

Organização e Conectividade de Sistemas Computacionais 1 - Prática de Laboratorio

Honda

13 de outubro de 2022

Sumário

1	NGINX, PHP e MySQL	5
1.1	Introdução	5
1.2	Verificações iniciais	5
1.3	Instale o NGINX!	6
1.4	Teste a sua página HTML no novo servidor	6
1.4.1	Passo 1:	6
1.4.2	Passo 2:	7
1.5	Sentindo a configuração do Nginx	7
1.6	Reiniciando o Nginx	8
1.6.1	O teste final	8
1.7	PHP 8	8
1.7.1	SQL	8
1.7.2	Banco de Dados Relacional	8
1.7.3	Chave primária	9
1.8	Instalando o PHP 8	9
1.8.1	No <i>shell</i> aplique o seguinte comando:	9
1.8.2	passo 1: Crie no raiz do servidor web a pasta para os arquivos .php	9
1.8.3	passo 2: Crie o arquivo de configuração que irá indicar a inicialização dos arquivos .php do servidor	9
1.8.4	passo 3 - Ative a configuração criando o seguinte atalho (link simbólico):	10
1.8.5	passo 4 - Apague a configuração anterior com o comando <code>unlink</code> :	10
1.8.6	passo 5 - Teste a sintaxe do arquivo de configuração:	10
1.8.7	passo 6 - Reinicie o Nginx:	10
1.8.8	passo 7:	10
1.8.9	passo 8:	11
1.9	Primeiros passos no PHP	11
1.9.1	<i>Hello World!</i>	11
1.9.2	Um exemplo mais útil:	12
1.9.3	Comentários	12
1.10	Variáveis no PHP	12
1.10.1	O operador de concatenação	12
1.10.2	Variável global	13
1.10.3	Variável local	13
1.11	Operações aritméticas	13
1.12	Estruturas de teste no PHP	13
1.12.1	<code>if</code>	13
1.12.2	<code>if .. else</code>	14
1.12.3	<code>if...elseif...else</code>	14
1.12.4	<code>switch - case</code>	14
1.13	Estruturas de repetição no PHP	15

1.13.1	while	15
1.13.2	do .. while	15
1.13.3	For	16
1.13.4	Foreach	16
1.14	Entrada e saída no PHP	16
1.14.1	Entrando dados em seu script:	17
1.15	Funções em PHP	17
1.15.1	Exercícios	18
1.16	Manipulando arquivos em PHP	18
1.16.1	feof()	19
1.16.2	fgets()	19
1.16.3	fgetc()	19
1.17	"Subindo" arquivos em PHP	20
1.17.1	Inserindo restrições ao "subir" arquivos	20
1.17.2	Como e onde salvar os arquivos enviados pelo <i>script</i> ?	21
1.18	MySQL	22
1.19	Como instalar o MySQL?	22
1.20	Como testar de forma básica o MySQL?	23
1.21	Exemplo prático introdutório com MySQL e PHP	24

Capítulo 1

NGINX, PHP e MySQL



1.1 Introdução

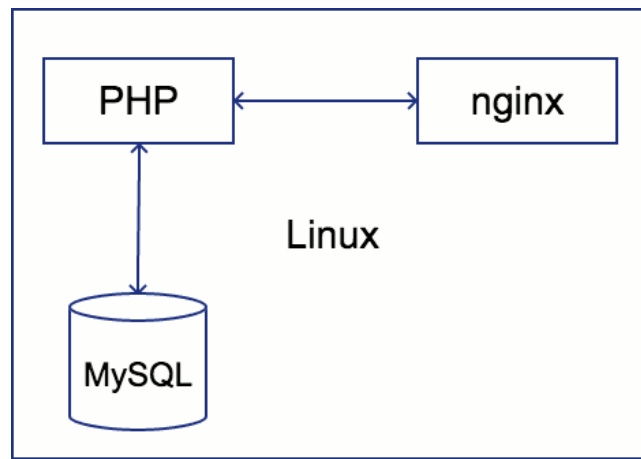
A trinca NGINX, PHP e MySQL se tornou um importante tripé para o desenvolvimento *web*. O NGINX é um servidor *web* que se tornou popular por utilizar poucos recursos de uma máquina. Um outro servidor web proprietário e também muito utilizado é o IIS da Microsoft. O NGINX é livre, gratuito e multiplataforma sendo também altamente configurável. O *web server* da Microsoft é pago e restrito somente aos servidores Windows.

