PHP

&

MySQL

SUMÁRIO

PHP

SITES ESTÁTICOS

- Utiliza-se freqüentemente apenas HTML
- Não interagem às solicitações dos usuários
- Apresentam uma estrutura “rígida”
- Atualização das páginas muito “trabalhosa”
- Não acessam Bancos de Dados

SITES DINÂMICOS

- Utiliza-se: HTML, PHP, ASP, JSP, XML, etc.
- Interagem às solicitações dos usuários
- Apresentam uma estrutura mais flexível
- Atualização das páginas muito “prática”
- Acessam Bancos de Dados

PHP

Originalmente: Personal Home Page

Hoje: Hipertext Preprocessor

CARACTERÍSTICAS

- Linguagem totalmente voltada à Internet
- Utilizada para criação de sites dinâmicos
- Gratuito: (download: www.php.net)
- Código aberto
- Pode ser “embutido” em páginas HTML
- Baseado no Servidor: Apache, IIS ...
- Suporte a vários SGBD’s: PostgreSQL, MySQL ...
- Portabilidade: Linux, Unix, Windows NT, 2000 ...

HISTÓRIA

1994: Rasmus Lerdorf
Aplicação: Informações sobre visitas ao site

1995: Primeira versão utlizada por outras pessoas
Livro de visitas, contador, etc.

1995 (2º semestre): PHP/FI (Form Interpreter)
Interpreta dados de formulários HTML e suporte a MySQL

1996: Interpretador reescrito: Zeev Suraski e Andi Gutmans

Atualmente: PHP4 (está sendo desenvolvida a versão 5).

RECURSOS NECESSÁRIOS

LINUX

- APACHE

WINDOWS 95/98/ME

- PHP TRIAD (inclui MySQL)
- EASY PHP (inclui MySQL)
- PWS (Personal Web Server) com módulo PHP

WINDOWS NT/2000/XP

- IIS (Internet Information Service) com modulo PHP

SITES PARA DOWNLOAD

PHP

http://www.php.net

PHP Triad

http://sourceforge.net/projects/phptriad

EasyPHP

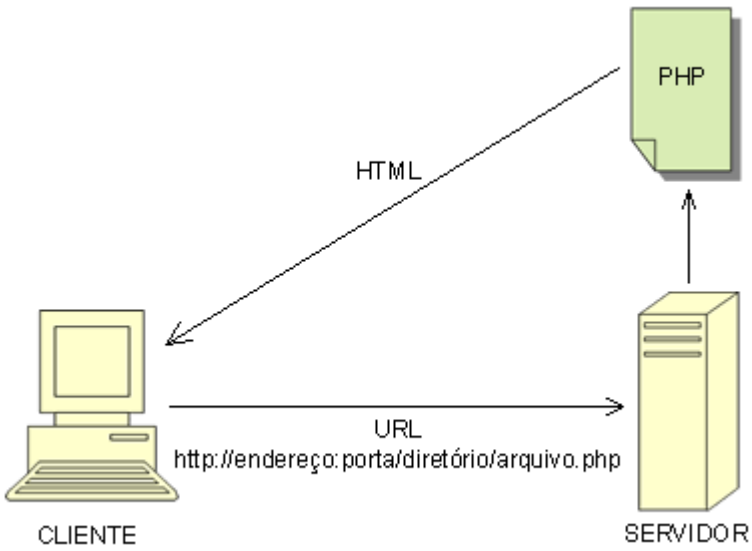
http://www.easyphp.org

MySQL

http://www.mysql.com

http://www.mysql.org

COMO FUNCIONA



CLIENT SIDE X SERVER SIDE

Scripts criados em JavaScript e as applets Java são “executadas” no lado cliente, ou seja, na máquina do usuário.

Os programas criados em PHP (assim como ASP e JSP) são “executados” no lado servidor. Quando o usuário requisita uma página PHP, o servidor interpreta o conteúdo da página e devolve para o cliente uma página HTML, embora com a extensão .php. Isso pode ser verificado no browser através da opção Exibir -> Código Fonte.

ESTRUTURA DE PÁGINAS HTML

```
<html>
<head>
<title>TÍTULO DA PÁGINA</title>
</head>
<body>
PÁGINA HTML
</body>
</html>
```

PÁGINAS HTML COM SCRIPTS PHP INCORPORADOS

```
<html>
<head>
<title>EXEMPLO</title>
</head>
<body>
Linha em HTML<br>
<?
echo "Linha em PHP";
?>
</body>
</html>
```

DELIMITADORES

O PHP deve geralmente estar “embutido” dentro do HTML

Para escrever o código PHP será necessária a utilização de tags no início e no fim de cada seção do PHP.

```
<?
comandos php;
?>
```

ENDEREÇOS PARA EXECUTAR PHP (na máquina local)

```
http://127.0.0.1/nome_do_arquivo.php
http://localhost/nome_do_arquivo.php
```

PROGRAMA DE TESTE USANDO phpinfo()

```
<?
phpinfo();
?>
```

COMENTÁRIOS

- Permitir que outra pessoa leia o seu código e entenda o que você estava pensando enquanto escrevia o código (ainda que esta pessoa seja você mesmo uma semana mais tarde)
- Desabilitar partes ou linhas de programas

Para comentar (ou desabilitar) uma linha:

```
// <? echo "teste1";//comentário ?>
#  <? echo "teste1";#comentário ?>
```

Para comentar (ou desabilitar) mais de uma linha:

```
<?
echo "teste3";
//echo "teste4";//linha desabilitada
/* linhas
comentadas */
?>
```

SEPARADOR DE INSTRUÇÕES

Para separar ou finalizar instruções PHP é necessário o uso de **;**

```
<?
$a=10;//final de instrução
$b="hello world";c=25+$a;//instrução separada por ;
?>
```

COMANDO DE IMPRESSÃO

Um programa em PHP normalmente tem como saída uma página HTML.

Para podemos "imprimir" estes resultados podemos usar os comandos **echo** ou **print**.

```
print(argumento)
echo argumento
echo "argumento"
```

CONSTANTES

DEFINIÇÃO

São valores predefinidos no início do programa e que não mudam ao longo de sua execução. Você pode definir suas próprias constrantes utilizando o comando define.

```
define (<constante>,<valor>);
```

```
<?
define("meunome","Marcos");
define("peso",95);
echo "O meu nome é " .meunome;
echo "<br>";
echo "O meu peso é " .peso. " quilos";
?>
```

VARIÁVEIS

APLICAÇÃO

Usadas para guardar informações que mudam ou variam no decorrer do script.

CARACTERÍSTICAS

- Todas as variáveis no PHP têm um sinal de cifrão ($) na frente
- O valor de uma variável é sempre o mais recente atribuído:

```
<?
$a=2;
$a="teste";
echo $a;//$a vale teste
?>
```

- Variáveis são atribuídas com o operador (=), ficando a variável do lado esquerdo e a expressão ou o valor, à direita:

```
$nome_da_variavel=valor;
```

NOMES

- O PHP é case sensitive, ou seja, ele diferencia maiúsculas de minúsculas

```
<?
$a="2";
$A="teste";
echo $a;//$a vale 2
?>
```

- A melhor maneira de programar em PHP é utilizar um padrão de digitação, para variáveis o recomendado é minúscula.

EXEMPLOS

Elabore um script em PHP para calcular a área de um triângulo.

```
<?
$base=5;
$altura=3;
$area=$base*$altura/2;
echo "A base é: $base<br>";
echo "A altura é: $altura<br>";
echo "A área é: $area<br>";
?>
```

Elabore um script em PHP para apresentar a data atual.

```
<html>
<body>
<?
$data_de_hoje=date("d/m/Y",time());
?>
<p align="center">Hoje é dia <? echo $data_de_hoje; ?></p>
</body>
</html>
```

TIPOS DE VARIÁVEIS

NUMÉRICAS

As variáveis numéricas apresentam valores inteiros ou reais (ponto flutuante).

