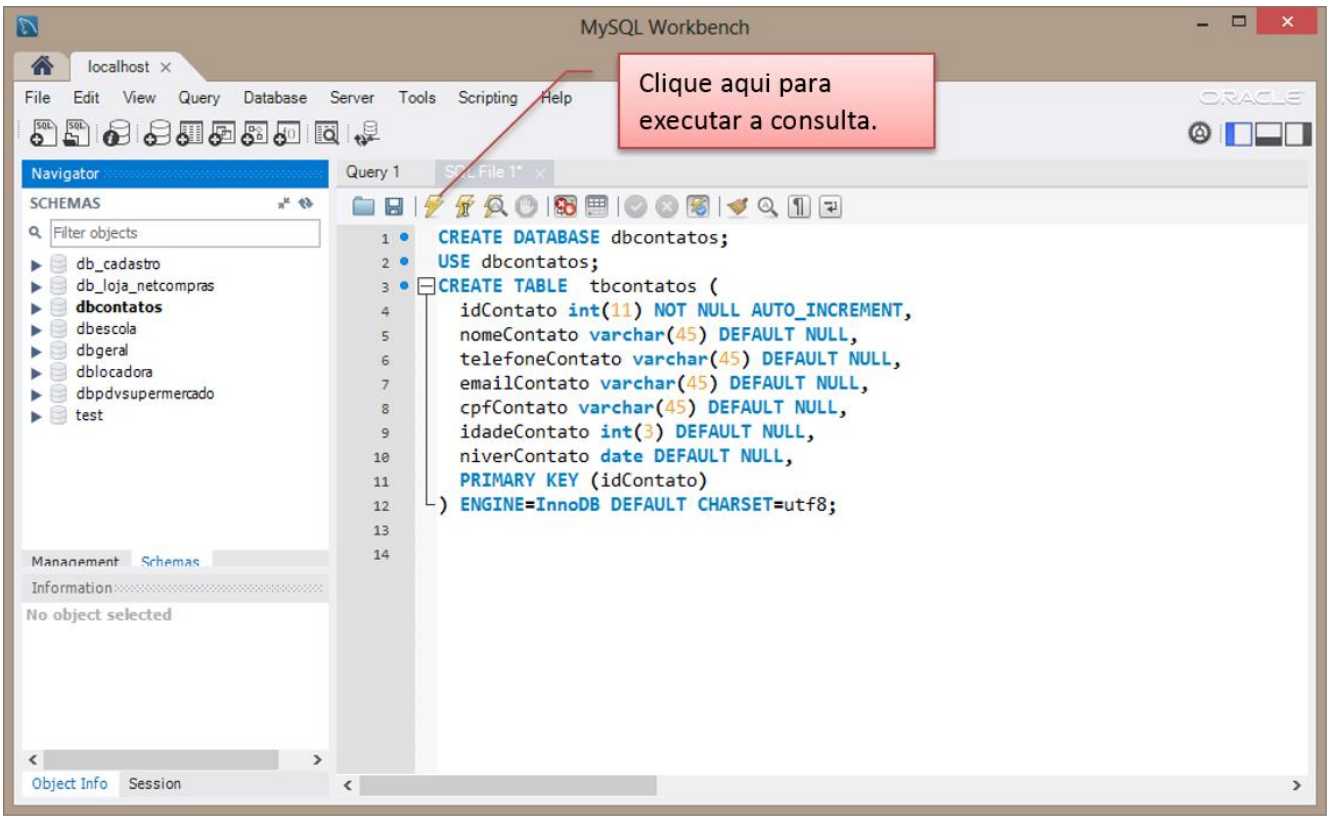
O sistema como falamos, é bastante simples, basicamente podemos cadastrar novos contatos, visualizar, atualizar e excluir registros.

Criando o banco dbContatos

Antes de começar, certifique-se de que seu servidor de banco de dados MySQL esteja ativo.