O PHP é uma linguagem de *script* interpretada. Extremamente popular no mundo *web*. Para construir *sites* dinâmicos, ou seja, que façam uma interação básica com o usuário é necessário uma linguagem de programação para *web*. As principais linguagens de programação para *web* que encontramos no mercado são o Javascript, Ruby, .Net, Perl, Node.JS, Python etc.

Para completar a trinca é necessário um gerenciador de base de dados. Exemplo: Qualquer sistema *e-commerce* precisa de uma base de dados para armazenar e com isso realizar a gestão dos cadastros de seus clientes. Um gerenciador de banco de dados muito utilizado para este tipo manobra é o MySQL.

1.2 Verificações iniciais

- Verifique se o seu sistema está atualizado
- Verifique se o servidor SSH está "aceso" e operacional



1.3 Instale o NGINX!

Como?

```
apt install nginx
```

1.4 Teste a sua página HTML no novo servidor

1.4.1 Passo 1:

O raiz do servidor *web* no *Ubuntu server* fica em:

```
/var/www/html/
```

Nesta pasta fica o arquivo `index.nginx-debian.html` de teste com o seguinte conteúdo:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed
    and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

1.4.2 Passo 2:

Visualize a página de teste através de qualquer navegador. Utilize uma máquina na mesma faixa de IP's e insira a seguinte url:

```
http://ip-da-maquina-servidora
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

1.5 Sentindo a configuração do Nginx

Onde ficam os arquivos de configuração do Nginx?

R:

```
/etc/nginx
```

1.6 Reiniciando o Nginx

Uma vez habilitado o módulo reinicie o servidor Nginx com o comando abaixo:

```
service nginx restart
```

1.6.1 O teste final

Teste o funcionamento do seu servidor web transferindo a página HTML realizada na prática 9 para a pasta `/var/www/html`.

1.7 PHP 8

O PHP (*Hypertext Preprocessor*) é uma linguagem de script para desenvolver aplicativos para *web*. Uma página dinâmica e com interação com ao usuário, pode fazer um excelente uso do PHP.

O PHP também é tecnicamente conhecido como uma linguagem de script *server side*; a Microsoft tem uma solução semelhante denominada ASP.NET (*Active Server Pages*).

A combinação mais popular é o PHP trabalhando em conjunto com um gerenciador de base de dados, muitos utilizam o MySQL para esta tarefa.

O gerenciador de base de dados é amplamente utilizado para armazenar informações sobre usuários ou qualquer outro tipo de informação coletável e armazenável em tabelas.

1.7.1 SQL

O SQL (*Structured Query Language*) é um padrão descritivo de base de dados. Existem muitas ferramentas e variações deste padrão. O MySQL é mais uma entre dezenas de ferramentas que gerenciam uma base de dados.

Um banco de dados SQL é um sistema de banco de dados que oferece uma interface SQL. Os bancos de dados SQL muitas vezes são usados em redes cliente/servidor em que vários clientes acessam um servidor central (por exemplo, um servidor SQL); por esse motivo, também são chamados de bancos de dados do servidor SQL ou, abreviadamente, de servidores SQL ¹

1.7.2 Banco de Dados Relacional

Um banco de dados relacional é uma coleção de itens de dados organizados como um conjunto de tabelas descritas formalmente, cujos dados podem ser acessados ou remontados de várias maneiras, sem precisar reorganizar as tabelas de bancos de dados. Um sistema de gerenciamento de bancos relacionais (SGBD) é um programa que permite a criação, a atualização e a administração de um banco de dados relacional. Um SGBD utiliza as instruções SQL (Structured Query Language) inseridas por um usuário ou contidas em um programa aplicativo e cria, atualiza ou fornece acesso ao banco de dados. Um bom exemplo de banco de dados relacional pode ser obtido com um banco de dados que contenha tabelas de clientes, compras e faturamento. Na tabela de faturamento, não há dados reais de

¹No LibreOffice, você pode integrar bancos de dados SQL externos. Esses bancos podem estar na sua máquina local assim ou na rede. O acesso é obtido por ODBC, JDBC ou por driver nativo integrado ao LibreOffice.

clientes e de compras; no entanto, a tabela contém referências por meio de um vínculo relacional, ou uma relação, com os respectivos campos das tabelas de clientes e de compras (por exemplo, o campo de ID do cliente na tabela de clientes).

1.7.3 Chave primária

Uma chave primária serve como identificador único dos campos de bancos de dados. A identificação única dos campos de bancos de dados é usada nos bancos de dados relacionais, para acessar dados em outras tabelas. Se for feita uma referência a uma chave primária de outra tabela, trata-se de uma chave externa. No LibreOffice, você define a chave primária na exibição de design de uma tabela escolhendo o comando relevante no menu de contexto de um cabeçalho de linha para o campo selecionado. [Fonte: Help do LibreOffice]

1.8 Instalando o PHP 8

1.8.1 No *shell* aplique o seguinte comando:

```
apt install php-fpm php-mysql
```

1.8.2 passo 1: Crie no raiz do servidor web a pasta para os arquivos **.php**

```
mkdir /var/www/enois
```

1.8.3 passo 2: Crie o arquivo de configuração que irá indicar a inicialização dos arquivos **.php** do servidor

```
vim /etc/nginx/sites-available/enois
```

Insira o código abaixo no arquivo `/etc/nginx/sites-available/enois`:

```
server {
    listen 80;
    server_name enois.org www.enois.org;
    root /var/www/enois;

    index index.html index.htm index.php;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ /\.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

```
}  
  
}
```

Entenda a sintaxe básica de alguns parâmetros do código acima:

- `listen`: define a porta que o servidor ficará "ouvindo"
- `root`: Define a raiz do servidor web para os arquivos do PHP
- `index`: Define a ordem de leitura dos arquivos `index`
- `server_name` — nome do servidor web
- `location /` — Mensagem para página não encontrada

1.8.4 passo 3 - Ative a configuração criando o seguinte atalho (link simbólico):

```
ln -s /etc/nginx/sites-available/enois /etc/nginx/sites-enabled/
```

1.8.5 passo 4 - Apague a configuração anterior com o comando `unlink`:

```
unlink /etc/nginx/sites-enabled/default
```

Caso for necessário restaurar a configuração `default` aplique o comando abaixo:

```
sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/
```

1.8.6 passo 5 - Teste a sintaxe do arquivo de configuração:

```
nginx -t
```

1.8.7 passo 6 - Reinicie o Nginx:

```
service nginx restart
```

1.8.8 passo 7:

Teste a funcionalidade do PHP8 com o seguinte procedimento:

Escreva um arquivo chamado `teste.php`, na pasta `/var/www/enois`, com o seguinte conteúdo:

```
<?php  
phpinfo();
```

Repare na extensão `.php`, a partir de agora todos os arquivos deverão possuir esta extensão.


Detalhe importante: Como na linguagem C a linha com um comando ou função completo deve terminar com ponto e vírgula.