- Inteiros

```
$x=12345;//inteiro positivo

$x=-123;//inteiro negativo
```

- Ponto flutuante

```
$nota=6.5;//ponto flutuante
```

A variável é definida como numérica no momento que atribuímos um valor numérico a ela.

```
$x=10;

$y=2.5;
```

STRING (ALFANUMÉRICAS)

São cadeias de caracteres que podem ser delimitados por aspas simples (‘) ou duplas (“). Porém, o resultado poderá ser diferente, note os exemplos a seguir:

- Aspas duplas:

```
<?
$texto1="exemplo";
$texto2="Esse $texto1 utiliza aspas duplas";
echo $texto2;
?>
```

- Aspas simples:

```
<?
$texto1="exemplo";
$texto2=’Esse $texto1 utiliza aspas simples’;
echo $texto2;
?>
```

CARACTERES DE ESCAPE

\" Insere no texto o caractere "
\n Nova linha
\r Retorno de carro (carriage return)
\t Tabulação
\$ Insere no texto o símbolo $
\\ Insere no texto o caractere \
Exemplo:

```
<?
$texto="Estou colocando \"aspas duplas\" dentro de uma string. ";
echo $texto;
?>
```

ARRAYS

As variáveis comuns, também chamadas de variáveis escalares, podem armazenar apenas um valor por vez. Um array (vetor) pode armazenar vários valores ao mesmo tempo, pois além de apresentar um identificador (como ocorre com as variáveis), há também um índice associado (que pode ser um número ou um texto). Cada índice indica uma posição de memória em que fica armazenado um elemento do array. O índice aparece entre colchetes “[ ]” logo após o identificador do array.

Os arrays são muito úteis quando precisamos realizar automatização de tarefas em nossos programas.

Analogia:

ARMÁRIO -> GAVETAS -> CORRESPONDÊNCIAS

IDENTIFICADOR -> ÍNDICES -> CONTEÚDO DAQUELE ELEMENTO

Exemplos:

```
<?
$futebol[0]="Sao Paulo";
$futebol[1]="Campeao";
$futebol[2]=2004;
echo $futebol[0];
echo "<br>";
echo $futebol[1];
echo "<br>";
echo $futebol[2];
?>
```

```
<?
$futebol=array(0=>"Sao Paulo",1=>"Campeao",2=>2004);
echo $futebol[0];
echo "<br>";
echo $futebol[1];
echo "<br>";
echo $futebol[2];
?>
```

OPERADORES

Utilizamos operadores para: atribuir valores a uma variável, efetuar operações aritméticas, comparar valores, etc.

TIPOS DE OPERADORES:

- Aritméticos
- Binários
- Comparação
- Atribuição
- Lógicos
- Ternário

ARITMÉTICOS

Utilizados para efetuar operações matemáticas (somar, subtrair, multiplicar, dividir, etc.)

OPERADOR	DESCRIÇÃO
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto de divisão

Há também os operadores unários (que atuam em apenas um operando). São utilizados para troca de sinal, incremento ou decremento de valor, etc.

OPERADOR	DESCRIÇÃO
-op	Troca o sinal do operando.
++op	Pré-incremento. Primeiro incrementa o valor do operando e depois realiza a operação.
--op	Pré-decremento. Primeiro decrementa o valor do operando e depois realiza a operação.
op++	Pós-incremento. Primeiro realiza a operação e depois incrementa o operando.
op--	Pós-decremento. Primeiro realiza a operação e depois decrementa o operando.

Exemplo:

```
<?
$a=2;
$b=3;
$x=++$b-$a;
$y=b--+$a;
echo "a=$a<br>b=$b<br>";
echo "x=$x<br>y=$y<br>";
?>
```

BINÁRIOS

Utilizados para fazer comparações binárias (bit a bit), inverter os bits de um operando, deslocar bits à direita (equivalente a uma divisão por dois), deslocar bits à esquerda (equivalente a multiplicar por dois).

OPERADOR	DESCRIÇÃO
~op1	Inverte os bits do operando.
op1&op2	Operação E (AND) bit a bit.
op1|op2	Operação OU (or) bit a bit.
op1^op2	Operação OU exclusivo (XOR).
op1>>n	Desloca o operando n bits à direita.
op1<<n	Desloca o operando n bits à esquerda.

Exemplo:

```
<?
$a=8;
$x=$a>>1; // desloca 1 bit a direita
echo $x;
?>
```

COMPARAÇÃO

Utilizados para executar comparações entre o valor de duas variáveis, uma variável e um texto (string) ou uma variável e um número. São também chamados de operadores condicionais.

OPERADOR	DESCRIÇÃO
op1==op2	Verdadeiro se op1 for igual a op2.
op1>op2	Verdadeiro se op1 for maior que op2.
op1<op2	Verdadeiro se op1 for menor que op2.
op1>=op2	Verdadeiro se op1 for maior ou igual a op2.
op1<=op2	Verdadeior se op1 for menor ou igual a op2.
op1!=op2	Verdadeiro se op1 for diferente de op2.
op1<>op2	Verdadeiro se op1 for diferente de op2.

ATRIBUIÇÃO

Utilizado para atribuir um valor a uma variável. A variável que receberá a atribuição encontra-se sempre do lado esquerdo do operador e recebe o valor gerado pela expressão ou operador que está à direita. As variações apresentadas a seguir são utilizadas para facilitar a programação.

OPERADOR	DESCRIÇÃO
op1=op2	op1 recebe o valor de op2
op1+=op2	Equivale a op1=op1+op2
op1-=op2	Equivale a op1=op1-op2
op1*=op2	Equivale a op1=op1*op2
op1/=op2	Equivale a op1=op1/op2
op1.=op2	Equivale a op1=op1.op2 (concatenação)
op1%=op2	Equivale a op1=op1%op2
op1<<=op2	Equivale a op1=op1<<op2
op1>>=op2	Equivale a op1=op1>>op2
op1&=op2	Equivale a op1=op1&op2
op1|=op2	Equivale a op1=op1|op2
op1^=op2	Equivale a op1=op1^op2

LÓGICOS

São operadores que retornam o valor verdadeiro (true) ou falso (false).

OPERADOR	DESCRIÇÃO
!op1	Verdadeiro se op1 for falso.
op1 AND op2	Verdadeiro se op1 E op2 forem verdadeiros. Precedência baixa.
op1 OR op2	Verdadeiro se op1 OU op2 forem verdadeiros. Precedência baixa.
op1 XOR op2	Verdadeiro se só op1 OU só op2 for verdadeiro.
op1 && op2	Verdadeiro se op1 E op2 forem verdadeiros. Precedência alta.
op1 || op2	Verdadeiro se op1 OU op2 forem verdadeiros. Precedência alta.

Resultado gerados pelos operadores de acordo com o tipo de expressão avaliada:

Operador AND		
exp1	exp2	resultado
V	V	V
V	F	F
F	V	F
F	F	F

Operador OR		
exp1	exp2	resultado
V	V	V
V	F	V
F	V	V
F	F	F

Operador XOR		
exp1	exp2	resultado
V	V	F
V	F	V
F	V	V
F	F	F

Operador ! (NOT)	
exp1	resultado
V	F
F	V

TERNÁRIO

Forma abreviada de utilizar o comando condicional IF. Uma condição é avaliada, se for verdadeira, atribui-se o primeiro valor à variável, se for falsa atribui-se o outro valor.

condicao ? expressao1 : expressao2

Exemplo:

```
<?
$nota = ($freq>=0.75) ? ($nota+1) : ($nota-1);
?>
```

Equivale a:

```
<?
if ($freq>=0.75)
  $nota = $nota+1;
else
  $nota = $nota-1;
?>
```

ESTRUTURAS DE CONTROLE

COMANDOS CONDICIONAIS:

- if
- switch

IF

Avalia uma expressão e, dependendo do resultado, é executado um conjunto diferente de instruções. Complementos: `elseif` e/ou `else`.

```
<?
$nota=7;
if ($nota<5)
  $avaliacao = "ruim";
elseif ($nota<7)
  $avalizacao = "regular";
elseif ($nota<9)
  $avalizacao = "bom";
else
  $avalizacao = "excelente";
echo "Resultado: $avaliacao";
?>
```

SWITCH

Também avalia o valor de uma expressão para escolher qual bloco de instrução deve ser executado. O `switch` trabalha basicamente com o operador de igualdade, enquanto o `if` trabalha com qualquer tipo de operador.

```
<?
$n=1;
switch ($n)
{
  case 0:
    echo "O número vale zero";
    break;
  case 1:
    echo "O número vale um";
    break;
  case 2:
    echo "O número vale dois";
    break;
}
?>
```

COMANDOS DE REPETIÇÃO:

- while
- do ... while
- for
- foreach

WHILE

Avalia uma expressão e, enquanto essa expressão retornar o valor verdadeiro, a execução do bloco de comando será repetida. Quando o valor retornado for falso, encerra-se o laço de repetição (loop), e a execução é transferida para o fim do comando while.

```
<?
$n=1;
while ($n<=50)
{
  echo "O valor atual é: $n <br>";
  $n++;
}
?>
```