Utilize o programa Workbench, digite o código abaixo e execute para criar o banco.

```
CREATE DATABASE dbcontatos;
USE dbcontatos;
CREATE TABLE  tbcontatos (
  idContato int(11) NOT NULL AUTO_INCREMENT,
  nomeContato varchar(45) DEFAULT NULL,
  telefoneContato varchar(45) DEFAULT NULL,
  emailContato varchar(45) DEFAULT NULL,
  cpfContato varchar(45) DEFAULT NULL,
  idadeContato int(3) DEFAULT NULL,
  niverContato date DEFAULT NULL,
  PRIMARY KEY (idContato)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```



Organizando a pasta raiz

Depois de ter criado o banco de dados, vamos começar o desenvolvimento do sistema em PHP organizando a estrutura de pastas a partir da pasta raiz.

Crie uma pasta com o nome “sistema_completo” dentro da pasta htdocs do servidor xampp.

Criando os arquivos PHP

Agora vamos criar todos os arquivos do sistema, um a um detalhadamente a começar pelos arquivos de "include" ou seja, que vão ser vinculados em outras páginas php do sistema.

Dois (2) arquivos serão utilizados como "include", são eles; conexao.php e menu.php.

conexao.php

O arquivo "conexao.php" como já vimos em outros exemplos neste livro, será utilizado como arquivo de conexão incluindo nas demais páginas do sistema que se comunicarem com o servidor. O arquivo como já demonstramos, será introduzido nas páginas através da função `include()` mais adiante.

```
<?php

/* informações para conexão à base de dados */

$servidor  = "localhost";        // host do mysql
$usuario   = "root";             // usuário
$senha             = "";                 // senha do usuário , informe a senha do seu servidor MySQL
$banco             = "dbcontatos";       // nome da base de dados

// conecta o mysql
$conexao = mysqli_connect($servidor, $usuario, $senha) or die ("<br><br><center>Problemas ao
conectar no servidor: " . mysqli_error($conexao) . "</center>");
// seleciona a base de dados

?>
```

menu.php

O arquivo menu.php também é um arquivo de inclusão nas demais páginas, basicamente é um arquivo com os links de navegação entre as páginas.

```
<h1><em>SISTEMA COMPLETO DE CADASTRO DE CLIENTES</em></h1>
<h2>
<a href="index.php">HOME</a>
 |  <a href="cad-contato.php">ADICIONAR CONTATO</a>
  | <a href="lista-contatos.php">LISTA DE CONTATOS</a>
</h2>
```

index.php

A página index.php (pagina principal) é também uma página muito simples que possui somente o arquivo de include do menu.php para que seja possível navegar entre as demais páginas dinâmicas php.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Sistema Completo de Clientes</title>
</head>

<body>
<?php
include("menu.php");
?>
<img src="CLIENTES.jpg" width="469" height="266" />
</body>
</html>
```

cad-contato.php

O arquivo “cad-contato.php” é basicamente um formulário para a entrada de dados que serão gravados no banco de dados.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Sistema Completo de cadastro de Contatos</title>
</head>

<body>
<?php
include("menu.php");
?>
<h2>Cadastro de Contatos </h2>
<form id="form1" name="form1" method="post" action="grava-contato.php">
  <p>Nome:
    <label for="nomeContato"></label>
  <input type="text" name="nomeContato" id="nomeContato" />
  </p>
  <p>Telefone:
    <label for="telefoneContato"></label>
    <input type="text" name="telefoneContato" id="telefoneContato" />
  </p>
  <p>E-mail:
    <label for="emailContato"></label>
    <input type="text" name="emailContato" id="emailContato" />
  </p>
  <p>CPF:
```

```
      <label for="cpfContato"></label>
      <input type="text" name="cpfContato" id="cpfContato" />
    </p>
    <p>Idade:
      <label for="idadeContato"></label>
      <input type="text" name="idadeContato" id="idadeContato" />
    </p>
    <p>Data de Aniv.:
      <label for="niverContato"></label>
      <input type="text" name="niverContato" id="niverContato" />
    </p>
    <p>
      <input type="submit" name="enviarContato" id="enviarContato" value="Salvar" />
    </p>
  </form>
  </body>
  </html>
```

grava-contato.php

Após digitar os dados que serão inseridos no banco de dados no servidor MySQL, para que os mesmos sejam realmente inseridos no banco quando pressionarmos o botão Salvar, é preciso criar o arquivo php que faça este processamento. A baixo é mostrado este código.