1.8.9 passo 8:

Teste em qualquer navegador digitando a seguinte url:

```
http://ip_do_servidor_web/teste.php
```

Caso aparecer uma descrição detalhada sobre o sistema instalado de seu *Linux box*, o PHP8 foi instalado com sucesso. Veja amostra abaixo:

PHP Version 8.1.2 	
System	Linux aho 5.15.0-48-generic #54-Ubuntu SMP Fri Aug 26 13:26:29 UTC 2022 x86_64
Build Date	Aug 8 2022 07:28:23
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/fpm
Loaded Configuration File	/etc/php/8.1/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/fpm/conf.d
Additional .ini files parsed	/etc/php/8.1/fpm/conf.d/10-mysqlnd.ini, /etc/php/8.1/fpm/conf.d/10-opcache.ini, /etc/php/8.1/fpm/conf.d/10-pdca, /etc/php/8.1/fpm/conf.d/20-calendar.ini, /etc/php/8.1/fpm/conf.d/20-ctype.ini, /etc/php/8.1/fpm/conf.d/20-exif.ini, /etc/php/8.1/fpm/conf.d/20-ffi.ini, /etc/php/8.1/fpm/conf.d/20-fileinfo.ini, /etc/php/8.1/fpm/conf.d/20-ftp.ini, /etc/php/8.1/fpm/conf.d/20-gettext.ini, /etc/php/8.1/fpm/conf.d/20-iconv.ini, /etc/php/8.1/fpm/conf.d/20-mysqli.ini, /etc/php/8.1/fpm/conf.d/20-pdo_mysql.ini, /etc/php/8.1/fpm/conf.d/20-phar.ini, /etc/php/8.1/fpm/conf.d/20-posix, /etc/php/8.1/fpm/conf.d/20-readline.ini, /etc/php/8.1/fpm/conf.d/20-shmop.ini, /etc/php/8.1/fpm/conf.d/20-sockets, /etc/php/8.1/fpm/conf.d/20-sysvmsg.ini, /etc/php/8.1/fpm/conf.d/20-sysvsem.ini, /etc/php/8.1/fpm/conf.d/20-sysvshm.ini, /etc/php/8.1/fpm/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no

1.9 Primeiros passos no PHP

Como sempre iremos adentrar no mundo do PHP com o clássico:

1.9.1 *Hello World!*

Um script PHP sempre começa com:

```
<?php
e termina com:
?>
```

O script PHP é comumente embutido dentro de uma descrição HTML:

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Isto serve para lembrar que o script em PHP pode estar em qualquer lugar do seu código HTML. É fundamental que o arquivo salvo tenha a extensão `.php`, caso contrário você não irá visualizar a saída do código no navegador.

1.9.2 Um exemplo mais útil:

O script abaixo exibe o navegador utilizado:

```
<?php
echo $_SERVER["HTTP_USER_AGENT"];
?>
```

No exemplo acima `$_SERVER` é o nome da variável a ser preenchida.

1.9.3 Comentários

Os comentários em PHP seguem a mesma sintaxe que os comentários em C padrão.

```
\* Isto eh um comentario *\
ou
\\ Isto eh um comentario
```

1.10 Variáveis no PHP

O tratamento com variáveis é praticamente idêntico ao realizado no shell (bash).

```
<?php
$txt="Hello World!";
$x=16;
echo $txt;
echo $x;
?>
```

1.10.1 O operador de concatenação

Um simples ponto confere o poder de concatenar expressões em PHP.

O questionamento a seguir procura explicar sucintamente a mecânica do operador de concatenação: Como eu insiro uma nova linha entre a variável `$txt` e `$x`?

```
<?php
$txt="Hello World!";
$x=16;
echo $txt . "<br />";
echo $x;
```

Inserindo na penúltima linha o operador de concatenação estaremos aplicando uma *new line* entre os dois comandos `echo` e melhorando a visualização do resultado.

1.10.2 Variável global

É o mesmo conceito estudado na linguagem C padrão.

1.10.3 Variável local

É o mesmo conceito estudado na linguagem C padrão.

1.11 Operações aritméticas

Idêntico ao C padrão.

1.12 Estruturas de teste no PHP

Como qualquer linguagem de programação, a capacidade de teste é fundamental para o desenvolvimento de qualquer código básico. Você perceberá facilmente que a semelhança com o C é intensa e por ser uma linguagem interpretada há um forte influência da linguagem Bash2 estudada nos primórdios do nosso curso.

