

PROGRAMANDO PARA WEB COM PHP/MySQL

autor: **Fred Cox Junior**

Agosto/2000





ÍNDICE

CAPÍTULO I - INTRODUÇÃO	
1. CLIENT-SIDE SCRIPTS	1
2. SERVER-SIDE SCRIPTS	1
3. O QUE É PHP	2
4. UMA BREVE HISTÓRIA DO PHP	3
5. ASP X PHP	3
6. INSTALAÇÃO EM AMBIENTE WINDOWS	4
6.1 SERVIDOR APACHE	4
6.2 PHP3	5
6.3 MYSQL	7
 CAPÍTULO II - MYSQL	
1.O BANCO DE DADOS MYSQL	8
2. CARACTERÍSTICAS DO MYSQL	9
3. HABILITANDO O MYSQL PARA CONEXÕES	9
4. SISTEMA DE SEGURANÇA DO MYSQL	10
5. GUIA DE REFERÊNCIA DO MYSQL	12
5.1 MYSQL MONITOR	12
5.2 COMO ESCREVER STRINGS E NÚMEROS	13
5.3 TIPOS DE COLUNAS SUPORTADOS PELO MYSQL	14
5.4 OPERADORES	16
5.5 COMANDOS	18
5.6 CRIANDO USUÁRIOS NO MYSQL	22
 CAPÍTULO III - A LINGUAGEM HTML	
1. INTRODUÇÃO	26
2. CRIANDO DOCUMENTOS HTML	26
2.1 HTML MÍNIMO	26
2.2 MARCAÇÕES BÁSICAS	27
3. INTERLIGANDO DOCUMENTOS	29
4. INTERLIGANDO DOCUMENTOS EM OUTRO DIRETÓRIO	29
5. FORMULÁRIOS HTML	31
5.1 CODIFICAÇÃO BÁSICA	32
5.2 ENTRADA DE TEXTO COMUM -TEXT	33
5.3 ENTRADA DE TEXTO PROTEGIDO - PASSWORD	34
5.4 ENTRADA DE VÁRIAS LINHAS DE TEXTO - TEXTAREA	35
5.5 LISTBOX E COMBO BOX	36
5.6 CHECKBOX	37
5.7 RADIO BUTTON	39
5.8 SUBMIT BUTTON E RESET BUTTONX	40
5.9 CONCLUSÃO	41

CAPÍTULO IV - A LINGUAGEM PHP	
1. SINTAXE BÁSICA.....	42
2. VARIÁVEIS.....	43
3. COMENTÁRIOS.....	43
4. TIPOS DE DADOS.....	43
4.1 INTEGER	43
4.2 FLOATING-POINT	43
4.3 ARRAY	44
4.4 STRING	44
4.5 CONSTANTES	44
4.6 OPERADORES	45
4.6.1 ARITMÉTICOS	45
4.6.2 STRINGS	45
4.6.3 LÓGICOS	45
4.6.4 COMPARAÇÃO	46
5. ESTRUTURAS DE CONTROLE.....	46
5.1 IF AND ELSE	46
5.2 LAÇO WHILE	47
5.3 LAÇO FOR	47
5.4 COMANDO BREAK	48
5.5 SWITCH	48
6. FUNÇÕES.....	49
7. GRAVANDO COOKIES.....	50
8. RECUPERANDO COOKIES.....	51
9. HEADER(LOCATION...).....	51
10. MYSQL FUNÇÕES.....	51
10.1 MYSQL_CONNECT().....	52
10.2 MYSQL_SELECT_DB().....	52
10.3 MYSQL_QUERY().....	53
10.4 MYSQL_NUM_ROWS().....	53
10.5 MYSQL_FETCH_ARRAY().....	54
11. TRABALHANDO COM ARQUIVOS.....	54
11.1 ABRINDO ARQUIVOS.....	55
11.2 LENDO ARQUIVOS.....	56
11.3 GRAVANDO DADOS.....	56
BIBLIOGRAFIA.....	58
APÊNDICE A - FUNÇÕES MATEMÁTICAS DO MYSQL.....	59
APÊNDICE B - FUNÇÕES DE STRING DO MYSQL.....	61
APÊNDICE C - FUNÇÕES DE DATA E HORA DO MYSQL.....	63
APÊNDICE D - OUTRAS FUNÇÕES IMPORTANTES DO MYSQL..	64
.....	
.....	

NOTAS DO AUTOR

Na maior parte do tempo, escrever um livro é desafiante, frustrante, tedioso, excitante e muito trabalhoso, tudo simultaneamente. Mas a criação de Programando para Web com PHP/MySQL teve um componente quase que intangível. Eu assumi a responsabilidade de criar uma fonte de referência para o programador que deseja migrar para tecnologias Intranet.

Este livro foi escrito para um público principiante, com o mínimo de conhecimento necessário para programar em PHP. Saber como funciona a linguagem html e possuir noções de lógica de programação é de fundamental importância para a compreensão dos capítulos.

Procurei resumir o conteúdo, da melhor forma possível, descartando tudo que for desnecessário, tornando assim, uma leitura acessível e de rápida assimilação e entendimento.

Críticas, dúvidas e sugestões serão bem-vindas.

O autor

meu email: fredcox@ig.com.br

minhas URL's: <http://membros.option-line.com/fredcox>

AGRADECIMENTOS

Inicialmente gostaria de agradecer a Maria José, minha mãe, pelo incentivo à publicação dessa obra.

Sinceros agradecimentos a todos da lista de discussão php-pt (<http://br.egroups.com/group/php-pt>).

Não se mede o valor de um homem
Pelas suas roupas
Ou pelos bens que possui.
O verdadeiro valor de um homem
É o seu caráter,
Suas idéias
E a nobreza do seus ideais.
Charles Chaplin



CAPÍTULO I

I N T R O D U Ç Ã O

1. Client-Side Scripts

São responsáveis pelas ações executadas no browser, sem contato com o servidor. Os exemplos mais comuns de aplicações client-side são imagens e textos que mudam com o passar do mouse e os java scripts.

Os scripts client-side são muito úteis para fazer validações de formulários sem utilizar processamento do servidor, com isso não provocando tráfego na rede.

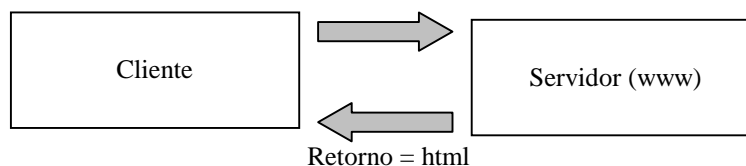
2. Server-Side Scripts

São responsáveis pelas ações executadas no servidor. Os exemplos mais comuns de aplicações server-side são os scripts cgi's e php.

No momento em que o usuário solicita uma URL, o servidor apresentará no browser um código html dinâmico, isto é muito útil para construções de aplicações baseadas em informações on-line.

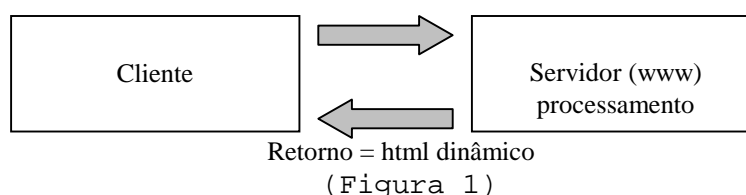
Cliente-Side Scripts

Requisição = URL



Server-Side Scripts

Requisição = URL



3. O QUE É PHP

PHP (Hiptertext PreProcessor) é uma linguagem de programação server-side scripts para criar sites dinâmicos. Sites dinâmicos são aqueles que retornam para o cliente uma página criada em tempo real. Um exemplo de server-side scripts são os sistemas de busca (www.yahoo.com, www.cade.com.br, etc...); nele, quando você digita a palavra chave da busca e clica no botão pesquisar o resultado da busca é processado on-line; outro exemplo são as salas de chat; nelas, quando você digita e clica no botão enviar, as informações são processadas em tempo real conjuntamente com a dos outros usuários, resultando num código HTML dinâmico gerado do servidor para o cliente.

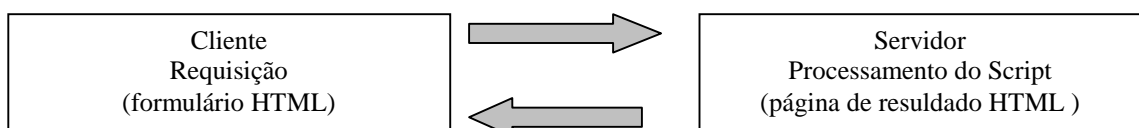
Um exemplo de um script PHP

```
1 <html>
2 <head>
3 <title>Exemplo de Script</title>
4 </head>
5 <body>
6     <?php echo "Oi, isto é um script!"; ?>
7 </body>
8 </html>
9
```

(Figura 2)

Utilizando a linguagem PHP você permite a interação direta do usuário com o site, através de formulários, contadores de acesso, estatísticas do site, ou criar aplicações para uma rede local baseadas numa Intranet.

Aplicações em PHP são geradas com excelente performance e automaticamente pelo servidor. O usuário não vê o código PHP, somente o HTML; isto é muito importante quando se está lidando com senhas.



(Figura 3)

O PHP suporta as seguintes bases de dados:

Adabas	Sybase	Solid
Dbase	MSQL	Interbase
MySQL	Oracle	Unix DBM
Informix	PostgreSQL	FilePro

4. UMA BREVE HISTÓRIA DO PHP

O PHP foi concebido no outono (lá nos EUA, aqui no Brasil seria primavera) de 1994 por Rasmus Lerdorf. As primeiras versões foram usadas na sua homepage para saber quem estava consultando o currículo online. A primeira versão, utilizada por outras pessoas, foi disponibilizada em meados de 1995, e era conhecida como Personal Home Page Tools (Ferramentas para Homepages Pessoais).