```
<?php
include("menu.php");
include("conexao.php");
if(isset($_POST["nomeContato"]) and isset($_POST["telefoneContato"]) and
isset($_POST["emailContato"]) and isset($_POST["cpfContato"])and
isset($_POST["idadeContato"])and isset($_POST["niverContato"])){

          if($_POST["nomeContato"] == "" or $_POST["telefoneContato"] == "" or
$_POST["emailContato"] == "" or $_POST["cpfContato"] == "" or $_POST["idadeContato"] ==
"" or $_POST["niverContato"] == ""){
                  echo "Você não preencheu algum dos campos!<br>";
                  echo "<a href=\"cad-contato.php\">Voltar</a>";
          }else{
                  $nomeContato        = $_POST["nomeContato"];
                  $telefoneContato    = $_POST["telefoneContato"];
                  $emailContato       = $_POST["emailContato"];
                  $cpfContato         = $_POST["cpfContato"];
                  $idadeContato       = $_POST["idadeContato"];
                  $niverContato       = str_replace('/','-',$_POST["niverContato"]);
                  $niverContato       = date('Y-m-d', strtotime($niverContato));
                  $sql = "insert into
tbcontatos(nomeContato,telefoneContato,emailContato,cpfContato,idadeContato,niverContato
)
values('$nomeContato','$telefoneContato','$emailContato','$cpfContato','$idadeContato','
$niverContato')";
                  $gravar = mysqli_query($conexao,$sql) or die("Aviso! ==> "
.mysqli_error($conexao));
                  echo "Dados gravados com sucesso!<br>";

          }
}
else{
          echo "Você entrou pelo local errado!";

   }
?>
```

lista-contatos.php

```
<?php
include("conexao.php");
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Sistema Cadastro de Contatos</title>
</head>
<body>
<?php
include("menu.php");
?>
<h2>Lista de Contatos </h2>
<table width="100%" border="1">
  <tr bgcolor="#FF9900">
    <td>Id</td>
```

```
    <td>Nome</td>
    <td>Telefone</td>
    <td>E-Mail</td>
    <td>CPF</td>
    <td>Idade</td>
    <td>Data de Aniv.</td>
    <td>Edição</td>
  </tr>
  <?php
  $quantidade    = 5;
  $pagina        = (isset($_GET["pagina"])) ? (int)$_GET["pagina"] : 1;
  $inicio        = ($quantidade * $pagina) - $quantidade;

  $sql    = "select * from tbContatos order by idContato asc limit $inicio,
$quantidade";

  $tb_Contatos = mysqli_query($conexao,$sql) or die("Não foi possível realizar a
consulta Erro: " .mysqli_error($conexao));
  $num_linhas          = mysqli_num_rows($tb_Contatos);
  for($i=0;$i<$num_linhas;$i++){

   $dados = mysqli_fetch_row($tb_Contatos) or die("Não foi possível realizar a consulta
Erro: " .mysqli_error($conexao));
   $idContato          = $dados[0];
   $nomeContato        = $dados[1];
   $telefoneContato    = $dados[2];
   $emailContato       = $dados[3];
   $cpfContato         = $dados[4];
   $idadeContato       = $dados[5];
   $niverContato       = str_replace('-','/',date('d-m-Y',strtotime($dados[6])));
  ?>
  <tr>
    <td><?php echo $idContato  ?> </td>
    <td><p id="nome" contenteditable="true"><?php echo $nomeContato ?></p></td>
    <td><p id="telefone" contenteditable="true"><?php echo $telefoneContato ?></p></td>
    <td><p id="email" contenteditable="true"><?php echo $emailContato ?></p></td>
    <td><p id="cpf" contenteditable="true"><?php echo $cpfContato ?></p></td>
    <td><p id="idade" contenteditable="true"><?php echo $idadeContato ?></p></td>
    <td><p id="niver" contenteditable="true"><?php echo $niverContato ?></p></td>

    <td><a href="editar-contato.php?idContato=<?php echo $idContato?>">Editar</a>
     - <a href="excluir-registro.php?idContato=<?php echo $idContato?>">Excluir</a></td>
  </tr>
  <?php
  }
  ?>
  <tr bgcolor="#FF9900">
    <td colspan="8">
    <?php
          $sqlTotal    = "select idContato from tbContatos";
   $qrTotal     = mysqli_query($conexao,$sqlTotal) or die(mysqli_error($conexao));
   $numTotal    = mysqli_num_rows($qrTotal);
   echo "Quant. de registros encontrados: $numTotal | ";
   $totalPagina = ceil($numTotal/$quantidade);
   echo "Tota de pag.:$totalPagina | ";

   echo " <a href=\"?pagina=1\">Primeira página</a> | ";
  for($i=1;$i <= $totalPagina; $i++){
```

```
        if($i == $pagina)
            echo "$i";
        else
            echo "<a href=\"?pagina=$i\">$i</a> ";
    }
    echo " | <a href=\"?pagina=$totalPagina\">Última página</a>";

    ?>
    </td>
    </tr>
  </table>
  </body>
  </html>
```

editar-contato.php

Este arquivo é também um formulário, porem já aparece com os campos já preenchidos com os dados do contato, para que possamos edita-los.