1.12.1 if

O if solitário funciona como uma "guia" ou "canaleta"

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

A função `date()` possui diversas opções; na opção mostrada anteriormente a função `date()` exibe o dia da semana em inglês de forma abreviada.

Outros exemplos de utilização da função `date()`:

```
<?php
echo date("d/m/Y") . "<br />";
echo date("d.m.Y") . "<br />";
echo date("d-m-Y");
?>
```

saída:

```
2022/09/04
2022.09.04
2022-09-04
```

1.12.2 if .. else

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
{
    echo "Have a nice weekend!";
}
else
{
    echo "Have a nice day!";
}
?>

</body>
</html>
```

1.12.3 if...elseif....else

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
{
    echo "Have a nice weekend!";
}
elseif ($d=="Sun")
{
    echo "Have a nice Sunday!";
}
else
{
    echo "Have a nice day!";
}
?>

</body>
</html>
```

1.12.4 switch - case

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

```
<?php
$x=1;
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
    break;
case 3:
    echo "Number 3";
    break;
default:
    echo "No number between 1 and 3";
}
?>

</body>
</html>
```

1.13 Estruturas de repetição no PHP

1.13.1 while

```
<html>
<body>

<?php
$i=1;
while($i<=5)
{
    echo "The number is " . $i . "<br />";
    $i++;
}
?>

</body>
</html>
```

1.13.2 do .. while

```
<html>
<body>

<?php
$i=1;
do
{
```

```
    $i++;  
    echo "The number is " . $i . "<br />";  
    }  
while ($i<=5);  
?>  
  
</body>  
</html>
```

1.13.3 For

```
<html>  
<body>  
  
<?php  
for ($i=1; $i<=5; $i++)  
{  
    echo "The number is " . $i . "<br />";  
}  
?>  
  
</body>  
</html>
```

1.13.4 Foreach

```
<html>  
<body>  
  
<?php  
$x=array("one", "two", "three");  
foreach ($x as $value)  
{  
    echo $value . "<br />";  
}  
?>  
  
</body>  
</html>
```

1.14 Entrada e saída no PHP

Uma vez realizada uma panorâmica básica do PHP crava-se uma dúvida ululante: Como o usuário insere dados num programa em PHP?

Até agora exemplificamos *scripts* que funcionavam sem nenhuma interação com o usuário. Portanto finalmente estudaremos como aplicar o *input* em um script PHP.

1.14.1 Entrando dados em seu script:

Existem 2 métodos para entrar dados em PHP:

- A variável pré-definida `$_GET` é usada para receber dados oriundos do usuário. A informação enviada é visível na url do navegador.

```
<html>
<body>
<form action="benvindo.php" method="get">
Name: <input type="text" name="nome" />
Age: <input type="text" name="idade" />
<input type="submit" />
</form>
</body>
</html>
```

Salve o código acima em seu `/var/www/php` e chame-o através do navegador. Quando o usuário clicar em enviar os dados, ou variáveis, poderão ser utilizados em um código em php semelhante ao mostrado abaixo:

```
What's up!<?php echo $_GET["nome"]; ?>.<br />
Voc tem <?php echo $_GET["idade"]; ?> anos!
```

O nome do arquivo que tem o código neste caso é `benvindo.php`. O nome do arquivo deve ser igual ao indicado em `<form action='benvindo.php' method='get'>`.

Você precisa deste código! Pois o mesmo irá receber as variáveis inseridas pelo usuário para a partir daí serem exibidas pelo navegador. O método GET não é adequado para valores muito grande de variáveis. O limite da variável GET é próximo de 2000 caracteres.

- A variável pré-definida `$_POST` é usada para receber dados oriundos do usuário. A informação enviada é **invisível** na url do navegador.

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

Mude o arquivo `welcome.php` para que ele fique como as linhas abaixo:

```
What's up!<?php echo $_POST["nome"]; ?>.<br />
Voc tem <?php echo $_POST["idade"]; ?> anos!
```

Caso for necessário inserir um tipo de dados numérico mude o tipo de entrada para: `input type="number"`
Quando utilizamos POST não há limites para o envio de informação como ocorre no método GET.