DO ... WHILE

While avalia a expressão no início do laço e do ... while avalia a expressão no final do laço. Portanto, utilizando do ... while o laço será executado pelo menos uma vez, mesmo que a condição não seja verdadeira.

```
<?
$n=5;
do
{
  echo "O valor atual é: $n <br>"; // escreve mesmo que a condição não seja V
  $n++;
}
while (n$<5);
?>
```

FOR

Utilizado quando queremos executar um conjunto de instruções um determinado número de vezes. Exemplo: imprimir todos os elementos de um array ou todos os registros retornados de uma consulta a um banco de dados.

```
for (inicialização; codição; operador)
{
 comandos
}
```

```
<?
$n=1;
for ($n=1;$n<=10;$n++)
{
  echo "O valor atual é: $n <br>";
}
?>
```

FOREACH

Oferece uma maneira prática de “navegar” entre os elementos de um array. Disponível apenas na PHP 4 ou superior.

```
foreach ($nome_array as $elemento)
{
 comandos
}
```

```
<?
/* foreach exemplo 1 */
$vetor=array(1,2,3);
foreach ($vetor as $n)
{
  print "O valor do elemento atual é: $n <br>";
}
?>
```

```
<?
/* foreach exemplo 2 */
$vetor = array(1=>2,2=>4,3=>6);
foreach ($vetor as $n)
{
  echo $n ."<br>";
}
?>

<?
/* foreach exemplo 3: chave e valor */
$a = array ("um" => 1,"dois" => 2,"tres" => 3,"quize" => 15);
foreach ($a as $k => $v) // $k foi atribuida a chave e $v foi atribuido o valor
{
  echo "$k => $v <br>";
}
?>

<?
/* foreach exemplo 4: array multi-dimensional */
$m[0][0] = "a";
$m[0][1] = "b";
$m[1][0] = "y";
$m[1][1] = "z";
foreach ($m as $v1)
{
  foreach ($v1 as $v2)
  {
    print "$v2 <br>";
  }
}
?>

<?
/* foreach exemplo 5: array dinâmico */
foreach (array(1, 2, 3, 4, 5) as $v)
{
  print "$v <br>";
}
```

FUNÇÕES

As funções tornam possível escrever código *reutilizável* e *modular*.

Uma função pode receber *argumentos* (variáveis passadas para a função para uso dentro da função) e retornar um valor (opcional).

Os argumentos são passados normalmente *por valor*, o que significa que *uma cópia dos dados é enviada para a função. Mudanças na cópia da variável não afetam a variável original*.

Alternativamente, os argumentos podem ser passados *por referência*, caso em que a função *não trabalha com uma cópia dos dados*, *mas sim com a própria variável original; assim mudanças na variável persistirão fora da função*.

```
<?
/* Argumentos passados por valor */
function calculo1($n1,$n2){
  $n1+=$n2;
}

$n1=2;
$n2=3;
calculo1($n1,$n2);
echo $n1; // escreve 2
?>
```

```
<?
/* Argumentos passados por referencia */
function calculo2(&$n1,$n2){
  $n1+=$n2;
}

$n1=2;
$n2=3;
calculo2($n1,$n2);
echo $n1; // escreve 5
?>
```

Os argumentos podem ser opcionais, atribuindo-se um valor a eles na declaração da função.

As *variáveis*, dentro de uma função, normalmente têm *alcance local*, significando que elas existem *somente dentro da função* e *não interferem com variáveis fora da função*, mesmo se elas tiverem o mesmo nome.

As variáveis podem acessar variáveis globais usando a instrução **`global`**:

ESCOPO DE UMA VARIÁVEL

Escopo de uma variável define a porção do programa onde ela pode ser utilizada.

Na maioria dos casos todas as variáveis têm escopo global. Entretanto, em funções definidas pelo usuário um ESCOPO LOCAL é criado.

Uma variável de ESCOPO GLOBAL não pode ser utilizada no interior de uma função sem que haja uma declaração explícita.

O exemplo a seguir não produzirá saída alguma, pois a variável $n é de escopo global, e não pode ser referenciada em um escopo local, mesmo que não haja outra com nome igual que cubra sua visibilidade.

```
<?
$n=10;
function numero(){
  print $n;
}

numero();
?>
```

Para que o script funcione, a variável global a ser utilizada deve ser declarada.

```
<?
$n=10;
function numero(){
  global $n;
  print $n;
}

numero();
?>
```

Uma declaração "global" pode conter várias variáveis, separadas por vírgulas.

Uma outra maneira de acessar variáveis de escopo global dentro de uma função é utilizando um array predefinido pelo PHP cujo nome é $GLOBALS.

```
<?
$n=10;
function numero(){
  print $GLOBALS ["n"]; // imprime 10
}

numero();
?>
```

As variáveis locais dentro de uma função são reinicializadas sempre que a função é chamada, a menos que a instrução **static** seja usada, caso em que ela irá reter o último valor da chamada anterior:

O MODIFICADOR static

Utilizando-se o modificador static o valor das variáveis declaradas como estáticas é mantido ao terminar a execução da função. Na próxima execução da função, ao encontrar novamente a declaração como static, o valor da variável é recuperado.

No exemplo abaixo, o último comando da função é inútil, pois assim que for encerrada a execução da função, a variável $n perde seu valor.

```
<?
function teste(){
  $n=10;
  print $n;
  $n++;
}

teste();
teste();
?>
```

No próximo exemplo, a cada chamada da função a variável $n terá seu valor impresso e será incrementada.

```
<?
function teste(){
  static $n=10;
  print $n;
  $n++;
}

teste();
teste();
?>
```

RECURSIVIDADE

Recursividade é quando uma função chama a si mesma. Isto pode levar a algoritmos muito elegantes que adotam a técnica de dividir um problema complexo em passos menores que podem ser repetidos.

Exemplos:

```
<?
function cubo($n) {
  return $n*$n*$n;
}
$n=10;
echo cubo(3)."<br>"; // escreve 27
echo $n; // escreve 10
?>
```

```
<?
function fatorial($n) {
  if ($n<0) {
    return -1;
  }
  if ($n<=1) {
    return 1;
  }
    return n*fatorial($n-1);
}
echo "O fatorial de 3 é: fatorial(3)"; // escreve 6
?>
```

TRABALHANDO COM ARQUIVOS

Arquivos de baixo nível são uma boa alternativa para situações em que precisamos armazenar informações, mas não temos acesso a um banco de dados.

Para utilizarmos os arquivos de baixo nível são necessários os seguintes passos:

1. Abrir o arquivo em modo leitura e/ou gravação;
2. Iniciar o processo de leitura ou gravação;
3. Fechar o arquivo.

Principais funções do PHP para arquivos de baixo nível:

- **fopen**: Abre um arquivo. Retornará false se ocorrer erro.
- **fwrite**: Escreve um número específico de bytes de uma string em um arquivo.
- **fgets**: Retorna uma string do arquivo a partir do ponteiro do arquivo até que (tamanho -1) bytes sejam lidos. A leitura termina quando encontra o fim de uma linha ou quando o arquivo termina.
- **feof**: Retorna true caso o ponteiro de arquivo esteja no fim do arquivo ou ocorra um erro, caso contrário retorna false.
- **fclose**: Fecha um arquivo aberto. Retorna true se obtiver sucesso e false em caso de erro.

A função **fopen** recebe como segundo argumento o ***modo*** de abertura:

- **r** Abre somente para leitura (read-only); coloca o ponteiro no começo do arquivo.
- **r+** Abre para leitura (read) e gravação (write); coloca o ponteiro no começo do arquivo.
- **w** Abre somente para gravação; coloca o ponteiro no começo do arquivo e ***apaga*** o conteúdo que já foi escrito. Se o arquivo não existir, tenta criá-lo.
- **w+** Abre para leitura e escrita; coloca o ponteiro no início do arquivo e apaga o conteúdo que já foi escrito. Se o arquivo não existir, tenta criá-lo.
- **a** Abre o arquivo somente para escrita; coloca o ponteiro no fim do arquivo. Se o arquivo não existir, tenta criá-lo.
- **a+** Abre o arquivo para leitura e gravação; coloca o ponteiro no fim do arquivo. Se o arquivo não existir, tenta criá-lo.

Se você abrir um arquivo, ler e visualizar o conteúdo, depois gravar uma versão editada com o mesmo nome de arquivo, você não pode depender do modo **w+**. Os modos **w** apagam o conteúdo do arquivo imediatamente ao abri-lo, por isso você só pode ler um arquivo **w+** depois de gravar nele.