Ela consistia num motor de interpretação bem simples, que entendia algumas macros especiais e alguns utilitários de uso comum nas homepages de então. Um livro de visitas, um contador e outras coisas. Em meados de 1995, o interpretador foi reescrito e batizado de PHP/FI Version 2. O sufixo FI veio de um outro pacote escrito por Ramus, que interpretava dados de formulário html. Ele combinou os scripts das Ferramentas para Homepages Pessoais com o Interpretador de Formulário e adicionou o suporte ao mSQL; o PHP/FI estava criado. O PHP/FI cresceu num ritmo incrível e as pessoas começaram a adicionar-lhe código.

É muito difícil estimar corretamente, mas, em fins de 1996, o PHP/FI estava sendo usado em pelo menos 15,000 web sites pelo mundo afora. Na metade de 1997, este número havia aumentado para mais de 50,000. Nesta época, o desenvolvimento do PHP também sofreu mudanças. De um projeto de estimação de Rasmus, com contribuições de um pequeno grupo de pessoas, se tornou um esforço de uma equipe mais organizada. O interpretador foi reescrito do zero por Zeev Suraski e Andi Gutmans, e este novo interpretador foi a base para o PHP Versão 3. Muito do código dos

utilitários do PHP/FI foi portado para o PHP3, e muito desse código foi totalmente reescrito.

Já na metade de 1999 PHP/FI e PHP3 eram oferecidos juntos com vários produtos comerciais, como o webserver StrongHold da C2 e o RedHat Linux. Uma estimativa conservadora baseada na extrapolação dos números fornecidos pela NetCraft diz que o PHP está em uso em mais de 150,000 sites em todo o mundo. Para se ter uma idéia do que isso significa, esse número é maior do que a quantidade de sites que rodam o Enterprise server da Netscape na Internet.

5. ASP x PHP

Enquanto o ASP, só é executado em plataformas micro\$oft, o PHP suporta a maioria das plataformas que proveêm acesso e serviços da internet, é distribuído sobre GPL (Licença Pública Geral), ou seja, não se precisa pagar para usar o PHP.

6. INSTALAÇÃO EM AMBIENTE WINDOWS

O servidor http, a ser utilizado neste curso, é o Apache, que está disponível para download em "<http://www.apache.org>". Para instalar o servidor de web siga os passos abaixo:

6.1 - Servidor Apache 1.3.X

Execute o utilitário de instalação (apache1.3.1.exe) e siga os passos de instalação normalmente.

Quando o programa de instalação solicitar o diretório de destino, clique o botão browse e digite "C:\Apache" na janela PATH. Isto garantirá uma performance considerável no acesso ao diretório htdocs (onde ficarão armazenadas as páginas html e php reconhecidas pelo apache), visto que, por default, o Apache será instalado em: "C:\Arquivos de Programas\Apache Group\Apache\".

O próximo passo é a configuração do servidor de páginas www. Começando pelo arquivo "httpd.conf" que fica localizado em "C:\Apache\conf", edite este arquivo com qualquer editor de textos de escrita rápida (Edit do DOS ou Bloco de Notas).

Adicione as seguintes linhas no final do arquivo "httpd.conf".

```
ServerName localhost
ScriptAlias /php3/ "c:/php3/"
AddType application/x-httpd-php3 .php3 .php
Action application/x-httpd-php3 "/php3/php.exe"
```

A primeira linha informa ao apache o nome do servidor. No caso localhost porque cada estação no decorrer do curso estará funcionando como um servidor de web independente. A segunda linha informa ao apache que execute scripts php. A terceira informa as extensões dos scripts php que serão executados pelo servidor de web, ou seja, qualquer arquivo com extensão .php3 ou php ativará o client side script. A quarta linha informa o caminho "path" do PHP.

6.2 - Instalação do PHP3

3.1 - Crie uma pasta: "C:\php3"

3.2 - Descompacte o arquivo "php-3.0.16-win32.zip" neste diretório.

3.3 - Copie o arquivo "php3.ini.dist.txt" para o diretório "C:\Windows" , renomeando-o para php3.ini.

3.4 - Procure pela "linha extension_dir" no arquivo "php3.ini" e inclua o seguinte parâmetro: "c:\php3"

Veja o trecho do arquivo abaixo como deve ficar:

```
;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;
include_path=; UNIX: "/path1:/path2"  Windows: "\path1;\path2"
```

```
doc_root=; the root of the php pages, used only if nonempty
user_dir=; the directory under which php opens the script using
/~username, used only if nonempty
;upload_tmp_dir=; temporary directory for HTTP uploaded files (will use
system default if not specified)
upload_max_filesize=2097152 ; 2 Meg default limit on file uploads
extension_dir=C:\PHP3 ./
```

Este é o parâmetro
que deve ser
adicionado!!!

3.5 - Procure pelo trecho "Dynamic Extensions" no arquivo de configuração do PHP3 e descomente as linhas. Obs: descomentar é só apagar o ponto e vírgula que antecede cada parâmetro de configuração.

```
;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;
; if you wish to have an extension loaded automaticly, use the
; following syntax: extension=modulename.extension
; for example, on windows,
; extension=mysql.dll
; or under UNIX,
; extension=mysql.so
; Note that it should be the name of the module only, no directory
information
; needs to go here. Specify the location of the extension with the
extension_dir directive above.
```

```
;Windows Extensions
extension=php3_mysql.dll
extension=php3_calendar.dll
extension=php3_dbase.dll
extension=php3_gd.dll
extension=php3_dbm.dll
```

Retirar o ponto e vírgula
dessas do início de cada
linha

```
extension=php3_mssql.dll  
extension=php3_zlib.dll  
extension=php3_filepro.dll  
extension=php3_imap4r1.dll  
extension=php3_ldap.dll  
extension=php3_crypt.dll  
extension=php3_mysql2.dll  
extension=php3_odbc.dll
```

Obs: Para testar se o Apache está corretamente instalado, execute o **Apache Server** no menu iniciar, carregue o browser e digite o endereço: <http://localhost>. Se o endereço carregar é porque o seu servidor de web está instalado corretamente; caso contrário, repita os passos acima.

6.3 - MySQL

A versão do MySQL, que será utilizada neste curso, é a 3.23.11-alpha. O MySQL é um robusto Servidor Banco de Dados, multiusuário, multitarefa que opera com a linguagem SQL (Structured Query Language), linguagem de consulta estruturada. O acesso e manipulação de dados no servidor MySQL será discutido posteriormente.

Este servidor de banco de dados pode ser facilmente conseguido pelo endereço <http://www.tcx.se> .

3.1 - Crie uma pasta temporária; por exemplo : "c:\tempo"

3.2 - Descompacte o MySQL para win32 neste diretório e execute o utilitário de instalação normalmente.



CAPÍTULO II

M y S Q L

1. O Banco de Dados MySQL

O MySQL é servidor de banco de dados multiusuário, multitarefa que trabalha com uma das linguagens de manipulação de dados mais popularizadas do mundo.

SQL é uma linguagem simples, em que você facilmente pode gravar, alterar e recuperar informações num web site com segurança e rapidez. Ela foi desenvolvida pelo Departamento de Pesquisas da IBM como forma de interface para o Sistema de Banco de Dados Relacionais **SYSTEM R**, no início dos anos 70; em 1996, a American National Institute (ANSI) publicou um padrão SQL. A SQL estabeleceu-se como linguagem padrão de Banco de Dados Relacional. A linguagem SQL tem como grande virtude sua capacidade de gerenciar índices sem a necessidade de controle individualizado de índice corrente, algo muito comum nos Sistemas Gerenciadores de Arquivos, o Dbase por exemplo. Nunca trabalhe com arquivos do Dbase (*.DBF)! Esses falsos bancos de dados não oferecem integridade alguma para os dados; uma simples recuperação de dados resulta num código complicado e extenso, visto que consiste numa busca de registro a registro, além de não passar de uma simples e frágil gravação sequencial de strings. Você foi avisado!

O MySQL foi originalmente desenvolvido pela empresa sueca **TCX**, que necessitava de um servidor de banco de dados que operasse com grandes escalas de dados rapidamente sem exigir caríssimas plataformas de hardware. A **TCX** opera desde 1996 com 40 bancos de dados, contendo 10.000 tabelas, sendo 500 delas com mais de 10 milhões de linhas.

2. Características do MySQL

✓ suporta diferentes plataformas: Win32, Linux, FreeBSD, Unix,

- ✓ etc...
- ✓ Suporte às API's das Seguintes linguagens: PHP, Perl, C, C++, Java, Python, etc...
- ✓ Suporte a múltiplos processadores
- ✓ Um sofisticado sistema de senhas criptografadas flexível e Seguro.
- ✓ Suporte à ODBC, você pode facilmente conectar o Access a um banco de dados do MySQL
- ✓ Suporta até 16 índices por tabela
- ✓ Código fonte escrito em C e C++ e testado com uma variedade de diferentes compiladores
- ✓ O Cliente conecta no MySQL através de conexões TCP/IP.
- ✓ Nenhum problema com o Y2K, visto que o MySQL usa o relógio do Unix que não apresentará problemas até 2069

3. Habilitando o MySQL para conexões

Para efetuar qualquer conexão com o MySQL é necessário que o daemon (demônio), programa que roda em standalone, esteja carregado na memória; para isso, execute o binário *mysqld-shareware.exe* pelo prompt do ms-dos. Veja figura abaixo e siga o esquema:

```
Microsoft(R) Windows 95
(C) Copyright Microsoft Corp 1981-1996.

C:\WINDOWS>cd ..

C:\>cd mysql

C:\mysql>cd bin

C:\mysql\bin>mysqld-shareware.exe

C:\mysql\bin>_
```

Pronto! O MySQL está habilitado para receber solicitações TCP/IP.

4. Sistema de segurança do MySQL

O MySQL possui um avançado sistema de segurança, a ser tratado neste capítulo.

Quando você se conecta a um MySQL Server, normalmente é solicitada uma senha de usuário. Esta informação poderá ser lida no momento em que ela passar do cliente para o servidor. O ideal é instalar o servidor de web com um protocolo de compressão e criptografia, o Apache web server para o Red Hat 6.1 já vem com mod_ssh e mod_ssl; com isto, a conexão TCP/IP entre o cliente e o servidor estará sendo uma transação criptografada.