1.15 Funções em PHP

O PHP possui centenas de funções a sintaxe destas é idêntica ao C padrão.

Como escrever uma simples função em PHP?

Exemplo 1:

```
<?php
function imprime()
{
echo "Oi.Tudo bom?";
}

echo "Apache cumprimentando:" . "<br/ >";
imprime();
?>
```

Utilizando a entrada com o método POST

Exemplo 2:

```
<form action="calcula.php" method="post">
Variavel: <input type="number" name="x" />
<input type="submit" />
</form>
```

Arquivo para processar o parâmetro do formulário:

```
<?php
function quadrado() {
echo $_POST["x"]*$_POST["x"];
}
echo "O valor do quadrado de " .$_POST["x"] . " eh:". "<br />";
quadrado();
?>
```

1.15.1 Exercícios

- Escreva um programa em PHP que calcula o fatorial. As entradas podem ser recebidas através do método GET.
- Escreva um programa em PHP que leia as medidas dos 3 lados a, b e c de um paralelepípedo, calcule e escreva o valor da sua diagonal.

1.16 Manipulando arquivos em PHP

A função utilizada para abrir um arquivo também é semelhante ao visto em C. Para abrir um arquivo utilize a função `fopen()`. Observe o exemplo a seguir:

```
<?php
$file=fopen("teste.txt","r");
?>
```

O segundo argumento da função indica o modo como o arquivo deve ser aberto.

- r - somente leitura. Começa no início do arquivo.

- r+ - leitura/escrita. Começa no início do arquivo.
- w+ - leitura escrita. Abre e apaga o conteúdo existente.
- a - somente leitura. Começa no início do arquivo.
- a+ - somente leitura. Começa no início do arquivo.
- x - somente leitura. Começa no início do arquivo.
- x+ - somente leitura. Começa no início do arquivo.

O exemplo abaixo gera uma mensagem caso não for possível abrir o arquivo:

```
<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
?>
```

As próximas seções mostrarão as principais funções para manipularmos arquivos em PHP.

1.16.1 feof()

Esta função checa se o fim do arquivo (EOF) foi alcançado.

```
if (feof($file)) echo "End of file";
```

1.16.2 fgets()

fgets() lê um arquivo linha por linha.

Exemplo:

O programa abaixo lê um arquivo linha por linha até o final do mesmo.

```
<?php
$file = fopen("teste.txt", "r") or exit("sujou!");
//Output a line of the file until the end is reached
while(!feof($file))
{
    echo fgets($file). "<br />";
}
fclose($file);
?>
```

1.16.3 fgetc()

Esta função lê um arquivo caracter por caracter.

Para os habituados em C padrão, vale dizer que o ponteiro do arquivo está sempre se movendo para o próximo caracter enquanto fgetc() é utilizado.

Exemplo:

```
<?php
$file=fopen("teste.txt","r") or exit("Sujou!");
while (!feof($file))
{
    echo fgetc($file);
}
```

```

    }
    fclose($file);
?>

```

1.17 "Subindo" arquivos em PHP

O código abaixo exemplifica um formulário para *upload* de um arquivo.

```

<form action="subindo.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>

```

Obviamente é necessário o *script* que faz o *upload* como mostrado a seguir:

```

<?php
if ($_FILES["file"]["error"] > 0)
{
    echo "Error: " . $_FILES["file"]["error"] . "<br>";
}
else
{
    echo "Upload: " . $_FILES["file"]["name"] . "<br>";
    echo "Type: " . $_FILES["file"]["type"] . "<br>";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
}
?>

```

O que é importante aqui é entender que o *array* `$_FILES` permite realizar o *upload* e o controle de arquivos.

- `$_FILES["file"]["name"]` - indica o nome do arquivo "subido"
- `$_FILES["file"]["type"]` - mostra o tipo de arquivo "subido"
- `$_FILES["file"]["size"]` - mostra o tamanho em *bytes* de arquivo "subido"
- `$_FILES["file"]["tmp_name"]` - o nome do arquivo de cópia temporária que será armazenada no servidor
- `$_FILES["file"]["error"]` - o erro que será exibido caso ocorra um erro na "subida" do arquivo.