Para contornar esse problema, você precisa abrir uma vez no modo de leitura e uma vez no modo de gravação:

Exemplo 1 - contador de visitas:

```
<?
$fp = fopen("contador.txt","r"); // abre o arquivo em modo leitura
$n = fgets($fp,255); // obtem a linha do arquivo e atribui seu numero a $n
$n++;
echo "Você é o visitante número: $n";
echo "<br>";
fclose($fp); // fecha o arquivo

$fp = fopen("contador.txt", "w+"); // abre o contador.txt em modo gravação
fwrite($fp,$n); // grava a variável $n no arquivo
fclose($fp); // fecha o arquivo
?>
```

Nota: Criar o arquivo *contador.txt* e gravar no mesmo diretório ou ../<dir>/contador.txt

Exemplo 2 – Registrando o IP do usuário, data e hora:

ip.php

```
<?
$arquivo="ip.txt";
$ip=$_SERVER['REMOTE_ADDR'];
$ver = date("d/m/Y H:i");
$arquivo=fopen($arquivo,"a");
fwrite($arquivo,$ver." - ".$ip."\n");
fclose($arquivo);
?>
```

Exemplo 3 – Leitura de um arquivo (noticias.txt):

learquivo.php

```
<?
// abre arquivo em modo leitura
$fp = fopen("noticias.txt","r");
while (! feof($fp)) { // looping para ler todo o conteúdo do arquivo
$linha = fgets($fp,100); //leitura de cada linha do arquivo
echo $linha;
echo "<br>";
} // fim do looping
fclose($fp); // fecha arquivo
?>
```

noticia.txt

```
Notícia 1
Notícia 2
Notícia 3
Notícia 4
Notícia 5
```

Outras funções importantes para manipulação de arquivos e diretórios:

explode()

divide uma string

http://www.php.net/manual/pt_BR/function.explode.php

opendir()

abre um diretório

http://www.php.net/manual/pt_BR/function.opendir.php

readdir()

lê os arquivos de um diretório

http://www.php.net/manual/pt_BR/function.readdir.php

closedir()

fecha um diretório

http://www.php.net/manual/pt_BR/function.closedir.php

file_exists()

verifica se um arquivo existe

http://www.php.net/manual/pt_BR/function.file-exists.php

FORMULÁRIOS

1° passo - Criar um formulário utilizando a linguagem HTML para enviar os dados:

envia.htm

```
<form action="recebe.php" method="POST">
<p>Digite o seu nome: <input type="text" name="nome"</p>
<p><input type="submit" value="Enviar"></p>
</form>
```

2° passo - Criar uma página PHP para receber os dados:

opção 1 - Tratar os dados enviados como variáveis, colocando o símbolo $ seguido do próprio nome do campo definido no formulário:

recebe.php

```
<?
echo "Seu nome é: $nome";
?>
```

*Para que esta opção funcione a diretiva **register_globals** (do arquivo **php.ini**) deverá conter o valor **on**.*

opção 2 - Utilizar os dois tipos de arrays definidos pelo PHP:

```
$_GET["$nome_do_campo"] // utilizada para method="GET"
$_POST["$nome_do_campo"] // utilizada para method="POST"
```

recebe.php

```
<?
echo "Seu nome é: " . $_POST["nome"];
?>
```

EXEMPLO: Sistema para guardar informações de usuários em arquivos de texto.

```
<!--cadastro.htm-->
<form action="cadastro.php" method="POST">
Nome: <input type="text" size="10" name="nome"><br><br>
E-mail: <input type="text" size="10" name="email"><br><br>
Endereço: <input type="text" size="10" name="endereco"><br><br>
Telefone: <input type="text" size="10" name="telefone"><br><br>
<input type="submit" value="cadastrar">
</form>
```

```
<?
// cadastro.php
// Verifica se todos os campos foram preenchidos
if (!$nome || !$email || !$endereco || !$telefone) {
    echo "preencha todos os campos";
} else {
    // Verifica se um usuário com o mesmo nome ja foi cadastrado
    if(!file_exists($nome . ".txt")) {
        // Cria o arquivo do usuário com w+
        $cria = fopen($nome . ".txt", "w+");
        // Declara as informações do usuário
        // São separadas por | para depois podermos recupera-las com explode
        $dados .= "$nome|";
        $dados .= "$email|";
        $dados .= "$endereco|";
        $dados .= "$telefone";
        // Escreve os dados no arquivo
        $escreve = fwrite($cria,$dados);
        // Fecha o arquivo
        fclose($cria);
        // Exibe a mensagem de usuário cadastrado
        echo "usuário cadastrado com sucesso!";
    } else {
        // Avisa se ja houver um usuário cadastrado com o mesmo nome
        echo "um usuário chamado $nome ja foi cadastrado";
    }
}
?>
```

```
<?
// exibe.php
// Define onde estão os arquivos
// ./ siginifica que os arquivos estão no diretório atual
$dir = "./";
// Abre o diretorio $dir
$abredir = opendir($dir);
// Faz o loop para a exibição de usuários
while ($arqs = readdir($abredir)) {
    // Retira "." e ".." que são "bugs" do readdir()
    // Também faz com que só sejam abertos arquivos de texto
    if ($arqs != "." && $arqs != ".." && is_file($arqs) && ereg(".txt", $arqs)) {
        // Abre arquivo por arquivo, e exibe os dados do usuário
        // Usamos o "r" pois somente queremos ler o arquivo
        $abre = fopen($arqs,"r");
        // Usa fread para ler o arquivo
        $le = fread($abre,filesize($arqs));
        // Separa os dados pelo "|" com explode
        $dado = explode("|",$le);
        // Define os registros
        $nome = $dado['0'];
        $email = $dado['1'];
        $endereco = $dado['2'];
        $telefone = $dado['3'];
        // Mostra os dados obtidos
        echo "Usuário: <b>$nome</b><br>";
        echo "nome: $nome<br>";
        echo "e-mail: $email<br>";
        echo "endereço: $endereco<br>";
        echo "telefone: $telefone<br><br>";
    }
}
// Fecha o diretorio
closedir($abredir);
?>
```

COOKIES

Definição:

Pequenos fragmentos de textos enviados ao browser do usuário.

Objetivo:

Armazenar informações (variáveis) na máquina do usuário e resgatá-las posteriormente.

Aplicação:

- Cesta de compras
- Cadastro de usuários
- Personalização de sites

DEFINIR UM COOKIE

```
setcookie ("name",["value"],["expiration"],["path"],["domain"],["security"]);
```

- **name** - Nome da variável global que será armazenada em HTTP_COOKIE_VARS e que será acessível nas páginas seguintes.
- **value** - Valor atribuído à variável name.
- **expiration** - Tempo durante o qual o cookie estará acessível.
- **path** - Se especificarmos / (barra simples), o cookie será válido para todos os diretórios do servidor web. Se especificarmos /nome_diretorio, o cookie será válido somente neste diretório.
- **domain** - Cookies são válidos somente para o host e domínio que o definiram. Se nenhum domínio for especificado, o valor default será o nome do servidor que o gerou.
- **security** - Se o parâmetro for 1, o cookie será transmitido somente via HTTPS.

```
<?
setcookie("codigo",100,time()+86400,"/");
?>
```

Atribuimos o valor 100 ao cookie codigo que expirará em 24 horas (86400 segundos).

IMPORTANTE: Os cookies fazem parte do cabeçalho HTTP, porisso precisamos chamar setcookie ANTES de enviar qualquer dado ao navegador.

LER UM COOKIE

```
$nome_variavel=$HTTP_COOKIE_VARS["name"];
```

```
<?
$teste=$HTTP_COOKIE_VARS["codigo"];
?>
```

EXEMPLO COMPLETO:

index.php

```
<?
$nome=$HTTP_COOKIE_VARS["nome"];
  if (! isset($nome))
  {
    include "cadastrar.htm";
  }
  else
  {
    include "cadastrado.php";
  }
?>
```

cadastrar.htm

```
<form action="definircookies.php" method="POST">
Nome: <input type="text" name="txnome"><br><br>
RA:   <input type="text" name="txra"><br><br>
<input type="submit" name="submit" value="enviar">
</form>
```

definircookies.php

```
<?
setcookie("nome","$txnome",time()+144400);
setcookie("ra","$txra",time()+144400);
?>
<p>Cookie gravado com sucesso!</p>
<a href="cadastrado.php">Consultar</a>
```

cadastrado.php

```
<?
$nome=$HTTP_COOKIE_VARS["nome"];
$ra=$HTTP_COOKIE_VARS["ra"];
?>
Nome: <? echo $nome; ?><br><br>
RA: <? echo $ra; ?><br><br>
<a href="apagarcookies.php">Apagar cookies</a>
```

apagarcookies.php

```
<?
/*
Para apagar os cookies devemos utilizar a função setcookie,
passando somente o nome do cookie como parâmetro.
*/
setcookie("nome");
setcookie("ra");
?>
<p>Os cookies foram apagados com sucesso!</p>
```

SESSIONS

Período durante o qual um usuário navega pelas páginas de um site. Quando este entra no site, abre-se uma sessão e registra-se nela algumas variáveis que ficarão gravadas no servidor e poderão ser acessadas em qualquer página do site, enquanto a sessão estiver aberta.