O MySQL criptografa as senhas dos usuários através de um algoritmo semelhante ao processo de autenticação de login do Unix

Quando se instala o MySQL para win32 pela primeira vez, ele por padrão, vem com dois usuários: um superusuário (root) e outro usuário padrão.

Para trocar a senha do root (superusuário) e apagar o usuário padrão, siga os passos a seguir, supondo que o daemon (mysqld-shareware.exe) já está ativado na memória:

No prompt do MS-DOS digite os comandos abaixo:

```
Microsoft(R) Windows 95
(C) Copyright Microsoft Corp 1981-1996.

C:\WINDOWS>cd ..

C:\>cd mysql

C:\mysql>cd bin

C:\mysql\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 68 to server version: 3.22.34-shareware-debug

Type 'help' for help.

mysql>
mysql>
```


Você acabou de se logar como root (superusuário) no MySQL monitor; é neste ambiente que você irá criar banco de dados, tabelas e usuários. Há outros tipos de ambientes com interfaces GUI's que não são abordados neste curso, ficando a critério do aluno a escolha. Vamos agora definir a senha do root: para isso, digite os comandos abaixo no MySQL monitor:

```
C:\mysql\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 68 to server version: 3.22.34-shareware-debug

Type 'help' for help.

mysql>
mysql> use mysql
Database changed
mysql> update user set password=password('sua_senha') where user='root';
Query OK, 2 rows affected (0.05 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.06 sec)

mysql> delete from user where user='';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.05 sec)

mysql>
```

Você apagou o usuário padrão e definiu a senha do root ('sua_senha'). Não se preocupe quanto à sintaxe SQL; isto será amplamente discutido no capítulo seguinte.

5. GUIA DE REFERÊNCIA DO MySQL

5.1 - MySQL MONITOR

O MySQL monitor é o cliente que vem no pacote do MySQL para win32. Através dele podemos criar tabelas, bancos de dados, usuários e estabelecer critérios de segurança para usuários.

UPE - POLI - Engenharia Eletrônica

Para acessar o MySQL você precisa acessar o prompt do ms-dos na pasta `c:\mysql\bin>`, e digitar os seguintes comandos.

```
c:\mysql\bin>mysql -u usuario -p banco_de_dados
```

ou

```
c:\mysql>bin>mysql -u usuario -psenha banco_de_dados
```

Veja figura abaixo:

```
C:\mysql\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 70 to server version: 3.22.34-shareware-debug

Type 'help' for help.

mysql>
mysql> _
```

O primeiro processo é mais seguro, visto que o MySQL Server solicitará a digitação da senha (ver figura). É extremamente necessário acessar o MySQL monitor para efetuar as consultas da linguagem SQL.

5.2 - COMO ESCREVER STRINGS E NÚMEROS

Strings

Qualquer sequência de caracteres delimitados por ' ' ou " ".

Exemplo:

`'um string'`

`"outro string"`

Números

Inteiros são representados por uma sequência de dígitos e ponto flutuante utiliza-se ' .' como separador decimal.

Exemplos de números inteiros válidos

1543

0

-48

Exemplos de números ponto flutuante válidos

294.42

-32032.6809e+10

148.00

5.3 - TIPO DE COLUNAS SUPORTADOS PELO MySQL

O MySQL suporta uma ampla variedade de colunas. Esta seção descreve os tipos disponíveis.

Os tipos de dados suportados pelo MySQL estão listados abaixo. O código das letras usado nas descrições é o seguinte:

- M - indica o tamanho máximo. O máximo valor que M pode assumir é 255
- D - utilizado para ponto flutuante, indicando o número de casas decimais. O valor máximo para D é 30.
- [] - indica um valor opcional.

** Note que se for especificado um parâmetro [ZEROFILL], o MySQL automaticamente atribui UNSIGNED para a coluna.*

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

Números inteiros muito pequenos. Pode assumir intervalo de valores entre -128 to 127. O intervalo de valores para unsigned é de 0 a 255.

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

Números inteiros pequenos. Intervalos de valores entre -32768 to 32767. O intervalo de valores para unsigned é de 0 a 65535.

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

Números inteiros de tamanho médio. O intervalo de valores está entre -8388608 a 8388607. O intervalo de valores para unsigned é de 0 a 16777215.

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

Inteiros de tamanho normal. O intervalo de valores está entre -2147483648 a 2147483647. Valores para unsigned é de 0 a 4294967295

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

Inteiros de tamanho grande. Assume intervalo de valores entre -9223372036854775808 a 9223372036854775807. Valores para unsigned está entre 0 a 18446744073709551615.

FLOAT[(M,D)] [ZEROFILL]

Números ponto flutuante pequenos (simples-precisão) . Assume valores entre -----3.402823466E+38 a -1.175494351E-38 . O valor de M corresponde ao tamanho e D ao número de casas decimais.

'DOUBLE[(M,D)] [ZEROFILL]'

Números ponto flutuante de tamanho normal. Valores assumidos entre -1.7976931348623157E+308' a '-2.2250738585072014E-308', O valor de M corresponde ao tamanho e D ao número de casas decimais.

`DATE'

Para armazenar valores de data. Assume valores entre`'1000-01-01'' a`'9999-12-31''.

Os *MySQL* são gravados no formato`'YYYY-MM-DD', porém há funções gravar e recuperar dados de data e hora que serão discutidas posteriormente.

`DATETIME`

Uma combinação de data e hora. Suporta valores entre `'1000-01-01 00:00:00'` a `'9999-12-31 23:59:59'`.

`TIME`

Para armazenar valores hora. Assume intervalo de valores entre `'838:59:59'` a `'838:59:59'`.

`CHAR(M) [BINARY]`

Valores de String. O valor de `M` indica o comprimento do campo string.

5.4 - OPERADORES

ARITMÉTICOS

`+`

Adição

```
mysql> select 3+5;  
-> 8
```

`-`

Subtração

```
mysql> select 3-5;  
-> -2
```

`*`

Multiplicação

```
mysql> select 3*5;  
-> 15  
mysql> select 18014398509481984*18014398509481984.0;  
-> 324518553658426726783156020576256.0
```

UPE - POLI - Engenharia Eletrônica

```
mysql> select 18014398509481984*18014398509481984;
-> 0
```

`/'

Divisão

```
mysql> select 3/5;
-> 0.60
```

```
mysql> select 102/(1-1);
-> NULL
```

LÓGICOS

Todas as operações lógicas no MySQL retornam `1' (Verdadeiro) ou `0' (Falso).

`NOT'

`!'

Operador lógico de negação NOT. Retorna `1' se o argumento é falso, caso verdadeiro retorna `0'.

The last example returns `1' because the expression evaluates the same way as `(!1)+1'.

`OR'

`||'

Operador lógico de escolha OR.

`AND'

`&&'

Operador lógico AND.

COMPARAÇÃO

`='

Igualdade

`<>'

`!='

Diferença

`<='

Menor ou igual

`<'

Menor que

`>='

Maior ou igual

`>'

Maior que

5.5 - COMANDOS

**Note que todo comando SQL termina com um ';'*

CREATE DATABASE

Cria um banco de dados. Este comando cria uma área lógica, diretório, onde estarão armazenadas todas as tabelas do banco de dados.

Syntax:

REATE DATABASE *banco_de_dados*;

Exemplo

mysql>CREATE DATABASE
funcionarios;

DROP DATABASE

Apaga um banco de dados.

UPE - POLI - Engenharia Eletrônica

Syntax:	Exemplo
DROP DATABASE <i>banco_de_dados</i> ;	mysql>DROP DATABASE funcionarios;

Obs: Muito cuidado com este comando. O usuário com garantia DROP pode apagar todos os dados do seu banco. Você foi avisado!

CREATE TABLE

Comando utilizado para criar tabelas.

Syntax:	Exemplo
CREATE	TABLE mysql>CREATE TABLE
<i>nome_tabela</i> (<i>nome_atributo1</i> <i>tipo</i> alunos(matricula UNSIGNED	
[NOT NULL], <i>nome_atributo2</i> <i>tipo</i> INT(10) NOT NULL, <i>nome</i> CHAR(40)	
[NOT NULL], ... , <i>nome_atributoN</i> NOT NULL,turma CHAR(20) NOT	
<i>tipo</i> [NOT NULL]);	NULL,
	PRIMARY KEY (matricula));

No exemplo acima foi definida uma chave primária para a coluna *matricula*. Isto impede que hajam repetições no número de matrícula do aluno na tabela. Uma chave primária indica que o valor armazenado no registro é único.

ALTER TABLE

Syntax:	Exemplo
ALTER TABLE <i>nome_tabela</i> ADD/DROP	mysql>ALTER TABLE alunos ADD
Especificação	COLUMN turno char(10) NOT NULL;
<i>Nome_atributo1</i> <i>tipo</i> [NOT NULL],	
<i>Nome_atributo2</i> <i>tipo</i> [NOT NULL],	
...	


```
nome_atributo N tipo [NOT NULL]);
```

especificações possíveis

```
ADD      INDEX      [nome_indice]
(coluna_indice,...)
ADD PRIMARY KEY (coluna,...)
ALTER [COLUMN]
CHANGE [COLUMN] antiga_coluna
```

O exemplo acima adicionou na tabela *alunos* uma coluna para cadastrar o turno, que é primordial na tabela, mas que foi esquecida com o intuito de utilizar este comando.

DROP TABLE

Apaga uma tabela.

Syntax:

```
DROP TABLE nome_tabela;
```

Exemplo

```
mysql>DROP TABLE alunos;
```

INSERT

Comando utilizado para inserir valores numa tabela.

Syntax:

```
INSERT INTO nome_tabela(coluna1,
Coluna2,...,
ColunaN)
VALUES
```

Exemplo

```
mysql>INSERT INTO
alunos(matricula,
nome,turma,turno)
values
```

UPE - POLI - Engenharia Eletrônica

```
(expressao1,
expressao2,
...,
expressaoN);
```

```
(127423,
'Fred Cox Junior',
'Eng. Eletrônica',
'Manhã');
```

SELECT

Comando usado para recuperar valores de uma tabela. Este poderoso comando, em conjunto com funções, possibilita a recuperação de qualquer valor de uma tabela.