1.17.1 Inserindo restrições ao "subir" arquivos

O *script* abaixo insere restrições como tipo e tamanho de arquivos que podem ser enviados para o servidor. É possível "subir" apenas arquivos do tipo `.gif`, `.jpg` e `.png`. Além disso o tamanho máximo do arquivo é restrito a algo menor que 20000 bytes.

```

<?php
$allowedExts = array("jpg", "jpeg", "gif", "png");
$extension = end(explode(".", $_FILES["file"]["name"]));
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000)
&& in_array($extension, $allowedExts))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Error: " . $_FILES["file"]["error"] . "<br />";
    }
    else
    {
        echo "Upload: " . $_FILES["file"]["name"] . "<br />";
        echo "Type: " . $_FILES["file"]["type"] . "<br />";
        echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
        echo "Stored in: " . $_FILES["file"]["tmp_name"];
    }
}
else
{
    echo "Invalid file";
}
?>

```

1.17.2 Como e onde salvar os arquivos enviados pelo *script*?

Todos os *scripts* anteriores salvam os arquivos em arquivos temporários. Uma vez que o *script* é finalizado este arquivo é apagado. O *script* a seguir testa se o arquivo existe e caso negativo ele o copia para um folder chamado *upload*.

```

<?php
$allowedExts = array("gif", "jpeg", "jpg", "png");
$temp = explode(".", $_FILES["file"]["name"]);
$extension = end($temp);
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/jpg")
|| ($_FILES["file"]["type"] == "image/pjpeg")
|| ($_FILES["file"]["type"] == "image/x-png")
|| ($_FILES["file"]["type"] == "image/png"))
&& ($_FILES["file"]["size"] < 20000)
&& in_array($extension, $allowedExts))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Return Code: " . $_FILES["file"]["error"] . "<br>";
    }
    else
    {

```

```

echo "Upload: " . $_FILES["file"]["name"] . "<br>";
echo "Type: " . $_FILES["file"]["type"] . "<br>";
echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br>";

if (file_exists("upload/" . $_FILES["file"]["name"]))
{
    echo $_FILES["file"]["name"] . " already exists. ";
}
else
{
    move_uploaded_file($_FILES["file"]["tmp_name"],
    "upload/" . $_FILES["file"]["name"]);
    echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
}
}
else
{
    echo "Invalid file";
}
?>

```

1.18 MySQL

O MySQL é um gerenciador de base de dados. Uma base de dados pode ser entendida como uma quantidade de informação armazenada de forma estruturada na memória do computador. A ideia de estruturar este armazenamento é motivada por dois fatores: a escalabilidade e desempenho otimizado no acesso a informação armazenada.

O SQL (*Structured Query Language*) é uma linguagem de programação padrão voltada para gerenciar dados do tipo RDBMS.

MySQL é um dentre dezenas de pacotes utilizados para realizar a tarefa de gerenciamento. Iremos utilizá-lo devido a sua popularidade, estabilidade e gratuidade (É um *software* livre)

Os dados no MySQL são armazenados através de tabelas. Uma tabela é um conjunto de linhas e colunas, como em uma planilha eletrônica, onde cada linha corresponde a uma entrada e cada coluna serviria como informação relacionada a esta entrada.

1.19 Como instalar o MySQL?

Procedimento:

- instale um servidor MySQL e um cliente MySQL:

```
apt install mysql-server
```

A instalação do `mysql-server`, no caso do *Ubuntu server*, instalará também o `mysql-client`

- Pense: Por que eu preciso de um *server* e um *client*?