- **session id** – identificador único para cada sessão. Para uma página ter acesso aos dados da sessão, ela deve conhecer o *session id*.

Podemos transmitir o session id entre as páginas de duas formas:

- Cookies: Armazena-se o session id em um cookie na máquina do usuário e este é utilizado como referência para acessar os dados da sessão no servidor.
- Propagação na URL: Pode ser feito de duas formas:
  - Ativando-se o parâmetro enable-trans-sid: a identificação é enviada transparentemente entre as páginas.
  - Caso o parâmetro acima não seja habilitado, deve-se acrescentar ao final da URL a constante SID, criada automaticamente pelo PHP no início da sessão.

```
echo '<a hef="pagina.php?' . SID. '">Link</a>;
```

IMPORTANTE: Sempre que possível utilize o primeiro método (cookies), pois permite melhor gerenciamento e proporciona maior segurança.

CRIAR UMA SESSÃO

- **session_start()** – função utilizada para criar uma sessão ou para restaurar os dados de uma sessão com base no identificador (passado por cookie ou pelos métodos GET ou POST).

IMPORTANTE: Chamar a função session_start *antes* de qualquer saída produzida pelo browser.

REGISTRAR VARIÁVEIS EM UMA SESSÃO

- **$_SESSION** – variável superglobal utilizada para registrar uma variável (disponível a partir da versão 4.1.0 do PHP).

EXEMPLO 1:

```
<?
session_start();
if (! isset($_session['contador']){
  $_SESSION['contador']=1;
} else {
  $_SESSION['contador']++;
}
?>
```

EXEMPLO 2:

```
<?
// pagina1.php
session_start();
echo 'pagina 1';
  $_SESSION['data']= date('d/m/Y', time());
  $_SESSION['nome']= 'Fulano';
  $_SESSION['ra']= '123';
echo '<br><a href="pagina2.php">Página 2</a>';
?>
```

Opcão: Para propagar o identificador através da URL deve-se alterar a última linha:

```
echo '<br><a href="pagina2.php'.SID.'">Página 2</a>';
```

```
<?
// pagina2.php
session_start(); // necessário para restaurar os dados da sessão atual
echo 'pagina 1';
echo $_SESSION['data']. "<br>";
echo $_SESSION['nome']. "<br>";
echo $_SESSION['ra']. "<br>";
echo '<br><a href="pagina1.php">Página 1</a>';
?>
```

MySQL

INTRODUÇÃO

Executar o EasyPHP ou Triad

c:\ ...\mysql\bin\mysql –u <usuario> –p <senha> <banco_de_dados>

c:\ ...\mysql\bin\mysql –u root –p

mysql>

PRINCIPAIS COMANDOS

mysql> \h

\h help	Mostra a tela de saída de ajuda (somente a relação de comandos).
? help	Sinônimo para help.
\c clear	Limpa o Buffer.
\r connect	Reconecta o servidor. Pode receber dois parâmetros <nome_db> e <host>.
\G ego	Envia um comando para o MySQL e exibe os resultados verticalmente.
\q exit	Encerra o MySQL. O mesmo que quit.
\g go	Envia o comando atual para o MySQL.
\t notee	Não envia para o arquivo de saída.
\p print	Imprime o comando atual.
\q quit	encerra o MySQL.
\# rehash	Refaz um Hash finalizado.
\. source	Executa uma relação de comandos SQL existente em um arquivo. Deve-se informar após o comando o nome do arquivo. Exemplo: \.script.sql
\s status	Exibe informações sobre o MySQL.
\T tee	Envia tudo para um arquivo de saída informado após o comando.
\u use	Indica o banco de dados que deve ser utilizado. Deve-se informar o nome do banco de dados. Exemplo: u\ cadastro

mysql> SELECT CURRENT_DATE();

```
2004-05-13
```

TIPOS DE DADOS

- Numérico
- Data / Hora
- Strings

NUMÉRICO

TINYINT [(M)] [UNSIGNED] [ZEROFILL]

Inteiro variando de -128 a 127 ou 0 a 255 se o parâmetro UNSIGNED for utilizado (neste caso aceitará somente números positivos).

SMALLINT [(M)] [UNSIGNED] [ZEROFILL]

Inteiro variando de -32768 a 32777 ou 0 a 65355 (UNSIGNED).

MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]

Inteiro variando de -8388608 a 8388607 ou 0 a 16777215 (UNSIGNED).

INT [(M)] [UNSIGNED] [ZEROFILL]

Inteiro variando de -2147483648 a 2147483647 ou 0 a 4294967295.

INTEGER [UNSIGNED] [ZEROFILL]

Sinônimo de INT.

BIGINT [(M)] [UNSIGNED] [ZEROFILL]

Inteiro variando de 9223372036854775808 a 9223372036854775807 ou 0 a 18446744073709551615

FLOAT [(M,D)] [ZEROFILL]

Flutuante de precisão simples, tendo os seguintes intervalos aceitos:
-3.402823466E+38 a -1.175494351E-38, 0 e 1.175494351E-38 a 3.402823466E+38

DOUBLE [(M,D)] [ZEROFILL]

Flutuante de precisão dupla, tendo os seguintes intervalos aceitos:
-1.7976931348623157E+308 a – 2.2250738585072014E-308, 0 e
2.2250738585072014E-308 a 1.7976931348623157E+308

REAL [(M,D)] [ZEROFILL]

Sinônimo de DOUBLE.

DECIMAL [(M[,D])] [ZEROFILL]

Flutuante com característica especial: cada número ocupa um byte (comporta-se como um dado do tipo CHAR).

NUMERIC [(M[,D])] [ZEROFILL]

Sinônimo de DECIMAL.

M – O tamanho máximo a ser exibido. O maior valor possível é 255.

D – O número máximo de dígitos decimais a serem exibidos. O valor máximo é 30, porém não pode ser maior que M-2.

ZEROFILL – Indica o preencimento de zeros à esquerda do valor: 5 será representado por 00005, dependendo do valor de M.

UNSIGNED – Indica que somente valores não negativos serão aceitos (maior ou igual a zero).

DATA / HORA

DATE

AAAA-MM-DD (intervalo aceito: 1000-01-01 A 9999-12-31).

DATETIME

AAAA-MM-DD HH-MM-SS (intervalo aceito: 1000-01-01 00:00:00 a 9999-12-31 23:59:59

TIMESTAMP

Armazena uma determinada data em segundos contados à partir de 1º de janeiro de 1970 às 00:00:00h (padrão UNIX).

TIME

HH:MM:SS (intervalo aceito: -838:59:59 a 838:59:59).

YEAR

Armazena um ano qualquer podendo ser de dois ou quatro dígitos.

STRING

CHAR

1 a 255 caracteres (tamanho fixo)

VARCHAR

1 a 255 caracteres (tamanho variável)

TINYTEXT

1 a 255 caracteres

TEXT

Até 65.535 caracteres

MEDIUMTEXT

Até 16.777.215 caracteres.

LONGTEXT

Até 4.294.967.295

TINYBLOB, BLOB, MEDIUMBLOB e LONGBLOB (Binary Large Object)

Armazenam dados em formato binário, i.e., são case sensitive na comparação e classificação de seus conteúdos (os tipos text não são case sensitive). Têm as mesmas limitações que seus tipos TEXT correspondentes.

ENUM (‘valor_1’,’valor_2’, ... ,’valor_n’)

Permite determinar uma lista de valores válidos para o campo, que somente aceitará um dos valores listados, o valor NULL ou o valor “” (em branco). O número máximo de itens da lista de valores é 65.535.