Sintax:	Exemplo
SELECT <i>coluna1</i> ,	mysql>SELECT * FROM alunos where
<i>Coluna2</i> ,...,	nome='Fred Cox Junior';
<i>ColunaN</i>	
FROM <i>tabela</i>	mysql>SELECT matricula,turno
WHERE <i>condição</i>	FROM alunos
ORDER BY <i>coluna</i> [ASC DESC]	ORDER BY matricula asc;

No primeiro exemplo, são mostradas todas as colunas da tabela *alunos* que possuem nome igual a 'Fred Cox Junior' (O asterisco * indica que devem ser mostradas todas as colunas). O segundo exemplo mostra somente as colunas *matricula* e *turno*, em ordem crescente numérica.

DELETE

Apaga registros em uma tabela. Se não for especificada a *condição where*, todos os dados serão apagados.

Sintax:	Exemplo
DELETE FROM <i>tabela</i>	Mysql>DELETE FROM alunos WHERE

```
WHERE condição;                turno='Manhã';
```

O exemplo acima apaga todos os alunos que estudam pela manhã.

UPDATE

Altera dados numa tabela.

Syntax:	Exemplo
UPDATE <i>tabela</i>	SET mysql>DROP TABLE alunos;
column1=expr1,col_name2=expr2,..	
..	
ColunaN=exprN	
WHERE condição;	

5.6 - CRIANDO USUÁRIOS NO MySQL

Criar um usuário no MySQL pode ser de duas formas: com o comando INSERT ou com o comando GRANT. A segunda forma é mais amigável e menos trabalhosa. Para criar e dar garantias a um usuário no MySQL você terá de especificar os privilégios e tabelas que este usuário pode acessar. Nunca garanta privilégios para um usuário no banco de dados mysql, pois somente o root (superusuário) deve possuir garantias para tal.

O superusuário pode especificar quais os privilégios que o usuário comum possui no banco ou tabela do banco de dados. Veja a lista de privilégios abaixo:

Tipo	Descrição
------	-----------

Select	Recuperar dados
Insert	Inserir dados
Update	Alterar dados
Delete	Apagar dados
Alter	Alterar estrutura da
Create	tabela
Drop	Criar tabelas
Grant	Apagar tabelas
all	Estabelecer
privileges	privilégios
	Todos os privilégios
	acima

Vamos criar um usuário para acessar o banco de dados *controle*.

- Acesse o MySQL monitor como root
- Crie um banco de dados com o nome *controle*. Utilize o comando CREATE DATABASE.

Veja a figura abaixo

UPE - POLI - Engenharia Eletrônica

```
C:\mysql\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 71 to server version: 3.22.34-shareware-debug

Type 'help' for help.

mysql>
mysql> CREATE DATABASE controle;
Query OK, 1 row affected (0.00 sec)

mysql> USE controle
Database changed
mysql> GRANT all privileges ON controle.* TO 'joao' IDENTIFIED BY 'joao';
Query OK, 0 rows affected (0.16 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.06 sec)

mysql>
```

Na figura acima foi criado um usuário com o login 'joao' e senha 'joao', que possui todos os privilégios no banco de dados controle em qualquer tabela (controle.*). O Comando USE possibilita configurar o banco de dados *controle* como o corrente. O FLUSH PRIVILEGES atualiza a tabela de privilégios do banco de dados MySQL, efetuando as mudanças e adicionando o usuário. Para fazer o teste, se o usuário foi corretamente cadastrado, digite no prompt do ms-dos:

```
C:\mysql\bin>mysql -u joao -p
```

Quando solicitar a senha, digite *joao*.

Observe outros exemplos de criação de usuários.

Exemplo	Descrição
mysql>GRANT select ON controle.alunos TO 'paulo' IDENTIFIED BY 'paulo'; mysql>FLUSH PRIVILEGES;	Usuário <i>paulo</i> de senha <i>paulo</i> com o privilégio somente de recuperar dados na tabela <i>alunos</i> do banco de dados <i>controle</i> .

mysql>GRANT select,insert,update, delete ON controle.* TO 'paula' IDENTIFIED by 'paula366'; mysql>FLUSH PRIVILEGES;	Usuária <i>paula</i> de senha <i>paula366</i> com os privilégios de inserir, apagar, recuperar e alterar dados em qualquer tabela do banco de dados <i>controle</i> .
---	---

Depois que o aluno aprender a linguagem de Script PHP, no decorrer do curso, o ideal é construir uma aplicação para gerenciar o processo de criação e manipulação de usuários, se for um caso de uma Intranet e o volume de usuários for extenso. A digitação da senha do usuário pelo administrador da rede não lhe agrada muito. Palavra de quem trabalha com isto. ☺



CAPÍTULO III

A L I N G U A G E M H T M L

1. INTRODUÇÃO

O propósito do curso resume-se a programação para web com linguagem de Script PHP e acesso a banco de dados MySQL. Pressupõe-se que o aluno esteja familiarizado com as tags html, visto que é requisito necessário para fazer o curso, e além do mais, estenderia o curso. Por isso, a abordagem desta linguagem será apenas no intuito de entender o conceito de páginas web dinâmicas, com comentários voltados a esclarecer a passagem de parâmetros de formulários html para os scripts em PHP.

A aparente sofisticação das páginas html, não passam de documentos de texto simples. Podem ser produzidos com qualquer editor de texto, como o Notepad, Emacs, vi, joe, etc... A diferença é que algumas páginas da web possuem características especiais de formatação de documento. Há programas especializados em fazer páginas html, tais como Dreamweaver 3.0, Hot Dog, Homesite, etc... Há uma ampla variedade de documentos, tutoriais e templates na Internet. Um bom endereço para começar a pesquisar sobre webdesign é : <http://www.tol.pro.br>

2. Criando Documentos HTML

2.1 - HTML Mínimo

Todo documento deve ser identificado como HTML (<html> </html>), ter uma área de cabeçalho (<head></head>) com o nome para o documento (<title> </title>), um título principal e uma área definida como corpo(<body></body>) do conteúdo do documento. Como o exemplo a seguir:

```
1 <HTML>
2 <HEAD>
3 <TITLE>Curso de PHP/MySQL</TITLE>
4 </HEAD>
5 <BODY>
6 <H1>Este é o primeiro nível de cabeçalho</H1>
7 Bem-vindo ao mundo do HTML.
8 Este é o primeiro parágrafo.<P>
9 E este é o segundo.<P>
10 </BODY>
11 </HTML>
```

2.2 - Marcações Básicas

Títulos

Todo documento em HTML deve possuir um título. De um modo geral o título aparece em lugar separado da página (por exemplo, alto da tela no Netscape), e é utilizado para identificar o documento em outros contextos (por exemplo, buscas Wais). É interessante que o título possa sugerir claramente o conteúdo do documento.

Atenção porque o conceito de título é diferente de cabeçalho. O título está mais para o nome do arquivo. Não é um elemento relevante na visualização do documento como acontece com o cabeçalho.

A marcação utilizada para títulos é <title> e seu par </title>.

Escrito desta forma:

```
1 <html>
2 <title> Este é o título</title>
3 <body>
4 <h2>E este o cabeçalho de nível 2</h2>
5 Aqui entra o texto do documento ...
6 </body>
7 </html>
```


Cabeçalhos

"Cabeçalhos" normalmente são usados para títulos e sub-títulos de uma página.

HTML possui seis níveis de cabeçalhos, numerados de 1 a 6, sendo o número 1 o de maior destaque. Cabeçalhos são exibidos em letras maiores e em negrito. O primeiro cabeçalho em cada documento deve estar marcado como <H1>.

ATENÇÃO: ao definir o tamanho de um cabeçalho, você não está definindo o tamanho da letra (fonte 10, fonte 14). Você apenas define que ele aparecerá com maior tamanho e destaque que o resto do texto. O tamanho exato com que ele será visualizado é definido pelo programa visualizador de html (browser) de cada pessoa que acessar a informação.

As notações relativas a cabeçalhos são:

```
<h1>Cabeçalho da Página</h1>
```

Parágrafos

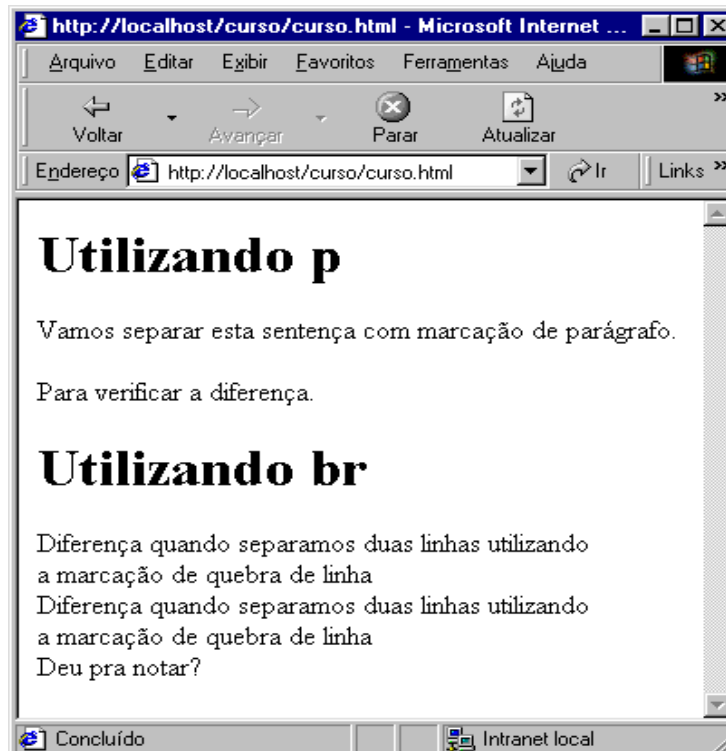
A marcação <p> é utilizada para definir o início de um novo parágrafo, deixando uma linha em branco entre cada parágrafo. HTML não reconhece o caracter de quebra de linha dos editores de texto. Mesmo que exista uma linha em branco, os clientes Web só reconhecem o início de um novo parágrafo mediante a marcação apropriada.