- Insira uma senha fácil de lembrar
- Acesse o MySQL server através da seguinte linha de comando:

```
mysql -h localhost -u root -p
```

- Preste atenção nos dizeres iniciais antes do prompt

```
mysql>
```

- Para sair do MySQL basta digitar: quit ou QUIT; O MySQL não é *sensitive case*.
- Altere o arquivo de configuração /etc/mysql/mysql.conf.d/mysqld.cnf:
Mude parâmetro bind-address = 127.0.0.0 para 0.0.0.0
- Reinicie o MySQL:

```
service mysql restart
```

1.20 Como testar de forma básica o MySQL?

- Acesse o MySQL server através da seguinte linha de comando:

```
mysql -h localhost -u root -p
```

- Crie um usuário no MySQL:

```
CREATE USER 'usuario' IDENTIFIED BY 'password';
```

- Apague um usuário no MySQL:

```
DROP USER usuario;
```

- É possível modificar permissões do usuário para diversos níveis:

- global
- base de dados
- tabela

- A sintaxe para modificar as permissões do usuário segue a forma:

```
GRANT tipo_privilegio ON {nome_tabela | * | *.* | nome_db.*} TO  
usuario;
```

Exemplo:

```
GRANT SELECT ON *.* TO jose;
```

- para revogar a permissão de um usuário faça:

```
REVOKE tipo_privilegio ON {nome da tabela |*|*.*| base de dados.*}  
FROM JOSE;
```

- Criando uma base de dados:

```
CREATE DATABASE nome-da-base;
```

- Removendo uma base de dados:

```
DROP DATABASE nome_da_base;
```

- Listando os bancos de dados disponíveis:

```
SHOW DATABASES;
```

1.21 Exemplo prático introdutório com MySQL e PHP

- Crie uma base de dados com o nome teste:

```
CREATE DATABASE teste;
```

- Crie um usuário da base de dados teste:

```
CREATE USER 'joseh' IDENTIFIED WITH mysql_native_password BY 'coltec'  
;
```

- Conceda acesso completo (*full privileges*) da base de dados teste para o user joseh:

```
GRANT ALL ON teste.* TO 'joseh';
```

- saia do MySQL:


```
exit
```

- Teste se o user joseh consegue de fato acessar a base de dados teste:

```
mysql -u joseh -p
```

```
SHOW DATABASES;
```

A saída será algo do tipo:

```
Type 'help;' or '\h' for help. Type

[mysql> Show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| performance_schema      |
| teste                   |
+-----+
3 rows in set (0.01 sec)

mysql> █
```

- Dentro da base de dados teste crie a seguinte **tabela**:

```
CREATE TABLE teste.listinha (  
    item_id INT AUTO_INCREMENT,  
    content VARCHAR(255),  
    PRIMARY KEY(item_id)  
);
```

- Insira a primeira linha da tabela com o seguinte comando:

```
INSERT INTO teste.listinha (content) VALUES ("Meu primeiro item");
```

- Insira a segunda linha da tabela com o seguinte comando:

```
INSERT INTO teste.listinha (content) VALUES ("Meu segundo item");
```

- Insira a terceira linha da tabela com o seguinte comando:

```
INSERT INTO example_database.todo_list (content) VALUES ("Meu  
terceiro item");
```

- Insira a quarta linha da tabela com o seguinte comando:

```
INSERT INTO teste.listinha (content) VALUES ("Meu quarto item");
```

- Confirme os dados inseridos na tabela com o seguinte comando:

```
SELECT * FROM teste.listinha;
```

- Saia do MySQL:

```
exit
```

- Agora teste o seguinte script PHP que faz uma conexão básica com o MySQL:

```
<?php  
$user = "joseph";  
$password = "coltec";  
$database = "teste";  
$table = "listinha";  
  
try {  
    $db = new PDO("mysql:host=localhost;dbname=$database", $user,  
        $password);  
    echo "<h2>TODO</h2><ol>";
```

```
foreach($db->query("SELECT content FROM $table") as $row) {  
    echo "<li>" . $row['content'] . "</li>";  
}  
echo "</ol>";  
} catch (PDOException $e) {  
    print "Error!: " . $e->getMessage() . "<br/>";  
    die();  
}
```

- Teste o script acima e aprecie o resultado.