SET (‘valor_1’,’valor_2’, ... ,’valor_n’)

Permite determinar uma lista de valores válidos para o campo, sendo que podem ser escolhidos 0, 1 ou vários dos itens da lista. O número máximo de itens da lista do tipo SET é de 64.

OPERADORES

ARITMÉTICOS

Realizam operações matemáticas básicas.

OPERADOR	DESCRIÇÃO
+	soma
-	subtração
*	multiplicação
/	divisão

LÓGICOS

Realizam comparações entre expressões, retornando verdadeiro ou falso.

OPERADOR	DESCRIÇÃO
NOT	Negação: retorna F se o resultado for V, V se o resultado for F. Retorna NULL se o resultado for NULL.
OR	OU: se uma das expressões for V, retorna V.
AND	E: se uma das expressões for F, retorna F.
!	O mesmo que NOT.
||	O mesmo que OR.
&&	O mesmo que AND.

BINÁRIOS

Realizam operações em nível de bits (0 e 1).

OPERADOR	DESCRIÇÃO
|	OU: caso um dos bits seja 1, retorna 1.
&	E: os dois bits precisam ser 1 para retornar 1.
<<	Deslocamento à esquerda. Equivale a multiplicar por dois.
>>	Deslocamento à direita. Equivale a dividir por dois.
~	Inversão: caso o bit seja 1 será convertido em O e vice-versa.

COMPARAÇÃO

Avalia valores literais, numéricos, strings, expressões e campos de uma tabela ou uma conjunção destes elementos.

OPERADOR	DESCRIÇÃO
=	IGUAL: op1=op2, retorna V se os dois operandos forem igual.
<>	DIFERENTE: op1<>op2, retorna V se os operandos não forem iguais.
!=	O mesmo que <>
>	MAIOR: op1>op2, retorna V se op1 for maior que op2.
>=	MAIOR OU IGUAL: op1>=op2, retorna V se op1 não for menor que op2.
<	MENOR: op1<op2, retorna V se op1 for menor que op2.
<=	MENOR OU IGUAL: op1<=op2, retorna V se op1 não for maior que op2.
<=>	IGUAL: considera o valor nulo (NULL) em sua comparação.
IS NULL	NULO: op1 IS NULL, retorna V se op1 for nulo.
IS NOT NULL	NÃO NULO: op1 IS NOT NULL, retorna V se op1 não for nulo.
BETWEEN ... AND	op1 BETWEEN op2 AND op3, retorna V se op1 for maior ou igual a op2 e menor ou igual a op3. Equivale a: op1>=op2 AND op1<=op3.
IN (<lista>)	op1 IN (valor1,valor2,...), retorna V se op1 estiver na lista.
NOT IN (<lista>)	op1 NOT IN (valor1,valor2,...), retorna V se op1 não estiver na lista.
ISNULL()	NULO: ISNULL(op1), retorna V se op1 for nulo.
COALESCE()	COALESCE(op1,op2,...), retorna o primeiro valor não nulo da lista.
INTERVAL()	INTERVAL(op,op1,op2,...), retorna zero se op<op1, 1 se op<op2, 2 se op<op3 ... A lista deve estar em ordem não decrescente: op1<=op2<=op3 ...

CONVERSÃO

OPERADOR	DESCRIÇÃO
BINARY	Converte uma string em uma string binária. Desta forma somente strings realmente idênticas serão comparadas como verdadeiras. `SELECT BINARY "Teste" = "teste";` Retorna 0 (falso).

CRIANDO UM BANCO DE DADOS

```
mysql> CREATE DATABASE nome_do_DB;

mysql> CREATE DATABASE exemplo;
```

CRIAR UM BANCO DE DADOS A PARTIR DE UM SCRIPT (ARQUIVO TEXTO)

```
mysql –u root –p nome_do_DB < c:\nome_arquivo.sql
```

EXCLUINDO UM BANCO DE DADOS

```
mysql> DROP DATABASE nome_do_DB;

mysql> DROP DATABASE exemplo;
```

LISTAR OS BANCO DE DADOS

```
mysql> SHOW DATABASES;
```

ESPECIFICAR UM BANCO DE DADOS

```
mysql> USE nome_do_DB;
```

CRIAR UMA TABELA

```
CREATE TABLE clientes (
  código VARCHAR(5) NOT NULL,
  nome VARCHAR (50)  NOT NULL,
  endereco VARCHAR (50) NULL,
  PRIMARY KEY(codigo)
);
```

LISTAR TODAS AS TABELAS

```
SHOW TABLES;
```

APAGAR UMA TABELA

```
DROP TABLE nome_da_tabela;
```

ALTERAR UMA TABELA

ADICIONAR COLUNA

```
ALTER TABLE nome_da_tabela
ADD COLUMN nome_nova_coluna tipo [(tamanho)] [FIRST | AFTER] nome_coluna;
```

ADICIONAR *PRIMARY KEY*

```
ALTER TABLE nome_da_tabela
ADD PRIMARY KEY (nome_coluna);
```

ADICIONAR *UNIQUE*

```
ALTER TABLE nome_da_tabela
ADD UNIQUE [nome_índice] (nome_coluna);
```

ADICIONAR *INDEX*

```
ALTER TABLE nome_da_tabela
ADD INDEX nome_indice(nome_coluna);
```

ADICIONAR *FOREIGN KEY*

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT simbolo FOREIGN KEY(nome_coluna)
REFERENCES nome_tabela(nome_coluna_pk);
```

Nota: Criará também um `INDEX` com o mesmo valor (`nome`) colocado em `simbolo`.

ELIMINAR *PRIMARY KEY*

```
ALTER TABLE nome_da_tabela
DROP PRIMARY KEY;
```

ELIMINAR *INDEX*

```
ALTER TABLE nome_da_tabela
DROP INDEX nome_indice;
```

ELIMINAR *FOREIGN KEY*

```
ALTER TABLE nome_da_tabela
DROP FOREIGN KEY simbolo;
```

Nota: Para saber o valor de simbolo utilizar o seguinte comando:

```
SHOW CREATE TABLE nome_da_tabela;
```

O valor é o que aparece ao lado direito de CONSTRAINT.

Exemplo: `CONSTRAINT 'fk_cliente_pedido'`

ALTERAR NOME DA COLUNA

```
ALTER TABLE nome_da_tabela
CHANGE nome_coluna novo_nome_coluna tipo [(tamanho)];
```

ELIMINAR UMA COLUNA

```
ALTER TABLE nome_da_tabela
DROP [COLUMN] nome_coluna;
```

ALTERAR NOME DA TABELA

```
ALTER TABLE nome_da_tabela
RENAME [TO] novo_nome_da_tabela;
```

APRESENTAR DETALHES (ESTRUTURA) DE UMA TABELA

```
DESCRIBE nome_da_tabela; ou DESC nome_da_tabela;
```

APRESENTAR OS ÍNDICES DE UMA TABELA

```
SHOW INDEX FROM nome_da_tabela;
```

INSERIR DADOS (POLULAR) NUMA TABELA

```
INSERT INTO nome_da_tabela (nome_coluna_1,...,nome_coluna_n) VALUES
("valor_11",..., "valor_n1"),
("valor_12",..., "valor_n2");

INSERT INTO cliente (cd_cliente,nm_cliente) VALUES
(1,"Antonio"),
(2,"Beatriz");
```

Nota: Dados do tipo *string* devem ser inseridos entre aspas simples ou duplas.

APRESENTAR TODOS OS DADOS DE UMA TABELA

```
SELECT * FROM clientes;
```

ELIMINAR DADOS DE UMA TABELA

```
DELETE FROM nome_da_tabela
WHERE condicao;

DELETE FROM cliente
WHERE cd_cliente=10;
```

REDIRECIONAR A SAÍDA PARA UM ARQUIVO HTML

```
mysql -u root -p -H -e "select * from nome_da_tabela" nome_do_DB > arquivo.html
```

CADASTRO DE ALUNOS

MySQL

---

1. Entrar no phpmyadmin:

   http://localhost/phpmyadmin

2. Criar o Banco de Dados: **teste**

3. Criar a tabela: **alunos**, com a seguinte estrutura:

ra	VARCHAR	04	PK
nome	VARCHAR	30	

4. Inserir cinco registros na tabela (utilizando o phpmyadmin).

PHP

```
<?
//exibir.php
$conexao=mysql_connect("localhost","root","");
$db=mysql_select_db("teste");
$resultado=mysql_query("SELECT * FROM alunos");
mysql_close($conexao);
while ($linha=mysql_fetch_array($resultado)){
  $ra = $linha["ra"];
  $nome = $linha["nome"];
echo "<font face=Verdana>";
echo "RA: $ra <br>";
echo "Nome: $nome <br>";
echo "<hr>";
}
?>
```