```
<?php
include("conexao.php");
$idContato = $_GET["idContato"];
$sql = mysqli_query($conexao,"select * from tbContatos where idContato = $idContato");
$dados = mysqli_fetch_row($sql);
$nomeContato           = $dados[1];
$telefoneContato       = $dados[2];
$emailContato    = $dados[3];
$cpfContato            = $dados[4];
$idadeContato    = $dados[5];
$niverContato    = $dados[6];
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Sistema Completo de cadastro de Contatos</title>
</head>
<body>
<?php
include("includes/menu.php");
?>
<h2>Editar  Contato</h2>
<form id="form1" name="form1" method="post" action="atualiza-contato.php">
<input type="hidden" name="idContato" value="<?php echo $idContato ?>">
  <p>Nome:
  <input type="text" name="nomeContato" id="nomeContato" value="<?php echo
$nomeContato?>"/>
  </p>
  <p>Telefone:
    <input type="text" name="telefoneContato" id="telefoneContato" value="<?php echo
$telefoneContato ?>"/>
  </p>
  <p>E-mail:
    <input type="text" name="emailContato" id="emailContato" value="<?php echo
$emailContato ?>" />
  </p>
  <p>CPF:
    <input type="text" name="cpfContato" id="cpfContato" value="<?php echo $cpfContato
?>" />
  </p>
  <p>Idade:
    <input type="text" name="idadeContato" id="idadeContato" value="<?php echo
$idadeContato ?>" />
  </p>
  <p>Data de Aniversário:
    <input type="text" name="niverContato" id="niverContato" value="<?php echo
$niverContato ?>" />
  </p>
```

```
    <p>
      <input type="submit" name="enviarContato" id="enviarContato" value="Atualizar" />
      <input type="reset" name="limparContato" id="limparContato" value="Limpar" />
    </p>
  </form>
  </body>
  </html>
```

atualiza-contato.php

Este arquivo complementa o arquivo anterior, pois após editarmos os campos com novas informações do contato, os novos dados serão alterados no banco de dados através deste arquivo que recupera estes dados e faz uma consulta UPDATE no banco.

```
<?php
include("menu.php");
include("conexao.php");
if(isset($_POST["idContato"]) and isset( $_POST["nomeContato"]) and
isset($_POST["telefoneContato"]) and isset($_POST["emailContato"]) and
isset($_POST["cpfContato"])and isset($_POST["idadeContato"]) and
isset($_POST["niverContato"])){

          if($_POST["nomeContato"] == "" or $_POST["telefoneContato"] == "" or
$_POST["emailContato"] == "" or $_POST["cpfContato"] == "" or $_POST["idadeContato"] ==
"" or $_POST["niverContato"] == ""){
                  echo "Você não preencheu algum dos campos!<br>";
                  echo "<a href=\"cad-Contato.php\">Voltar</a>";
          }else{
                  $idContato          = $_POST["idContato"];
                  $nomeContato = $_POST["nomeContato"];
                  $telefoneContato    = $_POST["telefoneContato"];
                  $emailContato= $_POST["emailContato"];
                  $cpfContato         = $_POST["cpfContato"];
                  $idadeContato= $_POST["idadeContato"];
                  $niverContato= str_replace('/','-',$_POST["niverContato"]);
                  $niverContato= date('Y-m-d',
strtotime($niverContato));//$_POST["niverContato"];
                  $sql = "update tbContatos set
nomeContato='$nomeContato',telefoneContato='$telefoneContato',emailContato='$emailContat
o',cpfContato='$cpfContato', idadeContato='$idadeContato',niverContato='$niverContato'
where idContato = $idContato";
                  $atualizar = mysqli_query($conexao,$sql) or die("Aviso! ==> "
.mysqli_error($conexao));
                  echo "Dados atualizados com sucesso!<br>";

          }
}
else{
          echo "Você entrou pelo local errado!";
   }
```

```
?>
```

excluir-registro.php

Este arquivo exclui o registro informado através do envio pelo método GET.

```
<?php
include("conexao.php");
$idContato      = $_GET["idContato"];
mysqli_query($conexao,"delete from tbContatos where idContato = $idContato") or die("Não
foi possivel excluir o registro. Erro: " .mysqli_error($conexao)) ;

header('Location:lista-contatos.php');
?>
```