Quebras de linha

A marcação
 faz uma quebra de linha sem acrescentar espaço extra entre as linhas. Veja a diferença do uso de <p> e
 ,nos exemplos a seguir:

```
1 <html>
2 <body>
3 <h1>Utilizando p</h1>
4 Vamos separar esta sentença com marcação de parágrafo.<p>
5 Para verificar a diferença.
6 </body>
7 </html>
8
9 <html>
10 <body>
11 <h1>Utilizando br</h1>
12 Diferença quando separamos duas linhas utilizando<br>
13 a marcação de quebra de linha<br>
14 Diferença quando separamos duas linhas utilizando<br>
15 a marcação de quebra de linha<br>
16 Deu pra notar?
17 </body>
18 </html>
```

Veja a diferença no uso das duas notações:



3. Interligando Documentos

O principal poder do HTML vem da sua capacidade de interligar partes de um texto (e também imagens) a outros documentos. Os clientes exibem em destaque estas áreas ou pontos chaves (normalmente com cores diferentes ou sublinhado) para indicar que se trata de um link, ou interligação, no hipertexto.

A marcação **<A>**, que define o ponto de partida para os links, é denominada de **âncora**. Para incluir uma âncora em seu documento:

1. Inicie a âncora com **<A** . (Há um espaço depois de A.)
2. Especifique o documento a ser interligado, inserindo parâmetro **HREF="arquivo"** seguido do sinal: **>**
3. Insira o texto que vai funcionar como link no documento corrente
4. Anote a marcação de final da âncora: ****.

Um exemplo de referência a um hipertexto:

```
<A HREF="ListaPraias.html">Praias</A>
```

A palavra **"Praias"** é definida como o marcador do link para se chegar ao documento `ListaPraias.html`, que está no mesmo diretório do documento corrente. Ou seja, **"Praias"** aparece em negrito e se eu clicar nessa palavra será exibido o documento apontado - `ListaPraias.html`

4. Interligando Documentos em Outros Diretórios

É possível interligar documentos em outro diretório especificando-se o caminho relativo a partir do documento corrente, em relação ao que está sendo interligado.

Por exemplo, um link para o arquivo **Sergipe.html** localizado no subdiretório **Estados** seria assim:

`Sergipe`

Estes são os denominados links relativos. É também possível usar o caminho completo (pathname absoluta) do arquivo desejado. Para isso, utiliza-se a sintaxe padrão do sistema UNIX.

Importante: se você quiser referenciar um diretório a partir da "raiz" do seu servidor www, inicie a notação com /. Isto é, se você tiver uma notação desta forma.

``, significa que o arquivo **incon1.gif** deverá ser buscado no diretório imagens que está imediatamente acima do diretório raiz do seu servidor WWW.

5. FORMULÁRIOS HTML

Esta constitui a seção mais importante deste capítulo, visto que, é a partir de um formulário html que o usuário interage, com o servidor enviando e recebendo informações.

Cada comando será comentado, a fim de que o aluno se familiarize o mais rápido possível com o conceito de passagem de parâmetros de formulários html para scripts php, visto que esta é uma das maiores dúvidas.

Existem vários tipos de campos de entrada de um formulário, como:

- campos de entrada de texto
- menus de múltipla escolha ou escolha única
- botões sim-ou-não
- botões para submissão ou limpeza de formulário

Cada um destes campos tem funcionalidade própria e você vai aprender em que situação utilizá-los em seus formulários.

Se você já conhece HTML, as marcações utilizadas em formulários seguem a mesma convenção, e portanto, será fácil aprender.

5.1 - Codificação básica

Um formulário começa com a marcação **<FORM>** e termina com **</FORM>**. Outros itens devem ser especificados:

- **Primeiro**, o formulário precisa saber como enviar a informação para o servidor. Existem dois métodos: GET e POST.

METHOD="GET"

A maioria dos documentos HTML são recuperados a partir da requisição de uma única URL ao servidor. Assim, um formulário que utilize este método, envia toda sua informação ao final da URL ativada.

METHOD="POST"

Este método transmite toda a informação fornecida via formulário, imediatamente após a URL ativada. Ou seja, quando o servidor recebe uma ativação de um formulário, utilizando POST, ele sabe que precisa continuar "ouvindo" para obter a informação. Este é o método que iremos utilizar para direcionar um script.

- **Segundo**: o formulário precisa saber para onde enviar a informação. Esta é a URL sendo ativada a partir do formulário, e ela é referenciada através da marcação **ACTION**. Esta URL apontará para um script PHP que irá receber e decodificar os resultados.

```
ACTION="teste.php3"
```

Após você construir estas marcações, seu formulário geralmente terá a seguinte estrutura:

```
<FORM METHOD="POST" ACTION="teste.php3">
```

Marcações de campos de entrada e HTML em geral

</FORM>

Observe que este formulário utiliza o método POST e envia as informações digitadas para um script PHP chamado *teste.php3* no diretório padrão do servidor.

Outra informação importante: cada marcação de entrada em um formulário tem uma opção **NAME** associada, de tal forma que o script que apontado pelo parâmetro ACTION receba uma variável com o valor digitado pelo usuário. Certamente você pode definir mais de um campo de entrada textual ou menu dentro de um formulário, mas certifique-se de que cada um possui um nome diferente; assim sendo, o script receberá campos digitados com variáveis diferentes.

5.2 - ENTRADA DE TEXTO COMUM - TEXT

A forma mais simples de campo de entrada é a marcação **text**. Este campo permite a digitação de uma única palavra ou linha de texto, e possui uma largura default de 20 caracteres.

Opções:

VALUE="" OPCIONAL

Utilizando a marcação VALUE você especifica que texto aparecerá no campo quando o formulário for exibido.

SIZE="" OPCIONAL

Esta marcação altera o tamanho deste campo exibido na tela.

Obs.: o usuário sempre poderá digitar mais caracteres do que o tamanho do campo na tela, pois o texto irá se deslocar à esquerda dentro do campo.

MAXLENGTH="" OPCIONAL

Se você deseja limitar o número de caracteres que o usuário pode digitar, basta usar esta marcação. O formulário irá emitir um bip de erro se o usuário tentar digitar além do permitido em MAXLENGTH.

Observe o exemplo abaixo. Note que o parâmetro NAME="primeiro_nome", indica que ao submeter este formulário, o script apontado pelo parâmetro ACTION="teste.php3" receberá uma variável chamada \$primeiro_nome (variáveis no PHP começam com '\$'), contendo o primeiro nome digitado pelo usuário ou o valor padrão especificado no parâmetro VALUE="Fred".

```
1 <html>
2 <title>Curso: PROGRAMANDO PARA WEB COM PHP/MySQL</title>
3 <body>
4 <h1>Utilizando formulários html</h1>
5 <FORM METHOD="POST" ACTION="teste.php3">
6 Entre com o seu primeiro nome:
7 <INPUT TYPE="text" NAME="primeiro_nome" VALUE="Fred" SIZE="10" MAXLENGHT="10">
8
9
10 </FORM>
11
12 </body>
13 </html>
```

5.3 - ENTRADA DE TEXTO PROTEGIDO - SENHA

Marcações de entrada do tipo **password** são idênticas aos campos do tipo text, exceto pelo fato de todos os caracteres serem exibidos como asteriscos (*). Apesar da máscara de entrada, o script receberá a variável especificada no parâmetro NAME do mesmo jeito do formulário anterior.

Opções:

VALUE="" OPCIONAL

A marcação VALUE especifica um valor default para este campo. *Obs: Esta opção não deve ser usada, é lógico!*

SIZE="" OPCIONAL

Esta troca o tamanho do campo de password exibido na tela.

MAXLENGTH="" OPCIONAL

Limita o número de caracteres que o usuário pode informar como password.

```
1 <html>
2 <title>Curso: PROGRAMANDO PARA WEB COM PHP/MySQL</title>
3 <body>
4 <h1>Utilizando formulários html</h1>
5 <FORM METHOD="POST" ACTION="teste.php3">
6 Entre com o seu primeiro nome:
7 <INPUT TYPE="text" NAME="primeiro_nome" VALUE="Fred" SIZE="10" MAXLENGTH="10">
8 <br>
9 Entre com a senha:
10 <INPUT TYPE="password" NAME="senha" SIZE="10" MAXLENGTH="10">
11
12 </FORM>
13
14 </body>
15 </html>
```

Observe que no exemplo acima o script *teste.php3* receberá duas variáveis, a primeira *\$primeiro_nome*, contendo o nome do usuário e a segunda *\$senha*, contendo a senha do usuário.

5.4 - ENTRADA DE VÁRIAS LINHAS DE TEXTO - TEXTAREA

A marcação **TEXTAREA** não utiliza o formato convencional `INPUT TYPE="text"` dos exemplos anteriores. Ao contrário, uma marcação `<TEXTAREA>` delimita o seu início e a marcação `</TEXTAREA>` o seu fim.

Opções:

ROWS="" OBRIGATÓRIO

Especifica o número de linhas da entrada textual.

COLS="" OBRIGATÓRIO

Especifica o número de colunas da entrada textual.

Texto default OPCIONAL

Se você deseja que um texto seja exibido no campo textual ao abrir o formulário, simplesmente coloque este texto entre as marcações de início e fim da TEXTAREA.

Observe o exemplo abaixo:

```
10 <TEXTAREA NAME="sugestoes" ROWS="5" COLS="20">
11 Digite aqui
12 suas sugestões...
13 </TEXTAREA>
```

5.5 - LISTBOX E COMBO BOX

COMBO BOX

Select

Esta entrada de formulário é muito importante, visto que é a partir dela que poderemos carregar base de dados inteiras através de comandos de recuperação SQL (SELECT) com suas respectivas cláusulas e sintaxe adequada para o contexto.

```
<select name="" size="">
  <option value="">texto</option>
</select>
```

Se você deixar de especificar o parâmetros "size" , aparecerá um COMBO BOX na tela; caso contrário, uma LISTBOX.