```
<?
//cadastra.php
echo "<h1><center>Sistema de Cadastro de Alunos</center></h1>";
echo "<hr><br>";
echo "<form action='inserir.php' method='post'>";
echo "RA: <input type='text' name='ra' size=5> *<br>";
echo "Nome: <input type='text' name='nome' size=30> *<br>";
echo "<input type='submit' value='Cadastrar'>";
echo "</form>";
echo "<br><hr>";
echo "<i>Campos marcados com <b>*</b> são obrigatórios";
?>
```

```
<?
//inserir.php
$conexao=mysql_connect("localhost","root","") or die ("Erro ...");
$db=mysql_select_db("teste") or die ("Erro ...");
$resultado=mysql_query("INSERT INTO alunos (ra, nome) VALUES ('$ra', '$nome')") or die
("Erro ...");
mysql_close($conexao);
echo "<center>CADASTRO EFETUADO COM SUCESSO!</center>";
?>
```

```
<?
//controle.php
$conexao=mysql_connect("localhost","root","");
$db=mysql_select_db("teste");
$resultado=mysql_query("SELECT * FROM alunos");
mysql_close($conexao);
echo "<table width=740 border=1 cellpadding=1 cellspacing=1>";
echo "<tr>";
echo "<th witdh=100>RA</th>";
echo "<th witdh=200>NOME</th>";
echo "<th witdh=100>ALTERAR</th>";
echo "<th witdh=100>EXCLUIR</th>";
echo "</tr>";
while ($linha=mysql_fetch_array($resultado)){
  $ra = $linha["ra"];
  $nome = $linha["nome"];
echo "<tr>";
echo "<th witdh=100>$ra<br></th>";
echo "<th witdh=200>$nome<br></th>";
echo "<th witdh=100><a href='alterar.php?ra=$ra'>Alterar</a><br></th>";
echo "<th witdh=100><a href='excluir.php?ra=$ra'>Excluir</a><br></th>";
echo "</tr>";
echo "<br>";
}
echo "</table>";
?>


<?
//excluir.php
$conexao=mysql_connect("localhost","root","");
$db=mysql_select_db("teste");
$resultado=mysql_query("DELETE FROM alunos WHERE ra='$ra'");
mysql_close($conexao);
echo "<center>O aluno foi excluido!</center>";
?>


<?
//alterar.php
$conexao=mysql_connect("localhost","root","");
$db=mysql_select_db("teste");
$resultado=mysql_query("SELECT * FROM alunos WHERE ra='$ra'");
mysql_close($conexao);
while ($linha=mysql_fetch_array($resultado)){
  $ra = $linha["ra"];
  $nome = $linha["nome"];
echo "<center><h1>Alterar Cadastro</h1></center>";
echo "<hr><br>";
echo "<form action='alterar_db.php?ra=$ra' method='post'>";
echo "RA: <input type='text' name='ra_novo' value='$ra' size=10><br>";
echo "Nome: <input type='text' name='nome_novo' value='$nome' size=30><br>";
echo "<input type='submit' value='Alterar'>";
echo "</form>";
echo "<br><hr>";
}
?>


<?
//alterar_db.php
$conexao=mysql_connect("localhost","root","");
$db=mysql_select_db("teste");
$resultado=mysql_query("UPDATE alunos SET ra='$ra_novo', nome='$nome_novo' WHERE
ra='$ra'");
echo "<center>Cadastro alterado com sucesso!</center>";
?>
```

```
<?
//pesquisar1.php
echo "<h1><center>Sistema de Cadastro de Alunos</center></h1>";
echo "<hr><br>";
echo "<form action='pesquisar2.php' method='post'>";
echo "RA: <input type='text' name='ra' size=5><br>";
echo "Nome: <input type='text' name='nome' size=20><br>";
echo "<input type='submit' value='Pesquisar'>";
echo "</form>";
echo "<br><hr>";
?>
```

```
<?
//pesquisar2.php
$conexao=mysql_connect("localhost","root","");
$db=mysql_select_db("teste");
if ($nome){
$resultado=mysql_query("SELECT * FROM alunos WHERE nome LIKE '%$nome%'");}
elseif ($ra){
$resultado=mysql_query("SELECT * FROM alunos WHERE ra = $ra");}
mysql_close($conexao);
while ($linha=mysql_fetch_array($resultado)){
$ra = $linha["ra"];
$nome = $linha["nome"];
echo "<font face=Verdana>";
echo "RA: $ra <br>";
echo "Nome: $nome <br>";
echo "<hr>"; }
?>
```

EXERCÍCIOS

1. Criar um programa em PHP para calcular a média de um determinado aluno, conforme a seguinte fórmula: media = (nota1 + nota2)/2.

2. Criar um **array** com os nomes dos meses para exibir a data atual no formato abaixo:

   10 de março de 2004

3. Criar um programa em PHP para calcular a média de um determinado aluno, conforme a seguinte fórmula: media = (nota1 + nota2)/2. Se a média for menor que 5 (cinco) imprimir REPROVADO, se for igual ou maior que 5 imprimir APROVADO.

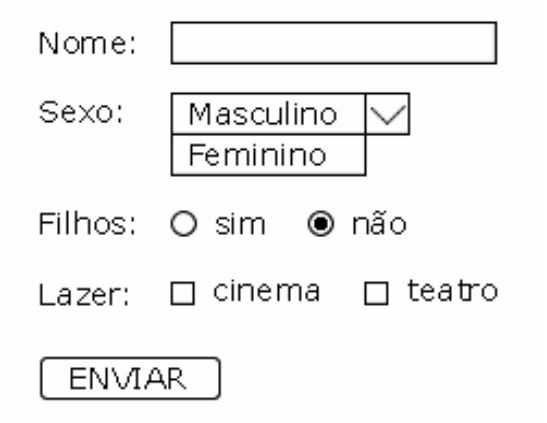
4. Fazer quatro programas em PHP utilizando os comandos de repetição **while**, **do ... while**, **for** e **foreach** para gerar a tabuada de um determinado número. A tela de saída deverá ser a seguinte:

   5 x 1 = 5
   5 x 2 = 10
   ...

   5 x 10 = 50

5. Crie uma página PHP para apresentar um menu de restaurante. Os itens e seus preços devem ser definidos através de um vetor associativo (array), cujas chaves (índices) serão os nomes dos itens. Devem ser apresentadas duas listas: uma ordenada por produtos e a outra por preços. Consulte alguma referência de PHP para descobrir que funções de ordenação (sorting) usar.

6. Criar o arquivo html conforme modelo abaixo e a página PHP para receber os dados do **formulário**.



7. Criar um catálogo de produtos. As páginas deverão ser compostas de um cabeçalho (no topo) com o nome e o logo da empresa. Um menu (à esquerda) deverá apresentar as categorias dos produtos vendidos (exemplo: informática, vídeo, som, etc.) Na seção principal (à direita do menu) deverão ser exibidos os seguintes detalhes: código, nome, descrição e foto de cada produto. Para página de abertura (home page) deverão aparecer 5 (cinco) ou 6 (seis) produtos que estarão em promoção.

APÊNDICE A

APACHE – CRIAR UM DIRETÓRIO PROTEGIDO POR SENHA

1. Criar o seguinte diretório:

```
c:\senhas
```

2. No prompt de comando, localizar o diretório onde encontra-se o arquivo *htpasswd.exe*

O caminho costuma ser o seguinte:

```
c:\apache\bin
```

3. Digitar o seguinte comando:

```
htpasswd -c c:\senhas\acesso aluno
```

Este comando cria no diretório *senhas* um arquivo chamado *acesso* e um usuário *aluno*.

Será solicitado que você insira a senha duas vezes.

4. Na pasta *htdocs* do Apache crie o diretório *teste*

Este é o diretório a ser protegido.

5. Abra o Windows Explorer e localize: *c:\ ... \apache\conf*

6. Abra com o bloco de notas o arquivo *httpd.conf*

7. Posicione o cursor ABAIXO do seguinte trecho:

```
<Directory "C:/xampplite/tmp">
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

A linha <*Directory "C:/xampplite/tmp"*> poderá ser diferente, dependendo do local onde o Apache foi instalado e do pacote escolhido (xampplite, easyphp, triad, etc.)