Parâmetros:

Size - número de linhas exibidas. Default: 1;

Multiple - parâmetro que, se presente, permite que sejam selecionadas duas ou mais linhas, através das teclas Control ou Shift;

option - Cada item do tipo "option" acrescenta uma linha ao select;

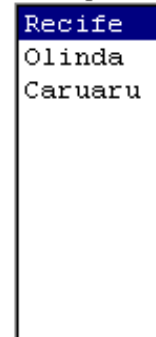
value - Valor a ser enviado ao servidor se aquele elemento for selecionado. Default: o texto do item;

text - valor a ser exibido para aquele item. Não é definido por um parâmetro, mas pelo texto que fica entre as tags <option> e </option>

Observe os exemplos abaixo:

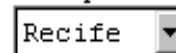
```
6 Exemplo de LISTBOX:
7 <BR>
8 <SELECT NAME="cidade" SIZE="10" >
9 <OPTION VALUE="Recife" >Recife </OPTION>
10 <OPTION VALUE="Olinda" >Olinda </OPTION>
11 <OPTION VALUE="Caruaru">Caruaru</OPTION>
12 <OPTION VALUE="Manaus" >
13 </SELECT>
```

Exemplo de LISTBOX:



```
6 Exemplo de COMBO BOX:
7 <BR>
8 <SELECT NAME="cidade">
9 <OPTION VALUE="Recife" >Recife </OPTION>
10 <OPTION VALUE="Olinda" >Olinda </OPTION>
11 <OPTION VALUE="Caruaru">Caruaru</OPTION>
12 <OPTION VALUE="Manaus" >
13 </SELECT>
```

Exemplo de COMBO BOX:



5.6 - CHECKBOX

Esta tag de formulário html é muito utilizada na internet. Você, com certeza, já deve ter aberto para um amigo uma conta de e-mail ou pedido um desses serviço grátis, onde são solicitadas pesquisas, tais como;

Você se interessa por:

☐ Cinema ☐ Artes

☐ Turismo ☒ Ciência e Tecnologia

☒ Outros...

Retornando ao formato de **INPUT TYPE=""**, a marcação **CHECKBOXES** é perfeita para escolher entre várias opções.

```
<input type="checkbox" name="" value="" checked>
```

Opções:

VALUE="" OBRIGATÓRIO

Especifica o valor da opção enviado ao script PHP. Esta opção deve conter o mesmo valor

CHECKED OPCIONAL

Esta marcação define a opção selecionada por default.

Observe o exemplo abaixo:

```
<p>
<input type="checkbox" name="pesquisa[]" value="Cinema">
Cinema
<input type="checkbox" name="pesquisa[]" value="Artes">
Artes</p>
<p>
<input type="checkbox" name="pesquisa[]" value="Turismo">
Turismo
<input type="checkbox" name="pesquisa[]" value="Ciência e Tecnologia">
Ciência e Tecnologia</p>
<p>
<input type="checkbox" name="pesquisa[]" value="Outros...">
Outros...</p>
```

Note que ao submeter o formulário acima, será enviado ao script PHP um array chamado `$pesquisa` numerado de 0 a 3, contendo os valores selecionados pelo usuário. Por exemplo: Vamos supor que o usuário marcou as opções *Cinema* e *Ciência e Tecnologia*. O script receberá os seguintes valores:

```
$pesquisa[0]='Cinema';
$pesquisa[1]='';
$pesquisa[2]='Ciência e Tecnologia'
$pesquisa[3]='';
```

5.7 - Radio Button

```
<input type="radio" name="" value="" checked>
```

Utilizado para campos de múltipla escolha, onde o usuário pode marcar apenas uma opção. Para agrupar vários elementos deste tipo, fazendo com que eles sejam exclusivos, basta atribuir o mesmo nome a todos do grupo.

Parâmetros:

Value - o valor que será enviado ao servidor quando o formulário for submetido, no caso do campo estar marcado

Checked - O estado inicial do elemento. Quando presente, o elemento já aparece marcado;

Observe o exemplo abaixo:

```
Você está discando de onde:  
<input type="radio" name="opcao" value="Recife" checked>Recife  
<input type="radio" name="opcao" value="Caruaru">Caruaru  
<input type="radio" name="opcao" value="Petrolina">Petrolina  
..
```

Note que, ao submeter o formulário, a variável `$opcao` conterá a escolha do usuário. Por exemplo: vamos supor que o usuário clicou na opção *Recife*; o formulário enviará ao script PHP uma variável chamada `$opcao` contendo o valor `'Recife'`.

5.8 - SUBMIT BUTTON E RESET BUTTON

Em vez de o usuário corrigir cada INPUT, um botão **RESET** pode ser utilizado para restaurar todos os campos a seus valores default, como se nenhuma informação houvesse sido digitada.

E finalmente, o FORM precisa de uma opção para enviar toda a informação digitada para o servidor, uma vez que o usuário terminou de preencher todos os campos de entrada. O botão **SUBMIT** transfere toda a informação para a URL especificada no elemento ACTION.

```
<INPUT TYPE="submit" NAME="botao" VALUE="Enviar">
```

Opções:

VALUE="" OPCIONAL

Especifica o texto a ser exibido no botão.

Se não for especificado, os textos default "Reset" e "Submit Query" serão colocados nos botões RESET e SUBMIT, respectivamente.

NAME="" OPCIONAL

Se NAME for definido em um botão SUBMIT, o formulário irá transmitir o valor do conteúdo do elemento VALUE, permitindo que você tenha múltiplos botões SUBMIT numa espécie de versão simplificada de um RADIOBUTTONS.

5.9 - Conclusão

O conhecimento da linguagem de marcação html é de extrema importância, bem como a forma que os formulários e as páginas da web interagem com o servidor, permitindo assim gerar páginas com código dinâmico.

O **Capítulo II** abrangeu um html simples, ou seja, o curso fornece o mínimo e necessário para o aluno comunicar-se com o servidor, gerando códigos html dinâmicos. Cabe ao aluno, caso não tenha conhecimento, se aprofundar na linguagem html. Na internet há um grande acervo de tutoriais, manuais e até mesmo livros, ensinando como elaborar páginas das mais simples às mais complexas. Você pode começar coletando manuais no endereço: <http://www.tol.pro.br> (Tutoriais On-line).

No próximo capítulo, estudaremos a linguagem de script PHP e ao final deste, o aluno estará preparado para escrever qualquer aplicação com tecnologia Intranet.

CAPÍTULO IV

A L I N G U A G E M P H P

1. Sintaxe Básica

O interpretador reconhece automaticamente scripts php delimitados da seguinte maneira:

```
<?
// código em php
?>
```

Veja o exemplo da página "curso.php3", no momento em que o apache web server verificar a incidência de "<? ?>", ele automaticamente iniciará o interpretador php, que construirá a página baseado no código php existente entre as devidas delimitações.

```
1 <html>
2 <title>Curso: PROGRAMANDO PARA WEB COM PHP/MySQL</title>
3 <body>
4 <h1>Introdução à PHP</h1>
5
6 <?
7
8 echo ('Este é um código PHP');
9
10 ?>
11
12
13 </body>
14 </html>
```

Observe que a marcação html utilizada é a mesma do capítulo anterior; o que muda é o trecho compreendido entre "<?" e "?>" no caso.

Note que todo comando php termina com ';', semelhante à linguagem C e Pascal. O simples esquecimento desse parâmetro resulta em erro no script, assim como programas em pascal e c.

2. Variáveis

As variáveis do PHP sempre começam com \$ e são declaradas quanto o tipo (inteiro,string,array, etc...) no momento em que é atribuído o seu valor, não sendo necessário indicar o nome e tipo da variável como na linguagem C. O php é **case sensitive**, portanto a variável \$fredcox é diferente da variável \$Fredcox.

3. Comentários

Os comentários podem ser de três tipos. Observe o exemplo abaixo:

```
$a=1237; #isto é um número inteiro  
$b='Fabiana Ferraz'; //isto é um string  
/* Isto é um comentário de  
várias linhas */
```

4. Tipos de Dados

Os tipos de dados do PHPs são:

4.1 - Integer

Variáveis inteiras são declaradas no PHP no momento da atribuição.

Exemplo:

```
$numero1=-12; #número inteiro negativo  
$numero2=64; #número inteiro positivo
```

4.2 - floating-point

Números com notação científica e decimais podem ser escritos da seguinte forma:

```
$a=1.12; #número decimal positivo
```


`$b=1.21e4` #número em notação científica

4.3 - Array

Você pode criar arrays usando as funções `list()` e `array()`, ou atribuindo valores aos seus respectivos elementos.

Exemplo:

```
$a[0]="Azul";  
$a[1]="Amarelo";  
$a[2]="Vermelho";
```

Veja a notação da função `array()`:

```
$a = array( "cor" => "Vermelho",  
           "gosto" => "Doce",  
           "formato" => "Redondo",  
           "nome" => "Maçã" );
```

Arrays serão amplamente utilizados com a cláusula `SELECT` do MySQL...

4.4 - String

Strings podem ser declarados delimitados por " " (aspas).

Exemplo:

```
$nome="Fred cox Junior";
```

4.5 - CONSTANTES

O php possui várias constantes pré-definidas, além de prover funções para criá-las em tempo de execução `define()`.

Algumas das constantes pré-definidas do php:

CONSTANTE	RETORNO
<code>__FILE__</code>	Nome do arquivo de script.
<code>__LINE__</code>	Número de linhas do script corrente.
<code>PHP_VERSION</code>	Versão do PHP que está sendo utilizada pelo servidor.

4.6 - Operadores

4.6.1 - Aritméticos

Exemplo	Nome	Resultado
<code>\$a+\$b</code>	Adição	Soma de \$a mais \$b.
<code>\$a-\$b</code>	Subtração	Diferença entre \$a e \$b
<code>\$a*\$b</code>	Multiplicação	Produto entre \$a e \$b
<code>\$a/\$b</code>	Divisão	Divide \$a por \$b

4.6.2 - Strings

O operador de concatenação de strings é ".".

```
$a="Fernanda";
$b="Ferraz";
$c=$a." ".$b;
echo $c;
```

O resultado será um string contendo o valor *"Fernanda Ferraz"*.