8. Digite o seguinte:

```
<Directory "C:/xampplite/htdocs/teste">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
    AuthAuthoritative on
    AuthName "Acesso Restrito"
    AuthType Basic
    AuthUserFile c:\senhas\acesso
    require user aluno
</Directory>
```

9. Salve o arquivo *httpd.conf*.

10. Encerre e re-inicialize o Apache. (Apache->Stop e Apache->Start)

11. Ao tentar acessar o *teste* será solicitado o usuário e a senha.

APÊNDICE B

WEB SERVICES

Um Web Service é uma aplicação que usa os protocolos padrão da Internet, para que se torne possível a comunicação transparente de máquina-máquina e aplicação-aplicação.

INTRODUÇÃO

Quando estamos trabalhando dentro de um ambiente controlado como as redes corporativas da nossa empresa, é possível criar aplicações que sejam compartilhadas por diversos servidores e desktops. Podemos inclusive escolher qual o melhor modelo de objetos para se construir tais aplicações, como o uso do COM+ da Microsoft, CORBA da OMG ou mesmo EJB (Enterprise JavaBeans) da Sun.

O problema aparece quando pretendemos utilizar a infra-estrutura da Internet para executar remotamente tais objetos. Como trabalham em formato binário, estes objetos exigem uma porta de comunicação exclusiva, o que gera problemas quando precisamos atravessar um firewall, por exemplo. Por questões de segurança estas portas são fechadas, pois oferecem riscos de invasão à rede interna.

Nos últimos anos, uma linguagem para troca de dados entre aplicativos vinha ganhando grande aceitação do mercado, a linguagem XML. O XML oferece uma série de benefícios em relação aos formatos antigos. Para começar, além do próprio dado, ele leva ainda a descrição do mesmo. Por ser arquivo texto, pode ser interpretado em qualquer plataforma ou sistema operacional e não apresenta problemas quando encontra um firewall, já que não apresenta risco de segurança. Por último, é um padrão controlado por um órgão independente, o W3C (World Wide Web Consortium).

Com tantos atributos o XML foi a base natural para criação do SOAP (Simple Object Application Protocol), modelo de objetos utilizados nos Web Services. As chamadas aos métodos públicos do objeto são realizados através de um arquivo XML, da mesma forma que a resposta a requisição também utiliza o mesmo formato. Contém diversas seções opcionais que descrevem os atendimentos do método (RPC) e que detalham a emissão de mensagens do SOAP sobre o HTTP.

O SOAP é um padrão aberto criado pela Microsoft, Ariba e IBM para padronizar a transferência de dados em diversas aplicações, por isso, utiliza XML.

Mais dois padrões foram criados em cima destes, o WSDL (Web Service Description Language), que descreve como o Web Service funciona (por sinal um arquivo XML), e o UDDI (Universal Description, Discovery and Integration), a "páginas amarelas" de Web Services, uma especificação para publicação e localização de informações sobre os serviços Web. Um serviço onde é possível cadastrar o seu serviço, categorizado, dentro dos serviços UDDI da Internet. Hoje este serviço é oferecido pela IBM, HP, SAP e Microsoft através do http://uddi.microsoft.com , mas novos serviços estão aparecendo a cada dia.

A especificação UDDI se baseia no protocolo de acesso a objeto simples (SOAP), a linguagem de marcação estensível (XML) e padrões de protocolo HTTP/S desenvolvidos pelo consórcio da World Wide Web (W3C) e a força tarefa de engenharia da Internet (IETF).

Resumindo, os Web Services são componentes de software que são chamados a partir de outros aplicativos. São "páginas web" para outros computadores e não para seres humanos como as páginas HTML tradicionais. É a tecnologia que permite que computadores na Internet conversem entre si sem a intervenção direta dos usuários.

Com toda esta infra-estrutura de tecnologia disponível, os Web Services estão se tornando uma nova opção de negócio para as empresas que planejam utilizar a Internet como meio de troca de informações. Os Web Services estão oferecendo uma nova proposição de negócios às empresas de tecnologia e abrindo um novo universo de oportunidades.

APLICAÇÕES E VANTAGENS

O uso de Web Service permite acessar:

- rotinas de validação de cartão de crédito
- endereçamento postal (CEP)
- cotação de preços
- calcular valores de fretes
- news de empresas, etc.

O Web Service é transparente para o Firewall de uma empresa, pois, como é uma string XML, então é interpretada como um arquivo "texto". Sendo assim, não é preciso pedir autorização do Firewall para entrar.

Exemplo: Servidor para consultar informações sobre CEP (Código de Endereçamento Postal).

1. MySQL - Criar o banco de dados e a tabela abaixo:

```
CREATE DATABASE cep;

USE cep;

CREATE TABLE cep (
  cep int(8) unsigned zerofill NOT NULL default '00000000',
  tipo varchar(10),
  logradouro varchar(30),
  bairro varchar(30),
  localidade varchar(30),
  uf varchar(2));

INSERT INTO cep VALUES (01311000,'Av','Paulista','Bela Vista','Sao Paulo','SP');
```

2. Baixar e descompactar o arquivo nusoap-0.6.4.zip:

http://dietrich.ganx4.com/nusoap/downloads/nusoap-0.6.4.zip

A classe NUSOAP prove conexão com o protocolo SOAP (Simple Object Access Protocol).

3. Criar o arquivo **server_cep.php**:

```
<?
/ SOAP Server (Web Service CEP´s).

// include nusoap classe
require('nusoap.php');

// instância do Server
$obServer = new soap_server();

$obServer->register('QueryByCep');

function QueryByCep($input_cep) {
    if (is_int($input_cep)){
        $host = 'localhost';
        $db = 'cep';
        $user = 'root';
        $passwd = '';

        $obDB = @mysql_connect($host, $user, $passwd);
        $DBResult = @mysql_db_query($db, 'SELECT tipo,logradouro, bairro, localidade, uf FROM
cep WHERE cep ="'.$input_cep.'" LIMIT 1');

        // Checa a conexão com Banco.
        if (!$DBResult ) {
            return new soap_fault('Server', '', 'Ocorreu um erro interno do servidor.');
        }

        // se não retornar nenhum registro
        if (!mysql_num_rows($DBResult)) {
            return new soap_fault('Server', '', "Não foi encontrado nenhum endereço com o CEP:
$input_cep");
        }
        mysql_close($obDB);

        // retorna dados
        return mysql_fetch_array($DBResult, MYSQL_ASSOC);

    } else{
        return new soap_fault('Client', '', 'Serviço requer somente numeros no parametro de
pesquisa. ');
    }
}
$obServer->service($HTTP_RAW_POST_DATA);

?>
```

4. Criar o arquivo **client.php**:

```
<?
/ SOAP Client (client.php).

// use form data
if ((string)$_GET['action'] == 'get_data') {
    // include nusoap classe
    require('nusoap.php');

    // seta paramento e instacia o client.
    $Param = array($_POST['cep']);
    $obClient  = new soapclient('http://127.0.0.1/webservice/server_cep.php');

    // Faz a chamada para o webmetodo(QueryByCep)
    $Result = $obClient->call('QueryByCep', $Param);

    // checa se houve erro
    if (!$obClient->getError()) {
        // print resultados
        print '<h1>Dados do CEP: ' .$Param[0]
            . '</h1><ul><li>tipo: '   .$Result['tipo']
            . '</li><li>Logradouro: '  .$Result['logradouro']
            . '</li><li>Bairro: '     .$Result['bairro']
            . '</li><li>Localidade: ' .$Result['localidade']
            . '</li><li>UF: '         .$Result['uf']
            . '</li></ul>';
    }
    // print descricao do erro
    else {
        echo '<h1>Error: ' . $obClient->getError() . '</h1>';
    }
}

print '<form name="input" action="'.$_SERVER['PHP_SELF'].'?action=get_data" method="POST">
CEP : <input type="text" name="cep">  <input type="submit" value="Procurar"> </form> ';
?>
```

SITES RECOMENDADOS

http://php.scriptbrasil.com

http://phpbrasil.com

http://php.planetmirror.com/manual/pt_BR/

BIBLIOGRAFIA

Niederauer, J. “Desenvolvendo Websites com PHP”, Novatec, 2004

_____ “Integrando PHP 5 com MySQL”, Novatec, 2005

_____ “PHP para Quem Conhece PHP”, Novatec, 2004

Muto, C. A. “PHP & MySQL – Guia Avançado”, Brasport, 2004

Ullman, L. “PHP para a World Wide Web”, Campus, 2001

CERLI, A. R. “Desenvolvendo Web Sites Dinâmicos: PHP, ASP, JSP”, Campus, 2003