4.6.3 - Lógicos

Exemplo	Nome	Resultado
<code>\$a and \$b</code>	And	verdadeiro se \$a e \$b são verdadeiros
<code>\$a or \$b</code>	Or	Verdadeiro se \$a ou \$b são verdadeiros
<code>!\$a</code>	Not	Verdadeiro se \$a for falso

4.6.4 - Comparação

Exemplo	Nome	Retorno
<code>\$a==\$b</code>	Igual	Verdadeiro se \$a for igual a \$b
<code>\$a!=\$b</code>	Não Igual	Verdadeiro se \$a for diferente de \$b
<code>\$a<\$b</code>	Menor que	Verdadeiro se \$a for menor que \$b
<code>\$a>\$b</code>	Maior que	Verdadeiro se \$a for maior que \$b
<code>\$a>=\$b</code>	Maior ou igual	Verdadeiro se \$a for maior ou igual a \$b
<code>\$a<=\$b</code>	Menor ou igual	Verdadeiro se \$a for menor ou igual a \$b

5. Estruturas de Controle

5.1 - If and Else

Frequentemente, o programador necessitará testar o valor de uma variável para decidir ou não pela execução de uma tarefa. O comando utilizado para tal é o **if**.

O "If" é uma das mais importantes estruturas de controle de muitas linguagens. O PHP possui a sintaxe desse comando semelhante à linguagem C.

```
if (expressão) {
    //Código se a expressão for verdadeira
}
else
    {
        //Código se for falsa a expressão
    }
```

Exemplo:

```
If ($a==$b) {
    Echo ("A é igual a B.");
}
else
    {
        echo ("B é diferente de A.");
    }
```

```
}
```

Traduzindo: se \$a for igual a \$b então imprima na tela "A é igual a B". De outro modo, imprima "B diferente de A".

5.2 - Laço While

O primeiro laço disponível em PHP é o laço **while**. A sua forma geral é:

```
While (condição) {  
    //Bloco de comandos...  
}
```

Exemplo:

```
$aux=0;  
while ($aux<=10){  
    echo $aux;  
    $aux+;  
}
```

```
1 <?  
2 //Esta sequência imprime os números de 0 a 10.  
3 $aux=0;  
4 while ($aux<=10){  
5     echo $aux;  
6     $aux+=1;  
7 }  
8 ?>
```

5.3 - Laço for

Esse comando permite que determinado processo seja executado várias vezes.

Sua sintaxe é a seguinte:

```
for (inicio; fim; incremento) {  
    //Bloco de comandos...  
}
```

Exemplo:

```
//Imprime os números de 1 a 10 com incremento de 1 em 1
For ($contador=1; $contador<=10; $contador++) {
    Echo $contador;
}
```

1. A variável \$contador pode ser um número inteiro (integer) ou real (float), sendo a utilização de números inteiros mais frequente.
2. A variável contador pode ser inicializada com qualquer valor positivo, negativo ou zero.

5.4 - Comando break

O comando **break** é utilizado para forçar uma terminação imediata de um laço, evitando o teste condicional do laço.

Quando o comando **break** é encontrado dentro de um laço, o laço é imediatamente terminado e o controle do script retorna no comando seguinte.

5.5 - Switch

O PHP tem um comando interno de seleção múltipla, **switch**, que testa sucessivamente o valor de uma expressão contra uma lista de constantes inteiras ou de caracteres. Quando o valor coincide, os comandos associados àquela constante são executados.

A sintaxe desse comando é a seguinte:

```
switch (variável) {
    case valor1:
        //Bloco de comandos Comandos...
        break;
    case valor2:
        //Bloco de comandos Comandos...
        break;
    case valor3:
        //Bloco de comandos Comandos...
        break;
}
```

O padrão ANSI especifica que um switch pode ter pelo menos 257 comandos case. Na prática, você deve limitar o número de comandos case em uma quantidade menor, para obter mais eficiência. Embora case seja um rótulo, ele não pode existir sozinho, fora de um switch.

6. Funções

Funções são blocos de comandos executados independentemente do script. A qualquer momento da execução você poderá solicitar uma função. Você pode passar argumento para as funções realizarem operações especificadas em tempo de programação. A sintaxe básica de construção de funções em php é a seguinte:

```
function nome_função ($arg_1, $arg_2, ..., $arg_n) {  
    //Bloco de comandos...  
    return $valor_retorno;  
}
```

Importante: Toda função em PHP tem de ser construída antes da sua respectiva chamada, a fim de que o interpretador reconheça a solicitação do script. Caso contrário, uma mensagem de erro é retornada.

Exemplo de funções:

```
1 <?  
2 //função de média aritmética entre dois números  
3 function media($a,$b) {  
4     $valor_retorno=($a+$b)/2;  
5     return $valor_retorno;  
6 }  
7 echo media(1,2); #chamada da função com passagem de argumentos  
8 ?>
```

```

1 <?
2 //função para calcular a impedância de um circuito RLC
3 //A partir da frequência em rad/s, resistor, indutor
4 //e capacitor do circuito
5 function impedancia($angular,$resistor,$indutor,$capacitor) {
6     $XL=$angular*$indutor; #calculando a reatância indutiva
7     $XC=1/($angular*$capacitor); #reatância capacitiva
8     $impedancia=sqrt(($resistor*$resistor)+(($XL-$XC)*($XL-$XC)));
9     return $impedancia;
10 }
11 echo impedancia(3.77e2,160,230e-3,15e-6); #chamada da função
12
13 ?>

```

7. Gravando Cookies

Cookies são variáveis gravadas remotamente pelo browser do usuário. É muito útil na hora em que um usuário executa uma rotina de login no sistema. Você pode gravar o login e a senha dele e recuperá-los de acordo com o tempo especificado na função; se o parâmetro tempo não for especificado, o cookie será gravado até o fim da aplicação, ou seja, até que o usuário feche o browser. Qualquer cookie enviado por um cliente é automaticamente transformado numa variável PHP. A função para gravar cookies é a `setcookie()`, cuja sintaxe é a seguinte:

```
setcookie(string_nome,string_valor,tempo);
```

Exemplos

```
setcookie("usuario",
$senha,time()+3600); #Este cookie expira em 1 hora
```

```
setcookie("senha",
$senha); #este cookie expira no momento do fechamento do browser.
```

8. Recuperando Cookies

Os cookies gravados pelo script PHP ficam armazenados no array `$HTTP_COOKIE_VARS[]`, cujo índice é o string nome especificado na função `setcookie`.

Para recuperar um cookie proceda da seguinte maneira:

```
echo $HTTP_COOKIE_VARS["usuario"];
```

O exemplo anterior mostrará na página html o nome do usuário.

9. header(Location...)

Esta função permite que um script php redirecione para outra página. A sintaxe é:

```
header("Location: endereço");
```

O exemplo a seguir redireciona o script para a página *index.php3* localizada no servidor *server* e diretório *controle*:

```
header("Location: http://server/controle/index.php3");
```

10. MySQL Funções

Veremos as funções mais importantes de comunicação entre um script php e um servidor de Banco de Dados MySQL. Colocar todas as funções e hipóteses neste livro o tornaria tedioso e cansativo, estendendo demais a leitura. O propósito é impulsionar um programador para a tecnologia de Intranet. As demais funções e parâmetros não contidos neste livro ficam a critério do leitor pesquisar no manual do PHP, disponível também gratuitamente no site: <http://www.php.net>

10.1 - mysql_connect()

Esta função habilita uma conexão com o servidor de banco de dados MySQL. Há indispensável necessidade de chamada dessa função antes de qualquer transação na base de dados. Esta função retorna o número inteiro 1 se a conexão for bem sucedida, caso contrário retorna 0. A sintaxe dessa função:

```
mysql_connect("host","seu_login","sua_senha");
```

onde;

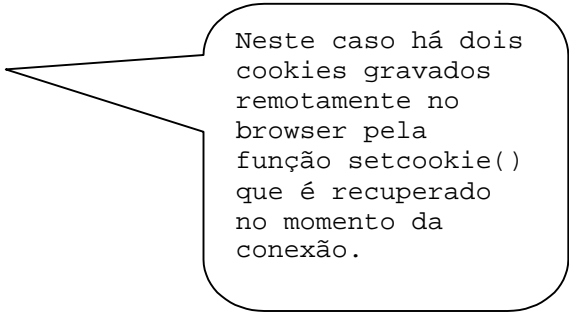
host - corresponde ao nome do computador que tem o servidor de Banco de Dados MySQL rodando. Pode ser o nome, por exemplo - localhost, ou o endereço de IP.

seu_login - é o nome de usuário do banco de dados

sua_senha - a senha do banco de dados

Exemplos do uso da função mysql_connect():

```
$conexao=mysql_connect("localhost","usuario","senha");  
mysql_connect("localhost",  
$HTTP_COOKIE_VARS["usuario"],  
$HTTP_COOKIE_VARS["senha"])
```



Neste caso há dois cookies gravados remotamente no browser pela função setcookie() que é recuperado no momento da conexão.

9.2 - mysql_select_db()

Esta função seleciona um banco de dados, retornando 1 se a transação foi bem sucedida e 0, caso contrário. Possui a seguinte sintaxe:

```
mysql_select_db("nome_do_db",inteiro_conexao);
```

onde;

nome_do_db - é o banco de dados que se quer utilizar.

inteiro_conexao - corresponde ao identificador retornado pela função `mysql_connect()`.

Exemplo:

```
<?
//Conectando com o MySQL Server
$conexao=mysql_connect("localhost","login","senha");
//Selecionando o banco de dados
$banco=mysql_select_db("usuarios",$conexao);
?>
```

10.3 - `mysql_query()`

Envia um comando SQL para o MySQL Server. Esta função retorna um identificador inteiro se o comando foi bem sucedido, caso contrário retorna o valor 0. A sintaxe desse comando é:

```
mysql_query(string_comando,inteiro_banco);
```

onde;

comando = string contendo o comando SQL.

inteiro_banco = identificador do banco de dados.

10.4 - `mysql_num_rows()`

Esta função retorna o número de linhas de uma query SQL.

Sintaxe:

```
mysql_num_rows(inteiro_query);
```

o parâmetro *inteiro_query* corresponde ao identificador de retorna da função `mysql_query()`.

10.5 - mysql_fetch_array()

Retorna um array contendo o resultado da query SQL.

Sintaxe:

```
mysql_fetch_array(inteiro_query);
```

onde *inteiro_query* corresponde ao identificador retornado pela função `mysql_query()`.

Exemplos:

```

1  <?
2  $conexao = mysql_connect('localhost',
3  $HTTP_COOKIE_VARS["usuario"],$HTTP_COOKIE_VARS["senha"]) or
4  die('Não consegui conectar com a base de dados');
5  //Selecionando a base de dados
6  $base=mysql_select_db('biblioteca',$conexao) or
7  die ('Não foi possível selecionar o banco de dados ...');
8  //Realizando a Consulta
9  $comando="select nome from alunos order by nome asc";
10 $consulta=mysql_query($comando,$base ) or die ('Não foi possível realizar a selecao
11 $aux=0;
12 echo '<select name="nome" size="10" multiple>';
13 while ($aux<mysql_num_rows($consulta)){
14     $campo=mysql_fetch_array($consulta);
15     echo '<option value="'. $campo[0]. ">'. $campo[0]. '</option>';
16     $aux=$aux+1;
17 }
18
19 echo '</select>';
20 ?>
```

11 - Trabalhando com Arquivos

Algumas vezes, o usuário precisa ler e gravar dados de arquivos no servidor. Por exemplo: um contador de página da web, seja gráfico ou modo texto, possui um arquivo guarda o resultado da última visita no site, um livro de visitas pode gravar os dados num arquivo de texto, um simples gerador de estatísticas de um site pode ser um arquivo texto contendo o IP, domínio e hora que o usuário solicitou àquela URL.

11.1 - Abrindo arquivos

Muito frequentemente, o usuário de uma aplicação desejará armazenar dados para posterior análise. Inicialmente, para que um arquivo possa ser manipulado, ele precisa ser aberto ou criado. Para isso, vamos utilizar o comando **fopen**.

O comando **fopen**, semelhante à linguagem C, retorna um identificador inteiro se a operação for bem sucedida ou 0, caso contrário. Esta função possui a seguinte sintaxe:

```
int fopen("arquivo",atributo);
```

Onde;

arquivo - string contendo o nome do arquivo

Modo - um dos especificadores abaixo.

R	Abre o arquivo com permissão apenas para leitura.
R+	Abre o arquivo com permissão para escrita e leitura, posicionando o ponteiro no início do mesmo.
W	Abre o arquivo com permissão apenas para escrita. Se o arquivo existir, todo o conteúdo é apagado. Se não existir, o PHP tenta criá-lo. O ponteiro é posicionado no início do arquivo
W+	Abre o arquivo com permissão para escrita e leitura. Se o arquivo existir, todo o conteúdo é apagado. Se não existir, o PHP tenta criá-lo. O ponteiro é posicionado no início do arquivo
A	Abre o arquivo com permissão apenas para escrita. Se o arquivo não existir, o PHP tenta criá-lo. O ponteiro é posicionado no final do arquivo
A+	Abre o arquivo com permissão para escrita e leitura. Se o arquivo não existir, o PHP tenta criá-lo. O ponteiro é posicionado no final do arquivo.

11.2 - Lendo Arquivos

O comando utilizado para leitura é o **fread**, cuja sintaxe é:

```
string fread(id,tamanho);
```

onde

id - corresponde ao identificador retornado pelo comando `fopen`.

tamanho - tamanho do arquivo a ser lido. Você poderá especificar a função `filesize("nome_arquivo")` neste parâmetro para indicar que corresponde ao valor máximo.

Exemplo de leitura de arquivo:

```
1 <?
2 //Nome do arquivo de texto
3 $arquivo='contador.txt';
4 //Testa se o arquivo existe
5 if (file_exists($arquivo)) {
6     //Abre o arquivo retornando o identificador $id
7     $id = fopen($arquivo, "r");
8     //le a informacao do arquivo
9     $conteudo = fread($id,filesize($arquivo));
10    //Imprime o conteúdo
11    echo $conteudo;
12    //Fecha arquivo
13    fclose($id);
14 }
15 else
16 {
17     echo ('Impossível localizar arquivo: contador.txt');
18 }
19
20 ?>
```

11.3 - Gravando Dados

Para gravar dados em arquivos utiliza-se o comando `fputs`. A sintaxe desse comando é:

```
int fputs(id,valor);
```

onde

id - corresponde ao identificador do arquivo

valor - o valor a ser armazenado no arquivo.

11 - Conclusão

Os comandos vistos anteriormente, juntamente com os capítulos anteriores, fornecem embasamento suficiente para qualquer programador inserir, alterar, apagar e recuperar dados de um MySQL Server através de scripts PHP. Os demais comandos poderão ser encontrados na documentação oficial no site <http://www.php.net>. O principal objetivo deste livro é fornecer conhecimento suficiente para que um programador se familiarize com a nova tecnologia.



BIBLIOGRAFIA

- 1 - LERDORF, Rasmus - PHP Manual, 1999 by PHP documentation group.
- 2 - CÔRTEZ, Pedro - C Auto Explicativo, São Paulo, Érica Editora Ltda 1992.
- 3 - SSHILDT, Herbert, C Completo e Total, São Paulo: Makron, McGraw-Hill, 1990.
- 4 - MySQL Manual - <http://www.tcx.se>, Suécia.

APENDICE A

FUNÇÕES MATEMÁTICAS DO MySQL

ABS(X)	Retorna o valor absoluto do número X
FLOOR(X)	Retorna o maior valor inteiro de um número. <pre> Mysql> select FLOOR(1.23); -> 1 mysql> select FLOOR(-1.23); -> -2 </pre>
ROUND(X)	Arredonda o argumento 'X', retornando um inteiro. <pre> mysql> select ROUND(-1.23); -> -1 mysql> select ROUND(-1.58); -> -2 mysql> select ROUND(1.58); -> 2 </pre>
LOG(X)	Retorna o logaritmo natural de 'X'
SQRT(X)	Retorna a raiz quadrada de 'X'. <pre> Mysql> select SQRT(4); -> 2.000000 mysql> select SQRT(20); -> 4.472136 </pre>
PI()	Retorna o valor de Pi <pre> Mysql> select PI(); -> 3.141593 </pre>
COS(X)	Retorna o cosseno de 'X'. Obs: X em radianos <pre> Mysql> select COS(PI()); -> -1.000000 </pre>

UPE - POLI - Engenharia Eletrônica

SIN(X)	Retorna o seno de `X`. Obs: X em radianos Mysql> select SIN(PI()); -> 0.000000
TAN(X)	Retorna a tangente de X. Obs: X em radianos.
ACOS(X)	Retorna o arco-cosseno de `X`.
ASIN(X)	Retorna o arco-seno de X.
ATAN(X)	Retorna o arco-tangente de X
DEGREES(X)	Converte o argumento X de radianos para graus.
RADIANS(X)	Converte o argumento X de graus para radianos.

APÊNDICE B

FUNÇÕES DE STRINGS DO MySQL

Função	Retorno
LOWER(str)	Força caracteres maiúsculos a aparecerem minúsculos.
UPPER(str)	Força caracteres minúsculos aparecerem maiúsculos.
CONCAT(str1,str2,...)	Concatena os strings
SUBSTRING(str,pos,len)	Extraí um substring começando em pos e terminando em len.
LTRIM(str)	Extraí espaços em branco à esquerda do string
RTRIN(str)	Extraí espaços em branco à direita do string
TRIM(str)	Extraí espaços em branco à esquerda e à direita do string.

APÊNDICE C

FUNÇÕES DE DATA E HORA DO MySQL

Função	Retorno
WEEKDAY(data)	Dia da semana (`0' = Monday, `1' = Tuesday, ... `6' = Sunday). mysql> select WEEKDAY('1997-10-04 22:23:00'); <div style="text-align: right;">-> 5</div> mysql> select WEEKDAY('1997-11-05'); <div style="text-align: right;">-> 2</div>
DAYOFMONTH(data)	Dia do mês. Assume valores de 1 a 31. mysql> select DAYOFMONTH('1998-02-03'); <div style="text-align: right;">-> 3</div>
DAYOFTIME(data)	Dia do ano. Assume valores de 1 a 366.
MONTH(data)	Mês do ano. Assume valores de 1 a 12
YEAR(data)	Ano referente a data. Assume valores entre 1000 a 9999
DATE_FORMAT(data,formato) Valores para o formato `%W' Dia da Semana `%Y' Ano, numerico, 4 digitos `%y' Ano, numerico, 2 digitos `%d' Dia do mês, numérico. (`00'..'31') `%m' Mês, numerico	Formata uma data. Exemplos: mysql> select DATE_FORMAT('1997-10-04 22:23:00', '%d-%m-%Y'); <div style="text-align: right;">-> '04-10-1997'</div> mysql> select DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s'); <div style="text-align: right;">-> '22:23:00'</div>

UPE - POLI - Engenharia Eletrônica

<code>(`01'..'12')</code>	
<code>CURDATE()</code>	Data atual. Formato (yyyy-mm-dd)
<code>CURTIME()</code>	Hora atual.
<code>NOW()</code>	Data e hora atual. mysql> select NOW(); -> '2000-08-01 23:50:26'

APÊNDICE D

OUTRAS FUNÇÕES IMPORTANTES DO MySQL

DATABASE()	Banco de dados corrente
USER()	Usuário corrente
VERSION()	String contendo a versão do MySQL Server
COUNT(expr)	Número de vezes que acontece a expressão
AVG(expr)	Média aritmética entre os valores da expressssão
MIN(expr)	Menor valor da expressão
MAX(expr)	Maior valor da expressão
SUM(expr)	Somatório da expressão